Markus Krötzsch
Umberto Straccia (Eds.)

LNCS 7497

Web Reasoning and Rule Systems

6th International Conference, RR 2012
Vienna, Austria, September 2012
Proceedings

RR2012

Springer

Markus Krötzsch   Umberto Straccia (Eds.)

Web Reasoning and Rule Systems

6th International Conference, RR 2012
Vienna, Austria, September 10-12, 2012
Proceedings

Springer

Volume Editors

Markus Krötzsch
University of Oxford
Department of Computer Science
Parks Road, Wolfson Building
Oxford OX1 3QD , UK
E-mail: markus.kroetzsch@cs.ox.ac.uk

Umberto Straccia
ISTI - CNR
Via G. Moruzzi, 1
56124 Pisa, Italy
E-mail: straccia@isti.cnr.it

ISSN 0302-9743 e-ISSN 1611-3349
ISBN 978-3-642-33202-9 e-ISBN 978-3-642-33203-6
DOI 10.1007/978-3-642-33203-6
Springer Heidelberg Dordrecht London New York

Library of Congress Control Number: 2012945946

CR Subject Classification (1998): D.4, H.2.4, I.2.4, H.3.5, H.5, D.1.6

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

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)

Preface

Web reasoning is the task of reasoning over Web-accessible data, which is typically heterogeneous, distributed, uncertain, contradictory, and – most of all – large. This leads to major challenges, which are critical not only for the Semantic Web, but also for modern information systems in general. A fruitful field of research has developed around the integration of rule and ontology languages, their computational analysis, and the development of robust and scalable reasoning algorithms. More recently, there is also renewed interest in the use of rules and ontologies in database applications, as witnessed by many new works on technologies such as Ontology-Based Data Access, existential rules, query rewriting, and even Datalog optimization.

The International Conference on Web Reasoning and Rule Systems has developed into a major forum for discussion and dissemination of new results on relevant topics within this area. Since the first edition of the conference in 2007, RR papers have been addressing a wide variety of problems in Web reasoning and rule systems, spanning research areas from computational logic and artificial intelligence to databases and Web technology. The conference is devoted to both theoretical and practical aspects of these subjects, and continues to attract the world's leading experts in the field. Further information can be found on the RR website at http://www.rr-conference.org/.

This volume contains the papers presented at RR 2012: The 6th International Conference on Web Reasoning and Rule Systems, held in Vienna, Austria, during September 10–12, 2012. The conference received 42 submissions (involving authors from 19 countries), of which 33 were research papers (RPs), and nine were technical communications (TCs), a more concise paper format that was introduced for the first time this year. TCs provide the opportunity to present a wider range of results and ideas that are of interest to the RR audience, including reports about recent own publications, position papers, and previews of ongoing work. In a rigorous reviewing process, 12 RPs were selected for publication (36% of 33 submitted RPs), while another 15 submissions were accepted as TCs.

The program of RR 2012 featured invited talks by Gerhard Brewka, Tommaso Di Noia, Phokion Kolaitis, and Robert Kowalski. Extended abstracts of the talks of Gerhard Brewka and Tommaso Di Noia are included in this volume. The talk of Phokion Kolaitis was a joint keynote with the Second Datalog 2.0 Workshop; the talk of Robert Kowalski was enabled in cooperation with the 4th International Conference on Computational Models of Argument (COMMA 2012) and Datalog 2.0. We want to thank all invited speakers for their contribution.

This year's conference also continued the successful cooperation with the Reasoning Web Summer School, which was again held in the week before RR. A Doctoral Consortium provided an additional opportunity for doctoral students in the area of Web reasoning to present their work and research plans, and to receive extensive feedback from leading scientist in the field. This volume includes five research summaries of doctoral students that were selected for presentation at the conference.

RR 2012 received the largest number of submissions since the first RR in 2007, while also featuring one of the shortest reviewing periods in the history of the conference. Managing this was only possible with the help of an exceptionally diligent and professional Program Committee. Every submission received at least three reviews, sometimes after extensive internal discussions. We would like to thank all reviewers for their invaluable efforts.

A number of people were involved in organizing this conference. We would like to thank our General Chair, Pascal Hitzler, as well as Alessandra Mileo, who chaired the Doctoral Consortium, and Francesco Calimeri, who acted as a Publicity Chair. The local organizers Thomas Eiter and Thomas Krennwallner and their team did a magnificent job with the organization of the event. We further thank all organizers of the co-located events COMMA, DEXA, Datalog 2.0, and the Reasoning Web Summer School, who supported the orchestration of these events in the context of the Vienna Logic Weeks 2012. We also thank Marco Maratea for his very successful work as Sponsorship Chair, and we gratefully acknowledge the support of our sponsors. In particular, we thank the main conference sponsors: the *Artificial Intelligence Journal*, the Association for Logic Programming, the Digital Enterprise Research Institute at the National University of Ireland in Galway, the *Dipartimento di Informatica, Sitemistica e Telematica* of the University of Genoa (DIST), the National Science Foundation, Siemens AG Austria, the Vienna Center for Logic and Algorithms, and the Wolfgang Pauli Institute Vienna. As usual, EasyChair provided a convenient and efficient platform for preparing the program and these proceedings. Finally, thanks are due to all authors and participants of RR 2012; we hope that their stay in Vienna was profitable and enjoyable.

September 2012

Markus Krötzsch
Umberto Straccia

Organization

General Chair

Pascal Hitzler	Kno.e.sis Center, Wright State University, USA

Program Chairs

Markus Krötzsch	University of Oxford, UK
Umberto Straccia	ISTI-CNR, Italy

Local Organization

Thomas Eiter	Vienna University of Technology, Austria
Thomas Krennwallner	Vienna University of Technology, Austria

Doctoral Consortium Chair

Alessandra Mileo	DERI, National University of Ireland, Galway

Publicity Chair

Francesco Calimeri	University of Calabria, Italy

Sponsorship Chair

Marco Maratea	University of Genoa, Italy

Program Committee

Darko Anicic	FZI Forschungszentrum Informatik, Germany
Jean-François Baget	INRIA *and* LIRMM, France
Chitta Baral	Arizona State University, USA
Fernando Bobillo	University of Zaragoza, Spain
Gerhard Brewka	Leipzig University, Germany
François Bry	University of Munich, Germany
Diego Calvanese	Free University of Bozen-Bolzano, Italy
Vinay Chaudhri	SRI International, USA
Kendall Clark	Clark & Parsia LLC, USA

Tommaso Di Noia	Politecnico di Bari, Italy
Francesco M. Donini	Università della Tuscia, Italy
Thomas Eiter	Vienna University of Technology, Austria
Michael Fink	Vienna University of Technology, Austria
Birte Glimm	Universität Ulm, Germany
Claudio Gutierrez	Universidad de Chile, Chile
Andreas Harth	Karlsruhe Institute of Technology, Germany
Aidan Hogan	DERI, National University of Ireland, Galway
Giovambattista Ianni	University of Calabria, Italy
Yevgeny Kazakov	Universität Ulm, Germany
Michael Kifer	State University of New York at Stony Brook, USA
Clemens Kupke	University of Oxford, UK
Georg Lausen	University of Freiburg, Germany
Domenico Lembo	Università di Roma “La Sapienza”, Italy
Francesca Alessandra Lisi	Università degli Studi di Bari, Italy
Thomas Lukasiewicz	University of Oxford, UK
Wolfgang May	Universität Göttingen, Germany
Alessandra Mileo	DERI, National University of Ireland, Galway
Ralf Moeller	Hamburg University of Technology, Germany
Marie-Laure Mugnier	LIRMM, France
Magdalena Ortiz	Vienna University of Technology, Austria
Jeff Z. Pan	University of Aberdeen, UK
Rafael Peñaloza	TU Dresden, Germany
Axel Polleres	Siemens AG Österreich, Austria *and* DERI, Ireland
Andrea Pugliese	University of Calabria, Italy
Guilin Qi	Southeast University, China
Sebastian Rudolph	Karlsruhe Institute of Technology, Germany
Steven Schockaert	Cardiff University, UK
Mantas Simkus	Vienna University of Technology, Austria
Sergio Tessaris	Free University of Bozen-Bolzano, Italy
Andrei Voronkov	University of Manchester, UK

Additional Reviewers

Francesco Calimeri
Simona Colucci
Felix Distel
Jianfeng Du
Cristina Feier
Marcel Karnstedt
Marcel Lippmann
Maria Vanina Martinez
Boris Motik
Nadeschda Nikitina
Philipp Obermeier
Yuan Ren
Patrik Schneider
František Simančík
Andreas Steigmiller
Yuting Zhao
Christoph Redl

Doctoral Consortium Program Committee

Pedro Cabalar	Corunna University, Spain
Diego Calvanese	Free University of Bozen-Bolzano, Italy
Thomas Eiter	Vienna University of Technology, Austria
Wolfgang Faber	University of Calabria, Italy
Claudio Gutierrez	Universidad de Chile, Chile
Pascal Hitzler	Kno.e.sis Center, Wright State University, USA
Tommie Meyer	CSIR Meraka *and* University of KwaZulu-Natal, South Africa
Jeff Z. Pan	University of Aberdeen, UK
Axel Polleres	Siemens AG Österreich, Austria *and* DERI, Ireland
Terrance Swift	State University of New York at Stony Brook, USA
Hans Tompits	Vienna University of Technology, Austria

Sponsors

Platinum Sponsors

Artificial Intelligence Journal

Association for Logic Programming

Digital Enterprise Research Institute,
National University of Ireland, Galway

Dipartimento di Informatica,
Sistemistica e Telematica;
Università Di Genova

National Science Foundation

Siemens AG Österreich

Vienna Center for
Logic and Algorithms

Wolfgang Pauli Institute

Silver Sponsors

DLVSYSTEM s.r.l.

IOS Press

Regular Sponsors

Kurt Gödel Society

Planet Data project

Doctoral Consortium Sponsors

Digital Enterprise Research Institute

National Science Foundation

Reasoning Web Summer School

Table of Contents

Multi-context Systems: Specifying the Interaction of Knowledge Bases Declaratively

Gerhard Brewka

Leipzig University, Informatics Institute, Postfach 100920, 04009 Leipzig, Germany
brewka@informatik.uni-leipzig.de

1 Motivation

Research in knowledge representation and, more generally, information technology has produced a large variety of formats and languages for representing knowledge. A wealth of tools and formalisms is now available, including rather basic ones like databases or the more recent triple-stores, and more expressive ones like ontology languages (e.g., description logics), temporal and modal logics, nonmonotonic logics, or logic programs under answer set semantics, to name just a few.

The diversity of formalisms poses some important challenges. Here is a simple illustrative example. Assume your home town's hospital has

– a patient database (e.g. an Oracle database),
– a disease ontology (written in a particular description logic),
– an ontology of the human body (using OWL),
– an expert system describing the effects of different medications (using a nonmonotonic reasoning formalism, say disjunctive logic programming).

There may be situations where the integration of the knowledge represented in such diverse formalisms is crucial. But how can this be achieved? Translating everything into a single all-purpose formalism is certainly not a solution. First of all, no standardized, universal knowledge representation language exists, and there are very good reasons for this (e.g. specific modeling needs or complexity considerations). Secondly, even if there were such a language, most probably remodeling the information would be too cumbersome and costly. What seems to be needed is a principled way of integrating knowledge expressed in different formats/languages/logics.

Nonmonotonic multi-context systems (MCS) [1] provide a promising way to address this issue. The basic idea is to leave the diverse formalisms and knowledge bases (called contexts in this approach for historical reasons [7]) untouched, and to equip each context with a collection of so-called bridge rules in order to model the necessary information flow among contexts.

Bridge rules are similar to logic programming rules (including default negation), with an important difference: they allow to access other contexts in their bodies. Using bridge rules has several advantages: the specification of the information flow is fully declarative; moreover, information - rather than simply being passed on as is - can be modified in various ways:

- we may translate a piece of information into the language/format of another context
- we may pass on an abstraction of the original information, leaving out unnecessary details
- we may select or hide information
- we may add conclusions to a context based on the absence of information in another one
- we may use simple encodings of preferences among parent contexts
- we can even encode voting rules, say based on majorities etc.

The semantics of MCS is defined in terms of equilibria: a belief state assigns a belief set to each context C_i. Intuitively, a belief state is an equilibrium whenever the belief set selected for each C_i is acceptable for C_i's knowledge base *augmented by the heads of C_i's applicable bridge rules.*

The history of MCS started in Trento. Advancing work in [6,8], the Trento School developed monotonic heterogeneous multi-context systems [7] with the aim to integrate different inference systems. Here reasoning within as well as across contexts is monotonic. The first, still somewhat limited attempts to include nonmonotonic reasoning were done in [9] and [4]. To allow for reasoning based on the *absence* of information from a context, in both papers default negation is allowed in the rules. In this way contextual and default reasoning are combined.

The nonmonotonic MCS of [1] substantially generalized these approaches, by accommodating *heterogeneous* and both *monotonic* and *nonmonotonic* contexts. They are thus capable of integrating "typical" monotonic logics like description logics or temporal logics, and nonmonotonic formalisms like Reiter's default logic, answer set programs, circumscription, defeasible logic, or theories in autoepistemic logic. The currently most general MCS variant, the so-called managed MCS (mMCS) [3] allow for arbitrary user-defined operations on the context knowledge bases, not just augmentations.

2 Nonmonotonic MCS More Formally

Multi-context systems as defined in [1] build on an abstract notion of a *logic* L as a triple (KB_L, BS_L, ACC_L), where KB_L is the set of admissible knowledge bases (KBs) of L, which are sets of KB-elements ("formulas"); BS_L is the set of possible belief sets, whose elements are beliefs; and $ACC_L : KB_L \to 2^{BS_L}$ is a function describing the semantics of L by assigning to each knowledge-base a set of acceptable belief sets.

A *multi-context system (MCS)* $M = (C_1, \ldots, C_n)$ is a collection of contexts $C_i = (L_i, kb_i, br_i)$ where L_i is a logic, $kb_i \in KB_{L_i}$ is a knowledge base and br_i is a set of bridge rules of the form:

$$s \leftarrow (c_1{:}p_1), \ldots, (c_j{:}p_j), not(c_{j+1}{:}p_{j+1}), \ldots, not(c_m{:}p_m). \qquad (1)$$

such that $kb \cup \{s\}$ is an element of KB_{L_i}, $c_\ell \in \{1, \ldots, n\}$, and p_ℓ is element of some belief set of BS_{c_ℓ}, for all $1 \leq \ell \leq m$. For a bridge rule r, we denote by $hd(r)$ the formula s while $body(r)$ denotes the set $\{(c_{\ell_1}{:}p_{\ell_1}) \mid 1 \leq \ell_1 \leq j\} \cup \{not(c_{\ell_2}{:}p_{\ell_2}) \mid j < \ell_2 \leq m\}$.

A belief state $S = (S_1, \ldots, S_n)$ for M consists of belief sets $S_i \in BS_i$, $1 \leq i \leq n$. A bridge rule r of form (1) is applicable wrt. S, denoted by $S \models \mathit{body}(r)$, iff $p_\ell \in S_{c_\ell}$ for $1 \leq \ell \leq j$ and $p_\ell \notin S_{c_\ell}$ for $j < \ell \leq m$. We use $\mathit{app}_i(S) = \{\mathit{hd}(r) \mid r \in \mathit{br}_i \wedge S \models \mathit{body}(r)\}$ to denote the heads of all applicable bridge rules of context C_i wrt. S.

The semantics of an MCS M is then defined in terms of equilibria, where an *equilibrium* is a belief state $(S_1, \ldots, S_n)$ such that $S_i \in \mathit{ACC}_i(\mathit{kb}_i \cup \mathit{app}_i(S))$, $1 \leq i \leq n$.

3 Beyond Information Flow

Although nonmonotonic MCS are, as we believe, an excellent starting point to address the problems discussed above, the way they integrate knowledge is still somewhat limited: if a bridge rule for a context is applicable, then the rule head is simply added to the context's knowledge base (KB). Although this covers the flow of information, it does not capture other operations one may want to perform on context KBs. For instance, rather than simply *adding* a formula ϕ, we may want to delete some information, or to *revise* the KB with ϕ to avoid inconsistency in the context's belief set. We are thus interested in generalizations of the MCS approach where specific predefined operations on knowledge bases can be performed.

A first step into this direction are argumentation context systems (ACS) [2]. They specialize MCS in one respect, and are more general in another. First of all, in contrast to nonmonotonic MCS they are homogeneous in the sense that all reasoning components in an ACS are of the same type, namely Dung-style argumentation frameworks [5]. The latter are widely used as abstract models of argumentation. However, ACS go beyond MCS in two important aspects: (1) The influence of an ACS module M_1 on another module M_2 can be much stronger than in an MCS. M_1 may not only provide information for M_2 and thus augment the latter, it may directly affect M_2's KB and reasoning mode: M_1 may invalidate arguments or attack relationships in M_2's argumentation framework, and even determine the semantics to be used by M_2. (2) A major focus in ACS is on *inconsistency handling*. Modules are equipped with additional components called *mediators*. The main role of the mediator is to take care of inconsistencies in the information provided by connected modules. It collects the information coming in from connected modules and turns it into a consistent update specification for its module, using a pre-specified consistency handling method which may be based on preference information about other modules.

Managed MCS (mMCS) push the idea of mediators even further. They allow additional operations on knowledge bases to be freely defined; this is akin to management functionality of database systems. We thus call the additional component *context manager*. In a nutshell (and somewhat simplifying) the features of mMCS are as follows:

- Each logic comes with a set of operations O.
- An operational statement is an operation applied to a formula (e.g. insert(p), delete(p), revise(p), ...).
- Bridge rules are as before, except for the heads which now are operational statements.
- A management function: $mng : 2^{Opst} \times KB \to 2^{KB}$, produces a collection of KBs out of set of operational statements and a KB.

- A managed context consists of a logic, a KB, a set of bridge rules (as before), together with the new part, a management function.
- An mMCS is just a collection of managed contexts.

Regarding the semantics, a belief state $S = (S_1, \dots S_n)$ contains - as before - a belief set for each context. To be an equilibrium S has to satisfy the following condition: the belief set chosen for each context must be acceptable for one of the KBs obtained by applying the management function to the heads of applicable bridge rules and the context's KB. More formally, for all contexts $C_i = (L_i, kb_i, br_i, mng_i)$: let S_i be the belief set chosen for C_i, and let Op_i be the heads of bridge rules in br_i applicable in S. Then S is an equilibrium iff, for $1 \leq i \leq n$,

$$S_i \in ACC_i(kb') \text{ for some } kb' \in mng_i(Op_i, kb_i).$$

Management functions allow us to model all sorts of modifications of a context's knowledge base and thus make mMCS a powerful tool for describing the influence contexts can have on each other.

4 Short CV

After studies in Bonn Gerhard Brewka received his PhD from University of Hamburg in 1989. After several years as a researcher at GMD, Sankt Augustin, and a year as a postdoc at ICSI, Berkeley, he became full professor for Knowledge Based Systems at TU Vienna in 1995 and professor for Intelligent Sytems in Leipzig in 1996. His research interests are in knowledge representation, in particular nonmonotonic reasoning, logic programming, preference and inconsistency handling and argumentation. Brewka was PC-chair of ECAI-2006, LPNMR-2007 and KR-2008 and general chair of KR-2012. He is an ECCAI fellow since 2002 and currently president of ECCAI and of KR, inc.

References

1. Brewka, G., Eiter, T.: Equilibria in heterogeneous nonmonotonic multi-context systems. In: Proc. AAAI 2007, pp. 385–390. AAAI Press (2007)
2. Brewka, G., Eiter, T.: Argumentation Context Systems: A Framework for Abstract Group Argumentation. In: Erdem, E., Lin, F., Schaub, T. (eds.) LPNMR 2009. LNCS, vol. 5753, pp. 44–57. Springer, Heidelberg (2009)
3. Brewka, G., Eiter, T., Fink, M., Weinzierl, A.: Managed multi-context systems. In: Proc. IJCAI 2011, pp. 786–791 (2011)
4. Brewka, G., Roelofsen, F., Serafini, L.: Contextual default reasoning. In: Proc. IJCAI 2007, pp. 268–273 (2007)
5. Dung, P.M.: On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games. Artif. Intell. 77(2), 321–358 (1995)
6. Giunchiglia, F.: Contextual reasoning. Epistemologia XVI, 345–364 (1993)
7. Giunchiglia, F., Serafini, L.: Multilanguage hierarchical logics or: How we can do without modal logics. Artif. Intell. 65(1), 29–70 (1994)
8. McCarthy, J.: Generality in artificial intelligence. Commun. ACM 30(12), 1029–1035 (1987)
9. Roelofsen, F., Serafini, L.: Minimal and absent information in contexts. In: Proc. IJCAI 2005 (2005)

Semantic Matchmaking and Ranking: Beyond Deduction in Retrieval Scenarios

Tommaso Di Noia

SisInf Lab – DEE – Politecnico di Bari – Via E. Orabona, 4, 70125 Bari, Italy
t.dinoia@poliba.it

Extended Abstract

Matchmaking can be basically seen as the process of computing a ranked list of resources with respect to a given query. *Semantic matchmaking* can be hence described as the process of computing such ordered list also taking into account the semantics of resources description and of the query, provided with reference to a logic theory (an ontology, a set of rules, etc.) [3]. A matchmaking step is fundamental in a number of retrieval scenarios spanning from (Web) service discovery and composition to e-commerce transactions up to recruitment in human resource management for task assignment, just to cite a few of them. Also in interactive exploratory tasks, matchmaking and ranking play a fundamental role in the selection of relevant resources to be presented to the user and, in case, further explored. In all the above mentioned frameworks, the user query may contain only hard (strict) requirements or may represent also her preferences. We will see how to handle both cases while computing the ranked list of most promising resources.

In semantic-based retrieval scenarios we have a logic theory represented, for example, as a set of ontological axioms that logically constrain the query and the description of a resource. If we consider both the query q and the description d expressed as Description Logics (DL)[1] statements (the logic language behind the semantics of `OWL`) and an ontology O we may check if the information encoded in d implies the one encoded in q or we may verify if d contains the description of some characteristics that are conflicting with the query. Let us consider the following simple example. Suppose we have a query q and four descriptions d_1, d_2, d_3 and d_4 such that the following relations hold[2]:

$$d_1 \sqcap q \not\sqsubseteq_O \bot \qquad d_2 \sqcap q \not\sqsubseteq_O \bot \qquad d_3 \sqcap q \sqsubseteq_O \bot \qquad d_4 \sqsubseteq_O q$$

By looking at the above relations we may say that, for sure, d_4 is the best choice for q since it *fully* satisfies the requirements expressed by the query. On the other hand, if we rely on deductive reasoning tasks, regarding d_1, d_2 and d_3 we may only say that the first two are compatible with q and the third one is not

1 In the following we will use DL notation to present our framework but the approaches can be easily adapted to other logic formalisms.

2 We use the notation $A \sqsubseteq_O B$ to represent $O \models A \sqsubseteq B$.

compatible with q since it contains some elements that make $q \sqcap d_3$ unsatisfiable. If we adopt an Open World Assumption (which is reasonable in many Web scenarios), we may say that d_1 and d_2 *potentially* satisfy q. After all, we do not have information on what is underspecified in both d_1 and d_2. Maybe, the one who wrote the description (imagine an advertisement on eBay) just did not care about some information to be explicitly represented. On the other hand, even though d_3 is conflicting with q, it may still contain some characteristics that have been expressed in q. In other words, d_3 *partially* satisfies q. The main questions here are: How do we rank d_1, d_2 and d_3 with respect to q? Is the classification in *potential* and *partial* matches[3] enough for ranking purposes? May we compute a similarity degree of d_1, d_2 and d_3 with respect to q based on their formal semantics [7]?

Our proposal is to use two non standard reasoning tasks, namely Concept Abduction and Concept Contraction [5], to compute an explanation on “the reason why” d does not *fully* satisfy q and then compute a score based on this explanation [4,6]. In a few words, we want to know the reason why $d \sqsubseteq_O q$ does not hold.

Concept Abduction. As for the Propositional case, the main aim of this task is to compute a (minimal) hypothesis on what is underspecified in d in order to *fully* satisfy q. More formally, given d, q and O such that $d \sqcap q \not\sqsubseteq_O \bot$, we say that h is a solution to a **C**oncept **A**bduction **P**roblem $\langle q, d, O\rangle^{CAP}$ if the following two conditions hold: $d \sqcap h \not\sqsubseteq_O \bot$ and $d \sqcap h \sqsubseteq_O q$. By solving a CAP on all those descriptions that *potentially* satisfy q, we have an explanation for *non-full* match. At this point, we may define a scoring function that assigns a weight to h, given q, d and O. Actually, the above formulation shows its limits with expressive languages and a more sophisticated definition is needed [1,8,2].

Concept Contraction. In case of partial match, we are interested in computing which part of the query q is conflicting with d and suggest to the user to revise and relax the query if she is interested in d. More formally, given d,q and O such that $d \sqcap q \sqsubseteq_O \bot$ we say that $\langle g, k\rangle$ is a solution to a **C**oncept **C**ontraction **P**roblem $\langle q, d, O\rangle^{CCP}$ if the following two conditions hold: $g \sqcap k \equiv_O q$ and $k \sqcap d \not\sqsubseteq_O \bot$. In a few words, g represents what has to be given up from q while k what has to be kept in order to *potentially* match q. Also in this case we may define a scoring function that assigns a weight to g and k, given q, d and O. Both for h and $\langle g, k\rangle$, some minimality criteria need to be defined in order to avoid trivial solutions.

Looking at the previous two reasoning tasks we observe that they could be used to move from a *partial* match to a *potential* match and from a *potential* match to a *full* match by solving a CCP and then a CAP. Indeed, we have the following matchmaking “path”:

$$\underset{\textbf{\textit{partial}}}{q \sqcap d \sqsubseteq_O \bot} \xrightarrow{\langle g,k\rangle = solve\ \langle q,d,O\rangle^{CCP}} \underset{\textbf{\textit{potential}}}{k \sqcap d \not\sqsubseteq_O \bot} \xrightarrow{h = solve\ \langle k,d,O\rangle^{CAP}} \underset{\textbf{\textit{full}}}{d \sqcap h \sqsubseteq_O k}$$

[3] These two match classes are also known as *intersection* and *disjoint* respectively [10].

We notice two important aspects related to h, g and k:

– they represent an **explanation** on the reason why d is not a *full* match for q. This is a very relevant aspect of modern retrieval systems and it is a very hot topic, for example, in the field of Recommender Systems [16].
– they can be used as a metric to compute a **semantic distance** of d from q. Hence we can use them all to evaluate the score we will use to rank a set of descriptions with respect to the query.

So far, we have represented the query as a single formula without taking into account user preferences. In fact, it may happen that a user is more satisfied by the retrieved d depending on which part of her query q is satisfied: the user may assign a different *utility* value to different parts of q. In these situations we need to borrow some notions and ideas from Utility Theory [9] to perform an efficient retrieval task. To model user preferences and their associated utility we represent them as weighted formulas. A preference is then formulated as $P = \langle p, v \rangle$, where p is a logic formula and v is a numerical value representing the utility gained by the user when d satisfies p. Given a set of preferences $\mathcal{P} = \{\langle p_i, v_i \rangle\}$ and a resource description d we define a utility function, in its basic form, as

$$u'(\mathcal{P}, d) = \sum \{v_i \mid \text{where } d \text{ satisfies } p_i\}$$

Actually, we still need to define what "where d satisfies p_i" really means. The most intuitive way of modelling such relation is $d \sqsubseteq_O p_i$. Unfortunately, this formulation may lead to some counter-intuitive situations. Suppose to have $d = A_1 \sqcup A_2$, $P_1 = \langle A_1, v_1 \rangle$ and $P_2 = \langle A_2, v_2 \rangle$. In this case, neither $d \sqsubseteq A_1$ nor $d \sqsubseteq A_2$ and then $u'(\{P_1, P_2\}, d) = 0$. This is not correct since we know, by the formal semantics of $\sqcup$, that d will satisfy A_1 or A_2 or them both. Hence, the final utility will be at least $min(\{v_1, v_2\})$. In order to solve such problems we need a more sophisticated utility function that takes into account the models of the formula d. Given an interpretation $\mathcal{I}$ of d, *i.e.*, $d^{\mathcal{I}} \neq \emptyset$, we say that $\mathcal{I}$ is a minimal model of d if the value

$$u^{min}(\mathcal{P}, \mathcal{I}) = \sum \{v_i \mid \langle p_i, v_i \rangle \in \mathcal{P} \text{ and } \mathcal{I} \models p_i\}$$

is minimal. In this case, we call $u^{min}(\mathcal{P}, \mathcal{I})$ a **minimal utility value** associated to d with respect to $\mathcal{P}$. As we would expect, things become a little bit more complicated when we need to consider also an ontology [14,15].

In all the proposed examples we always had only one active user who formulated the query q (as a set of preferences). Nevertheless, we can model negotiating actors where both the user who expresses q and the one who describes d have an active role in the matchmaking and they both want to maximize their expected utility. Imagine the case of an online marketplace where the seller and the buyer have some preferences on the configuration of an item to sell and to buy respectively, and there is a mediator (in this case the marketplace itself) that tries to find an agreement that is mutually beneficial for both the buyer and the seller. We have successfully investigated such frameworks with particular reference to multi-issue *bilateral one-shot negotiation*, a special case of *bilateral matchmaking.*

Among all possible Pareto efficient [11] agreements we are obviously interested in those either maximizing the sum of utilities – maximum welfare – or maximizing their product – Nash-bargaining solution [12,13] and we show how to compute them.

References

1. Colucci, S., Di Noia, T., Di Sciascio, E., Donini, F.M., Mongiello, M.: A Uniform Tableaux-Based Method for Concept Abduction and Contraction in Description Logics. In: ECAI 2004, pp. 975–976 (2004)
2. Colucci, S., Di Noia, T., Di Sciascio, E., Donini, F.M., Ragone, A.: A unified framework for non-standard reasoning services in description logics. In: ECAI 2010, pp. 479–484 (2010)
3. Colucci, S., Di Noia, T., Pinto, A., Ragone, A., Ruta, M., Tinelli, E.: A non-monotonic approach to semantic matchmaking and request refinement in e-marketplaces. IJEC 12(2), 127–154 (2007)
4. Di Noia, T., Di Sciascio, E., Donini, F.M.: Extending Semantic-Based Matchmaking via Concept Abduction and Contraction. In: Motta, E., Shadbolt, N.R., Stutt, A., Gibbins, N. (eds.) EKAW 2004. LNCS (LNAI), vol. 3257, pp. 307–320. Springer, Heidelberg (2004)
5. Di Noia, T., Di Sciascio, E., Donini, F.M.: Semantic Matchmaking as Non-Monotonic Reasoning: A Description Logic Approach. JAIR 29, 269–307 (2007)
6. Di Noia, T., Di Sciascio, E., Donini, F.M.: Semantic matchmaking via non-monotonic reasoning: the mamas-tng matchmaking engine. Communications of SIWN 5, 67–72 (2008)
7. Di Noia, T., Di Sciascio, E., Donini, F.M., Mongiello, M.: A system for principled Matchmaking in an electronic marketplace. IJEC 8(4), 9–37 (2004)
8. Di Noia, T., Di Sciascio, E., Donini, F.M.: Computing information minimal match explanations for logic-based matchmaking. In: WI/IAT 2009, pp. 411–418 (2009)
9. Fishburn, P.C.: Interdependence and additivity in multivariate, unidimensional expected utility theory. International Economic Review 8, 335–342 (1967)
10. Li, L., Horrocks, I.: A Software Framework for Matchmaking Based on Semantic Web Technology. IJEC 8(4), 39–60 (2004)
11. Myerson, R.B., Satterthwaite, M.A.: Efficient mechanisms for bilateral trading. Journal of Economic Theory 29(2), 265–281 (1983)
12. Ragone, A., Di Noia, T., Di Sciascio, E., Donini, F.M.: Logic-based automated multi-issue bilateral negotiation in peer-to-peer e-marketplaces. J. AAMAS 16(3), 249–270 (2008)
13. Ragone, A., Straccia, U., Di Noia, T., Di Sciascio, E., Donini, F.M.: Fuzzy matchmaking in e-marketplaces of peer entities using Datalog. FSS 10(2), 251–268 (2009)
14. Ragone, A., Di Noia, T., Donini, F.M., Di Sciascio, E., Wellman, M.P.: Computing Utility from Weighted Description Logic Preference Formulas. In: Baldoni, M., Bentahar, J., van Riemsdijk, M.B., Lloyd, J. (eds.) DALT 2009. LNCS, vol. 5948, pp. 158–173. Springer, Heidelberg (2010)
15. Ragone, A., Di Noia, T., Donini, F.M., Di Sciascio, E., Wellman, M.P.: Weighted Description Logics Preference Formulas for Multiattribute Negotiation. In: Godo, L., Pugliese, A. (eds.) SUM 2009. LNCS, vol. 5785, pp. 193–205. Springer, Heidelberg (2009)
16. Tintarev, N., Masthoff, J.: Designing and Evaluating Explanations for Recommender Systems. In: Recommender Systems Handbook, pp. 479–510 (2011)

A Tableau Algorithm for Fuzzy Description Logics over Residuated De Morgan Lattices

Stefan Borgwardt and Rafael Peñaloza

Theoretical Computer Science, TU Dresden, Germany
{stefborg,penaloza}@tcs.inf.tu-dresden.de

Abstract. Fuzzy description logics can be used to model vague knowledge in application domains. This paper analyses the consistency and satisfiability problems in the description logic $\mathcal{SHI}$ with semantics based on a complete residuated De Morgan lattice. The problems are undecidable in the general case, but can be decided by a tableau algorithm when restricted to finite lattices. For some sublogics of $\mathcal{SHI}$, we provide upper complexity bounds that match the complexity of crisp reasoning.

1 Introduction

Description Logics (DLs) [1] are a family of knowledge representation formalisms that are widely used to model application domains. In DLs, knowledge is represented with the help of *concepts* (unary predicates) and *roles* (binary predicates) that express the relationships between concepts. They have been successfully employed to formulate ontologies–especially in the medical domain–like Galen[1] and serve as the underpinning for the current semantic web language OWL 2.[2] Standard reasoning in these logics includes *concept satisfiability* (is a given concept non-contradictory?) and *ontology consistency* (does a given ontology have a model?). These and other reasoning problems have been studied for DLs, and several algorithms have been proposed and implemented.

One of the main challenges in knowledge representation is the correct modeling and use of imprecise or vague knowledge. For example, medical diagnoses from experts are rarely clear-cut and usually depend on concepts like HighBloodPressure that are necessarily vague. Fuzzy variants of description logics were introduced in the nineties as a means to tackle this challenge. Their applicability to the representation of medical knowledge was studied in [22].

Fuzzy DLs generalize (crisp) DLs by providing a *membership degree* semantics for their concepts. Thus, e.g. 130/85 belongs to the concept HighBloodPressure with a lower degree than, say 140/80. In their original form, membership degrees are elements of the real-number interval $[0, 1]$, but this was later generalized to lattices [21,26]. The papers [21,26] consider only a limited kind of semantics over lattices, where conjunction and disjunction are interpreted through the lattice operators meet and join, respectively.

[1] http://www.opengalen.org/
[2] http://www.w3.org/TR/owl2-overview/

In this paper, we consider a more general lattice-based semantics that uses a *triangular norm* (t-norm) and its residuum as interpretation functions for the logical constructors. We study fuzzy variants of the standard reasoning problems like concept satisfiability and ontology consistency in this setting.

We show that concept satisfiability in $\mathcal{ALC}$ under this semantics is undecidable in general, even if we restrict ourselves to a very simple class of infinite lattices. However, we show with the help of a tableaux-based algorithm that decidability of reasoning can be regained—even for the more expressive DL $\mathcal{SHI}$—if the underlying lattice is required to be finite. Moreover, we describe a black-box method that can be used to transform any decision algorithm for (a small generalization of) satisfiability into a decision procedure for consistency.

Due to space constraints, some of the technical proofs have been left out of this paper; they can be found in the technical report [12].

2 Preliminaries

We start with a short introduction to residuated lattices, which will be the base for the semantics of the fuzzy DL L-$\mathcal{SHI}$. For a more comprehensive view on these lattices, we refer the reader to [15,17].

2.1 Lattices

A *lattice* is a triple $(L, \vee, \wedge)$, consisting of a *carrier set* L and two idempotent, associative, and commutative binary operators *join* $\vee$ and *meet* $\wedge$ on L that satisfy the absorption laws $\ell_1 \vee (\ell_1 \wedge \ell_2) = \ell_1 = \ell_1 \wedge (\ell_1 \vee \ell_2)$ for all $\ell_1, \ell_2 \in L$. These operations induce a partial order $\leq$ on L: $\ell_1 \leq \ell_2$ iff $\ell_1 \wedge \ell_2 = \ell_1$. As usual, we write $\ell_1 < \ell_2$ if $\ell_1 \leq \ell_2$ and $\ell_1 \neq \ell_2$. A subset $T \subseteq L$ is called an *antichain (in L)* if there are no two elements $\ell_1, \ell_2 \in T$ with $\ell_1 < \ell_2$. Whenever it is clear from the context, we will use the carrier set L to represent the lattice $(L, \vee, \wedge)$.

The lattice L is *distributive* if $\vee$ and $\wedge$ distribute over each other, *finite* if L is finite, and *bounded* if it has a *minimum* and a *maximum* element, denoted as $\mathbf{0}$ and $\mathbf{1}$, respectively. It is *complete* if joins and meets of arbitrary subsets $T \subseteq L$, $\bigvee_{t \in T} t$ and $\bigwedge_{t \in T} t$, respectively, exist. Clearly, every finite lattice is also complete, and every complete lattice is bounded.

A *De Morgan lattice* is a bounded distributive lattice L extended with an involutive and anti-monotonic unary operation $\sim$, called *(De Morgan) negation*, satisfying the De Morgan laws $\sim(\ell_1 \vee \ell_2) = \sim\ell_1 \wedge \sim\ell_2$ and $\sim(\ell_1 \wedge \ell_2) = \sim\ell_1 \vee \sim\ell_2$ for all $\ell_1, \ell_2 \in L$.

Given a lattice L, a *t-norm* is an associative and commutative binary operator on L that is monotonic and has $\mathbf{1}$ as its unit. A *residuated lattice* is a lattice L with a t-norm $\otimes$ and a binary operator $\Rightarrow$ (called *residuum*) such that for all $\ell_1, \ell_2, \ell_3 \in L$ we have $\ell_1 \otimes \ell_2 \leq \ell_3$ iff $\ell_2 \leq \ell_1 \Rightarrow \ell_3$. A simple consequence is that for all $\ell_1, \ell_2 \in L$ we have $\mathbf{1} \Rightarrow \ell_1 = \ell_1$, and $\ell_1 \leq \ell_2$ iff $\ell_1 \Rightarrow \ell_2 = \mathbf{1}$.

A t-norm $\otimes$ over a complete lattice L is *continuous* if for all $\ell \in L$ and $T \subseteq L$ we have $\ell \otimes (\bigvee_{\ell' \in T} \ell') = \bigvee_{\ell' \in T} (\ell \otimes \ell')$. Every continuous t-norm $\otimes$ has the unique

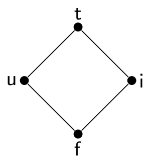


Fig. 1. The De Morgan residuated lattice L_4 with $\sim\mathsf{u} = \mathsf{u}$ and $\sim\mathsf{i} = \mathsf{i}$

residuum $\Rightarrow$ defined by $\ell_1 \Rightarrow \ell_2 = \bigvee\{x \mid \ell_1 \otimes x \leq \ell_2\}$ for all $\ell_1, \ell_2 \in L$. If L is a distributive lattice, then the meet operator $\ell_1 \wedge \ell_2$ always defines a continuous t-norm, often called the *Gödel t-norm*. In a residuated De Morgan lattice L, the *t-conorm* $\oplus$ is defined as as $\ell_1 \oplus \ell_2 := \sim(\sim\ell_1 \otimes \sim\ell_2)$. The t-conorm of the Gödel t-norm is the join operator $\ell_1 \vee \ell_2$.

For example, consider the finite lattice L_4, with the elements f, u, i, and t as shown in Figure 1. This lattice has been used for reasoning about incomplete and contradictory knowledge [5] and as a basis for a paraconsistent rough DL [28]. In our blood pressure scenario, the two degrees i and u may be used to express readings that are *potentially* and *partially* high blood pressures, respectively. The incomparability of these degrees reflects the fact that none of them can be stated to belong *more* to the concept $\mathsf{HighBloodPressure}$ than the other.

For the rest of this paper, L denotes a complete residuated De Morgan lattice with t-norm $\otimes$ and residuum $\Rightarrow$, unless explicitely stated otherwise.

2.2 The Fuzzy DL L-$\mathcal{SHI}$

The fuzzy DL L-$\mathcal{SHI}$ is a generalization of the crisp DL $\mathcal{SHI}$ that uses the elements of L as truth values, instead of just the Boolean *true* and *false*. The syntax of L-$\mathcal{SHI}$ is the same as in $\mathcal{SHI}$ with the addition of the constructor $\rightarrow$.

Definition 1 (syntax of L-$\mathcal{SHI}$). *Let $\mathsf{N_C}$, $\mathsf{N_R}$, and $\mathsf{N_I}$ be pairwise disjoint sets of* concept-, role-, *and* individual names, *respectively, and $\mathsf{N_R^+} \subseteq \mathsf{N_R}$ a set of* transitive role names. *The set of* (complex) roles *is $\mathsf{N_R} \cup \{r^- \mid r \in \mathsf{N_R}\}$. The set of* (complex) concepts *C is obtained through the following syntactic rule, where $A \in \mathsf{N_C}$ and s is a complex role:*

$$C ::= A \mid C_1 \sqcap C_2 \mid C_1 \sqcup C_2 \mid C_1 \rightarrow C_2 \mid \neg C \mid \exists s.C \mid \forall s.C \mid \top \mid \bot.$$

The inverse *of a complex role s (denoted by $\overline{s}$) is s^- if $s \in \mathsf{N_R}$ and r if $s = r^-$. A complex role s is* transitive *if either s or $\overline{s}$ belongs to $\mathsf{N_R^+}$.*

The semantics of this logic is based on functions specifying the *membership degree* of every domain element in a concept C.

Definition 2 (semantics of L-$\mathcal{SHI}$). *An* interpretation *is a pair $\mathcal{I} = (\Delta^\mathcal{I}, \cdot^\mathcal{I})$ where $\Delta^\mathcal{I}$ is a non-empty* domain, *and $\cdot^\mathcal{I}$ is a function that assigns to every individual name a an element $a^\mathcal{I} \in \Delta^\mathcal{I}$, to every concept name A a function*

$A^\mathcal{I} : \Delta^\mathcal{I} \to L$, and to every role name r a function $r^\mathcal{I} : \Delta^\mathcal{I} \times \Delta^\mathcal{I} \to L$, where $r^\mathcal{I}(x,y) \otimes r^\mathcal{I}(y,z) \leq r^\mathcal{I}(x,z)$ holds for all $r \in \mathsf{N}_\mathsf{R}^+$ and $x,y,z \in \Delta^\mathcal{I}$.

The function $\cdot^\mathcal{I}$ is extended to L-$\mathcal{SHI}$ concepts as follows for every $x \in \Delta^\mathcal{I}$:

- $\top^\mathcal{I}(x) = \mathbf{1}, \quad \bot^\mathcal{I}(x) = \mathbf{0}$,
- $(C \sqcap D)^\mathcal{I}(x) = C^\mathcal{I}(x) \otimes D^\mathcal{I}(x), \quad (C \sqcup D)^\mathcal{I}(x) = C^\mathcal{I}(x) \oplus D^\mathcal{I}(x)$,
- $(C \to D)^\mathcal{I}(x) = C^\mathcal{I}(x) \Rightarrow D^\mathcal{I}(x)$,
- $(\neg C)^\mathcal{I}(x) = {\sim} C^\mathcal{I}(x)$,
- $(\exists s.C)^\mathcal{I}(x) = \bigvee_{y \in \Delta^\mathcal{I}} \big(s^\mathcal{I}(x,y) \otimes C^\mathcal{I}(y)\big)$,
- $(\forall s.C)^\mathcal{I}(x) = \bigwedge_{y \in \Delta^\mathcal{I}} \big(s^\mathcal{I}(x,y) \Rightarrow C^\mathcal{I}(y)\big)$,

where $(r^-)^\mathcal{I}(x,y) = r^\mathcal{I}(y,x)$ for all $x,y \in \Delta^\mathcal{I}$ and $r \in \mathsf{N}_\mathsf{R}$.

The semantics of the existential and value restrictions is just the direct application of the semantics of quantification of fuzzy first-order logic [18,19] to fuzzy DLs.

Notice that, unlike in crisp $\mathcal{SHI}$, existential and universal quantifiers are not dual to each other, i.e. in general, $(\neg\exists s.C)^\mathcal{I}(x) = (\forall s.\neg C)^\mathcal{I}(x)$ does not hold. Likewise, the implication constructor $\to$ cannot be expressed in terms of the negation $\neg$ and conjunction $\sqcap$.

The axioms of this logic are those of crisp $\mathcal{SHI}$, but with associated lattice values, which express the degree to which the restrictions must be satisfied.

Definition 3 (axioms). *An* assertion *can be a* concept assertion *of the form $\langle a : C \rhd \ell\rangle$ or a* role assertion *of the form $\langle (a,b) : s \rhd \ell\rangle$, where C is a concept, s is a complex role, a, b are individual names, $\ell \in L$, and $\rhd \in \{=, \geq\}$. If $\rhd$ is $=$, then it is called an* equality assertion. *A* general concept inclusion (GCI) *is of the form $\langle C \sqsubseteq D, \ell\rangle$, where C, D are concepts, and $\ell \in L$. A* role inclusion *is of the form $s \sqsubseteq s'$, where s and s' are complex roles.*

An ontology *$(\mathcal{A}, \mathcal{T}, \mathcal{R})$ consists of a finite set $\mathcal{A}$ of assertions (*ABox*), a finite set $\mathcal{T}$ of GCIs (*TBox*), and a finite set $\mathcal{R}$ of role inclusions (*RBox*). The ABox $\mathcal{A}$ is called* local *if there is an individual $a \in \mathsf{N}_\mathsf{I}$ such that all assertions in $\mathcal{A}$ are of the form $\langle a : C = \ell\rangle$, for some concept C and $\ell \in L$.*

An interpretation $\mathcal{I}$ satisfies *the assertion $\langle a : C \rhd \ell\rangle$ if $C^\mathcal{I}(a^\mathcal{I}) \rhd \ell$ and the assertion $\langle (a,b) : s \rhd \ell\rangle$ if $s^\mathcal{I}(a^\mathcal{I}, b^\mathcal{I}) \rhd \ell$. It satisfies the GCI $\langle C \sqsubseteq D, \ell\rangle$ if $C^\mathcal{I}(x) \Rightarrow D^\mathcal{I}(x) \geq \ell$ holds for every $x \in \Delta^\mathcal{I}$. It satisfies the role inclusion $s \sqsubseteq s'$ if for all $x, y \in \Delta^\mathcal{I}$ we have $s^\mathcal{I}(x,y) \leq s'^\mathcal{I}(x,y)$.*

$\mathcal{I}$ is a model *of the ontology $(\mathcal{A},\mathcal{T},\mathcal{R})$ if it satisfies all axioms in $\mathcal{A}$, $\mathcal{T}$, $\mathcal{R}$.*

Given an RBox $\mathcal{R}$, the *role hierarchy* $\sqsubseteq_\mathcal{R}$ on the set of complex roles is the reflexive and transitive closure of the relation

$$\{(s, s') \mid s \sqsubseteq s' \in \mathcal{R} \text{ or } \overline{s} \sqsubseteq \overline{s'} \in \mathcal{R}\}.$$

Using reachability algorithms, the role hierarchy can be computed in polynomial time in the size of $\mathcal{R}$. An RBox $\mathcal{R}$ is called *acyclic* if it contains no cycles of the form $s \sqsubseteq_\mathcal{R} s'$, $s' \sqsubseteq_\mathcal{R} s$ for two roles $s \neq s'$.

The fuzzy DL L-$\mathcal{ALC}$ is the sublogic of L-$\mathcal{SHI}$ where no role inclusions, transitive roles, or inverse roles are allowed. $\mathcal{SHI}$ is the sublogic of L-$\mathcal{SHI}$ where the underlying lattice contains only the elements $\mathbf{0}$ and $\mathbf{1}$, which may be interpreted as *false* and *true*, respectively, and the t-norm and t-conorm are conjunction and disjunction, respectively.

Recall that the semantics of the quantifiers require the computation of a supremum or infimum of the membership degrees of a possibly infinite set of elements of the domain. To obtain effective decision procedures, reasoning is usually restricted to a special kind of models, called witnessed models [19].

Definition 4 (witnessed model). *Let $n \in \mathbb{N}$. A model $\mathcal{I}$ of an ontology $\mathcal{O}$ is n-*witnessed *if for every $x \in \Delta^\mathcal{I}$, every role s and every concept C there are $x_1, \dots, x_n, y_1, \dots, y_n \in \Delta^\mathcal{I}$ such that*

$$(\exists s.C)^\mathcal{I}(x) = \bigvee_{i=1}^{n} \left(s^\mathcal{I}(x, x_i) \otimes C^\mathcal{I}(x_i)\right), \quad (\forall s.C)^\mathcal{I}(x) = \bigwedge_{i=1}^{n} \left(s^\mathcal{I}(x, y_i) \Rightarrow C^\mathcal{I}(y_i)\right).$$

In particular, if $n = 1$, the suprema and infima from the semantics of $\exists s.C$ and $\forall s.C$ are maxima and minima, respectively, and we say that $\mathcal{I}$ is witnessed.

The reasoning problems for $\mathcal{SHI}$ generalize to the fuzzy semantics of L-$\mathcal{SHI}$.

Definition 5 (decision problems). *Let $\mathcal{O}$ be an ontology, C, D be two concepts, $a \in \mathsf{N_I}$, and $\ell \in L$. $\mathcal{O}$ is* consistent *if it has a (witnessed) model. C is* strongly ℓ-satisfiable *if there is a (witnessed) model $\mathcal{I}$ of $\mathcal{O}$ and $x \in \Delta^\mathcal{I}$ with $C^\mathcal{I}(x) \geq \ell$. The individual a is an* ℓ-instance *of C if $\langle a : C \geq \ell \rangle$ is satisfied by all (witnessed) models of $\mathcal{O}$. C is* ℓ-subsumed *by D if $\langle C \sqsubseteq D, \ell \rangle$ is satisfied by all (witnessed) models of $\mathcal{O}$.*

Example 6. It is known that coffee drinkers and salt consumers tend to have a higher blood pressure. On the other hand, bradycardia is highly correlated with a lower blood pressure. This knowledge can be expressed through the TBox

$$\{\langle \mathsf{CoffeeDrinker} \sqsubseteq \mathsf{HighBloodPressure}, \mathsf{i} \rangle, \ \langle \mathsf{SaltConsumer} \sqsubseteq \mathsf{HighBloodPressure}, \mathsf{i} \rangle, \\ \langle \mathsf{Bradycardia} \sqsubseteq \neg\mathsf{HighBloodPressure}, \mathsf{i} \rangle\},$$

over the lattice L_4 from Figure 1. The degree i in these axioms expresses that the relation between the causes and $\mathsf{HighBloodPressure}$ is not absolute. Consider the patients ana, who is a coffee drinker, and bob, a salt consumer with bradycardia, as expressed by the ABox

$$\{\langle \mathsf{ana} : \mathsf{CoffeeDrinker} = \mathsf{t} \rangle, \ \langle \mathsf{bob} : \mathsf{SaltConsumer} \sqcap \mathsf{Bradycardia} = \mathsf{t} \rangle\}.$$

We can deduce that both patients are an i-instance of $\mathsf{HighBloodPressure}$, but only bob is an i-instance of $\neg\mathsf{HighBloodPressure}$. Notice that if we changed all the degrees from the GCIs to the value t, the ontology would be inconsistent.

We will focus first on a version of the consistency problem where the ABox is required to be a local ABox; we call this problem *local consistency*. We show in Section 5 that local consistency can be used for solving other reasoning problems in L-$\mathcal{SHI}$ if L is finite. Before that, we show that satisfiability and (local) consistency are undecidable in L-$\mathcal{ALC}$, and hence also in L-$\mathcal{SHI}$, in general.

3 Undecidability

To show undecidability, we use a reduction from the Post Correspondence Problem [24] to strong satisfiability in L-$\mathcal{ALC}$ over a specific infinite lattice. The reduction uses ideas that have been successfully applied to showing undecidability of reasoning for several fuzzy description logics [2,3,14].

Although the basic idea of the proof is not new, it is interesting for several reasons. First, previous incarnations of the proof idea focused on decidability of ontology consistency [3,13,14], while we are concerned with strong ℓ-satisfiability. Second, most of the previous undecidability results only hold for reasoning w.r.t. witnessed models, but the current proof works for both witnessed and general models. Finally, in contrast to an earlier version of this proof [11], the employed lattice has a quite simple structure in the sense that it is a total order that has only the two limit points $-\infty$ and ∞ instead of infinitely many. Note that any distributive lattice without limit points is already finite and reasoning in finite residuated De Morgan lattices is decidable (see Sections 4 and 5).

Definition 7 (PCP). *Let $\mathcal{P} = \{(v_1, w_1), \dots, (v_n, w_n)\}$ be a finite set of pairs of words over the alphabet $\Sigma = \{1, \dots, s\}$ with $s > 1$. The* Post Correspondence Problem (PCP) *asks for a finite non-empty sequence $i_1 \dots i_k \in \{1, \dots, n\}^+$ such that $v_{i_1} \dots v_{i_k} = w_{i_1} \dots w_{i_k}$. If this sequence exists, it is called a* solution *for $\mathcal{P}$.*

For $\nu = i_1 \cdots i_k \in \{1, \dots, n\}^*$, we define $v_\nu := v_{i_1} \cdots v_{i_k}$ and $w_\nu := w_{i_1} \cdots w_{i_k}$.

We consider the lattice $\mathbb{Z}_\infty$ whose domain is $\mathbb{Z} \cup \{-\infty, \infty\}$ with the usual ordering over the integers and $-\infty$ and ∞ as the minimal and maximal element, respectively. Its De Morgan negation is $\sim\ell = -\ell$ if $\ell \in \mathbb{Z}$, $\sim\infty = -\infty$, and $\sim(-\infty) = \infty$. The t-norm $\otimes$ is defined as follows for all $\ell, m \in \mathbb{Z}_\infty$:

$$\ell \otimes m := \begin{cases} \ell + m & \text{if } \ell, m \in \mathbb{Z} \text{ and } \ell, m \leq 0 \\ \min\{\ell, m\} & \text{otherwise.} \end{cases}$$

This is in fact a residuated lattice with the following residuum:

$$\ell \Rightarrow m := \begin{cases} \infty & \text{if } \ell \leq m \\ m & \text{if } \ell > m \text{ and } \ell \geq 0 \\ m - \ell & \text{if } \ell > m \text{ and } \ell < 0. \end{cases}$$

Given an instance $\mathcal{P}$ of the PCP, we will construct a TBox $\mathcal{T}_\mathcal{P}$ such that the designated concept name S is strongly ∞-satisfiable iff $\mathcal{P}$ has no solution. Recall that the alphabet Σ consists of the first s positive integers. Thus, every word in Σ^+ can be seen as a positive integer written in base $s + 1$; we extend this intuition and denote the empty word by 0. We encode each word $u \in \Sigma^*$ with the number $-u \leq 0$.

The idea is that the TBox $\mathcal{T}_\mathcal{P}$ describes the *search tree* of $\mathcal{P}$ with the nodes $\{1, \dots, n\}^*$. At its root ε, it encodes the value $v_\varepsilon = w_\varepsilon = \varepsilon$, which is represented by 0, using the concept names V and W. These concept names are used throughout the tree to express the values v_ν and w_ν at every node $\nu \in \{1, \dots, n\}^*$.

Additionally, we will use the auxiliary concept names V_i and W_i to encode the constant words v_i and w_i, respectively, for each $i \in \{1, \dots, n\}$. These will be used to compute the concatenation $v_{\nu i} = v_\nu v_i$ at each node.

To simplify the reduction, we will use some abbreviations. Given two L-$\mathcal{ALC}$ concepts C and D and $r \in \mathsf{N_R}$, $\langle C \equiv D\rangle$ abbreviates the axioms $\langle C \sqsubseteq D, \infty\rangle$, $\langle D \sqsubseteq C, \infty\rangle$; and $\langle C \overset{r}{\rightsquigarrow} D\rangle$ stands for the axioms $\langle C \sqsubseteq \forall r.D, \infty\rangle$, $\langle \exists r.D \sqsubseteq C, \infty\rangle$. For $n \geq 1$, the concept C^n is inductively defined by $C^1 := C$ and $C^{n+1} := C^n \sqcap C$.

Proposition 8. *Let $\mathcal{I}$ be an interpretation and $x \in \Delta^\mathcal{I}$.*

- *If $\mathcal{I}$ satisfies $\langle C \equiv D\rangle$, then $C^\mathcal{I}(x) = D^\mathcal{I}(x)$.*
- *If $\mathcal{I}$ satisfies $\langle C \overset{r}{\rightsquigarrow} D\rangle$ and $C^\mathcal{I}(x) \leq 0$, then $C^\mathcal{I}(x) = D^\mathcal{I}(y)$ holds for all $y \in \Delta^\mathcal{I}$ with $r^\mathcal{I}(x, y) \geq 1$.*
- *If $C^\mathcal{I}(x) \in \mathbb{Z}$, $C^\mathcal{I}(x) \leq 0$, and $n \geq 1$, then $(C^n)^\mathcal{I}(x) = n \cdot C^\mathcal{I}(x)$.*

We now introduce the TBox $\mathcal{T}_0 := \bigcup_{i=0}^n \mathcal{T}_\mathcal{P}^i$ that encodes the search tree of the instance $\mathcal{P}$ of the PCP:

$$\begin{aligned}
\mathcal{T}_\mathcal{P}^0 &:= \{\langle S \sqsubseteq V, 0\rangle, \langle S \sqsubseteq \neg V, 0\rangle, \langle S \sqsubseteq W, 0\rangle, \langle S \sqsubseteq \neg W, 0\rangle\},\\
\mathcal{T}_\mathcal{P}^i &:= \{\langle \top \sqsubseteq \exists r_i.\top, 1\rangle,\\
&\qquad \langle \top \sqsubseteq V_i, -v_i\rangle, \langle \top \sqsubseteq \neg V_i, v_i\rangle, \langle \top \sqsubseteq W_i, -w_i\rangle, \langle \top \sqsubseteq \neg W_i, w_i\rangle,\\
&\qquad \langle (V^{(s+1)^{|v_i|}} \sqcap V_i) \overset{r_i}{\rightsquigarrow} V\rangle, \langle (W^{(s+1)^{|w_i|}} \sqcap W_i) \overset{r_i}{\rightsquigarrow} W\rangle\},
\end{aligned}$$

where $|u|$ denotes the length of the word u.

The TBox $\mathcal{T}_\mathcal{P}^0$ initializes the search tree by ensuring for every model $\mathcal{I}$ and every domain element $x \in \Delta^\mathcal{I}$ that satisfies $S^\mathcal{I}(x) = \infty$ that the values of V and W are both 0, which is the encoding of the empty word. Each TBox $\mathcal{T}_\mathcal{P}^i$ ensures the existence of an r_i-successor for every domain element and describes the constant pair (v_i, w_i) using the concepts V_i and W_i, that is, it forces that $V_i^\mathcal{I}(x) = -v_i$ and $W_i^\mathcal{I}(x) = -w_i$ for every $x \in \Delta^\mathcal{I}$. Using the last two axioms, the search tree is then extended by concatenating the words v and w produced so far with v_i and w_i, respectively. In the following, we will describe this in more detail.

Consider the interpretation $\mathcal{I}_\mathcal{P}$ over the domain $\Delta^{\mathcal{I}_\mathcal{P}} = \{1, \dots, n\}^*$, where for all $\nu, \nu' \in \{1, \dots, n\}^*$ and $i \in \{1, \dots, n\}$,

- $V^{\mathcal{I}_\mathcal{P}}(\nu) = -v_\nu, \quad W^{\mathcal{I}_\mathcal{P}}(\nu) = -w_\nu$,
- $V_i^{\mathcal{I}_\mathcal{P}}(\nu) = -v_i, \quad W_i^{\mathcal{I}_\mathcal{P}}(\nu) = -w_i$,
- $r_i^{\mathcal{I}_\mathcal{P}}(\nu, \nu i) = \infty$ and $r_i^{\mathcal{I}_\mathcal{P}}(\nu, \nu') = -\infty$ if $\nu' \neq \nu i$,
- $S^{\mathcal{I}_\mathcal{P}}(\varepsilon) = \infty$ and $S^{\mathcal{I}_\mathcal{P}}(\nu') = -\infty$ if $\nu' \neq \varepsilon$.

It is easy to see that $\mathcal{I}_\mathcal{P}$ is in fact a model of $\mathcal{T}_0$ and it strongly satisfies S with degree ∞. Moreover, *every* model of this TBox that strongly ∞-satisfies S must "include" $\mathcal{I}_\mathcal{P}$ in the following sense.

Lemma 9. *Let $\mathcal{I}$ be a model of $\mathcal{T}_0$ such that $S^\mathcal{I}(x_0) = \infty$ for some $x_0 \in \Delta^\mathcal{I}$. Then there exists a function $g : \Delta^{\mathcal{I}_\mathcal{P}} \to \Delta^\mathcal{I}$ such that $A^{\mathcal{I}_\mathcal{P}}(\nu) = A^\mathcal{I}(g(\nu))$ and $r_i(g(\nu), g(\nu i)) \geq 1$ hold for every concept name $A \in \{V, W, V_1, W_1, \dots, V_n, W_n\}$, every $\nu \in \Delta^{\mathcal{I}_\mathcal{P}}$, and every $i \in \{1, \dots, n\}$.*

Proof. We construct the function g by induction on ν and set $g(\varepsilon) := x_0$. Since $\mathcal{I}$ is a model of $\mathcal{T}_\mathcal{P}^0$ and $S^\mathcal{I}(x_0) = \infty$, we have $V^\mathcal{I}(x_0) \geq 0$ and $\sim V^\mathcal{I}(x_0) \geq 0$, i.e. $V^\mathcal{I}(x_0) = 0$, and similarly $W^\mathcal{I}(x_0) = 0$. In the same way, for every $i \in \{1, \dots, n\}$, $V_i^\mathcal{I}(x_0)$ and $W_i^\mathcal{I}(x_0)$ are restricted by $\mathcal{T}_\mathcal{P}^i$ to be $-v_i$ and $-w_i$, respectively.

Let now $\nu \in \{1, \dots, n\}^*$ and assume that $g(\nu)$ already satisfies the condition. For each $i \in \{1, \dots, n\}$, the first axiom of $\mathcal{T}_\mathcal{P}^i$ ensures that $\bigvee_{y \in \Delta^\mathcal{I}} r_i^\mathcal{I}(g(\nu), y) \geq 1$. Thus, there is $y_i \in \Delta^\mathcal{I}$ such that $r_i^\mathcal{I}(g(\nu), y_i) \geq 1$. We define $g(\nu i) := y_i$. By Proposition 8, we have

$$V^\mathcal{I}(y_i) = (V^{(s+1)^{|v_i|}} \sqcap V_i)^\mathcal{I}(g(\nu)) = -\left((s+1)^{|v_i|} v_\nu + v_i\right) = -v_\nu v_i = -v_{\nu i},$$

and similarly for $W^\mathcal{I}(y_i)$. The claim for V_i and W_i can be shown as above. □

This proposition shows that every model of $\mathcal{T}_0$ encodes a description of the search tree for a solution of $\mathcal{P}$. Thus, to decide the PCP, it suffices to detect whether there is a node $\nu \in \{1, \dots, n\}^+$ of $\mathcal{I}_\mathcal{P}$ where $V^{\mathcal{I}_\mathcal{P}}(\nu) = W^{\mathcal{I}_\mathcal{P}}(\nu)$. We accomplish this using the TBox

$$\mathcal{T}' := \{\langle \top \sqsubseteq \forall r_i.\neg((V \to W) \sqcap (W \to V)), 0\rangle \mid 1 \leq i \leq n\}.$$

The interpretation $\mathcal{I}_\mathcal{P}$ is a model of $\mathcal{T}'$ iff $V^{\mathcal{I}_\mathcal{P}}(\nu) \neq W^{\mathcal{I}_\mathcal{P}}(\nu)$ holds for every $\nu \in \{1, \dots, n\}^+$.

Lemma 10. *$\mathcal{P}$ has a solution iff S is not ∞-satisfiable w.r.t. $\mathcal{T}_\mathcal{P} := \mathcal{T}_0 \cup \mathcal{T}'$.*

Proof. For any two values $\ell, m \leq 0$, we have $\ell \neq m$ iff $(\ell \Rightarrow m) \otimes (m \Rightarrow \ell) \leq 0$.

Assume now that S is not ∞-satisfiable w.r.t. $\mathcal{T}_\mathcal{P}$. Then, in particular, $\mathcal{I}_\mathcal{P}$ does not satisfy $\mathcal{T}'$, i.e. we have $(\forall r_i.\neg((V \to W) \sqcap (W \to V)))^{\mathcal{I}_\mathcal{P}}(\nu) < 0$ for some $\nu \in \{1, \dots, n\}^*$ and $i \in \{1, \dots, n\}$. There must be a $\nu \in \{1, \dots, n\}^+$ with $(\neg((V \to W) \sqcap (W \to V)))^{\mathcal{I}_\mathcal{P}}(\nu) < 0$; thus, $-v_\nu = V^{\mathcal{I}_\mathcal{P}}(\nu) = W^{\mathcal{I}_\mathcal{P}}(\nu) = -w_\nu$. This shows that $v_\nu = w_\nu$, i.e. $\mathcal{P}$ has a solution.

For the other direction, let $\mathcal{I}$ be a model of $\mathcal{T}_\mathcal{P}$ and $x_0 \in \Delta^\mathcal{I}$ such that $S^\mathcal{I}(x_0) = \infty$. In particular, we have

$$r_i^\mathcal{I}(g(\nu), g(\nu i)) \Rightarrow (\neg((V \to W) \sqcap (W \to V)))^\mathcal{I}(g(\nu i)) \geq 0$$

for every $\nu \in \{1, \dots, n\}^*$ and $i \in \{1, \dots, n\}$, where g is the function constructed in Lemma 9. Thus, $((V \to W) \sqcap (W \to V))^\mathcal{I}(g(\nu)) \leq 0$ for every $\nu \in \{1, \dots, n\}^+$, which implies $-v_\nu = V^\mathcal{I}(g(\nu)) \neq W^\mathcal{I}(g(\nu)) = -w_\nu$. This shows that $v_\nu \neq w_\nu$ for all $\nu \in \{1, \dots, n\}^+$, i.e. $\mathcal{P}$ has no solution. □

As mentioned before, since the interpretation $\mathcal{I}_\mathcal{P}$ is witnessed, undecidability holds even if we restrict reasoning to n-witnessed models, for any $n \in \mathbb{N}$.

Theorem 11. *Strong satisfiability is undecidable in L-$\mathcal{ALC}$, for some countable total order L with at most two limit points, even if reasoning is over n-witnessed models only.*

This theorem also shows that (local) consistency is undecidable in $\mathbb{Z}_\infty$-$\mathcal{ALC}$ since S is strongly ∞-satisfiable w.r.t. $\mathcal{T}_\mathcal{P}$ iff $(\{\langle a : S = \infty\rangle\}, \mathcal{T}_\mathcal{P})$ is locally consistent, where a is an arbitrary individual name.

Notice that this does not exclude the existence of classes of infinite lattices for which reasoning in L-$\mathcal{SHI}$ is decidable. In fact, there exists a large class of infinite total orders for which consistency is decidable [9]. What Theorem 11 shows is that there exist lattices for which this problem is undecidable. If we restrict to finite lattices, then a tableau algorithm can be used for reasoning.

4 A Tableaux Algorithm for Local Consistency

Before presenting a tableau algorithm [4] that decides local consistency by constructing a model of a given L-$\mathcal{SHI}$ ontology, we discuss previous approaches to deciding consistency of fuzzy DLs over finite residuated De Morgan lattices in the presence of GCIs.

A popular method is the reduction of fuzzy ontologies into crisp ones, which has so far only been done for finite total orders [7,8,26]. Reasoning can then be performed through existing optimized reasoners for crisp DLs. The main idea is to translate every concept name A into finitely many crisp concept names $A_{\geq\ell}$, one for each truth value ℓ, where $A_{\geq\ell}$ collects all those individuals that belong to A with a truth degree $\geq \ell$. The lattice structure is expressed through GCIs of the form $A_{\geq\ell_2} \sqsubseteq A_{\geq\ell_1}$, where ℓ_2 is a minimal element above ℓ_1, and analogously for the role names. All axioms are then recursively translated into crisp axioms that use only the introduced crisp concept and role names. The resulting crisp ontology is consistent iff the original fuzzy ontology is consistent.

In general such a translation is exponential in the size of the concepts that occur in the fuzzy ontology. The reason is that, depending on the t-norm used, there may be many possible combinations of values ℓ_1, ℓ_2 for C, D, respectively, that lead to $C \sqcap D$ having the value $\ell = \ell_1 \otimes \ell_2$, and similarly for the other constructors. All these possibilities have to be expressed in the translation. Since ontology consistency in crisp $\mathcal{SHI}$ is ExpTime-hard, this yields a 2-ExpTime reasoning procedure. Moreover, DL reasoners usually implement tableaux algorithms with a worst-case complexity above NExpTime; in that case, one gets a 2-NExpTime reasoning procedure. In contrast, our tableau algorithm has a worst-case complexity of NExpTime, matching the behaviour of crisp $\mathcal{SHI}$ reasoners.

To the best of our knowledge, at the moment there exists only one (correct) tableaux algorithm that can deal with a finite total order of truth values and GCIs [25],[3] but it is restricted to the Gödel t-norm. The main difference between this algorithm and ours is that we non-deterministically guess the degree of membership of each individual to every relevant concept, while the approach from [25] sets only lower and upper bounds for these degrees; this greatly reduces

[3] Several tableau algorithms for fuzzy DLs over infinite total orders exist, but they are either restricted to acyclic TBoxes or are not correct in the presence of GCIs, as shown in [2,6].

the amount of non-determinism encountered, but introduces several complications when a t-norm different from the Gödel t-norm is used.

We present a straightforward tableaux algorithm with a larger amount of nondeterminism that nevertheless matches the theoretical worst-case complexity of tableaux algorithms for crisp $\mathcal{SHI}$. It is loosely based on the crisp tableaux algorithm in [20]. A first observation that simplifies the algorithm is that since L is finite, we can w.l.o.g. restrict reasoning to n-witnessed models.

Proposition 12. *If the maximal cardinality of an antichain of L is n, then every interpretation in L-$\mathcal{SHI}$ is n-witnessed.*

For simplicity, we consider only the case $n = 1$. For $n > 1$, the construction is similar, but several witnesses have to be produced for satisfying each existential and value restriction. The necessary changes in the algorithm are described at the end of this section. We can also assume w.l.o.g. that the RBox is acyclic. The proof of this follows similar arguments as for crisp $\mathcal{SHI}$ [27].

Proposition 13. *Deciding local consistency in L-$\mathcal{SHI}$ is polynomially equivalent to deciding local consistency in L-$\mathcal{SHI}$ w.r.t. acyclic RBoxes.*

In the following, let $\mathcal{O} = (\mathcal{A}, \mathcal{T}, \mathcal{R})$ be an ontology where $\mathcal{A}$ is a local ABox that contains only the individual name a and $\mathcal{R}$ is an acyclic RBox. We first show that $\mathcal{O}$ has a model if we can find a *tableau*; intuitively, a possibly infinite "completed version" of $\mathcal{A}$. Later we describe an algorithm for constructing a finite representation of such a tableau.

Definition 14. *A* tableau *for $\mathcal{O}$ is a set $\mathbf{T}$ of equality assertions over a set Ind of individuals such that $a \in \mathsf{Ind}$, $\mathcal{A} \subseteq \mathbf{T}$, and the following conditions are satisfied for all $C, C_1, C_2 \in \mathsf{sub}(\mathcal{O})$, $x, y \in \mathsf{Ind}$, $r, s \in \mathsf{N_R}$, and $\ell \in L$:*

Clash-free: *If $\langle x : C = \ell \rangle \in \mathbf{T}$ or $\langle (x, y) : r = \ell \rangle \in \mathbf{T}$, then there is no $\ell' \in L$ such that $\ell' \neq \ell$ and $\langle x : C = \ell' \rangle \in \mathbf{T}$ or $\langle (x, y) : r = \ell' \rangle \in \mathbf{T}$, respectively.*

Complete: *For every row of Table 1, the following condition holds:*

*"If $\langle$*trigger*$\rangle$ is in $\mathbf{T}$, there are $\langle$*values*$\rangle$ such that $\langle$*assertions*$\rangle$ are in $\mathbf{T}$."*

These conditions help to abstract from the interplay between transitive roles and existential and value restrictions. We prove in [12] that it suffices to satisfy the above conditions to make certain that $\mathcal{O}$ has a model.

Lemma 15. *$\mathcal{O}$ is locally consistent iff it has a tableau.*

We now present a tableaux algorithm for deciding local consistency. The algorithm starts with the local ABox $\mathcal{A}$, and nondeterministically expands it to a tree-like ABox $\widehat{\mathcal{A}}$ that represents a model of $\mathcal{O}$. It uses the conditions from Table 1 and reformulates them into *expansion rules* of the form:

"If there is $\langle$trigger$\rangle$ in $\widehat{\mathcal{A}}$ and there are no $\langle$values$\rangle$ such that $\langle$assertions$\rangle$ are in $\mathcal{A}$, then introduce $\langle$values$\rangle$ and add $\langle$assertions$\rangle$ to $\widehat{\mathcal{A}}$."

Table 1. The tableau conditions for L-$\mathcal{SHI}$

	⟨trigger⟩	⟨values⟩	⟨assertions⟩
$\sqcap$	$\langle x : C_1 \sqcap C_2 = \ell \rangle$	$\ell_1, \ell_2 \in L$ with $\ell_1 \otimes \ell_2 = \ell$	$\langle x : C_1 = \ell_1 \rangle$, $\langle x : C_2 = \ell_2 \rangle$
$\sqcup$	$\langle x : C_1 \sqcup C_2 = \ell \rangle$	$\ell_1, \ell_2 \in L$ with $\ell_1 \oplus \ell_2 = \ell$	$\langle x : C_1 = \ell_1 \rangle$, $\langle x : C_2 = \ell_2 \rangle$
$\rightarrow$	$\langle x : C_1 \rightarrow C_2 = \ell \rangle$	$\ell_1, \ell_2 \in L$ with $\ell_1 \Rightarrow \ell_2 = \ell$	$\langle x : C_1 = \ell_1 \rangle$, $\langle x : C_2 = \ell_2 \rangle$
$\neg$	$\langle x : \neg C = \ell \rangle$		$\langle x : C = \sim \ell \rangle$
$\exists$	$\langle x : \exists r.C = \ell \rangle$	$\ell_1, \ell_2 \in L$ with $\ell_1 \otimes \ell_2 = \ell$, individual y	$\langle (x, y) : r = \ell_1 \rangle$, $\langle y : C = \ell_2 \rangle$
$\exists_{\leq}$	$\langle x : \exists r.C = \ell \rangle$, $\langle (x, y) : r = \ell_1 \rangle$	$\ell_2 \in L$ with $\ell_1 \otimes \ell_2 \leq \ell$	$\langle y : C = \ell_2 \rangle$
$\exists_{+}$	$\langle x : \exists s.C = \ell \rangle$, $\langle (x, y) : r = \ell_1 \rangle$ with r transitive and $r \sqsubseteq_{\mathcal{R}} s$	$\ell_2 \in L$ with $\ell_1 \otimes \ell_2 \leq \ell$	$\langle y : \exists r.C = \ell_2 \rangle$
$\forall$	$\langle x : \forall r.C = \ell \rangle$	$\ell_1, \ell_2 \in L$ with $\ell_1 \Rightarrow \ell_2 = \ell$, individual y	$\langle (x, y) : r = \ell_1 \rangle$, $\langle y : C = \ell_2 \rangle$
$\forall_{\geq}$	$\langle x : \forall r.C = \ell \rangle$, $\langle (x, y) : r = \ell_1 \rangle$	$\ell_2 \in L$ with $\ell_1 \Rightarrow \ell_2 \geq \ell$	$\langle y : C = \ell_2 \rangle$
$\forall_{+}$	$\langle x : \forall s.C = \ell \rangle$, $\langle (x, y) : r = \ell_1 \rangle$ with r transitive and $r \sqsubseteq_{\mathcal{R}} s$	$\ell_2 \in L$ with $\ell_1 \Rightarrow \ell_2 \geq \ell$	$\langle y : \forall r.C = \ell_2 \rangle$
inv	$\langle (x, y) : r = \ell_1 \rangle$		$\langle (y, x) : \overline{r} = \ell_1 \rangle$
$\sqsubseteq_{\mathcal{R}}$	$\langle (x, y) : r = \ell_1 \rangle$, $r \sqsubseteq_{\mathcal{R}} s$	$\ell_2 \in L$ with $\ell_1 \leq \ell_2$	$\langle (x, y) : s = \ell_2 \rangle$
$\sqsubseteq_{\mathcal{T}}$	individual x, $\langle C_1 \sqsubseteq C_2, \ell \rangle$ in $\mathcal{T}$	$\ell_1, \ell_2 \in L$ with $\ell_1 \Rightarrow \ell_2 \geq \ell$	$\langle x : C_1 = \ell_1 \rangle$, $\langle x : C_2 = \ell_2 \rangle$

The rules $\exists$ and $\forall$ always introduce new individuals y that do not appear in $\widehat{\mathcal{A}}$. Initially, the ABox $\mathcal{A}$ contains the single individual a. It is expanded by the rules in a tree-like way: role connections are only created by adding new successors to existing individuals. If an individual y was created by a rule $\exists$ or $\forall$ that was applied to an assertion involving an individual x, then we say that y is a *successor* of x, and x is the *predecessor* of y; *ancestor* is the transitive closure of *predecessor*. Note that the presence of an assertion $\langle (x, y) : r = \ell \rangle$ in $\widehat{\mathcal{A}}$ does not imply that y is a successor of x—it could also be the case that this assertion was introduced by the inv-rule. We further denote by $\widehat{\mathcal{A}}_x$ the set of all concept assertions from $\widehat{\mathcal{A}}$ that involve the individual x, i.e. are of the form $\langle x : C = \ell \rangle$ for some concept C and $\ell \in L$. To ensure that the application of the rules terminates, we need to add a blocking condition. We use *anywhere blocking* [23], which is based on the idea that it suffices to examine each set $\widehat{\mathcal{A}}_x$ only once in the whole ABox $\widehat{\mathcal{A}}$.

Let $\succ$ be a total order on the individuals of $\widehat{\mathcal{A}}$ that includes the ancestor relationship, i.e. whenever y is a successor of x, then $y \succ x$. An individual y is *directly blocked* if for some other individual x in $\widehat{\mathcal{A}}$ with $y \succ x$, $\widehat{\mathcal{A}}_x$ is equal to $\widehat{\mathcal{A}}_y$ modulo the individual names; in this case, we write $\widehat{\mathcal{A}}_x \equiv \widehat{\mathcal{A}}_y$ and also say that x *blocks* y. It is *indirectly blocked* if its predecessor is either directly or indirectly blocked. A node is *blocked* if it is either directly or indirectly blocked.

The rules $\exists$ and $\forall$ are applied to $\widehat{\mathcal{A}}$ only if the node x that triggers their execution is not blocked. All other rules are applied only if x is not indirectly blocked.

The total order $\succ$ avoids cycles in the blocking relation. One possibility is to simply use the order in which the individuals were created by the expansion rules. Note that the only individual a that occurs in $\mathcal{A}$, which is the root of the tree-like structure represented by $\widehat{\mathcal{A}}$, cannot be blocked since it is an ancestor of all other individuals in $\widehat{\mathcal{A}}$. With this blocking condition, we can show that the size of $\widehat{\mathcal{A}}$ is bounded exponentially in the size of $\mathcal{A}$, as in the crisp case [23].

Lemma 16. *Every application of expansion rules to $\mathcal{A}$ terminates after at most exponentially many rule applications.*

We say that $\widehat{\mathcal{A}}$ contains a *clash* if it contains two assertions that are equal except for their lattice value (see Definition 14). $\widehat{\mathcal{A}}$ is *complete* if it contains a clash or none of the expansion rules are applicable. The algorithm is correct in the sense that it produces a clash iff $\mathcal{O}$ is not locally consistent (see [12] for details).

Lemma 17. *$\mathcal{O}$ is locally consistent iff some application of the expansion rules to $\mathcal{A}$ yields a complete and clash-free ABox.*

Since the tableau rules are nondeterministic, Lemmata 16 and 17 together imply that the tableaux algorithm decides local consistency in NExpTime.

Theorem 18. *Local consistency in L-$\mathcal{SHI}$ w.r.t. witnessed models can be decided in* NExpTime.

As explained before, L-$\mathcal{SHI}$ has the n-witnessed model property for some $n \geq 1$. We have so far restricted our description to the case where $n = 1$. If $n > 1$, it does not suffice to generate only one successor for every existential and universal restriction, but one must produce n different successors to ensure that the degrees guessed for these complex concepts are indeed witnessed by the model. The only required change to the algorithm is in the rows $\exists$ and $\forall$ of Table 1, where we have to introduce n individuals $y_1, \dots, y_n$, and $2n$ values $\ell_1^1, \ell_2^1, \dots, \ell_1^n, \ell_2^n \in L$ that satisfy $\bigvee_{i=1}^n \ell_1^i \otimes \ell_2^i = \ell$ or $\bigwedge_{i=1}^n \ell_1^i \Rightarrow \ell_2^i = \ell$, respectively.

5 Local Completion and other Black-Box Reductions

In the following, we assume that we have a black-box procedure that decides local consistency in a sublogic of L-$\mathcal{SHI}$. This procedure can be, e.g. the tableau-based algorithm from the previous section, or any other method for solving this decision problem. We show how to employ such a procedure to solve other reasoning problems for this sublogic.

5.1 Consistency

To reduce consistency of an arbitrary ontology $\mathcal{O} = (\mathcal{A}, \mathcal{T}, \mathcal{R})$ to local consistency, we first make sure that the information contained in $\mathcal{A}$ is consistent "in itself", i.e. if we only consider the individuals occurring in $\mathcal{A}$. It then suffices to check a local consistency condition for each of the individuals.

Let $\mathsf{Ind}_\mathcal{A}$ denote the set of individual names occurring in $\mathcal{A}$ and $\mathsf{sub}(\mathcal{A},\mathcal{T})$ the set of all subconcepts of concepts occurring in $\mathcal{A}$ or $\mathcal{T}$. We first guess a set $\widehat{\mathcal{A}}$ of equality assertions of the forms $\langle a : C = \ell\rangle$ and $\langle (a,b) : r = \ell\rangle$ with $a, b \in \mathsf{Ind}_\mathcal{A}$, $C \in \mathsf{sub}(\mathcal{A},\mathcal{T})$, $r \in \mathsf{N_R}$, and $\ell \in L$. We then check whether $\widehat{\mathcal{A}}$ is clash-free and satisfies the tableau conditions listed in Table 1, except the witnessing conditions $\exists$ and $\forall$. Additionally, we impose the following condition on $\widehat{\mathcal{A}}$:

> "If there is an assertion $\langle \alpha \rhd \ell\rangle$ in $\mathcal{A}$, then there is $\ell' \in L$ such that $\ell' \rhd \ell$ and $\langle \alpha = \ell'\rangle$ is in $\widehat{\mathcal{A}}$."

We call $\widehat{\mathcal{A}}$ *locally complete* iff it is of the above form and satisfies all of the above conditions. Guessing this set and checking whether it is locally complete can be done in polynomial time in the size of $\mathcal{O}$.

Lemma 19. *An ontology $\mathcal{O} = (\mathcal{A},\mathcal{T},\mathcal{R})$ is consistent iff there is a locally complete set $\widehat{\mathcal{A}}$ such that $\mathcal{O}_x = (\widehat{\mathcal{A}}_x,\mathcal{T},\mathcal{R})$ is locally consistent for every $x \in \mathsf{Ind}_\mathcal{A}$.*

The proof of this lemma can be found in [12] and uses similar methods as the proofs for the results of the previous section.

Theorem 20. *If local consistency in L-$\mathcal{SHI}$ can be decided in a complexity class* C*, then consistency in L-$\mathcal{SHI}$ can be decided in any complexity class that contains both* NP *and* C*.*

This means that consistency in L-$\mathcal{SHI}$ is decidable in NExpTime. In [10], an automata-based algorithm was presented that can decide satisfiability and subsumption in L-$\mathcal{ALCI}$ in ExpTime. Moreover, if the TBox is acyclic, then this bound can be improved to PSpace. The algorithm can easily be adapted to decide local consistency. With the above reduction, this shows that consistency in L-$\mathcal{ALCI}$ w.r.t. general and acyclic TBoxes can be decided in ExpTime and PSpace, respectively. The same argument applies to any sublogic of L-$\mathcal{SHI}$ for which local consistency can be decided in ExpTime or PSpace.

5.2 Satisfiability, Instance Checking, and Subsumption

To decide whether a concept C is strongly ℓ-satisfiable w.r.t. $\mathcal{O} = (\mathcal{A},\mathcal{T},\mathcal{R})$, we can simply check whether $(\mathcal{A} \cup \{a : C \geq \ell\},\mathcal{T},\mathcal{R})$ is consistent for a new individual name a not occurring in $\mathcal{A}$. Thus, strong ℓ-satisfiability is in the same complexity class as consistency. Moreover, we can easily compute the set of *all* values $\ell \in L$ such that the ontology $(\mathcal{A}\cup\{a : C \geq \ell\},\mathcal{T},\mathcal{R})$ is consistent by calling the decision procedure for consistency a constant number of times, i.e. once for each $\ell \in L$. We can use this set to compute the best bound for the satisfiability of C. Formally, the *best satisfiability degree* of a concept C is the supremum of all $\ell \in L$ such that C is ℓ-satisfiable w.r.t. $\mathcal{O}$. Since we can compute the set of all elements of L satisfying this property, obtaining the best satisfiability degree requires only a supremum computation. As the lattice L is fixed, this adds a constant factor to the complexity of checking consistency.

To check ℓ-instances, we can exploit the fact that a is not an ℓ-instance of C w.r.t. $\mathcal{O}$ iff there is a model $\mathcal{I}$ of $\mathcal{O}$ and a domain element $x \in \Delta^{\mathcal{I}}$ such that $C^{\mathcal{I}}(a^{\mathcal{I}}) \not\geq \ell$. This is the case iff there is a value $\ell' \not\geq \ell$ such that the ontology $(\mathcal{A} \cup \{a : C = \ell'\}, \mathcal{T}, \mathcal{R})$ is consistent. Thus, ℓ-instances can be decided by calling the decision procedure for consistency a constant number of times, namely at most once for each $\ell' \in L$ with $\ell' \not\geq \ell$. We can also compute the *best instance degree* for a and C, which is the supremum of all $\ell \in L$ such that a is an ℓ-instance of C w.r.t. $\mathcal{O}$. Let $\mathfrak{L}$ denote the set of all ℓ' such that $(\{a : C = \ell'\}, \mathcal{T}, \mathcal{R})$ is consistent. The best instance degree for a and C is the infimum of all $\ell' \in \mathfrak{L}$ since

$$\begin{aligned}\bigvee\{\ell \in L \mid a \text{ is an } \ell\text{-instance of } C\} &= \bigvee\{\ell \in L \mid \forall \ell' \not\geq \ell : \ell' \notin \mathfrak{L}\} \\ &= \bigvee\{\ell \in L \mid \forall \ell' \in \mathfrak{L} : \ell \leq \ell'\} = \bigwedge \mathfrak{L}.\end{aligned}$$

Finally, note that C is ℓ-subsumed by D iff a is an ℓ-instance of $C \to D$, where a is a new individual name. Thus, deciding ℓ-subsumption and computing the *best subsumption degree* can be done using the same approach as above.

Lemma 21. *If local consistency in L-$\mathcal{SHI}$ can be decided in a complexity class* C*, then strong satisfiability, instance checking, and subsumption in L-$\mathcal{SHI}$ can be decided in any complexity class that contains both* NP *and* C*.*

This shows that strong satisfiability, instance checking, and subsumption in L-$\mathcal{SHI}$ are in NExpTime. This bound reduces to ExpTime or PSpace if we consider L-$\mathcal{ALCI}$ w.r.t. general or acyclic TBoxes, respectively [10].

6 Conclusions

We have studied fuzzy description logics with semantics based on complete residuated De Morgan lattices. We showed that even for the fairly inexpressive DL L-$\mathcal{ALC}$, strong satisfiability w.r.t. general TBoxes is undecidable for some infinite lattices. For finite lattices, decidability is regained. In fact, local consistency can be decided with a nondeterministic tableaux-based procedure in exponential time. We conjecture that this upper bound can be improved to ExpTime either by an automata-based algorithm or with the help of advanced caching techniques [16]. However, automata-based approaches [10] can only deal with local consistency and concept satisfiability.

Our reduction shows that any algorithm deciding local consistency suffices for deciding consistency of ontologies, through the tableau-based local completion described in Section 5.1. In particular, this yields tight complexity bounds for deciding consistency in L-$\mathcal{ALCI}$ w.r.t. acyclic and general TBoxes–PSpace and ExpTime, respectively. Other decision and computation problems can also be solved using a local consistency reasoner as a black box.

The presented tableaux algorithm has highly nondeterministic rules, and as such is unsuitable for an implementation. Most of the optimizations developed for tableaux algorithms for crisp DLs, like the use of an optimized rule-application ordering, can be transfered to our setting. However, the most important task is to reduce the search space created by the choice of lattice values in most of the rules. We plan to study these optimizations in the future.

References

1. Baader, F., Calvanese, D., McGuinness, D.L., Nardi, D., Patel-Schneider, P.F. (eds.): The Description Logic Handbook: Theory, Implementation, and Applications, 2nd edn. Cambridge University Press (2007)
2. Baader, F., Peñaloza, R.: Are fuzzy description logics with general concept inclusion axioms decidable? In: Proc. FUZZ-IEEE 2011, pp. 1735–1742 (2011)
3. Baader, F., Peñaloza, R.: On the Undecidability of Fuzzy Description Logics with GCIs and Product T-norm. In: Tinelli, C., Sofronie-Stokkermans, V. (eds.) FroCoS 2011. LNCS, vol. 6989, pp. 55–70. Springer, Heidelberg (2011)
4. Baader, F., Sattler, U.: An overview of tableau algorithms for description logics. Studia Logica 69(1), 5–40 (2001)
5. Belnap, N.D.: A useful four-valued logic. In: Modern Uses of Multiple-Valued Logic, pp. 7–37. Reidel Publishing Company (1977)
6. Bobillo, F., Bou, F., Straccia, U.: On the failure of the finite model property in some fuzzy description logics. Fuzzy Set. Syst. 172(1), 1–12 (2011)
7. Bobillo, F., Straccia, U.: Finite fuzzy description logics: A crisp representation for finite fuzzy $\mathcal{ALCH}$. In: Proc. URSW 2010, vol. 654, pp. 61–72. CEUR (2010)
8. Bobillo, F., Straccia, U.: Reasoning with the finitely many-valued Łukasiewicz fuzzy description logic $\mathcal{SROIQ}$. Inform. Sciences 181, 758–778 (2011)
9. Borgwardt, S., Distel, F., Peñaloza, R.: How Fuzzy Is My Fuzzy Description Logic? In: Gramlich, B., Miller, D., Sattler, U. (eds.) IJCAR 2012. LNCS (LNAI), vol. 7364, pp. 82–96. Springer, Heidelberg (2012)
10. Borgwardt, S., Peñaloza, R.: Finite lattices do not make reasoning in $\mathcal{ALCI}$ harder. In: URSW 2011, vol. 778, pp. 51–62. CEUR (2011)
11. Borgwardt, S., Peñaloza, R.: Fuzzy ontologies over lattices with t-norms. In: Proc. DL 2011, vol. 745, pp. 70–80. CEUR (2011)
12. Borgwardt, S., Peñaloza, R.: Consistency in fuzzy description logics over residuated De Morgan lattices. LTCS-Report 12-04. Chair for Automata Theory, TU Dresden (2012), http://lat.inf.tu-dresden.de/research/reports.html
13. Borgwardt, S., Peñaloza, R.: Undecidability of fuzzy description logics. In: Proc. KR 2012. AAAI Press (2012)
14. Cerami, M., Straccia, U.: On the undecidability of fuzzy description logics with GCIs with Łukasiewicz t-norm. Tech. rep., (2011), arXiv:1107.4212v3 [cs.LO]
15. De Cooman, G., Kerre, E.E.: Order norms on bounded partially ordered sets. J. of Fuzzy Math. 2, 281–310 (1993)
16. Goré, R.P., Nguyen, L.A.: EXPTIME Tableaux with Global Caching for Description Logics with Transitive Roles, Inverse Roles and Role Hierarchies. In: Olivetti, N. (ed.) TABLEAUX 2007. LNCS (LNAI), vol. 4548, pp. 133–148. Springer, Heidelberg (2007)
17. Grätzer, G.: General Lattice Theory, 2nd edn. Birkhäuser Verlag (2003)
18. Hájek, P.: Metamathematics of Fuzzy Logic (Trends in Logic). Springer (2001)

19. Hájek, P.: Making fuzzy description logic more general. Fuzzy Set. Syst. 154(1), 1–15 (2005)
20. Horrocks, I., Sattler, U.: A description logic with transitive and inverse roles and role hierarchies. J. Logic and Computation 9(3), 385–410 (1999)
21. Jiang, Y., Tang, Y., Wang, J., Deng, P., Tang, S.: Expressive fuzzy description logics over lattices. Knowl.-Based Syst. 23(2), 150–161 (2010)
22. Molitor, R., Tresp, C.B.: Extending description logics to vague knowledge in medicine. In: Fuzzy Systems in Medicine. STUDFZZ, vol. 41, pp. 617–635. Springer (2000)
23. Motik, B., Shearer, R., Horrocks, I.: Optimized Reasoning in Description Logics Using Hypertableaux. In: Pfenning, F. (ed.) CADE 2007. LNCS (LNAI), vol. 4603, pp. 67–83. Springer, Heidelberg (2007)
24. Post, E.L.: A variant of a recursively unsolvable problem. Bull. AMS 52(4), 264–268 (1946)
25. Stoilos, G., Straccia, U., Stamou, G., Pan, J.Z.: General concept inclusions in fuzzy description logics. In: Proc. ECAI 2006, pp. 457–461. IOS Press (2006)
26. Straccia, U.: Description logics over lattices. Int. J. Uncert. Fuzz. 14(1), 1–16 (2006)
27. Tobies, S.: Complexity Results and Practical Algorithms for Logics in Knowledge Representation. Ph.D. thesis, RWTH Aachen (2001)
28. Viana, H., Alcântara, J., Martins, A.T.C.: Paraconsistent rough description logic. In: Proc. DL 2011, vol. 745. CEUR (2011)

Ontology-Based Governance of Data-Aware Processes*

Diego Calvanese[1], Giuseppe De Giacomo[2], Domenico Lembo[2], Marco Montali[1], and Ario Santoso[1]

[1] Free University of Bozen-Bolzano
{calvanese,montali,santoso}@inf.unibz.it
[2] Sapienza Università di Roma
{deGiacomo,lembo}@dis.uniroma1.it

Abstract. In this paper we show how one can use the technology developed recently for Ontology-Based Data Access (OBDA) to govern data-aware processes through ontologies. In particular, we consider processes executed over a relational database which issue calls to external services to acquire new information and update the data. We equip these processes with an OBDA system, in which an ontology modeling the domain of interest is connected through declarative mappings to the database, and that consequently allows one to understand and govern the manipulated information at the conceptual level. In this setting, we are interested in verifying first-order μ-calculus formulae specifying temporal properties over the evolution of the information at the conceptual level. Specifically, we show how, building on first-order rewritability of queries over the system state that is typical of OBDA, we are able to reformulate the temporal properties into temporal properties expressed over the underlying database. This allows us to adopt notable decidability results on verification of evolving databases that have been established recently.

1 Introduction

Recent work in business processes, services and databases brought the necessity of considering both data and processes simultaneously while designing the system. This holistic view of considering data and processes together has given rise to a line of research under the name of *artifact-centric business processes* [19,16,1,2] that aims at avoiding the notorious discrepancy of traditional approaches where these aspects are considered separately [9]. Recently, interesting decidability results for verification of temporal properties over such systems have been obtained in the context of so-called Data-centric Dynamic Systems (DCDSs) based on relational technology [14,7,5,6]. In a DCDS, processes operate over the data of the system and evolve it by executing actions that may issue calls to external services. The data returned by such external services is injected into the system, effectively making it infinite state. There has been also some work on a form of DCDS based on ontologies, where the data layer is represented in

* This research has been partially supported by the EU under the ICT Collaborative Project ACSI (Artifact-Centric Service Interoperation), grant agreement n. FP7-257593.

a rich ontology formalism, and actions perform a form of instance level update of the ontology [4]. The use of an ontology allows for a high-level conceptual view of the data layer that is better suited for a business level treatment of the manipulated information.

Here we introduce Semantically-Governed Data-Aware Processes (SGDAP), in which we merge these two approaches by enhancing a *relational layer* constituted by a DCDS-based system, with an ontology, constituting a *semantic layer*. The ontology captures the domain in which the SGDAP is executed, and allows for seeing the data and their manipulation at a conceptual level through an ontology-based data access (OBDA) system [10,21]. Hence it provides us with a way of semantically governing the underlying DCDS. Specifically, an SGDAP is constituted by two main components: *(i)* an *OBDA system* [10] which includes (the intensional level of) an ontology, a relational database schema, and a mapping between the ontology and the database; *(ii)* a *process component*, which characterizes the evolution of the system in terms of a process specifying preconditions and effects of action execution over the relational layer.

The ontology is represented through a Description Logic (DL) TBox [3], expressed in a lightweight ontology language of the *DL-Lite* family [12], a family of DLs specifically designed for efficiently accessing to large amounts of data. The mapping is defined in terms of a set of assertions, each relating an arbitrary (SQL) query over the relational layer to a set of atoms whose predicates are the concepts and roles of the ontology, and whose arguments are terms built using specific function symbols applied to the answer variables of the SQL query. Such mappings specify how to populate the elements of the ontology from the data in the database, and function symbols are used to construct (abstract) objects (*object terms*) from the concrete values retrieved from the database. As an example, let us consider a fragment of a university information system in which data about students and their degree is stored and manipulated. An ontology records the fact that both bachelor and master students are students, and that some of the students are graduated. The actual data about students is maintained in a relational database containing, among others, a table storing for currently enrolled students their id, name, surname, type of degree, and for previously enrolled students also the graduation date. The mappings relating the database to the ontology specify that the concepts for bachelor and master students are populated using a simple query that extracts from the enrollment table name, surname, and degree type of each students stored therein, and uses this information to create student objects. This reflects the fact that the combination of these three properties is considered sufficient to identify a student in the modeled domain.

When an SGDAP evolves, each snapshot of the system is characterized by a database instance at the relational layer, and by a corresponding virtual ABox, which together with the TBox provides a conceptual view of the relational instance at the semantic layer. When the system is progressed by the process component, we assume that at every time the current instance can be arbitrarily queried, and can be updated through action executions, possibly involving external service calls to get new values from the environment. Hence the process component relies on three main notions: *actions*, which are the atomic progression steps for the data layer; *external services*, which can be called during the execution of actions; and a *process*, which is essentially a non-deterministic program that uses actions as atomic instructions. In our example, we might have an

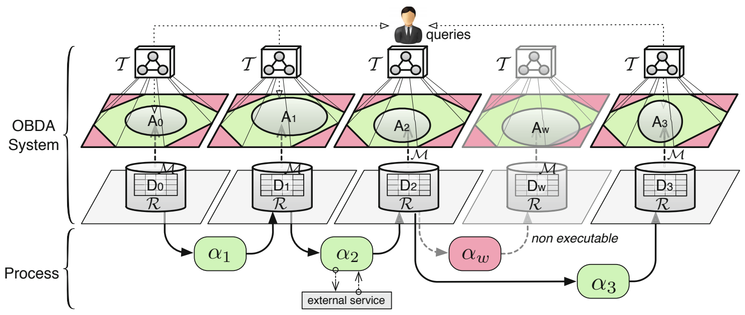


Fig. 1. Overview of a semantically-governed data-aware process

action to graduate a student with a given id, that extracts from the enrollment table the student, provided her graduation date is `NULL` (indicating that the student is not graduated yet), and after obtaining the graduation mark by calling an external service, inserts her in the table of graduated students. During the execution, the snapshots of the relational layer can be virtually mapped as ABoxes in the semantic layer. This enables to: *(i) understand* the evolution of the system at the conceptual level, and *(ii) govern* it at the semantic level, rejecting those actions that, executed at the relational layer, would lead to a new semantic snapshot that is inconsistent with the semantic layer's TBox. Figure 1 gives an intuition about the components of a SGDAP and the usages of the ontology to understand and govern the system execution. The subsequent technical development details the various components of the depicted framework, and the role they play in the system.

In this work, we are in particular interested in verifying dynamic properties specified in a variant of μ-calculus [18], one of the most powerful temporal logics, expressed over the semantic layer of an SGDAP. We consider properties expressed as μ-calculus formulae whose atoms are queries built over the semantic layer. In our running example, we can verify a property stating that every evolution of the system leads to a state in which all students present in that state have graduated. By relying on techniques for query answering in *DL-Lite* OBDA systems, which exploit FOL rewritability of query answering and of ontology satisfiability, we reformulate the temporal properties expressed over the semantic layer into analogous properties over the relational layer. Given that our systems are in general infinite-state, verification of temporal properties is undecidable. However, we show how we can adapt to our setting recent results on the decidability of verification of DCDSs based on suitable finite-state abstractions [6].

2 Preliminaries

In this section we introduce the description logic (DL) *DL-Lite*$_{\mathcal{A},id}$ and describe the ontology-based data access (OBDA) framework.

***DL-Lite*$_{\mathcal{A},id}$** [13,10] allows for specifying *concepts*, representing sets of objects, *roles*, representing binary relations between objects, and *attributes*, representing binary relations between objects and values. For simplicity, in this paper we consider a unique domain for all values used in the system. The syntax of concept, role and attribute *expressions* in *DL-Lite$_{\mathcal{A},id}$* is as follows:

$$B \longrightarrow N \mid \exists R \mid \delta(U) \qquad\qquad R \longrightarrow P \mid P^-$$

Here, N, P, and U respectively denote a *concept name*, a *role name*, and an *attribute name*, P^- denotes the *inverse of a role*, and B and R respectively denote *basic concepts* and *basic roles*. The concept $\exists R$, also called *unqualified existential restriction*, denotes the *domain* of a role R, i.e., the set of objects that R relates to some object. Similarly, the concept $\delta(U)$ denotes the *domain* of an attribute U, i.e., the set of objects that U relates to some value. Note that we consider here a simplified version of *DL-Lite$_{\mathcal{A},id}$* where we distinguish between objects and values, but do not further deal with different datatypes; similarly, we consider only a simplified version of identification assertions.

A *DL-Lite$_{\mathcal{A},id}$ ontology* is a pair $(\mathcal{T}, A)$, where $\mathcal{T}$ is a TBox, i.e., a finite set of *TBox assertions*, and A is an Abox, i.e., a finite set of *ABox assertions*. *DL-Lite$_{\mathcal{A},id}$* TBox assertions have the following form:

$$\begin{array}{ccc} B_1 \sqsubseteq B_2 & R_1 \sqsubseteq R_2 & U_1 \sqsubseteq U_2 \\ B_1 \sqsubseteq \neg B_2 & R_1 \sqsubseteq \neg R_2 & U_1 \sqsubseteq \neg U_2 \\ (\mathsf{id}\ B\ Z_1, \ldots, Z_n) & (\mathsf{funct}\ R) & (\mathsf{funct}\ U) \end{array}$$

From left to right, assertions of the first row respectively denote *inclusions* between basic concepts, basic roles, and attributes; assertions of the second row denote *disjointness* between basic concepts, basic roles, and attributes; assertions of the last row denote *identification (assertions)* (IdA), and global *functionality* on roles and attributes. In the IdA, each Z_i denotes either an attribute or a basic role. Intuitively, an IdA of the above form asserts that for any two different instances o, o' of B, there is at least one Z_i such that o and o' differ in the set of their Z_i-fillers, that is the set of objects (if Z_i is a role) or values (if Z_i is an attribute) that are related to o by Z_i. As usual, in *DL-Lite$_{\mathcal{A},id}$* TBoxes we impose that roles and attributes occurring in functionality assertions or IdAs cannot be specialized (i.e., they cannot occur in the right-hand side of inclusions).

DL-Lite$_{\mathcal{A},id}$ ABox assertions have the form $N(t_1)$, $P(t_1, t_2)$, or $U(t_1, v)$, where t_1 and t_2 denote individual objects and v denotes a value.

The semantics of *DL-Lite$_{\mathcal{A},id}$* is given in [13]. We only recall here that we interpret objects and values over distinct domains, and that for both we adopt the Unique Name Assumption, i.e., different constants denote different objects (or values). The notions of *entailment*, *satisfaction*, and *model* are as usual [13]. We also say that A is *consistent wrt* $\mathcal{T}$ if $(\mathcal{T}, A)$ is satisfiable, i.e., admits at least one model.

Next we introduce queries. As usual (cf. OWL 2), answers to queries are formed by terms denoting individuals appearing in the ABox. The *domain of an ABox A*, denoted by $\textsc{adom}(A)$, is the (finite) set of terms appearing in A. A *union of conjunctive queries* (UCQ) q over a TBox $\mathcal{T}$ is a FOL formula of the form $\exists \vec{y}_1.conj_1(\vec{x}, \vec{y}_1) \vee \cdots \vee \exists \vec{y}_n.conj_n(\vec{x}, \vec{y}_n)$, with free variables $\vec{x}$ and existentially quantified variables $\vec{y}_1, \ldots, \vec{y}_n$. Each $conj_i(\vec{x}, \vec{y_i})$ in q is a conjunction of atoms of the form $N(z)$, $P(z, z')$, $U(z, z')$ where N, P and U respectively denote a concept, role and attribute name of $\mathcal{T}$, and z, z' are constants in a set $\mathcal{C}$ or variables in $\vec{x}$ or $\vec{y_i}$, for some $i \in \{1, \ldots, n\}$. The

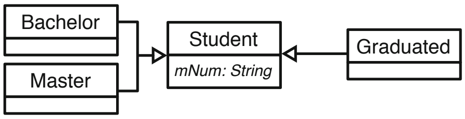


Fig. 2. UML conceptual schema for our running example

(certain) answers to q over an ontology $(\mathcal{T}, A)$ is the set $ans(q, \mathcal{T}, A)$ of substitutions[1] σ of the free variables of q with constants in $\text{ADOM}(A)$ such that $q\sigma$ evaluates to true in every model of $(\mathcal{T}, A)$. If q has no free variables, then it is called *boolean*, and its certain answers are true or false. Computing $ans(q, \mathcal{T}, A)$ of a UCQ q over a *DL-Lite*$_{\mathcal{A},id}$ ontology $(\mathcal{T}, A)$ is in AC^0 in the size of A [13]. This is actually a consequence of the fact that *DL-Lite*$_{\mathcal{A},id}$ enjoys the *FOL rewritability* property, which in our setting says that for every UCQ q, $ans(q, \mathcal{T}, A)$ can be computed by evaluating the UCQ $\text{REW}(q, \mathcal{T})$ over A considered as a database. $\text{REW}(q, \mathcal{T})$ is the so-called perfect reformulation of q w.r.t. $\mathcal{T}$ [13]. We also recall that, in *DL-Lite*$_{\mathcal{A},id}$, ontology satisfiability is FOL rewritable. In other words, we can construct a boolean FOL query $\mathsf{q}_{\mathsf{unsat}}(\mathcal{T})$ that evaluates to true over an ABox A iff the ontology $(\mathcal{T}, A)$ is unsatisfiable.

In our framework, we consider an extension of UCQs, called ECQs, which are queries of the query language *EQL-Lite*(UCQ) [11]. Formally, an *ECQ* over a TBox $\mathcal{T}$ is a possibly open *domain independent* formula of the form:

$$Q \longrightarrow [q] \mid \neg Q \mid Q_1 \wedge Q_2 \mid \exists x.Q \mid x = y$$

where q is a UCQ over $\mathcal{T}$ and $[q]$ denotes that q is evaluated under the (minimal) knowledge operator (cf. [11]). To compute the certain answers $\text{ANS}(Q, \mathcal{T}, A)$ to an ECQ Q over an ontology $(\mathcal{T}, A)$, we can compute the certain answers over $(\mathcal{T}, A)$ of each UCQ embedded in Q, and evaluate the first-order part of Q over the relations obtained as the certain answers of the embedded UCQs. Hence, also computing $\text{ANS}(Q, \mathcal{T}, A)$ of an ECQ Q over a *DL-Lite*$_{\mathcal{A},id}$ ontology $(\mathcal{T}, A)$ is in AC^0 in the size of A [11].

Ontology-Based Data Access (OBDA). In an OBDA system, a relational database is connected to an ontology that represents the domain of interest by a mapping, which relates database values with values and (abstract) objects in the ontology (c.f. [10]). In particular, we make use of a countably infinite set $\mathcal{V}$ of values and a set Λ of function symbols, each with an associated arity. We also define the set $\mathcal{C}$ of constants as the union of $\mathcal{V}$ and the set $\{f(d_1, \ldots, d_n) \mid f \in \Lambda \text{ and } d_1, \ldots, d_n \in \mathcal{V}\}$ of *object terms*.

Formally, an OBDA system is a structure $\mathcal{O} = \langle \mathcal{R}, \mathcal{T}, \mathcal{M} \rangle$, where: *(i)* $\mathcal{R} = \{R_1, \ldots, R_n\}$ is a database schema, constituted by a finite set of relation schemas; *(ii)* $\mathcal{T}$ is a *DL-Lite*$_{\mathcal{A},id}$ TBox; *(iii)* $\mathcal{M}$ is a set of mapping assertions, each of the form $\Phi(\vec{x}) \rightsquigarrow \Psi(\vec{y}, \vec{t})$, where: (a) $\vec{x}$ is a non-empty set of variables, (b) $\vec{y} \subseteq \vec{x}$, (c) $\vec{t}$ is a set of object terms of the form $f(\vec{z})$, with $f \in \Lambda$ and $\vec{z} \subseteq \vec{x}$, (d) $\Phi(\vec{x})$ is an arbitrary SQL query over $\mathcal{R}$, with $\vec{x}$ as output variables, and (e) $\Psi(\vec{y}, \vec{t})$ is a CQ over $\mathcal{T}$ of arity $n > 0$ without non-distinguished variables, whose atoms are over the variables $\vec{y}$ and the object terms $\vec{t}$. Without loss of generality, we use the special function symbol $val/1$ to map values from the relational layer to the range of attributes in the semantic layer.

[1] As customary, we can view each substitution simply as a tuple of constants, assuming some ordering of the free variables of q.

Example 1. We formalize our running example presented in Section 1, and consider a simple university information system that stores and manipulates data concerning students and their degree. In particular, we define an OBDA system $\mathcal{O} = \langle \mathcal{R}, \mathcal{T}, \mathcal{M} \rangle$ to capture the conceptual schema of such a domain, how data are concretely maintained in a relational database, and how the two information levels are linked through mappings. The TBox $\mathcal{T}$, shown in Figure 2 using the notation of UML class diagrams, is constituted by the following assertions:

$$\begin{array}{rlrll} \mathsf{Bachelor} \sqsubseteq \mathsf{Student} & & \delta(\mathsf{MNum}) \sqsubseteq \mathsf{Student} & & (\mathsf{funct}\ \mathsf{MNum}) \\ \mathsf{Master} \sqsubseteq \mathsf{Student} & & \mathsf{Student} \sqsubseteq \delta(\mathsf{MNum}) & & (\mathsf{id}\ \mathsf{Student}\ \mathsf{MNum}) \\ \mathsf{Graduated} \sqsubseteq \mathsf{Student} & & & & \end{array}$$

The conceptual schema states that Bachelor, Master, and Graduated are subclasses of Student, and that MNum (representing the matriculation number) is an attribute of Student. The conceptual schema also expresses that: *(i)* each Student has *exactly one* matriculation number (by composing the assertion stating that each Student must be in the domain of MNum with the assertion stating that MNum is functional); *(ii)* matriculation numbers can be used to *identify* Students (i.e., each MNum is associated to *at most one* Student). Data related to students are maintained in a concrete underlying data source that obeys the database schema $\mathcal{R}$, constituted by the following relation schemas: *(i)* $\mathsf{ENROLLED}(\mathsf{id}, \mathsf{name}, \mathsf{surname}, \mathsf{type}, \mathsf{endDate})$ stores information about students that are currently (endDate=`NULL`) or were enrolled in a bachelor (type=`"BSc"`) or master (type=`"MSc"`) course. *(ii)* $\mathsf{GRAD}(\mathsf{id}, \mathsf{mark}, \mathsf{type})$ stores data of former students who have been graduated. *(iii)* $\mathsf{TRANSF\_M}(\mathsf{name}, \mathsf{surname})$ is a temporary relation used to maintain information about master students that have been recently transferred from another university, and must still complete the enrollment process. The interconnection between $\mathcal{R}$ and $\mathcal{T}$ is specified through the following set $\mathcal{M}$ of mapping assertions:

m_1 : `SELECT` name, surname, type `FROM` ENROLLED `WHERE` type = `"BSc"`
$\rightsquigarrow$ Bachelor (stu$_1$ (name,surname,type))
m_2 : `SELECT` name, surname, type `FROM` ENROLLED `WHERE` type = `"MSc"`
$\rightsquigarrow$ Master (stu$_1$ (name,surname,type))
m_3 : `SELECT` name, surname, type, id `FROM` ENROLLED
$\rightsquigarrow$ MNum (stu$_1$ (name,surname,type), *val*(id))
m_4 : `SELECT` name, surname `FROM` TRANSF_M
$\rightsquigarrow$ Master (stu$_1$ (name,surname, `"MSc"`))
m_5 : `SELECT` e. name, e. surname, e. type `FROM` ENROLLED e, GRAD g `WHERE` e. id = g. id
$\rightsquigarrow$ Graduated (stu$_1$ (name,surname,type))

Intuitively, m_1 (resp., m_2) maps every id in ENROLLED with type `"BSc"` (`"MSc"`) to a bachelor (master) student. Such a student is constructed by "objectifying" the name, surname and course type using variable term $\mathsf{stu}_1/3$. In m_3, the MNum attribute is instead created using directly the value of id to fill in the target of the attribute. Notice the use of the *val* function symbol for mapping id to the range of MNum. Mapping m_4 leads to create further master students by starting from the temporary TRANSF_M table. Since such students are not explicitly associated to course type, but it is intended that they are `"MSc"`, objectification is applied to students' name and surname, adding `"MSc"` as a constant in the variable term. Notice that, according to the TBox $\mathcal{T}$, such students have a matriculation number, but its value is not known (and, in fact, no mapping exists to generate their MNum attribute). Finally, m_5 generates graduated students by selecting only those students in the ENROLLED table whose matriculation number is also contained in the GRAD table. □

Given a database instance D made up of values in $\mathcal{V}$ and conforming to schema $\mathcal{R}$, and given a mapping $\mathcal{M}$, the *virtual ABox* generated from D by a mapping assertion

$m = \Phi(x) \rightsquigarrow \Psi(y,t)$ in $\mathcal{M}$ is $m(D) = \bigcup_{v \in eval(\Phi,D)} \Psi[x/v]$, where $eval(\Phi, D)$ denotes the evaluation of the SQL query Φ over D, and where we consider $\Psi[x/v]$ to be a set of atoms (as opposed to a conjunction). Then, the ABox generated from D by the mapping $\mathcal{M}$ is $\mathcal{M}(D) = \bigcup_{m \in \mathcal{M}} m(D)$. Notice that $\textsc{adom}(\mathcal{M}(D)) \subseteq \mathcal{C}$. As for ABoxes, the active domain $\textsc{adom}(D)$ of a database instance D is the set of values occurring in D. Notice that $\textsc{adom}(D) \subseteq \mathcal{V}$. Given an OBDA system $\mathcal{O} = \langle \mathcal{R}, \mathcal{T}, \mathcal{M} \rangle$ and a database instance D for $\mathcal{R}$, a *model* for $\mathcal{O}$ wrt D is a model of the ontology $(\mathcal{T}, \mathcal{M}(D))$. We say that $\mathcal{O}$ wrt D is satisfiable if it admits a model wrt D.

Example 2. Consider a DB instance $D = \{\mathsf{ENROLLED}(123, \mathsf{john}, \mathsf{doe}, \texttt{"BSc"}, \texttt{NULL})\}$. The corresponding virtual ABox obtained from the application of the mapping $\mathcal{M}$ is $\mathcal{M}(D) = \{\mathsf{Bachelor}(\mathsf{stu_1}(\mathsf{john}, \mathsf{doe}, \texttt{"BSc"})), \mathsf{MNum}(\mathsf{stu_1}(\mathsf{john}, \mathsf{doe}, \texttt{"BSc"}), \mathit{val}(123))\}$. □

A UCQ q over an OBDA system $\mathcal{O} = \langle \mathcal{R}, \mathcal{T}, \mathcal{M} \rangle$ is simply an UCQ over $\mathcal{T}$. To compute the certain answers of q over $\mathcal{O}$ wrt a database instance D for $\mathcal{R}$, we follow a three-step approach: *(i)* q is *rewritten* to compile away $\mathcal{T}$, obtaining $q_r = \textsc{rew}(q, \mathcal{T})$; *(ii)* the mapping $\mathcal{M}$ is used to *unfold* q_r into a query over $\mathcal{R}$, denoted by $\textsc{unfold}(q_r, \mathcal{M})$, which turns out to be an SQL query [20]; *(iii)* such a query is executed over D, obtaining the certain answers. For an ECQ, we can proceed in a similar way, applying the rewriting and unfolding steps to the embedded UCQs. It follows that computing certain answers to UCQs/ECQs in an OBDA system is FOL rewritable. Applying the unfolding step to $\mathsf{q_{unsat}}(\mathcal{T})$, we obtain also that satisfiability in $\mathcal{O}$ is FOL rewritable.

3 Semantically-Governed Data-Aware Processes

A Semantically-Governed Data-Aware Process (SGDAP) $\mathcal{S} = \langle \mathcal{O}, \mathcal{P}, D_0 \rangle$ is formed by an OBDA System $\mathcal{O} = \langle \mathcal{R}, \mathcal{T}, \mathcal{M} \rangle$ by a process component $\mathcal{P}$, and by an initial database instance D_0 that conforms to the relational schema $\mathcal{R}$ in $\mathcal{O}$. Intuitively, the OBDA system keeps all the data of interest, while the process component modifies and evolves such data, starting from the initial database D_0.

The process component $\mathcal{P}$ constitutes the progression mechanism for the SGDAP. Formally, $\mathcal{P} = \langle \mathcal{F}, \mathcal{A}, \pi \rangle$, where: *(i)* $\mathcal{F}$ is a finite set of *functions* representing calls to *external services*, which return values; *(ii)* $\mathcal{A}$ is a finite set of *actions*, whose execution progresses the data layer, and may involve external service calls; *(iii)* π is a finite set of *condition-action rules* that form the specification of the overall *process*, which tells at any moment which actions can be executed.

An *action* $\alpha \in \mathcal{A}$ has the form $\alpha(p_1, \ldots, p_n) : \{e_1, \ldots, e_m\}$, where: *(i)* $\alpha(p_1, \ldots, p_n)$ is the *signature* of the action, constituted by a name α and a sequence $p_1, \ldots, p_n$ of *input parameters* that need to be substituted with values for the execution of the action, and *(ii)* $\{e_1, \ldots, e_m\}$ is a set of *effect specifications*, whose specified effects are assumed to take place simultaneously. Each e_i has the form $q_i^+ \wedge Q_i^- \rightsquigarrow E_i$, where: *(a)* $q_i^+ \wedge Q_i^-$ is a query over $\mathcal{R}$ whose terms are variables $\vec{x}$, action parameters, and constants from $\textsc{adom}(D_0)$. The query q_i^+ is a UCQ, and the query Q_i^- is an arbitrary FOL formula whose free variables are included in those of q_i^+. Intuitively, q_i^+ selects the tuples to instantiate the effect, and Q_i^- filters *away* some of them[2]. *(b)* E_i is

[2] To convey this intuition, we use the "$^-$" superscript.

the effect, i.e., a set of facts for $\mathcal{R}$, which includes as terms: terms in $\textsc{adom}(D_0)$, input parameters, free variables of q_i^+, and in addition Skolem terms formed by applying a function $f \in \mathcal{F}$ to one of the previous kinds of terms. Such Skolem terms involving functions represent external service calls and are interpreted so as to return a value chosen by an external user/environment when executing the action.

The *process* π is a finite set of *condition-action rules* $Q \mapsto \alpha$, where α is an action in $\mathcal{A}$ and Q is a FOL query over $\mathcal{R}$ whose free variables are exactly the parameters of α, and whose other terms can be quantified variables or values in $\textsc{adom}(D_0)$.

Example 3. Consider the OBDA system $\mathcal{O}$ defined in Example 1. We now define a process component $\mathcal{P} = \langle \mathcal{F}, \mathcal{A}, \pi \rangle$ over the relational schema $\mathcal{R}$ of $\mathcal{O}$, so as to obtain a full SGDAP. In particular, π is constituted by the following condition-action rules ('_' denotes existentially quantified variables that are not used elsewhere):

- $\mathsf{ENROLLED}(\mathsf{id}, \_, \_, \_, \text{NULL}) \rightsquigarrow \textsc{graduate}(\mathsf{id})$
- $\mathsf{TRANSF\_M}(\mathsf{name}, \mathsf{surname}) \rightsquigarrow \textsc{compl-enr}(\mathsf{name}, \mathsf{surname})$

The first rule extracts a matriculation number id of a currently enrolled student (endDate=NULL) from the ENROLLED relation and graduates the student, whereas the second rule selects a pair name surname in TRANSF_M and use them to complete the enrollment of that student. In order to be effectively executed, the involved actions rely on the following set $\mathcal{F}$ of service calls: *(i)* $\mathsf{today}()$ returns the current date; *(ii)* $\mathsf{getMark}(\mathsf{id}, \mathsf{type})$ returns the final mark received by student id; *(iii)* $\mathsf{getID}(\mathsf{name}, \mathsf{surname}, \mathsf{type})$ returns the matriculation number for the name-surname pair of a student. The two actions GRADUATE and COMPL-ENR are then defined as follows:

$$
\begin{aligned}
\textsc{graduate}(\mathsf{id}) : \{\ & \mathsf{GRAD}(\mathsf{id}_2, \mathsf{m}, \mathsf{t}) \rightsquigarrow \mathsf{GRAD}(\mathsf{id}_2, \mathsf{m}, \mathsf{t}),\\
& \mathsf{TRANSF\_M}(\mathsf{n}, \mathsf{s}) \rightsquigarrow \mathsf{TRANSF\_M}(\mathsf{n}, \mathsf{s}),\\
& \mathsf{ENROLLED}(\mathsf{id}_2, \mathsf{n}, \mathsf{s}, \mathsf{t}, \mathsf{d}) \wedge \mathsf{id}_2 \neq \mathsf{id} \rightsquigarrow \mathsf{ENROLLED}(\mathsf{id}_2, \mathsf{n}, \mathsf{s}, \mathsf{t}, \mathsf{d}),\\
& \mathsf{ENROLLED}(\mathsf{id}, \mathsf{n}, \mathsf{s}, \mathsf{t}, \text{NULL}) \rightsquigarrow \mathsf{ENROLLED}(\mathsf{id}, \mathsf{n}, \mathsf{s}, \mathsf{t}, \mathsf{today}()),\\
& \mathsf{ENROLLED}(\mathsf{id}, \_, \_, \mathsf{t}, \text{NULL}) \rightsquigarrow \mathsf{GRAD}(\mathsf{id}, \mathsf{getMark}(\mathsf{id}, \mathsf{t}), \mathsf{t})\ \}\\
\textsc{compl-enr}(\mathsf{n}, \mathsf{s}) : \{\ & \mathsf{GRAD}(\mathsf{id}, \mathsf{m}, \mathsf{t}) \rightsquigarrow \mathsf{GRAD}(\mathsf{id}, \mathsf{m}, \mathsf{t}),\\
& \mathsf{ENROLLED}(\mathsf{id}, \mathsf{n}_2, \mathsf{s}_2, \mathsf{t}, \mathsf{d}) \rightsquigarrow \mathsf{ENROLLED}(\mathsf{id}, \mathsf{n}_2, \mathsf{s}_2, \mathsf{t}, \mathsf{d}),\\
& \mathsf{TRANSF\_M}(\mathsf{n}_2, \mathsf{s}_2) \wedge (\mathsf{n}_2 \neq \mathsf{n} \vee \mathsf{s}_2 \neq \mathsf{s}) \rightsquigarrow \mathsf{TRANSF\_M}(\mathsf{n}_2, \mathsf{s}_2),\\
& \mathsf{TRANSF\_M}(\mathsf{n}, \mathsf{s})\\
& \quad \rightsquigarrow \mathsf{ENROLLED}(\mathsf{getID}(\mathsf{n}, \mathsf{s}, \texttt{"MSc"}), \mathsf{n}, \mathsf{s}, \texttt{"MSc"}, \mathsf{NULL})\}
\end{aligned}
$$

Given a matriculation number id, action GRADUATE inserts a new tuple for id in GRAD, updating at the same time the enrollment's end date for id in ENROLLED to the current date, while keeping all other entries in TRANSF_M, GRAD and ENROLLED. Given a name and surname, action COMPL-ENR has the effect of moving the corresponding tuple in TRANSF_M to a new tuple in ENROLLED, for which the matriculation number is obtained by interacting with the getID service call; all other entries TRANSF_M, GRAD and ENROLLED are preserved. □

4 Execution Semantics

This work focuses on the semantics of SGDAP assuming that *external services behave nondeterministically*, i.e., two calls of a service with the same arguments may return different results during the same run. This captures both services that model a truly nondeterministic process (e.g., human operators), and services that model stateful servers.

Let $\mathcal{S} = \langle \mathcal{O}, \mathcal{P}, D_0 \rangle$ be a SGDAP where $\mathcal{O} = \langle \mathcal{R}, \mathcal{T}, \mathcal{M} \rangle$ and $\mathcal{P} = \langle \mathcal{F}, \mathcal{A}, \pi \rangle$. The semantics of $\mathcal{S}$ is defined in terms of a possibly infinite transition system (TS). More specifically, two possible transition systems can be constructed to describe the execution semantics of $\mathcal{S}$: *(i)* a *relational layer transition system* (RTS), representing all allowed computations that, starting from D_0, the process component can do over the data of the relational layer, according to the constraints imposed at the semantic layer (semantic governance); *(ii)* a *semantic layer transition system* (STS), representing the same computations at the semantic layer. To construct these transition systems, we first define the semantics of *action execution*. Let α be an action in $\mathcal{A}$ of the form $\alpha(\vec{p}) : \{e_1, \ldots, e_n\}$ with effects $e_i = q_i^+ \wedge Q_i^- \rightsquigarrow E_i$, and let σ be a substitution of $\vec{p}$ with values in $\mathcal{V}$. The evaluation of the effects of α on a database instance D using a substitution σ is captured by the following function:

$$\textsc{do}(D, \alpha, \sigma) \ = \ \bigcup_{q_i^+ \wedge Q_i^- \rightsquigarrow E_i \text{in } \alpha} \bigcup_{\theta \in \textsc{ans}((q_i^+ \wedge Q_i^-)\sigma, D)} E_i \sigma \theta$$

which returns a database instance made up of values in $\mathcal{V}$ and Skolem terms representing service calls. We denote with $\textsc{calls}(\textsc{do}(D, \alpha, \sigma))$ such service calls, and with $\textsc{evals}(D, \alpha, \sigma)$ the set of substitutions that replace these service calls with values in $\mathcal{V}$:

$$\textsc{evals}(D, \alpha, \sigma) = \{\theta \mid \theta : \textsc{calls}(\textsc{do}(D, \alpha, \sigma)) \to \mathcal{V} \text{ is a total function}\}.$$

We then say that the database instance D' is *produced* from D by the application of action α using substitution σ if $D' = \textsc{do}(D, \alpha, \sigma)\theta$, where $\theta \in \textsc{evals}(D, \alpha, \sigma)$.

Relational Layer Transition System (RTS). Let $\mathcal{S} = \langle \mathcal{O}, \mathcal{P}, D_0 \rangle$ be a SGDAP with $\mathcal{O} = \langle \mathcal{R}, \mathcal{T}, \mathcal{M} \rangle$. The RTS $\Upsilon_{\mathcal{S}}^{\mathrm{R}}$ of $\mathcal{S}$ is formally defined as $\langle \mathcal{R}, \Sigma, s_0, db, \Rightarrow \rangle$, where Σ is a (possibly infinite) set of states, s_0 is the initial state, db is a total function from states in Σ to database instances made up of values in $\mathcal{V}$ and conforming to $\mathcal{R}$, and $\Rightarrow \subseteq \Sigma \times \Sigma$ is a transition relation. Σ, $\Rightarrow$ and db are defined by simultaneous induction as the smallest sets such that $s_0 \in \Sigma$, with $db(s_0) = D_0$, and satisfying the following property: Given $s \in \Sigma$, for each condition-action rule $Q(\vec{p}) \mapsto \alpha(\vec{p}) \in \pi$, for each substitution σ of $\vec{p}$ such that $\sigma \in \textsc{ans}(Q, D)$, consider every database instance D' produced from D by the application of α using σ. Then: *(i)* if there exists $s' \in \Sigma$ such that $db(s') = D'$, then $s \Rightarrow s'$; *(ii)* otherwise, if $\mathcal{O}$ is *satisfiable* wrt D', then $s' \in \Sigma$, $s \Rightarrow s'$ and $db(s') = D'$, where s' is a fresh state. We observe that the satisfiability check done in the last step of the RTS construction accounts for *semantic governance*.

Semantic Layer Transition System (STS). Given a SGDAP $\mathcal{S}$ with $\mathcal{O} = \langle \mathcal{R}, \mathcal{T}, \mathcal{M} \rangle$ and with RTS $\Upsilon_{\mathcal{S}}^{\mathrm{R}} = \langle \mathcal{R}, \Sigma, s_0, db, \Rightarrow \rangle$, the STS $\Upsilon_{\mathcal{S}}^{\mathrm{S}}$ of $\mathcal{S}$ is a "virtualization" of the RTS in the semantic layer. In particular, $\Upsilon_{\mathcal{S}}^{\mathrm{S}}$ maintains the structure of $\Upsilon_{\mathcal{S}}^{\mathrm{R}}$ unaltered, reflecting that the process component is executed over the relational layer, but it associates each state to a virtual ABox obtained from the application of the mapping $\mathcal{M}$ to the database instance associated by $\Upsilon_{\mathcal{S}}^{\mathrm{R}}$ to the same state. Formally, $\Upsilon_{\mathcal{S}}^{\mathrm{S}} = \langle \mathcal{T}, \Sigma, s_0, abox, \Rightarrow \rangle$, where *abox* is a total function from Σ to ABoxes made up of individual objects in $\mathcal{C}$ and conforming to $\mathcal{T}$, such that for each $s \in \Sigma$ with $db(s) = D$, $abox(s) = \mathcal{M}(D)$.

5 Dynamic Constraints Formalism

Let $\mathcal{S} = \langle \mathcal{O}, \mathcal{P}, D_0 \rangle$ be an SGDAP where $\mathcal{O} = \langle \mathcal{R}, \mathcal{T}, \mathcal{M} \rangle$ and $\mathcal{P} = \langle \mathcal{F}, \mathcal{A}, \pi \rangle$. We are interested in the verification of *conceptual temporal properties* over $\mathcal{S}$, i.e., properties that constrain the dynamics of $\mathcal{S}$ understood at the semantic layer. Technically, this means that properties are verified over the SGDAP's STS $\Upsilon_{\mathcal{S}}^{\mathrm{S}}$, combining temporal operators with queries posed over the ontologies obtained by combining the TBox $\mathcal{T}$ with the ABoxes associated to the states of $\Upsilon_{\mathcal{S}}^{\mathrm{S}}$. More specifically, we adopt ECQs [11] to query the ontologies of $\Upsilon_{\mathcal{S}}^{\mathrm{S}}$, and μ-calculus [18] to predicate over the dynamics of $\Upsilon_{\mathcal{S}}^{\mathrm{S}}$.

We use a variant of μ-calculus [18], one of the most powerful temporal logics subsuming LTL, PSL, and CTL* [15], called $\mu\mathcal{L}_{\mathrm{C}}^{\mathrm{EQL}}$, whose formulae have the form:

$$\Phi ::= Q \mid Z \mid \neg\Phi \mid \Phi_1 \vee \Phi_2 \mid \exists x \in \mathcal{C}_0.\Phi \mid \langle - \rangle \Phi \mid \mu Z.\Phi$$

where Q is an ECQ over $\mathcal{T}$, $\mathcal{C}_0 = \textsc{adom}(\mathcal{M}(D_0))$ is the set of object terms appearing in the initial virtual ABox (obtained by applying the mapping $\mathcal{M}$ over the database instance D_0), and Z is a predicate variable. As usual, syntactic monotonicity is enforced to ensure existence of unique fixpoints. Beside the usual FOL abbreviations, we also make use of the following ones: $[-]\Phi = \neg\langle - \rangle(\neg\Phi)$ and $\nu Z.\Phi = \neg\mu Z.\neg\Phi[Z/\neg Z]$. The subscript C in $\mu\mathcal{L}_{\mathrm{C}}^{\mathrm{EQL}}$ stands for "closed", and attests that ECQs are closed queries. In fact, $\mu\mathcal{L}_{\mathrm{C}}^{\mathrm{EQL}}$ formulae only support the limited form of quantification $\exists x \in \mathcal{C}_0.\Phi$, which is a convenient, compact notation for $\bigvee_{c \in \textsc{adom}(\mathcal{M}(D_0))} \Phi[x/c]$. We make this assumption for simplicity, but actually, with some care, our result can be extended to a more general form of quantification over time [6].

In order to define the semantics of $\mu\mathcal{L}_{\mathrm{C}}^{\mathrm{EQL}}$ we resort to STSs. Let $\Upsilon = \langle \mathcal{T}, \Sigma, s_0, \mathit{abox}, \Rightarrow \rangle$ be an STS. Let V be a predicate and individual variable valuation on Υ, i.e., a mapping from the predicate variables Z to subsets of the states Σ, and from individual variables to constants in $\textsc{adom}(\mathcal{M}(D_0))$. Then, we assign meaning to $\mu\mathcal{L}_{\mathrm{C}}^{\mathrm{EQL}}$ formulas by associating to Υ and V an *extension function* $(\cdot)_V^{\Upsilon}$, which maps $\mu\mathcal{L}_{\mathrm{C}}^{\mathrm{EQL}}$ formulas to subsets of Σ. The extension function $(\cdot)_V^{\Upsilon}$ is defined inductively as:

$$\begin{array}{ll}
(Q)_V^{\Upsilon} & = \{s \in \Sigma \mid \textsc{ans}(QV, \mathcal{T}, \mathit{abox}(s)) = \mathsf{true}\} \\
(Z)_V^{\Upsilon} & = V(Z) \subseteq \Sigma \\
(\neg\Phi)_V^{\Upsilon} & = \Sigma - (\Phi)_V^{\Upsilon} \\
(\Phi_1 \vee \Phi_2)_V^{\Upsilon} & = (\Phi_1)_V^{\Upsilon} \cup (\Phi_2)_V^{\Upsilon} \\
(\exists x \in \mathcal{C}_0.\Phi)_V^{\Upsilon} & = \bigcup\{(\Phi)_{V[x/c]}^{\Upsilon} \mid c \in \textsc{adom}(\mathcal{M}(D_0))\} \\
(\langle - \rangle\Phi)_V^{\Upsilon} & = \{s \in \Sigma \mid \exists s'.\ s \Rightarrow s' \text{ and } s' \in (\Phi)_V^{\Upsilon}\} \\
(\mu Z.\Phi)_V^{\Upsilon} & = \bigcap\{\mathcal{E} \subseteq \Sigma \mid (\Phi)_{V[Z/\mathcal{E}]}^{\Upsilon} \subseteq \mathcal{E}\}
\end{array}$$

Intuitively, the extension function $(\cdot)_V^{\Upsilon}$ assigns to the various $\mu\mathcal{L}_{\mathrm{C}}^{\mathrm{EQL}}$ constructs the following meanings. The boolean connectives have the expected meaning, while quantification is restricted to constants in $\textsc{adom}(\mathcal{M}(D_0))$. The extension of $\langle - \rangle\Phi$ consists of the states s such that for *some* state s' with $s \Rightarrow s'$, we have that Φ holds in s', The extension of $\mu Z.\Phi$ is the *smallest subset* $\mathcal{E}_\mu$ of Σ such that, assigning to Z the extension $\mathcal{E}_\mu$, the resulting extension of Φ is contained in $\mathcal{E}_\mu$. When Φ is a closed formula, $(\Phi)_V^{\Upsilon}$ does not depend on V, and we denote it by $(\Phi)^{\Upsilon}$. We are interested in the *model* *checking* problem, i.e., verify whether a $\mu\mathcal{L}_C^{EQL}$ *closed formula* Φ *holds for the SGDAP* $\mathcal{S}$. This problem is defined as checking whether $s_0 \in (\Phi)^{\Upsilon_\mathcal{S}^{\mathrm{S}}}$, that is, whether Φ is true in the initial state s_0 of $\Upsilon_\mathcal{S}^{\mathrm{S}}$. If it is the case, we write $\Upsilon_\mathcal{S}^{\mathrm{S}} \models \Phi$.

Example 4. An example of dynamic property in our running example is $\Phi = \mu Z.((\forall s.[\mathsf{Student(s)}] \rightarrow [\mathsf{Graduated(s)}]) \vee [-]Z)$, which expresses that every evolution of the system leads to a state in which all students present in that state are graduated. □

6 Rewriting μ-Calculus Formulae

Let $\mathcal{S} = \langle \mathcal{O}, \mathcal{P}, D_0 \rangle$ be an SGDAP where $\mathcal{O} = \langle \mathcal{R}, \mathcal{T}, \mathcal{M} \rangle$ and $\mathcal{P} = \langle \mathcal{F}, \mathcal{A}, \pi \rangle$. In this section, we show how verification of $\mu\mathcal{L}_{\mathrm{C}}^{\mathrm{EQL}}$ properties over the STS $\Upsilon_\mathcal{S}^{\mathrm{S}}$ can be reduced to verification of $\mu\mathcal{L}_C$ properties over the corresponding RTS $\Upsilon_\mathcal{S}^{\mathrm{R}}$.

$\mu\mathcal{L}_C$ properties are μ-calculus properties whose atoms are closed, domain-independent FO queries over a database schema. More specifically, the semantics of $\mu\mathcal{L}_C$ is defined over an RTS $\Upsilon^{\mathrm{R}} = \langle \mathcal{R}, \Sigma, s_0, db, \Rightarrow \rangle$, following exactly the same line given in Section 5 for $\mu\mathcal{L}_{\mathrm{C}}^{\mathrm{EQL}}$ and STSs, except for local queries, whose semantics is:

$$(Q)_V^{\Upsilon^{\mathrm{R}}} = \{s \in \Sigma \mid eval(QV, db(s)) = \mathsf{true}\}$$

where $eval(QV, db(s)) = \mathsf{true}$ iff QV is true in $db(s)$, considered as a FOL interpretation. Let Φ be a $\mu\mathcal{L}_{\mathrm{C}}^{\mathrm{EQL}}$ dynamic property specified over the TBox $\mathcal{T}$ of $\mathcal{S}$. The reduction is realized by providing a translation mechanism from Φ into a corresponding $\mu\mathcal{L}_C$ property Φ' specified over $\mathcal{R}$, and then showing that $\Upsilon_\mathcal{S}^{\mathrm{S}} \models \Phi$ if and only if $\Upsilon_\mathcal{S}^{\mathrm{R}} \models \Phi'$.

Before dealing with the translation of Φ, we substitute each subformula of the form $\exists x \in \mathcal{C}_0.\Psi$ into the equivalent form $\bigvee_{c \in \textsc{adom}(\mathcal{M}(D_0))} \Psi[x/c]$. This means that when such a form of quantification is used, the initial ABox must be *materialized* in order to compute the active domain of the initial ABox in the semantic layer. We then deal with the translation of Φ, by separating the treatment of the dynamic part and of the embedded ECQs. Since the dynamics of an SGDAP is completely determined at the relational layer, the dynamic part is maintained unaltered. ECQs are instead manipulated as defined in Section 2, performing in particular the following two-step translation: (1) the TBox $\mathcal{T}$ used by the property is compiled away, rewriting the original formula into a "self-contained" equivalent formula $\Phi_r = \textsc{rew}(\Phi, \mathcal{T})$, obtained by replacing each embedded ECQ with its corresponding rewriting wrt $\mathcal{T}$ [20]; (2) by using the information contained in the mapping $\mathcal{M}$, Φ_r is unfolded to the relational layer into $\textsc{unfold}(\Phi_r, \mathcal{M})$, by replacing each embedded ECQ with its corresponding unfolding wrt $\mathcal{M}$ [20].

As for the unfolding, the interesting case to be discussed is hence the one of (existential) quantification: the other cases are simply managed by pushing the unfolding down to the subformula(e). Given an ECQ of the form $\exists x.Q$, we have:

$$\textsc{unfold}(\exists x.Q, \mathcal{M}) = \bigvee_{(f/n) \in \textsc{fs}(\mathcal{M})} \exists x_1, \ldots, x_n.\textsc{unfold}(Q[x/f(x_1, \ldots, x_n)], \mathcal{M})$$

where $\textsc{fs}(\mathcal{M})$ is the set of function symbols contained in $\mathcal{M}$, including the special function symbol *val* used for attribute values. This unfolding reflects that quantification over object terms and values in the ontology must be properly rephrased as a corresponding quantification over those values in the relational layer that could lead to produce such object terms and values through the application of $\mathcal{M}$. This is done by unfolding $\exists x.Q$ into a disjunction of formulae, where each of the formulae is obtained from Q by replacing x with one of the possible variable terms constructed from function symbols in $\mathcal{M}$, and quantifying over the existence of values that could form a corresponding object term. We observe that one of the formulae, namely the one using the *val* function symbol, tackles the case in which x appears in the range of an attribute.

When the unfolding is applied to a UCQ of the query, the atoms in the UCQ are unified with the heads of the mapping assertions in $\mathcal{M}$. For each possible unifier, each atom is replaced with an auxiliary view predicate, which corresponds to a view defined in terms of the SQL query that constitutes the body of the matching mapping assertion. The UCQ's unfolding is then obtained as the union of all queries obtained in this way.

Example 5. Consider the $\mu\mathcal{L}_{\mathrm{C}}^{\mathrm{EQL}}$ property Φ described in Example 4, together with the TBox $\mathcal{T}$ and mapping $\mathcal{M}$ in Example 1. $\Phi_r = \textsc{rew}(\Phi, \mathcal{T})$ results in $\mu Z.(\forall s.Q_r(s)) \vee [-]Z$, where

$$Q_r(s) = [\mathsf{Student}(s) \vee \mathsf{Bachelor}(s) \vee \mathsf{Master}(s) \vee \mathsf{MNum}(s, \_)] \supset [\mathsf{Graduated}(s)]$$

As for the unfolding, we first observe that $\textsc{fs}(\mathcal{M}) = \{\mathsf{stu}_1/3, \mathit{val}/1\}$. This means that $\textsc{unfold}(\forall s.Q_r(s), \mathcal{M})$ results in

$$\forall v.\textsc{unfold}(Q_r(\mathit{val}(v)), \mathcal{M}) \wedge \forall x_1, x_2, x_3.\textsc{unfold}(Q_r(\mathsf{stu}_1(x_1, x_2, x_3)), \mathcal{M})$$

The first conjunct corresponds to true, because there are no matching mapping assertions for the UCQ $\mathsf{Student}(\mathit{val}(x)) \vee \mathsf{Bachelor}(\mathit{val}(x)) \vee \mathsf{Master}(\mathit{val}(x)) \vee \mathsf{MNum}(\mathit{val}(x), \_)$, which is on the left-hand side of the implication in $Q_r(\mathit{val}(v))$. As for the second conjunct, when unfolding the UCQ $\mathsf{Student}(\mathsf{stu}_1(x_1, x_2, x_3)) \vee \mathsf{Bachelor}(\mathsf{stu}_1(x_1, x_2, x_3)) \vee \mathsf{Master}(\mathsf{stu}_1(x_1, x_2, x_3)) \vee \mathsf{MNum}(\mathsf{stu}_1(x_1, x_2, x_3), \_)$, we notice that the involved mapping assertions are m_1, m_2, and m_3, but we only consider m_3, because the query on its left-hand side contains the ones on the left-hand side of m_1 and m_2. The unfolding then results in:

$$\mu Z.\big(\forall x_1, x_2, x_3.\textsc{aux}_{m_3}(x_1, x_2, x_3, \_) \supset \textsc{aux}_{m_5}(x_1, x_2, x_3)\big) \vee [-]Z$$

where m_3 and m_5 are the mapping assertions whose right-hand side respectively matches with $\mathsf{MNum}(\mathsf{stu}_1(x_1, x_2, x_3), \_)$ and $\mathsf{Graduated}(\mathsf{stu}_1(x_1, x_2, x_3))$, and where $\textsc{aux}_{m_3}(\mathsf{name}, \mathsf{surname}, \mathsf{type}, \mathsf{id})$ and $\textsc{aux}_{m_5}(\mathsf{name}, \mathsf{surname}, \mathsf{type})$ represent the auxiliary view predicates of mapping assertions m_3 and m_5 respectively, whose defining queries are the SQL queries in the left-hand side of the mapping assertions themselves. □

We are now ready to state our main result: verification of $\mu\mathcal{L}_{\mathrm{C}}^{\mathrm{EQL}}$ properties over an STS can be faithfully reduced to verification of $\mu\mathcal{L}_C$ properties over the corresponding RTS.

Theorem 1. *Let $\mathcal{S} = \langle \mathcal{O}, \mathcal{P}, D_0 \rangle$ be an SGDAP with $\mathcal{O} = \langle \mathcal{R}, \mathcal{T}, \mathcal{M} \rangle$, and let Φ be a $\mu\mathcal{L}_C^{EQL}$ dynamic property specified over $\mathcal{T}$. Then:*

$$\Upsilon_{\mathcal{S}}^{S} \models \Phi \quad \textit{if and only if} \quad \Upsilon_{\mathcal{S}}^{R} \models \textsc{unfold}(\textsc{rew}(\Phi, \mathcal{T}), \mathcal{M})$$

Proof. From Section 4, we know that $\Upsilon_{\mathcal{S}}^{\mathrm{R}} = \langle \mathcal{R}, \Sigma, s_0, \mathit{db}, \Rightarrow \rangle$ and $\Upsilon_{\mathcal{S}}^{\mathrm{S}} = \langle \mathcal{T}, \Sigma, s_0, \mathit{abox}, \Rightarrow \rangle$, where $\mathit{abox}(\cdot)$ is defined as follows: for every $s \in \Sigma$, $\mathit{abox}(s) = \mathcal{M}(\mathit{db}(s))$. We prove the following more general result: given a state $s \in \Sigma$, we have

$$s \in (\Phi)^{\Upsilon_{\mathcal{S}}^{\mathrm{S}}} \quad \text{if and only if} \quad s \in (\textsc{unfold}(\textsc{rew}(\Phi, \mathcal{T}), \mathcal{M}))^{\Upsilon_{\mathcal{S}}^{\mathrm{R}}}$$

For simplicity, below use $\textsc{ur}(\Phi)$ as an abbreviation for $\textsc{unfold}(\textsc{rew}(\Phi, \mathcal{T}), \mathcal{M})$.

We start by observing that we can drop first-order quantification from the language replacing $\exists x \in \mathcal{C}_0.\Phi$ with $\bigvee_{c \in \textsc{adom}(abox(s_0))} \Phi[x/c]$. This allows us to consider valuations only for the predicate variables, used in fixpoint formulae. The proof is then organized in three parts: (1) We prove the theorem for formulae of $\mathcal{L}_C^{\text{EQL}}$, obtained from $\mu\mathcal{L}_C^{\text{EQL}}$ by dropping the predicate variables and the fixpoint constructs. $\mathcal{L}_C^{\text{EQL}}$ corresponds to a first-order variant of the Hennessy Milner logic, and its semantics does not depend on the second-order valuation. (2) We extend the results to the infinitary logic obtained by extending $\mathcal{L}_C^{\text{EQL}}$ with arbitrary countable disjunction. (3) We recall that fixpoints can be translated into this infinitary logic, thus proving that the theorem holds for $\mu\mathcal{L}_C^{\text{EQL}}$.

Proof for $\mathcal{L}_C^{\text{EQL}}$. We proceed by induction on the structure of Φ, without considering the case of predicate variable and of fixpoint constructs, which are not part of $\mathcal{L}_C^{\text{EQL}}$.

(Base case: $\Phi = Q$) We have to show that $\textsc{ans}(Q, \mathcal{T}, abox(s)) = \mathtt{true}$ if and only if $\textsc{ans}(\textsc{ur}(Q), db(s)) = \mathtt{true}$. By definition, $abox(s) = \mathcal{M}(db(s))$, hence the proof is obtained from the soundness and completeness of ECQ rewriting [11].

(Inductive step: $\Phi = \neg\Psi$) By induction hypothesis, for every $s \in \Sigma$ we have $s \in (\Psi)^{\Upsilon_S^S}$ if and only if $s \in (\textsc{ur}(\Psi))^{\Upsilon_S^R}$. Hence, $s \notin (\Psi)^{\Upsilon_S^S}$ if and only if $s \notin (\textsc{ur}(\Psi))^{\Upsilon_S^R}$, which in turn implies that $s \in (\neg\Psi)^{\Upsilon_S^S}$ if and only if $s \in (\neg\textsc{ur}(\Psi))^{\Upsilon_S^R}$. The proof is then obtained by observing that, by definition, $\neg\textsc{ur}(\Psi) = \textsc{ur}(\neg\Psi)$.

(Inductive step: $\Phi = \Phi_1 \vee \Phi_2$) By induction hypothesis, for every $s \in \Sigma$ we have $s \in (\Phi_1)^{\Upsilon_S^S}$ if and only if $s \in (\textsc{ur}(\Phi_1))^{\Upsilon_S^R}$, and $s \in (\Phi_2)^{\Upsilon_S^S}$ if and only if $s \in (\textsc{ur}(\Phi_2))^{\Upsilon_S^R}$. Hence, $s \in (\Phi_1)^{\Upsilon_S^S}$ or $s \in (\Phi_2)^{\Upsilon_S^S}$ if and only if $s \in (\textsc{ur}(\Phi_1))^{\Upsilon_S^R}$ or $s \in (\textsc{ur}(\Phi_2))^{\Upsilon_S^R}$, which in turn implies that $s \in (\Phi_1 \vee \Phi_2)^{\Upsilon_S^S}$ if and only if $s \in (\textsc{ur}(\Phi_1) \vee \textsc{ur}(\Phi_2))^{\Upsilon_S^R}$. The proof is then obtained by observing that, by definition, $\textsc{ur}(\Phi_1) \vee \textsc{ur}(\Phi_2) = \textsc{ur}(\Phi_1 \vee \Phi_2)$.

(Inductive step: $\Phi = \langle - \rangle\Psi$) By induction hypothesis, for every $s' \in \Sigma$ we have $s' \in (\Psi)^{\Upsilon_S^S}$ if and only if $s' \in (\textsc{ur}(\Psi))^{\Upsilon_S^R}$. Now consider that, by definition, $s \in (\langle - \rangle\Psi)^{\Upsilon_S^S}$ if and only if there exists a transition $s \Rightarrow s'$ such that $s' \in (\Psi)^{\Upsilon_S^S}$. By construction, Υ_S^S and Υ_S^R have the same transition relation. Therefore, we have that $s \in (\langle - \rangle\Psi)^{\Upsilon_S^S}$ if and only if $s \in (\langle - \rangle\textsc{ur}(\Psi))^{\Upsilon_S^R}$. The proof is then obtained by observing that, by definition, $\langle - \rangle\textsc{ur}(\Psi) = \textsc{ur}(\langle - \rangle\Psi)$.

Extension to Arbitrary Countable Disjunction. Let Ψ be a countable set of $\mathcal{L}_C^{\text{EQL}}$ formulae. Given an STS $\Upsilon^S = \langle \mathcal{T}, \Sigma, s_0, abox, \Rightarrow \rangle$, the semantics of $\bigvee \Psi$ is $(\bigvee \Psi)^{\Upsilon^S} = \bigcup_{\psi \in \Psi} (\psi)^{\Upsilon^S}$ (similarly for RTSs). Therefore, given a state $s \in \Sigma$ we have $s \in (\bigvee \Psi)^{\Upsilon^S}$ if and only if there exists $\psi \in \Psi$ such that $s \in (\psi)^{\Upsilon^S}$. Arbitrary countable conjunction is obtained for free because of negation.

Let $\Upsilon_S^R = \langle \mathcal{R}, \Sigma, s_0, db, \Rightarrow \rangle$ and $\Upsilon_S^S = \langle \mathcal{T}, \Sigma, s_0, abox, \Rightarrow \rangle$. By induction hypothesis, we can assume that for every $s \in \Sigma$ and formula $\psi \in \Psi$, we have $s \in (\psi)^{\Upsilon_S^S}$ if and only if $s \in (\textsc{ur}(\psi))^{\Upsilon_S^R}$. Given the semantics of $\bigvee \Psi$ above, this implies that $s \in (\bigvee \Psi)^{\Upsilon_S^S}$ if and only if $s \in (\bigvee \textsc{ur}(\Psi))^{\Upsilon_S^R}$, where $\textsc{ur}(\Psi) = \{\textsc{ur}(\psi) \mid \psi \in \Psi\}$. The proof is then obtained by observing that $\bigvee \textsc{ur}(\Psi) = \textsc{ur}(\bigvee \Psi)$.

Extension to Full $\mu\mathcal{L}_{\mathrm{C}}^{\mathrm{EQL}}$. In order to extend the result to the whole $\mu\mathcal{L}_{\mathrm{C}}^{\mathrm{EQL}}$, we resort to the well-known result stating that fixpoints of the μ-calculus can be translated into the infinitary Hennessy Milner logic by iterating over *approximants*, where the approximant of index α is denoted by $\mu^{\alpha} Z.\Phi$ (resp. $\nu^{\alpha} Z.\Phi$). This is a standard result that also holds for $\mu\mathcal{L}_{\mathrm{C}}^{\mathrm{EQL}}$. In particular, approximants are built as follows:

$$\begin{array}{rlrl} \mu^0 Z.\Phi &= \mathsf{false} & \nu^0 Z.\Phi &= \mathsf{true} \\ \mu^{\beta+1} Z.\Phi &= \Phi[Z/\mu^{\beta} Z.\Phi] & \nu^{\beta+1} Z.\Phi &= \Phi[Z/\nu^{\beta} Z.\Phi] \\ \mu^{\lambda} Z.\Phi &= \bigvee_{\beta<\lambda} \mu^{\beta} Z.\Phi & \nu^{\lambda} Z.\Phi &= \bigwedge_{\beta<\lambda} \nu^{\beta} Z.\Phi \end{array}$$

where λ is a limit ordinal, and where fixpoints and their approximants are connected by the following properties: given an STS or RTS Υ and a state s of Υ

- $s \in (\mu Z.\Phi)_V^{\Upsilon}$ if and only if there exists an ordinal α such that $s \in (\mu^{\alpha} Z.\Phi)_V^{\Upsilon}$ and, for every $\beta < \alpha$, it holds that $s \notin (\mu^{\beta} Z.\Phi)_V^{\Upsilon}$;
- $s \notin (\nu Z.\Phi)_V^{\Upsilon}$ if and only if there exists an ordinal α such that $s \notin (\nu^{\alpha} Z.\Phi)_V^{\Upsilon}$ and, for every $\beta < \alpha$, it holds that $s \in (\nu^{\beta} Z.\Phi)_V^{\Upsilon}$. □

7 Decidability Results

Given an SGDAP $\mathcal{S}$, in Section 6 we have shown how to reduce verification of $\mu\mathcal{L}_{\mathrm{C}}^{\mathrm{EQL}}$ properties over $\Upsilon_{\mathcal{S}}^{\mathrm{S}}$ into verification of $\mu\mathcal{L}_C$ properties over $\Upsilon_{\mathcal{S}}^{\mathrm{R}}$. However, due to the injection of new, fresh data into the system due to call to external services, $\Upsilon_{\mathcal{S}}^{\mathrm{R}}$ (as well as $\Upsilon_{\mathcal{S}}^{\mathrm{S}}$) is in general infinite-state. This causes verification to be undecidable in general, even for the very simple case of an SGDAP in which the TBox contains no assertions and directly reflects the database schema via simple one-to-one mappings, and where the temporal formula to be verified is a propositional reachability property [6].

An extensive study concerning some decidability boundaries for the verification of Data-Centric Dynamic Systems (DCDSs) with non-deterministic external services has been provided in [6]. One of the most interesting conditions for decidability that have been studied so far is *state-boundedness*. Let $\mathcal{S} = \langle \mathcal{O}, \mathcal{P}, D_0 \rangle$ be an SGDAP with $\mathcal{O} = \langle \mathcal{R}, \mathcal{T}, \mathcal{M} \rangle$ and RTS $\Upsilon_{\mathcal{S}}^{\mathrm{R}} = \langle \mathcal{R}, \Sigma, s_0, db, \Rightarrow \rangle$. We say that $\mathcal{S}$ is *state-bounded* if there exists a bound b such that for each $s \in \Sigma$, $|\textsc{adom}(db(s))| < b$. Intuitively, state-boundedness imposes that the database associated to the state of the RTS $\Upsilon_{\mathcal{S}}^{\mathrm{R}}$ remains bounded, although it may acquire arbitrarily many new values in the course of the evolution of the system (forgetting old ones, to keep the bound on the state). Leveraging on the result on state-boundedness, we can exploit our rewriting result above to get the following theorem.

Theorem 2. *Verification of $\mu\mathcal{L}_C^{EQL}$ properties over state-bounded SGDAPs is decidable, and can be reduced to conventional finite-state model checking.*

Proof (sketch). The proof is based on a reduction to DCDSs. For a formal definition of a DCDS, the interested reader can refer to [6]. Intuitively, DCDSs are tightly related to SGDAPs, with some key differences in the data component: *(i)* the process component is identical in the two frameworks; *(ii)* DCDSs are constituted by a relational layer (i.e., no ontology nor mapping are present); *(iii)* while SGDAPs define constraints over the data at the semantic layer, DCDSs are equipped with denial constraints posed directly over the database schema.

Let $\mathcal{S} = \langle \mathcal{O}, \mathcal{P}, D_0 \rangle$, with $\mathcal{O} = \langle \mathcal{R}, \mathcal{T}, \mathcal{M} \rangle$. By exploiting FOL rewritability in *DL-Lite*$_\mathcal{A}$, the consistency check used to generate $\Upsilon_\mathcal{S}^\mathrm{S}$ can be rewritten as a denial constraint over $\mathcal{R}$. This means that $\Upsilon_\mathcal{S}^\mathrm{R}$ can be generated by a purely relational DCDS. Technically, starting from $\mathcal{S}$ we can construct a corresponding DCDS with nondeterministic services $\mathcal{S}_\mathrm{REL} = \langle \mathcal{D}, \mathcal{P} \rangle$, where $\mathcal{D} = \langle \mathcal{V}, \mathcal{R}, \{\textsc{unfold}(\mathsf{q}_\mathsf{unsat}(\mathcal{T}), \mathcal{M}) \rightarrow \mathsf{false}\}, D_0 \rangle$, such that $\Upsilon_\mathcal{S}^\mathrm{R} \equiv \Upsilon_{\mathcal{S}_\mathrm{REL}}^\mathrm{DCDS}$, where $\Upsilon_{\mathcal{S}_\mathrm{REL}}^\mathrm{DCDS}$ is the RTS constructed for the DCDS $\mathcal{S}_\mathrm{REL}$ following the definition in [6]. This also means that $\mathcal{S}$ is state-bounded if and only if $\mathcal{S}_\mathrm{REL}$ is state-bounded.

Let us now consider a $\mu\mathcal{L}_\mathrm{C}^\mathrm{EQL}$ property Φ. From Theorem 1, we know that $\Upsilon_\mathcal{S}^\mathrm{S} \models \Phi$ if and only if $\Upsilon_\mathcal{S}^\mathrm{R} \models \Phi'$, where $\Phi' = \textsc{unfold}(\textsc{rew}(\Phi, \mathcal{T}), \mathcal{M})$. By recalling that $\Upsilon_\mathcal{S}^\mathrm{R} \equiv \Upsilon_{\mathcal{S}_\mathrm{REL}}^\mathrm{DCDS}$, we get that $\Upsilon_\mathcal{S}^\mathrm{S} \models \Phi$ if and only if $\Upsilon_{\mathcal{S}_\mathrm{REL}}^\mathrm{DCDS} \models \Phi'$. The proof is then obtained from the decidability of verification of $\mu\mathcal{L}_P$ properties for state-bounded DCDSs with non-deterministic services [6], by recalling that Φ' is a $\mu\mathcal{L}_C$ property, and by observing that $\mu\mathcal{L}_C$ is trivially contained in $\mu\mathcal{L}_P$ [6]. □

Example 6. Consider the SGDAP $\mathcal{S} = \langle \mathcal{O}, \mathcal{P}, D_0 \rangle$, where $\mathcal{O}$ is the OBDA system defined in Example 1, and $\mathcal{P}$ the process component defined in Example 3. It is easy to see that the resulting RTS $\Upsilon_\mathcal{S}^\mathrm{R}$ is state-bounded. Intuitively, this follows from the facts that the actions of $\mathcal{S}$ either move tuples from the TRANSF_M table to the ENROLLED one, or copy tuples from the ENROLLED table to the GRAD one. Hence, the size of each database instance appearing in $\Upsilon_\mathcal{S}^\mathrm{R}$ is at most twice the size of D_0, thus verification of $\mu\mathcal{L}_\mathrm{C}^\mathrm{EQL}$ properties over the STS $\Upsilon_\mathcal{S}^\mathrm{S}$ is decidable. □

We close this section by mentioning that the sufficient syntactic conditions for state-boundedness of DCDSs given in [6] can be easily applied to SGDAPs as well, given that the structure of the process component remains unchanged.

8 Conclusion

In this paper, we have introduced semantically-governed data-aware processes, where an ontology in *DL-Lite*$_{\mathcal{A},id}$ is used to capture the information manipulated by the process at the conceptual level, and to understand and govern the process itself. Our key result is the ability of extending FOL rewritability, typical of *DL-Lite*, to arbitrary temporal formulae expressed in μ-calculus. Indeed, in this paper we have used this result to show decidability of temporal verification in our setting for an interesting class of data-aware processes. The exact computational complexity of verification remains to be investigated.

However, the result appears to be much more general, and can be exploited to lift results obtained lately for (relational) data-aware processes [8,6,17], to the case in which an ontology-based governance component is introduced. We plan to investigate this further in the future, and study the requirements on the languages used to express the ontology and the mappings that make this lifting feasible. The framework described in this paper is being applied in the context of the EU project ACSI[3] to two significant real-world case studies.

[3] http://www.acsi-project.eu

References

1. van der Aalst, W.M.P., Barthelmess, P., Ellis, C.A., Wainer, J.: Proclets: A Framework for Lightweight Interacting Workflow Processes. Int. J. of Cooperative Information Systems 10(4), 443–481 (2001)
2. Abiteboul, S., Bourhis, P., Galland, A., Marinoiu, B.: The AXML Artifact Model. In: Proc. of TIME, pp. 11–17 (2009)
3. Baader, F., Calvanese, D., McGuinness, D., Nardi, D., Patel-Schneider, P.F. (eds.): The Description Logic Handbook: Theory, Implementation and Applications. Cambridge University Press (2003)
4. Bagheri Hariri, B., Calvanese, D., De Giacomo, G., De Masellis, R.: Verification of Conjunctive-Query Based Semantic Artifacts. In: Proc. of DL, vol. 745. CEUR (2011), ceur-ws.org (2011)
5. Bagheri Hariri, B., Calvanese, D., De Giacomo, G., De Masellis, R., Felli, P.: Foundations of Relational Artifacts Verification. In: Rinderle-Ma, S., Toumani, F., Wolf, K. (eds.) BPM 2011. LNCS, vol. 6896, pp. 379–395. Springer, Heidelberg (2011)
6. Bagheri Hariri, B., Calvanese, D., De Giacomo, G., Deutsch, A., Montali, M.: Verification of Relational Data-Centric Dynamic Systems with External Services. CoRR Technical Report arXiv:1203.0024, arXiv.org e-Print archive (2012), http://arxiv.org/abs/1203.0024
7. Belardinelli, F., Lomuscio, A., Patrizi, F.: Verification of Deployed Artifact Systems via Data Abstraction. In: Kappel, G., Maamar, Z., Motahari-Nezhad, H.R. (eds.) Service Oriented Computing. LNCS, vol. 7084, pp. 142–156. Springer, Heidelberg (2011)
8. Belardinelli, F., Lomuscio, A., Patrizi, F.: An Abstraction Technique for the Verification of Artifact-Centric Systems. In: Proc. of KR, pp. 319–328 (2012)
9. Bhattacharya, K., Gerede, C., Hull, R., Liu, R., Su, J.: Towards Formal Analysis of Artifact-Centric Business Process Models. In: Alonso, G., Dadam, P., Rosemann, M. (eds.) BPM 2007. LNCS, vol. 4714, pp. 288–304. Springer, Heidelberg (2007)
10. Calvanese, D., De Giacomo, G., Lembo, D., Lenzerini, M., Poggi, A., Rodríguez-Muro, M., Rosati, R.: Ontologies and Databases: The *DL-Lite* Approach. In: Tessaris, S., Franconi, E., Eiter, T., Gutierrez, C., Handschuh, S., Rousset, M.-C., Schmidt, R.A. (eds.) Reasoning Web. LNCS, vol. 5689, pp. 255–356. Springer, Heidelberg (2009)
11. Calvanese, D., De Giacomo, G., Lembo, D., Lenzerini, M., Rosati, R.: EQL-Lite: Effective First-Order Query Processing in Description Logics. In: Proc. of IJCAI, pp. 274–279 (2007)
12. 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. of Automated Reasoning 39(3), 385–429 (2007)
13. Calvanese, D., De Giacomo, G., Lembo, D., Lenzerini, M., Rosati, R.: Path-Based Identification Constraints in Description Logics. In: Proc. of KR, pp. 231–241 (2008)
14. Cangialosi, P., De Giacomo, G., De Masellis, R., Rosati, R.: Conjunctive Artifact-Centric Services. In: Maglio, P.P., Weske, M., Yang, J., Fantinato, M. (eds.) ICSOC 2010. LNCS, vol. 6470, pp. 318–333. Springer, Heidelberg (2010)
15. Clarke, E.M., Grumberg, O., Peled, D.A.: Model Checking. The MIT Press, Cambridge (1999)
16. Cohn, D., Hull, R.: Business Artifacts: A Data-Centric Approach to Modeling Business Operations and Processes. IEEE Bull. on Data Engineering 32(3), 3–9 (2009)
17. De Masellis, R., De Giacomo, G., Rosati, R.: Verification of Conjunctive Artifact-Centric Services. Int. J. of Cooperative Information Systems (to appear, 2012)

18. Emerson, E.A.: Automated Temporal Reasoning About Reactive Systems. In: Moller, F., Birtwistle, G. (eds.) Logics for Concurrency. LNCS, vol. 1043, pp. 41–101. Springer, Heidelberg (1996)
19. Nigam, A., Caswell, N.S.: Business Artifacts: An Approach to Operational Specification. IBM Systems Journal 42(3), 428–445 (2003)
20. Poggi, A., Lembo, D., Calvanese, D., De Giacomo, G., Lenzerini, M., Rosati, R.: Linking Data to Ontologies. In: Spaccapietra, S. (ed.) Journal on Data Semantics X. LNCS, vol. 4900, pp. 133–173. Springer, Heidelberg (2008)
21. Rodriguez-Muro, M., Calvanese, D.: High Performance Query Answering over *DL-Lite* Ontologies. In: Proc. of KR, pp. 308–318 (2012)

Query Patterns for Existential Rules

Cristina Civili and Riccardo Rosati

Dipartimento di Ingegneria Informatica, Automatica e Gestionale
Sapienza Università di Roma, Italy
{Civili,Rosati}@dis.uniroma1.it

Abstract. In this paper we study query answering over ontologies expressed in Datalog+/–, i.e., datalog with existential variables in rule heads. Differently from previous proposals, we focus on subclasses of unions of conjunctive queries (UCQs), rather than on the whole class of UCQs. To identify subclasses of UCQs, we introduce the notion of *conjunctive query pattern*. Given a class of queries $\mathcal{Q}$ expressed by a conjunctive query pattern, we study decidability and complexity of answering queries in $\mathcal{Q}$ over a Datalog+/– program. In particular, we define an algorithm that, given a Datalog+/– program P and a class of queries $\mathcal{Q}$, is able to compute a simplified Datalog+/– program P' that is equivalent to P with respect to answering queries in $\mathcal{Q}$. We show that such an algorithm constitutes both a theoretical and a practical interesting tool for studying query answering over ontologies expressed in terms of Datalog+/– rules.

1 Introduction

A lot of interesting recent work on Datalog extensions [4,1,2,14,7,12,15] are based on the idea of extending Datalog rules with existential variables in rule heads. Following [4], we call Datalog+/– this extension of the Datalog language (recent papers also use the terms *existential rules* or Datalog$^\exists$ for this formalism).

The renewed interest in this kind of rules is mainly due to its relationship with ontology languages: as shown in [5], extending Datalog with existential head variables allows for expressing several Description Logics, in particular the logics of the DL-Lite family [9], which can be seen as the logical underpinnings of the OWL 2 QL ontology specification language.

Reasoning under this kind of existential rules is in general undecidable. Therefore, much effort has been devoted to the identification of decidable fragments of the general formalism, and the computational analysis of reasoning in such fragments. Notable fragments are, for instance, linear Datalog+/–, multi-linear Datalog+/–, and guarded Datalog+/– [5], but many other fragments have been recently defined (see, e.g., [1,2,7,14,15]).

Almost all the recent approaches in this direction focus on *conjunctive query answering* under existential rules, i.e., the problem of answering a conjunctive query (or a union of conjunctive queries) over a set of existential rules and a database instance (see, e.g., [4]). We recall that many other problems involving existential rules can be reduced in a straightforward way to this problem, e.g., implication of existential rules, conjunctive query containment under existential rules, etc.

In this paper we study query answering over ontologies expressed in Datalog+/−. Differently from previous proposals, we focus on *subclasses* of unions of conjunctive queries (UCQs), rather than on the whole class of UCQs. In fact, in many applications it seems realistic to assume that only a subclass of UCQs will be used to query the program. In such cases, the current results and reasoning techniques might not be an optimal choice.

In fact, let P be a program that does not belong to any known decidable Datalog+/− fragment. This implies that answering *arbitrary* UCQs under P is not decidable (or is not known to be decidable). But, answering a *subclass* of UCQs under P might actually be decidable. In particular, this problem might be equivalent to answering such queries to a program P' simpler than P, and we might use known query answering algorithms for this task. In this paper we follow this idea, and provide a technique that is able to realize such a form of program simplification.

More precisely, the main contributions of this paper are the following:

– To identify subclasses of UCQs, we introduce the notion of *conjunctive query pattern*, which we see as a natural way of expressing restrictions on query variable bindings in the context of Datalog+/−.
– Given a class of queries $\mathcal{Q}$ expressed by a conjunctive query pattern, we study decidability and complexity of answering queries in $\mathcal{Q}$ over a Datalog+/− program. In particular, we define an algorithm that, given a Datalog+/− program P and a class of queries $\mathcal{Q}$, is able to compute a simplified Datalog+/− program P' that is equivalent to P with respect to answering queries in $\mathcal{Q}$. We show that such an algorithm constitutes both a theoretical and a practical interesting tool for studying query answering over ontologies expressed in terms of Datalog+/− rules.

The idea of conjunctive query pattern shares similarities with the magic sets technique for Datalog programs [3]. However, besides main technical differences, the purpose of conjunctive query patterns in the scenario of rules with existential variables in the head is actually different from magic sets, since we use query patterns to *identify classes of queries* for which a given program admits decidable (or tractable) query answering.

In the following section, we first briefly recall the Datalog+/− formalism. Then, in Section 3 we introduce a query-based notion of equivalence between Datalog+/− programs, and introduce the notion of conjunctive query pattern. In Section 4, we present our technique for the reduction of a Datalog+/− program with respect to a conjunctive query pattern. Finally, in Section 5 we present an example of execution of the above technique, and conclude in Section 6.

2 Preliminaries

2.1 Syntax of Datalog+/–

We start from three pairwise disjoint alphabets: (i) a relational schema $\mathcal{R}$, where, as usual, every relation symbol is associated with an arity, denoted by $\mathit{Arity}(r)$, which is a non-negative integer; (ii) a countably infinite domain of constants; (iii) a countably infinite domain of variables.

An atom is an expression of the form $r(t_1, \ldots, t_k)$ where $k = \mathit{Arity}(r)$ and where every t_i is either a constant symbol or a variable symbol. A database is a (possibly infinite) set of ground atoms. Given an atom γ, we denote by $\mathit{Rel}(\gamma)$ the relation symbol occurring in γ.

A Datalog+/– rule R is an expression of the form α :- $\beta_1, \ldots, \beta_n$., where $\alpha, \beta_1, \ldots, \beta_n$ are atoms and $i \geq 1$ (we omit the existential quantification in the head of the rule). We call the atom α the *head* of R and call the expression $\beta_1, \ldots, \beta_n$ the *body* of R.

We call *distinguished variables of* R the variables occurring both in the head and in the body of R. We call *existential body variables* of R the variables that occur only in the body of R, *ebj-variables* (existential body join variables) of R the existential body variables occurring at least twice in the body of R, and call *existential head variables* of R the variables that occur only in the head of R.

A *Datalog+/– program* is a finite set of Datalog+/– rules. Given a Datalog+/– program P, the *signature of* P is the set of relation symbols occurring in P.

A *conjunctive query (CQ)* is an existentially quantified conjunction of positive atoms (possibly with free variables): in this paper, we use a Datalog notation for CQs, i.e., a CQ q is an expression of the form $q(\mathbf{x})$:- $\alpha_1, \ldots, \alpha_n$, where $\alpha_1, \ldots, \alpha_n$ is a sequence of atoms, called the *body* of q, the variables $\mathbf{x}$ are the *distinguished variables* of q and every variable of $\mathbf{x}$ occurs at least once in the body of q; the non-distinguished variables occurring in the body of q are called *existential variables* of q; finally, we call *join variables* the variables occurring at least twice in the body of q. The number of variables of $\mathbf{x}$ is called the *arity of* q. A *union of conjunctive queries (UCQ)* is a set of CQs of the same arity.

2.2 Semantics of Datalog+/–

Given a Datalog+/– rule R of the form α :- $\beta_1, \ldots, \beta_n$. and a database B, we say that B *satisfies* R if the first-order interpretation $\mathcal{I}^B$ (i.e., the first-order interpretation isomorphic to B) satisfies the first-order sentence

$$\forall \boldsymbol{x}.\beta_1 \wedge \ldots \wedge \beta_n \rightarrow \exists \boldsymbol{y}.\alpha$$

where $\boldsymbol{x}$ denotes all the variables occurring in the body of R and $\boldsymbol{y}$ denotes the existential head variables of R.

Given a Datalog+/– program P and a database D, we say that a database B *satisfies* (P, D) if $B \supseteq D$ and B satisfies every rule in P. Moreover, we denote by $\mathit{sem}(P, D)$ the set of all databases B such that B satisfies (P, D).

Let q be a FOL query and let B be a database. We denote by $ans(q, B)$ the set of tuples of constants $\mathbf{c}$ such that $\mathcal{I}^B$ satisfies $q(\mathbf{c})$, where $q(\mathbf{c})$ is the first-order sentence obtained from q by replacing its free variables with the constants $\mathbf{c}$.

We are interested in posing unions of conjunctive queries (UCQs) over Datalog+/– programs. More precisely, let P be a program, let q be a UCQ and let D be a database. The *certain answers to q under P and D*, denoted by $cert(q, P, D)$, are the set of tuples of constants t such that $t \in \bigcap_{B \in sem(P,D)} ans(q, B)$.

2.3 Linear, Multi-linear, and Guarded Datalog+/–

Finally, we briefly recall some classes of Datalog+/– programs, i.e., linear Datalog+/–, multi-linear Datalog+/–, and guarded Datalog+/–.

A Datalog+/– program P is *linear* if, for every rule R in P, there is a single atom in the body of R.

A Datalog+/– program P is *multi-linear* if, for every rule R in P, every atom α in the body of R is such that all distinguished variables and all existential body variables of R occur as arguments in α.

A Datalog+/– program P is *guarded* if, for every rule R in P, there exists an atom α (called the *guard*) in the body of R such that all distinguished variables and all existential body variables of R occur in α.

It has been shown that, with respect to data complexity, answering UCQs under linear and multi-linear Datalog+/– programs is in AC_0, while answering UCQs under guarded Datalog+/– programs is PTIME-complete [5].

3 Query-Equivalent Programs and Conjunctive Query Patterns

In this section we first introduce a notion of equivalence between Datalog+/– programs with respect to a class of queries. Then, we define conjunctive query patterns.

3.1 Query-Equivalent Programs

We start by defining a query-based notion of program equivalence.

Definition 1. *Given a class $\mathcal{Q}$ of UCQs and two Datalog+/– programs P_1, P_2, we say that P_1* is $\mathcal{Q}$-equivalent to *P_2 if, for every UCQ $q \in \mathcal{Q}$ and for every database D, $ans(q, P_1, D) = ans(q, P_2, D)$.*

Informally, the above notion of $\mathcal{Q}$-equivalence is in terms of certain answers to the queries of the class $\mathcal{Q}$: Two programs are $\mathcal{Q}$-equivalent if the certain answers to the queries in the class $\mathcal{Q}$ over every database D are the same for the two programs.

Notice that the above notion of equivalence is somehow similar to the notion of uniform equivalence studied in Datalog [16] and answer set programming [11], however, differently from uniform equivalence, it is query-dependent.

It can be easily shown that, in general, checking $\mathcal{Q}$-equivalence between two Datalog+/− programs is undecidable. For instance, let $\mathcal{Q}$ be the whole class of conjunctive queries. Then, it is easy to see that deciding $\mathcal{Q}$-equivalence of two arbitrary Datalog programs corresponds to check uniform equivalence between Datalog programs, which is already undecidable [16,10].

Based on the concept of $\mathcal{Q}$-equivalence, we now define the notion of $\mathcal{Q}$-expansion of a set of programs $\mathcal{P}$.

Definition 2. *Given a class $\mathcal{P}$ of programs, a program P and a class $\mathcal{Q}$ of UCQs, we define the $\mathcal{Q}$-*expansion *of $\mathcal{P}$, denoted by $\mathcal{P}[\mathcal{Q}]$, as the class of programs constituted by every program P such that there exists a program $P' \in \mathcal{P}$ such that P' is $\mathcal{Q}$-equivalent to P.*

Therefore, the class $\mathcal{P}[\mathcal{Q}]$ "expands" the initial class of programs $\mathcal{P}$ by adding all programs that are $\mathcal{Q}$-equivalent to some program in $\mathcal{P}$. Thus, the less constrained the class of queries $\mathcal{Q}$ is, the larger the expansion $\mathcal{P}[\mathcal{Q}]$ is.

An informal explanation of the above definition is the following. If a program P is in the class $\mathcal{P}[\mathcal{Q}]$ but not in the class $\mathcal{P}$, then the program P can be considered as a member of the class $\mathcal{P}$, *as long as we are interested in answering queries from the class $\mathcal{Q}$.*

Of course, for every class of programs $\mathcal{P}$ and for every class of UCQs $\mathcal{Q}$, we have that $\mathcal{P}[\mathcal{Q}] \supseteq \mathcal{P}$ (since $\mathcal{Q}$-equivalence is a reflexive relation).

3.2 Conjunctive Query Patterns

We now introduce the notion of conjunctive query pattern (CQ pattern).

We start by introducing QP-atoms. A *QP-atom* is an atom whose arguments are occurrences of elements of the set $\{B, U, BU\}$. The symbols B, U, BU, where B stands for bound (term), U stands for unbound and BU stands for bound or unbound, are called *pattern arguments*. We assume that pattern arguments are disjoint from both the alphabet of constants and the alphabet of variables.

The informal meaning of the pattern arguments of QP-atoms is the following. The pattern argument B is used to represent a *bound* argument position, i.e., a position where distinguished variables and constants may occur as arguments. The pattern argument U is used to represent an *unbound* argument position, i.e., a position where only non-join existential variables may occur as arguments. The pattern argument BU is used to represent an arbitrary argument position, i.e., a position where all kinds of terms (distinguished variables, constants, and both join and non-join existential variables) may occur as arguments.

A *conjunctive query pattern (CQP)* π is a set of QP-atoms.

From now on, we will call *query atoms* the standard atoms that may occur in conjunctive queries (to distinguish such atoms from QP-atoms).

Given an atom γ occurring in the body of a conjunctive query q, we denote by $\tau_{qp}(\gamma, q)$ the function that returns the QP-atom obtained from γ by replacing: (i) every occurrence of a distinguished variable or a constant in γ with B, (ii) every occurrence of a non-join existential variable with U, (iii) every other argument (i.e., every join existential variable) with BU. Moreover, given a CQ q, we denote by $\tau_{qp}(q)$ the set of QP-atoms $\bigcup_{\gamma \in body(q)} \tau_{qp}(\gamma, q)$.

Definition 3. *Given a CQP π and a conjunctive query q, we say that q* is an instance of π *if $\tau_{qp}(q) \subseteq \pi$.*

Informally, an instance of a CQP π is any conjunctive query whose body matches with a subset of the pattern π, under the assumption that the pattern argument B can only match with distinguished variables or constants, and the pattern argument U can only match with non-join existential variables.

Given a CQP π, we denote by $\mathcal{Q}_\pi$ the set constituted by every UCQ Q such that every CQ in Q is an instance of π.

Given a QP-atom α and an atom γ of a CQ q, we say that γ is *compatible with α* if $\tau_{qp}(\gamma, q) = \alpha$.

Example 1. Let π be the CQP $\{r(B, BU, U), s(U, BU)\}$. Then:

- Let q_1 be the CQ $q_1(X)$:- $r(X, Y, Z), s(W, Y)$. Then, q_1 is an instance of π: in fact, the atom $r(X, Y, Z)$ matches with the QP-atom $r(B, BU, U)$ (i.e., $\tau_{qp}(r(X, Y, Z), q_1) = r(B, BU, U)$) because variable X is distinguished and variable Z is a non-join existential variable in q_1; moreover, the atom $s(W, Y)$ matches with the QP-atom $s(U, BU)$ (i.e., $\tau_{qp}(s(W, Y), q_1) = s(U, BU)$) because variable W is a non-join existential variable in q_1. Therefore, $\tau_{qp}(q) \subseteq \pi$.
- Let q_2 be the CQ $q_2(X)$:- $r(X, Y, Z), r(c, Y, V), s(W, X)$. Then, q_2 is also an instance of π: in particular, the atom $r(c, Y, V)$ matches with the QP-atom $r(B, BU, U)$ (i.e., $\tau_{qp}(r(c, Y, V), q_2) = r(B, BU, U)$) because c is a constant, Y is a join existential variable, and V is a non-join existential variable in q_2.
- Let q_3 be the CQ $q_3(X, Z)$:- $r(X, Y, Z), s(W, Y)$. Then, q_3 is not an instance of π, because the atom $r(X, Y, Z)$ does not match with the QP-atom $r(B, BU, U)$ (because variable Z is distinguished in q_3), i.e., $\tau_{qp}(r(X, Y, Z), q_3) = r(B, BU, B)$.

□

4 Query-Pattern Based Program Reduction

In this section we present an algorithm that, given a program P and a CQP π, produces a new Datalog+/– program that is $\mathcal{Q}_\pi$-equivalent to P.

We start by defining a notion of applicability of a rule to a QP-atom.

Definition 4. *We say that a rule R is* QP-applicable *to a QP-atom α if $Rel(\alpha) = Rel(head(R))$ and for every argument of α in which B occurs, the corresponding argument of $head(R)$ is a distinguished variable or a constant.*

The above definition can be explained as follows. Roughly speaking, some of the known query rewriting techniques for Datalog+/– (e.g., [8,6,13]) are based on the application of a resolution step between a query atom α and a rule R. This resolution step is based on a specialized unification rule between α and $\mathit{head}(R)$: such a unification succeeds only if all the existential head variables of R occurring in $\mathit{head}(R)$ match with existential variables of the query in α.

Now, a QP-atom represents an approximated representation of a query atom. Therefore, Definition 4 corresponds to an approximation of the above applicability condition between query atoms and rules, based on the structure of QP-atoms.

Definition 5. *Let R be a rule and α a QP-atom. We define the* specialization of R with respect to α *(and denote it by $\mathit{Spec}(R, \alpha)$) as the rule obtained from R by replacing every distinguished argument of $\mathit{head}(R)$ with a new head existential variable if the corresponding argument of α is U.*

Moreover, given a program P and a QP-atom α, we denote by $\mathit{RulesQP}(\alpha, P)$ the set of rules constituted by every rule R' such that there exists a rule R of P such that:

1. *R has $\mathit{Rel}(\alpha)$ as head predicate;*
2. *R is QP-applicable to α;*
3. *$R' = \mathit{Spec}(R, \alpha)$.*

Informally, the specialization of R with respect to α is a transformation of rule R which is obtained by turning every distinguished variable X of R which unifies with a pattern argument U of α into an existential variable. For instance, if $\alpha = r(B, U)$ and R is the rule $r(x, y) \text{ :- } r(x, z), r(z, y)$, then $\mathit{Spec}(R, \alpha)$ is the rule $r(x, y') \text{ :- } r(x, z), r(z, y)$.

Of course, the transformed rule $\mathit{Spec}(R, \alpha)$ is weaker than the initial rule R, in the sense that R logically entails $\mathit{Spec}(R, \alpha)$ while the converse in general does not hold. Anyway, with respect to the QP-atom α (and all the query atoms compatible with α), $\mathit{Spec}(R, \alpha)$ can actually be considered as a faithful representation of R: in fact, it can be shown that, with respect to the above mentioned resolution step, for every query atom γ compatible with α, the result of applying R to γ is the same as the result of applying $\mathit{Spec}(R, \alpha)$ to γ.

The above simplification of a rule with respect to a QP-atom is the crux of our program reduction technique. In fact, our idea is to simplify the program with respect to a set of QP-atoms that represents the set of all possible query atoms which can be generated by resolution during the rewriting of the query.

To compute the above set of QP-atoms, we can actually start from an initial CQ-pattern π and perform a rewriting of π, through a specialization of the rewriting steps applied in the above mentioned query rewriting methods for Datalog+/–, in particular [8,6].

In the following, we call *redundant* a rule R such that there exists a substitution σ of the existential head variables of R such that there exists an atom in the body of R that is equal to $\sigma(\mathit{head}(R))$. Indeed, it is immediate to verify that a redundant rule can always be removed from every program P without changing the semantics of P.

We are now able to present the algorithm *QP-Projection*.

```
Algorithm. QP-Projection
Input: program $P$, CQP $\pi$
Output: program $P'$
begin
  $\mathcal{A} := \pi$;
  repeat
    for each $\alpha \in \mathcal{A}$ do
      for each $R \in P$ do
        if $R$ is QP-applicable to $\alpha$
        then $\mathcal{A} := \mathcal{A} \cup$ QP-Atom-Rewrite $(\alpha, R)$;
    for each $\alpha_1, \alpha_2 \in \mathcal{A}$ do
      if $Rel(\alpha_1) = Rel(\alpha_2)$
      then $\mathcal{A} := \mathcal{A} \cup$ QP-Reduce $(\alpha_1, \alpha_2)$;
  until a fixpoint is reached;
  $P' := \emptyset$;
  for each $\alpha \in \mathcal{A}$ do
    $P' := P' \cup RulesQP(\alpha, P)$;
    delete all redundant rules from $P'$;
  return $P'$;
end
```

The algorithm takes as input a program P and a CQ-pattern π, and returns a new program P'. The algorithm is divided in two parts:

- In its first part, the algorithm computes the set $\mathcal{A}$ of QP-atoms. Such a set represents all the QP-atoms which are relevant for the CQ-pattern π with respect to the program P. The set $\mathcal{A}$ is computed starting from the initial CQ-pattern π and increasing such a set through the application of two expansion steps, called *QP-Atom-Rewrite* and *QP-Reduce* (and described below), until a fixpoint is reached.
- In its second part, the algorithm computes the set of rules $RulesQP(\alpha, P)$ for every QP-atom α in the set $\mathcal{A}$ previously computed. The union of all such rules constitutes the program returned by the algorithm (except for redundant rules which are filtered out).

We now present the algorithms *QP-Atom-Rewrite* and *QP-Reduce*. The algorithm *QP-Atom-Rewrite* is the following:

```
Algorithm. QP-Atom-Rewrite
Input: QP-atom $\alpha = r(t_1, ..., t_k)$, rule $R$
Output: set of QP-atoms $\beta_1, ..., \beta_n$
begin
  let $head(R)$ be of the form $r(t'_1, ..., t'_k)$;
```

```
    let $z_1, ..., z_k$ be variable symbols not occurring in $R$;
    $R' := R$;
    **for each** $i$ s.t. $1 \leq i \leq k$ **do**
      **if** $t_i = U$ and $t'_i$ is distinguished
      **then** $t''_i := z_i$
      **else** $t''_i := t'_i$;
    let $head(R')$ be of the form $r(t''_1, ..., t''_k)$;
    **if** $R'$ is redundant
    **then** return $\emptyset$
    **else** return $body(\tau_{qp}(R', \alpha))$;
**end**
```

The algorithm *QP-Reduce* is the following:

```
**Algorithm.** *QP-Reduce*
**Input:** QP-atoms $\alpha_1 = r(t_1, ..., t_k)$, $\alpha_2 = r(t'_1, ..., t'_k)$
**Output:** QP-atom $\beta$
**begin**
    **for each** $i$ s.t. $1 \leq i \leq k$ **do**
      **if** $t_i = t'_i = U$
      **then** $t''_i := U$
      **else**
        **if** $t_i = t'_i = B$
        **then** $t''_i := B$
        **else** $t''_i := BU$;
    **return** $r(t''_1, ..., t''_k)$;
**end**
```

Essentially, such algorithms are very similar to the analogous rewriting steps of the above cited query rewriting techniques for Datalog+/−: in particular, *QP-Atom-Rewrite* is a specialization to the case of QP-atoms of step (b) of the rewriting algorithm presented in [6], while *QP-Reduce* is an analogous specialization of step (a) of the same algorithm.

Through the above algorithms *QP-Atom-Rewrite* and *QP-Reduce*, the first part of the algorithm *QP-Projection* executes a rewriting of the initial CQ-pattern π by using a technique analogous to the query rewriting algorithm in [6]. However, besides the differences due to the different nature of CQ-patterns with respect to CQs, we point out that, differently from the algorithm in [6], *QP-Projection* only focuses on rewriting of *single atoms* rather than *conjunctions of atoms*. That is, every QP-atom is always considered in isolation when it is processed.

Then, it is easy to verify that *QP-Projection* always terminates. In particular, the first part of the algorithm *QP-Projection* always terminates, because the number of distinct QP-atoms that can be generated by the algorithm is finite.

We are now able to prove that the program returned by *QP-Projection* is $\mathcal{Q}_\pi$-equivalent to the initial program P (recall that $\mathcal{Q}_\pi$ is the class of UCQs constituted by CQs which are instances of the CQ-pattern π).

Theorem 1. *Let P be a Datalog+/– program, let π be a CQP, and let P' be the program returned by QP-Projection(P, π). Then, P' is $\mathcal{Q}_\pi$-equivalent to P.*

Proof (sketch). In the following, we will make use of the algorithm *TGD-rewrite* of [12], reported below:

```
Algorithm. TGD-rewrite
Input: a schema $R$, a set of TGDs $\Sigma$ and a BCQ $q$ over $R$
Output: the FO-rewriting $Q_r$ of $q$ under $\Sigma$
begin
  $Q_{new} := \{q\}$;
  $Q_r := \emptyset$;
  repeat
    $Q_r := Q_r \cup Q_{new}$;
    $Q_{temp} := Q_{new}$;
    $Q_{new} := \emptyset$;
    for all $q \in Q_{temp}$ do
      for all $\sigma \in \Sigma$ do
        - $factorization$ -
        $q' := factorize(q, \sigma)$;
        - $rewriting$ -
        for all $\underline{a} \in body(q')$ do
          - $atom\text{-}rewrite$ -
          if $\sigma$ is applicable to $\underline{a}$
            then $q'' := \gamma_{\underline{a},\sigma}(q'[\underline{a}/body(\sigma)])$;
          if $q'' \notin Q_r$
            then $Q_n := Q_n \cup \{q''\}$;
  until $Q_{new} = \emptyset$;
  return $Q_r$;
end
```

Let $q \in \mathcal{Q}_\pi$. Let Q be the UCQ returned by *TGD-rewrite* on input P and q and let Q' be the UCQ returned by the algorithm *TGD-rewrite* on input P' and q. We prove that Q and Q' are equivalent UCQs. In particular, we prove the following key property. Suppose that q is a CQ, suppose that α is an atom of $body(q)$ and suppose that the atom-rewrite step can be applied to α using a rule $R \in P$. Let q' be the CQ returned by such an atom-rewrite step. Then, there exists a rule $R_i \in P'$ such that the atom-rewrite step can be applied to α using a rule R_i and the query returned by such an atom-rewrite step is q'.

Then, it can be shown that the above property in turn implies that $Q \subseteq Q'$, which obviously implies that Q is contained in Q'.

On the other hand, it is easy to see that the program P' is logically entailed by P. Therefore, from the correctness of the algorithm *TGD-rewrite* (see Theorem 1 of [12]), it follows that Q' is contained in Q. Hence, Q and Q' are equivalent.□

As a corollary of the above theorem, we get the following important property.

Corollary 1. *Let P be a Datalog+/– program, let π be a CQP, let $\mathcal{P}$ be a class of Datalog+/– programs, and let P' be the program returned by QP-Projection(P, π). If $P' \in \mathcal{P}$, then $P \in \mathcal{P}[\mathcal{Q}_\pi]$.*

Informally, the meaning of the above corollary is that, if the program P' computed by *QP-Projection* on input P and π is in the class $\mathcal{P}$, then program P can be considered in the class of programs $\mathcal{P}$ as well, but only with respect to the class of queries $\mathcal{Q}_\pi$ (i.e., P is in the $\mathcal{Q}_\pi$-expansion of $\mathcal{P}$).

As we will show in the next section, the above corollary has very interesting consequences, since it may allow for query answering over programs that do not belong to known decidable classes of Datalog+/– programs; or, it may allow for using polynomial algorithms for query answering over Datalog+/– programs that do not belong to tractable classes of programs; and so on.

5 Example

In this section, in order to show the power of query patterns, we present a comprehensive example of an application of the program reduction algorithm to a Datalog+/– program P. In particular, we will focus on a program which does not belong to the class of guarded Datalog+/– programs (and hence does not belong to the class of multi-linear programs too). Finally, we conclude the section with a brief comment on the possible usages of the techniques presented in this paper.

5.1 Program

Let us consider a program P concerning the university domain. P is composed by the following set of rules.

$R_1 = courseMates(X, Y)$:- $courseMates(X, Z), courseMates(Z, Y)$.
$R_2 = courseMates(X, Y)$:- $student(X), attendCourse(X, W), course(W)$.
$R_3 = friends(X, Y)$:- $courseMates(X, Y), studyTogether(X, Y)$.
$R_4 = course(W)$:- $heldAt(W, J, K)$.
$R_5 = heldAt(W, J, K)$:- $department(J), hasDepartment(K, J),$
$university(K)$.
$R_6 = hasDepartment(K, J)$:- $heldAt(W, J, K)$.

We can observe that the rule R_1, which states that if two students are both course mates with a third student they are course mates too, is a typical example of transitive closure rule, thus obviously it is not guarded (and thus R_1 is not multi-linear too).

The rule R_2, which states that if a student attends a course he has at least one course mate, is guarded, since it contains a guard atom, $attendCourse(X, W)$, which contains all the universally quantified variables of the rule; however, R_2 is not multi-linear.

The rule R_3, which states that two course mates that study together are friends too, is multi-linear (and thus guarded too), since all the body atoms contain all the variables of the rule.

The rule R_4, which states that the first argument of the relation *heldAt* is a course, is linear, and thus is both multi-linear and guarded.

The rule R_5, which states that at least one course is held at each department, is guarded, since it contains the guard atom $\mathit{hasDepartment}(K, J)$; however, R_5 is not multi-linear.

Finally, the rule R_6, which states that if a course is held at a department d of a university u, then department d belongs to university u, is linear, and thus is both multi-linear and guarded.

Hence, the overall program P is neither multi-linear nor guarded.

5.2 Query 1

Let us now consider the following query over the program P:

$$q_1(X) \text{ :- } \mathit{courseMates}(X, Y).$$

The query pattern for q_1 (i.e., $\tau_{qp}(q_1)$) is:

$$\pi_1 = \{\mathit{courseMates}(B, U)\}$$

Let P and π_1 be the input elements for the *QP-Projection* algorithm, and P_{π_1} the output program.

At the first iteration of the *QP-Projection* algorithm we have that $\mathcal{A} = \pi_1$.

The rule R_1 is QP-applicable to the QP-atom $\alpha = \mathit{courseMates}(B, U)$, however the *QP-Atom-Rewrite* algorithm does not return new QP-atoms because the rule obtained by projecting R_1 on α is $\mathit{Spec}(R_1, \alpha) = R_1' = \mathit{courseMates}(X, Y') \text{ :- } \mathit{courseMates}(X, Z), \mathit{courseMates}(Z, Y)$, which is redundant. So the set $\mathcal{A}$ does not change.

Also the rule R_2 is QP-applicable to the QP-atom $\alpha = \mathit{courseMates}(B, U)$, and since the rule obtained by projecting R_2 on α, $\mathit{Spec}(R_2, \alpha) = R_2' = R_2$ is not redundant, the body atoms resulting from the *QP-Atom-Rewrite* step are added to $\mathcal{A}$, such that:

$$\mathcal{A} = \{\mathit{courseMates}(B, U), \mathit{student}(B), \mathit{attendCourse}(B, BU), \mathit{course}(BU)\}.$$

Now we have that the rule R_4 is QP-applicable to the QP-atom $\beta = \mathit{course}(BU)$, and since the rule obtained by projecting R_4 on β, $\mathit{Spec}(R_4, \beta) = R_4' = R_4$ is not redundant, the body atom resulting from the *QP-Atom-Rewrite* step is added to $\mathcal{A}$, such that:

$$\mathcal{A} = \{\mathit{courseMates}(B, U), \mathit{student}(B), \mathit{attendCourse}(B, BU), \mathit{course}(BU), \mathit{heldAt}(BU, U, U)\}.$$

Since the rule R_5 is QP-applicable to the QP-atom $\gamma = \mathit{heldAt}(BU, U, U)$, and since the rule obtained by projecting R_5 on γ, $\mathit{Spec}(R_5, \gamma) = R_5' = R_5$ is not redundant, the body atoms resulting from the *QP-Atom-Rewrite* step are added to $\mathcal{A}$, such that:

$$\begin{aligned}\mathcal{A} = \{&courseMates(B,U), student(B), attendCourse(B,BU), course(BU),\\ &heldAt(BU,U,U), department(U), hasDepartment(U,U),\\ &university(U)\}.\end{aligned}$$

Finally, since the rule R_6 is QP-applicable to the QP-atom $\delta = hasDepartment(U,U)$, and since the rule obtained by projecting R_6 on δ, $Spec(R_6,\delta) = R'_6 = R_6$ is not redundant, the body atom resulting from the *QP-Atom-Rewrite* step is added to $\mathcal{A}$, such that:

$$\begin{aligned}\mathcal{A} = \{&courseMates(B,U), student(B), attendCourse(B,BU), course(BU),\\ &heldAt(BU,U,U), department(U), hasDepartment(U,U),\\ &university(U), heldAt(U,U,U)\}.\end{aligned}$$

The *QP-Reduce* step between the QP-atoms $heldAt(BU,U,U)$ and $heldAt(U,U,U)$ produces the QP-atom $heldAt(BU,U,U)$, which already belongs to $\mathcal{A}$ and, since there are no other rules which are QP-applicable to the atoms of $\mathcal{A}$, the *QP-Projection* algorithm produce the following rewriting of P with respect to π_1: $P_{\pi_1} = \{R_2, R_4, R_5, R_6\}$.

Notice that the rules belonging to P_{π_1} are only the ones for which the *QP-Atom-Rewrite* algorithm produced new QP-atoms in $\mathcal{A}$. In particular, as we observed before, the rule generated by projecting R_1 on α is $R'_1 = courseMates(X,Y')$:- $courseMates(X,Z), courseMates(Z,Y)$. Since such a rule is redundant, R'_1 does not belong to P_{π_1}.

The rules R_2, R_4, R_5 and R_6 are all guarded, thus the resulting program P_{π_1} is obviously guarded, hence query answering under this program is PTIME-complete.

Consequently, Corollary 1 implies that answering the initial query, as well as every query in the set $\mathcal{Q}_{\pi_1}$, over P is in PTIME.

5.3 Query 2

Let us now consider the following query over the program P:

$$\begin{aligned}q_2(X) \text{ :- } &course(X), attendCourse(Y,X), attendCourse(Z,X),\\ &studyTogether(Y,Z).\end{aligned}$$

The query pattern for q_2 (i.e., $\tau_{qp}(q_2)$) is:

$$\pi_2 = \{course(B), attendCourse(BU,B), studyTogether(BU,BU)\}$$

Let P and π_2 be the input elements for the *QP-Projection* algorithm, and P_{π_2} the output program.

In this case we have that, at the first iteration of the *QP-Projection* algorithm, $\mathcal{A} = \pi_2$.

The rule R_4 is QP-applicable to the atom $\beta = \mathit{course}(B)$, and since, as before, the rule obtained by projecting R_4 on β, $\mathit{Spec}(R_4, \beta) = R_4' = R4 = \mathit{course}(W)$:- $\mathit{heldAt}(W, J, K)$. is not redundant, the body atom resulting from the *QP-Atom-Rewrite* step is added to $\mathcal{A}$, such that:

$$\mathcal{A} = \{\mathit{course}(B), \mathit{attendCourse}(BU, B), \mathit{studyTogether}(BU, BU), \\ \mathit{heldAt}(B, U, U)\}.$$

Now, there are no rules that are QP-applicable to the atoms $\mathit{attendCourse}(BU, B)$, $\mathit{studyTogether}(BU, BU)$ and $\mathit{heldAt}(B, U, U)$ (notice that R_5 is not QP-applicable to $\mathit{heldAt}(B, U, U)$ because in the head of the rule the first argument is unbound), and the *QP-Reduce* step cannot be applied to any QP-atom of $\mathcal{A}$, thus the *QP-Projection* algorithm produce the following rewriting of P with respect to π_2: $P_{\pi_2} = \{R_4\}$.

Since, as observed before, R_4 is a multi-linear rule (linear, actually), the resulting program P_{π_2} is multi-linear, thus query answering under this program is in AC_0.

Consequently, Corollary 1 implies that answering query q_2, as well as every query in the set $\mathcal{Q}_{\pi_2}$, over P is in AC_0.

We conclude this section with a remark on the practical usage of query patterns. While, for ease of exposition, the above example always starts from a single query q and derives the query pattern corresponding to q, this is not the only interesting usage of query patterns. For instance, a more general use of query patterns would consist of directly starting from a query patten π defining a whole class of UCQs, and then check, through the algorithm *QP-Projection*, the program over the whole class of queries corrsponding to π.

Another very interesting usage of query patterns would be to start from a program P and a given class of Datalog+/− programs $\mathcal{C}$ (such that $P \notin \mathcal{C}$), and try to identify the maximal query patterns π (i.e. maximal subclasses of UCQs) such that the program P' returned by *QP-Projection*(P, π) belongs to $\mathcal{C}$. Indeed, such maximal query patterns can be identified through an iterative algorithm which starts from the universal query pattern containing the QP-atom $r(BU, BU, BU)$ for every relation r occurring in P, and then tries to "specialize" such a query pattern in all possible ways (by eliminating QP-atoms or replacing existing QP-atoms with more specialized ones) until *QP-Projection*(P, π) returns a program belonging to the class $\mathcal{C}$.

6 Conclusions

In this paper we have studied query answering under existential rules. We have focused on subclasses of UCQs and have presented a program reduction technique that is able to simplify a given program with respect to a subclass of UCQs.

Our results clearly show that, in many cases, restricting to a subclass of UCQs may significantly simplify query answering: in particular, our technique in principle allows for effectively processing queries under Datalog+/– programs for which no query answering technique is currently known.

We plan to extend the present work along different directions. First, we believe that both the notion of conjunctive query pattern and the program reduction algorithm can be further refined, to produce even more significant program simplifications. Then, although the present technique does not constitute *per se* a new query answering method, it would be nice to experimentally evaluate whether (and how much) pairing the program reduction technique with known query answering algorithms for Datalog+/– is able to speed-up the query answering process. Finally, it would be very interesting to generalize this approach to other ontology languages (e.g., Description Logics).

Acknowledgments. This research has been partially supported by the ICT Collaborative Project ACSI (Artifact-Centric Service Interoperation), funded by the EU under FP7 ICT Call 5, 2009.1.2, grant agreement n. FP7-257593. The authors wish to thank the anonymous reviewers for their useful comments, and in particular for suggesting the idea of extracting query patterns from a given program.

References

1. Baget, J.-F., Leclère, M., Mugnier, M.-L., Salvat, E.: On rules with existential variables: Walking the decidability line. Artificial Intelligence 175(9-10), 1620–1654 (2011)
2. Baget, J.-F., Mugnier, M.-L., Rudolph, S., Thomazo, M.: Walking the complexity lines for generalized guarded existential rules. In: Proc. of the 22st Int. Joint Conf. on Artificial Intelligence (IJCAI 2011), pp. 712–717 (2011)
3. Bancilhon, F., Maier, D., Sagiv, Y., Ullman, J.D.: Magic sets and other strange ways to implement logic programs. In: Proc. of the 5th ACM SIGACT SIGMOD Symp. on Principles of Database Systems (PODS 1986) (1986)
4. Calì, A., Gottlob, G., Lukasiewicz, T.: A general Datalog-based framework for tractable query answering over ontologies. In: Proc. of the 28th ACM SIGACT SIGMOD SIGART Symp. on Principles of Database Systems (PODS 2009), pp. 77–86 (2009)
5. Calì, A., Gottlob, G., Lukasiewicz, T.: A general Datalog-based framework for tractable query answering over ontologies. Technical Report CL-RR-10-21. Oxford University Computing Laboratory (2010)
6. Calì, A., Gottlob, G., Pieris, A.: Query rewriting under non-guarded rules. In: Proc. of the 4th Alberto Mendelzon Int. Workshop on Foundations of Data Management (AMW 2010) (2010)
7. Calì, A., Gottlob, G., Pieris, A.: New expressive languages for ontological query answering. In: Proc. of the 25th AAAI Conf. on Artificial Intelligence, AAAI 2011 (2011)
8. Calì, A., Lembo, D., Rosati, R.: Query rewriting and answering under constraints in data integration systems. In: Proc. of the 18th Int. Joint Conf. on Artificial Intelligence (IJCAI 2003), pp. 16–21 (2003)

9. 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. of Automated Reasoning 39(3), 385–429 (2007)
10. Cosmadakis, S.S., Gaifman, H., Kanellakis, P.C., Vardi, M.Y.: Decidable optimization problems for database logic programs. In: Proc. of the 20th ACM SIGACT Symp. on Theory of Computing (STOC 1988), pp. 477–490 (1988)
11. Eiter, T., Fink, M., Woltran, S.: Semantical characterizations and complexity of equivalences in answer set programming. ACM Trans. on Computational Logic 8(3) (2007)
12. Gottlob, G., Orsi, G., Pieris, A.: Ontological queries: Rewriting and optimization. In: Proc. of the 27th IEEE Int. Conf. on Data Engineering (ICDE 2011), pp. 2–13 (2011)
13. Gottlob, G., Orsi, G., Pieris, A.: Ontological Query Answering via Rewriting. In: Eder, J., Bielikova, M., Tjoa, A.M. (eds.) ADBIS 2011. LNCS, vol. 6909, pp. 1–18. Springer, Heidelberg (2011)
14. Krötzsch, M., Rudolph, S.: Extending decidable existential rules by joining acyclicity and guardedness. In: Proc. of the 22st Int. Joint Conf. on Artificial Intelligence (IJCAI 2011), pp. 963–968 (2011)
15. Leone, N., Manna, M., Terracina, G., Veltri, P.: Efficiently computable Datalog$^{\exists}$ programs. In: Proc. of the 13th Int. Conf. on the Principles of Knowledge Representation and Reasoning, KR 2012 (to appear, 2012)
16. Sagiv, Y.: Optimizing Datalog programs. In: Proc. of the 6th ACM SIGACT SIGMOD SIGART Symp. on Principles of Database Systems (PODS 1987), pp. 349–362 (1987)

Semantic Independence in DL-Programs[⋆]

Thomas Eiter, Michael Fink, and Daria Stepanova

Institute of Information Systems
Vienna University of Technology
Favoritenstraße 9-11, A-1040 Vienna, Austria
{dasha,eiter,fink}@kr.tuwien.ac.at

Abstract. Description Logic programs (DL-programs) are a prominent approach for a loose coupling of rules and ontologies, which has become a topic of increased interest. When computing answer sets of a DL-program, special DL-atoms, which provide query interface to an ontology, are evaluated under a possibly changing input that gives a context for the evaluation. Many different such contexts may exist and evaluating a DL-atom may be costly even for one context. Thus a natural question to ask is when the evaluation is independent of the context. Such information has immediate applications in optimization of DL-programs, but is also beneficial for other reasoning tasks, like inconsistency diagnosis and program repair. We provide an answer to this question based on a semantic notion of independence and provide a complete characterization of independent DL-atoms. We then extend the characterization to independence under additional information about inclusions among rule predicates. Moreover, we develop an axiomatization which allows one to derive all tautological DL-atoms in the general case and under predicate inclusions. A complexity analysis reveals that checking whether a DL-atom is independent, can be done efficiently.

1 Introduction

DL-programs are a prominent approach for the loose coupling of rules and ontologies, in which the rules and the ontology part exchange information via a well-defined interface.[1] In general, a DL-atom specifies an update of an ontology prior to querying it; e.g. $\mathrm{DL}[C \uplus p;\, C](t)$; means that assertions $C(t)$ are made for each individual t such that p is true for t in the rules part. Several semantics of such programs have been defined, cf. [4,9,13,19,17,3], and the concept of DL-atom has been adopted and generalized by other formalisms e.g. [18,10,6].

Irrespective of a particular semantics, for the evaluation of DL-programs in practice (i.e., when computing models or answer sets) individual DL-atoms have to be evaluated under varying input in general. Thus, the possible ontology updates specified by a DL-atom define respective contexts for their evaluation, and many different such contexts may need to be considered. Moreover, even for one context evaluating the query

⋆ This work is partially supported by the Austrian Science Fund (FWF) project P20840, and by the EC FP7 project OntoRule (IST-2009-231875).

[1] For further discussion of loose and strong couplings and their strengths, cf. [15,4,8].

specified by the DL-atom in this context may be costly. Therefore, developing optimization techniques, e.g. caching techniques [7], partial evaluation and atom merging [6], is necessary for the development of effective solvers.

Caching techniques, for instance, aim at memorizing the value of a DL-atom for some inputs, and to conclude about its value on a new input. However, the very question whether its value is on all inputs the same has not been considered so far; we call DL-atoms with this property *independent*. The identification of independent DL-atoms has immediate applications in optimization, as such atoms respectively rules involving them can be removed from the DL-program.

However, information about independence has also other uses. The loose coupling by DL-programs may result in inconsistency, that is, that no answer set (i.e., suitable model) of a DL-program exists. To remedy the situation, an inconsistency-tolerating approach was developed in [16,8]. In this approach, one distills a set of DL-atoms (a "diagnosis") which has the "wrong value" in establishing an answer set, meaning that if these atoms and rules involving them are ignored, then an answer set exists. Based on such diagnoses, one can think of repairing the ontology part of the DL-program by changing the axioms such that consistency is gained. However, the value of a DL-atom can be independent of the underlying ontology (or the initial one modulo a set of changes); thus some of the diagnoses are false positives, i.e., the opposite value can never be established.

This problem can be avoided by identifying independent DL-atoms, for instance tautologic ones. However, it is not always obvious that a certain DL-atom is tautologic. Let us illustrate this by an example.

Example 1. Consider the following DL-program representing information in the fruit domain: $P = (\Phi, \Pi)$ with underlying ontology Φ and the rules part

$$\Pi = \left\{ \begin{array}{llll} (1) & so(pineapple, chile). & (2) & vi(X) \leftarrow ex(X). \\ (3) & sw(X) \leftarrow ex(X), not\ bi(X). & (4) & ex(X) \leftarrow so(X,Y). \\ (5) & \multicolumn{3}{l}{no(X) \leftarrow \mathrm{DL}[H \uplus vi, H \uminus sw, A \cap\!\!\!\!- \; ex; \neg A](X).} \end{array} \right\},$$

where predicate so stands for Southern fruit with its country of origin, vi for vitamin, ex for exotic, bi for bitter, sw for sweet, and no for non-African fruit, respectively. Moreover, H stands for the concept of healthiness and A for the concept of African fruit. Here (1) is the fact that pineapple is a Southern fruit possibly from Chile, rule (2) states that exotic fruits are rich of vitamin and rule (3) that exotic fruits are sweet, unless they are known to be bitter. Rule (4) says that Southern fruits are exotic. Finally, rule (5) contains a DL-atom in its body. Informally, it selects all objects o into no such that $\neg A(o)$, i.e., that it is a not an African fruit is provable from the ontology Φ, upon the (temporary) assertions that vitamin objects are healthy, sweet ones are unhealthy, and the restriction that only fruit known to be exotic may be African.

It is not straightforward for this DL-atom, nor for any of its instances, that it is tautologic; this however will be shown in Section 4.

If we adopt the reasonable assumption that the underlying ontology is satisfiable, another kind of independence is possible: DL-atoms which are *contradictory*, i.e., always evaluate to false.

Our contributions on identifying independent DL-atoms briefly are as follows:

- Based on a semantic notion of independence, we provide a syntactic characterization of independent DL-atoms. While tautologic DL-atoms have a rich structure, contradictory DL-atoms are simple and only possible without ontology update prior to query evaluation.
- We also consider relaxed forms of tautologies, relative to additional information on rule predicates (acting as constraints on the possible updates to the ontology). In particular, we study inclusion among rule predicates.
- We develop a complete axiomatization for deriving all tautologies by means of simple rules of inference, in the general case as well as under separable inclusion constraints, i.e., without projective input inclusions.
- We determine the complexity of the calculus. It turns out that tautology checking is feasible in polynomial time (more precisely, in NLogSpace in general, and in LogSpace, in fact it is first-order expressible, for non-negative queries), also relative to separable inclusion constraints (in this case, it is NLogSpace-complete). Thus, we establish that checking whether a given DL-atom is independent can be done efficiently.

Our results provide further insight into the nature of DL-programs. In particular, they might be useful for DL-programs that are automatically constructed (like the ones encoding a fragment of Baader and Hollunder's terminological default logic over ontologies [2]). They can be applied to simplify DL-programs, as well as in inconsistency analysis, e.g., to refine inconsistency-tolerating semantics of DL-programs [16].

2 Preliminaries

2.1 Description Logics

We assume that the reader is familiar with the basics of Description Logics (DLs) and their syntax and semantics [1]. We will consider DL knowledge bases defined over signatures $\Sigma_o = \langle \mathcal{F}, \mathcal{P}_o \rangle$ with a set $\mathcal{F}$ of individuals (constants) c and a set $\mathcal{P}_o = \mathcal{P}_c \cup \mathcal{P}_r$ of (atomic) concept names $\mathcal{P}_c$ and role names $\mathcal{P}_r$; concept expressions, role expressions are defined as usual, as well as concept inclusion axioms $C \sqsubseteq D$, role axioms (if any are available), and assertion axioms. A *DL knowledge base (or ontology)* in a DL $\mathcal{L}$ is then a (finite) set Φ of axioms in $\mathcal{L}$.

We do not commit to a particular DL $\mathcal{L}$ here, but as for DL-programs assume that

- assertions $C(a)$, $\neg C(a)$, $R(a, b)$, $\neg R(a, b)$ with $C \in \mathcal{P}_c$, $R \in \mathcal{P}_r$ and $a, b \in \mathcal{F}$ are admissible in $\mathcal{L}$ (or can be simulated), and
- $\Phi \models \phi$ denotes, under the usual model-based semantics of $\mathcal{L}$, logical entailment of a formula ϕ from Φ, i.e., each model Φ satisfies ϕ.

For instance, the DLs $\mathcal{SHIF}$, $\mathcal{SHOIN}$, and $\mathcal{SROIQ}$, which provide the logical underpinnings of OWL-Lite, OWL DL and OWL 2, respectively (see, e.g., [11,12,14]), and the lightweight DL $DL\text{-}Lite_{\mathcal{R}}$ fulfill this. [2]

In particular, we consider *DL-queries*, i.e., formulas $\phi = Q(\boldsymbol{t})$ such that Q is either

[2] If negative role assertions can not be simulated, as e.g. basic DLs of the *DL-Lite* family or in $\mathcal{EL}^{++}$, the syntax of DL-atoms can be accordingly restricted.

(a) a concept inclusion $C \sqsubseteq D$ or its negation $\neg(C \sqsubseteq D)$, with $\boldsymbol{t} = \epsilon$ (void),
(b) a concept instance $C(t_1)$ or its negation $\neg C(t_1)$, with $\boldsymbol{t} = t_1$ a term (i.e., a constant from $\mathcal{F}$ or a variable), or
(c) a role instance $R(t_1, t_2)$ or its negation $\neg R(t_1, t_2)$, with $\boldsymbol{t} = t_1, t_2$ two terms,

where $C, D \in \mathcal{P}_c$, respectively $C, D \in \mathcal{P}_c \cup \{\top, \bot\}$ in case of a), and $R \in \mathcal{P}_r$. Satisfaction of a ground, i.e., variable free, $Q(\boldsymbol{t})$ in a model $\mathcal{I}$ of Φ, is defined by (a) $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ resp. $C^{\mathcal{I}} \not\subseteq D^{\mathcal{I}}$, (b) $t_1^{\mathcal{I}} \in C^{\mathcal{I}}$ resp. $t_1^{\mathcal{I}} \notin C^{\mathcal{I}}$, and (c) $(t_1^{\mathcal{I}}, t_2^{\mathcal{I}}) \in R^{\mathcal{I}}$ resp. $(t_1^{\mathcal{I}}, t_2^{\mathcal{I}}) \notin R^{\mathcal{I}}$. A general $Q(\boldsymbol{t})$ is satisfied by $\mathcal{I}$, if each of its ground instances $Q(\boldsymbol{t'})$ (obtained by replacing variables with constants from $\mathcal{F}$) is satisfied.

Note that entailment $\Phi \models Q(\boldsymbol{t})$ is monotonic, (in particular, $\neg(C \sqsubseteq D)$ is $C \not\sqsubseteq D$).

2.2 DL-Programs

Informally, a DL-program consists of a DL ontology Φ over Σ_o and a normal logic program Π over Σ_p, which may contain DL-queries to Φ in rule bodies. The latter are evaluated subject to hypothetical updates of Φ with assertions determined from the predicate extensions under an interpretation of Π.

Syntax. A signature $\Sigma = \langle \mathcal{F}, \mathcal{P}_o, \mathcal{P}_p \rangle$ for DL-programs consists of a set $\mathcal{F}$ of constant symbols and sets $\mathcal{P}_o$, $\mathcal{P}_p$ of predicate symbols such that $\Sigma_o = \langle \mathcal{F}, \mathcal{P}_o \rangle$ is a DL-signature and $\Sigma_p = \langle \mathcal{F}, \mathcal{P}_p \rangle = \langle \mathcal{F}, \mathcal{P} \rangle$ is an LP-signature, i.e., a function-free first-order signature over a nonempty finite set $\mathcal{F}$ of constant symbols and a nonempty finite set $\mathcal{P}$ of predicate symbols of arities ≥ 0. Terms over $\mathcal{F}$ and a set $\mathcal{V}$ of variables, and ordinary atoms $p(t_1, \ldots, t_n)$ are defined as usual, where $p \in \mathcal{P}$; a *classical literal* is either an ordinary atom a or its negation $\neg a$.

A *DL-atom* $a(\mathbf{t})$ has the form

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

where 1. either $S_i \in \mathcal{P}_c$ and $p_i \in \mathcal{P}_p$ is unary, or $S_i \in \mathcal{P}_r$ and $p_i \in \mathcal{P}_p$ is binary, 2. $op_i \in \{\uplus, \cup\!\!\!\!-, \cap\!\!\!\!-\}$, and 3. $Q(\boldsymbol{t})$ is a DL-query. We call $\lambda = S_1\ op_1\, p_1, \ldots, S_m\ op_m\, p_m$, the *input signature* and $p_1, \ldots, p_m$ the *input predicates* of $a(\boldsymbol{t})$. We regard λ as unordered list—thus for any permutation π of $\{1, \ldots, n\}$, the DL-atom $\mathrm{DL}[\lambda_\pi;\ Q](\boldsymbol{t})$ where $\lambda_\pi = S_{\pi(1)}\ op_{\pi(1)}\ p_{\pi(1)}, \ldots, S_{\pi(m)}\ op_{\pi(m)}\ p_{\pi(m)}$ is a syntactic variant of (1)—and assume that its elements $S_i\ op_i\ p_i$ are pairwise different. We also write $S_i\ op_i\ p_i \in \lambda$. Intuitively, $op_i = \uplus$ (resp., $op_i = \cup\!\!\!\!-$) increases S_i (resp., $\neg S_i$) by the extension of p_i, while $op_i = \cap\!\!\!\!-$ constrains S_i to p_i.

A *DL-rule* r has the form

$$a_0 \leftarrow a_1, \ldots, a_k, not\ \ a_{k+1}, \ldots, not\ \ a_m\,, \qquad m \geq k \geq 0\,, \tag{2}$$

where a_0 is a classical literal, and every a_i is a classical literal or a DL-atom, $1 \leq i \leq m$, where a_0 may be absent (written as $\bot$). A *DL-program* $P = (\Phi, \Pi)$ consists of a DL ontology Φ and a finite set Π of DL-rules.

Example 2. Consider a DL-program $P = (\Phi, \Pi)$, such that $\Phi = \{C \sqsubseteq D\}$ and Π is given by:

$$\{p(a).;\ q(a).;\ r(b).;\ v(X) \leftarrow \text{DL}[C \uplus p, D \cap\!\!\!\!- q;\ D](X), \mathit{not}\ \text{DL}[C \cup\!\!\!\!- r;\ \neg C](X).\}$$

In the first DL-atom intuitively the concept C is extended by the predicate p and the concept $\neg D$ is restricted by predicate q. Then, all instances of D are retrieved from the resulting ontology. The second DL-atom extends $\neg C$ by the extension of r and queries all instances of $\neg C$ from the respectively extended Φ.

Semantics. In what follows, let $P = (\Phi, \Pi)$ be a DL-program over $\Sigma = \langle \mathcal{F}, \mathcal{P}_o, \mathcal{P}_p \rangle$. By $gr(\Pi)$ we denote the grounding of Π wrt. $\mathcal{F}$, i.e., the set of ground rules originating from DL-rules in Π by replacing, per DL-rule, each variable by each possible combination of constants in $\mathcal{F}$.

An *interpretation* I (over Σ_p) is a consistent set of ground literals over Σ_p; I satisfies (i) a classical ground literal l under Φ, denoted $I \models^{\Phi} l$, iff $l \in I$, and (ii) a ground DL-atom a of the form (1), denoted $I \models^{\Phi} a$, iff $\Phi \cup \tau^I(a) \models Q(\mathbf{c})$, where $\tau^I(a) = \bigcup_{i=1}^{m} A_i(I)$, the DL-update of Φ under I by a, is defined as

- $A_i(I) = \{S_i(\mathbf{e}) \mid p_i(\mathbf{e}) \in I\}$, for $op_i = \uplus$;
- $A_i(I) = \{\neg S_i(\mathbf{e}) \mid p_i(\mathbf{e}) \in I\}$, for $op_i = \cup\!\!\!\!-$;
- $A_i(I) = \{\neg S_i(\mathbf{e}) \mid p_i(\mathbf{e}) \notin I\}$, for $op_i = \cap\!\!\!\!-$.[3]

We say that I *satisfies* a ground DL-rule r of form (2), denoted $I \models^{\Phi} r$, if either $I \models a_0$, $I \models a_j$ for some $k < j \leq m$, or $I \not\models a_i$ for some $1 \leq i \leq k$. I satisfies (is a model of) $P = (\Phi, \Pi)$, denoted $I \models P$, iff $I \models^{\Phi} r$ for all $r \in gr(\Pi)$.

Finally, an interpretation I is an *answer set* of P, iff I is a minimal (wrt. $\subseteq$) model of the FLP-reduct $P^I_{FLP} = \langle \Phi, \Pi^I_{FLP} \rangle$ of P wrt. I, where Π^I_{FLP} contains all ground DL-rules r of form (2) from $gr(\Pi)$ such that $I \models a_i$ for all $1 \leq i \leq k$, and $I \not\models a_j$, for all $k < j \leq m$. This is the *FLP-semantics* of DL-programs; several other semantics have been proposed, cf. [4,13,19,17,3], but the evaluation of DL-atoms is the same.

Example 3. Reconsider P from Example 2. It has one answer set $I = \{p(a), q(a), r(b), v(a)\}$. The DL-update of Φ under I by the DL-atom $\text{DL}[C \uplus p, D \cap\!\!\!\!- q;\ D](a)$ results in $A_1(I) \cup A_2(I)$, where $A_1(I) = \{C(a)\}$ and $A_2(I) = \{\neg D(b)\}$. Due to $C \sqsubseteq D$ in Φ, it holds that $\Phi \cup A_1(I) \cup A_2(I) \models D(a)$. Thus $\text{DL}[C \uplus p, D \cap\!\!\!\!- q;\ D](a)$ evaluates to true. On the other hand, the DL-update of Φ under I by $\text{DL}[C \cup\!\!\!\!- r;\ \neg C](a)$ is $A_3(I) = \{\neg C(b)\}$, and $\Phi \cup A_3(I) \not\models \neg C(a)$. Therefore, $\text{DL}[C \cup\!\!\!\!- r;\ \neg C](a)$ evaluates to false, and the FLP-reduct of P wrt. I contains the ground rule

$$v(a) \leftarrow \text{DL}[C \uplus p, D \cap\!\!\!\!- q;\ D](a), \mathit{not}\ \text{DL}[C \cup\!\!\!\!- r;\ \neg C](a).$$

Finally, one can verify that I is the only answer set of P. Adding the "guessing" rules $v(c) \leftarrow \mathit{not}\ v(b)$ and $v(b) \leftarrow \mathit{not}\ v(c)$ to Π, however, results in two answer sets, namely $I_1 = I \cup \{v(b)\}$ and $I_2 = I \cup \{v(c)\}$.

3 Independent DL-atoms

We call a DL-atom a *independent*, if it always has the same truth value, regardless of the underlying ontology and the context in which it is evaluated, i.e, the interpretation

[3] If $\neg S_i(\mathbf{e})$ can not be expressed, the use of $\cup\!\!\!\!-$ and $\cap\!\!\!\!-$ is excluded, cf. Footnote 2.

I of the rules. This means that a amounts to one of the logical constants $\bot$ (false, i.e., is a contradiction) or $\top$ (true, i.e., is a tautology).

In formalizing this notion, we take into account that independence trivializes for unsatisfiable underlying ontologies, and thus restrict to satisfiable ones.

Definition 1 (independent DL-atom). *A ground DL-atom a is* independent*, if for all satisfiable ontologies Φ, Φ' and all interpretations I, I' it holds that $I \models^{\Phi} a$ iff $I' \models^{\Phi'} a$.*

Furthermore, we call a tautologic *(resp.,* contradictory*), if for all satisfiable ontologies Φ and all interpretations I, it holds that $I \models^{\Phi} a$ (resp., $I \not\models^{\Phi} a$).*

Example 4. A DL-atom of the form $a = \mathrm{DL}[\,;\neg(C \sqsubseteq C)]()$ is contradictory. Indeed, the query $\neg(C \sqsubseteq C)$ is unsatisfiable, hence there does not exist any satisfiable ontology Φ, s.t. $\phi \models \neg(C \sqsubseteq C)$. Hence regardless of I, always $I \not\models^{\Phi} \neg(C \sqsubseteq C)$.

On the other hand consider a DL-atom $b = \mathrm{DL}[C \cap\!\!\!\!- \, p, C \uplus p; \neg C](c)$. It is tautologic, because under any interpretation I of p, it holds that $\neg C(c) \in \tau^{I}(b)$. Hence, it is true that $I \models^{\Phi} \neg C(c)$ for any ontology Φ (and any interpretation I).

In the following, we aim at a characterization of independent DL-atoms.

3.1 Contradictory DL-atoms

We defined above contradictory DL-atoms relative to satisfiable ontologies (otherwise, trivially no contradictory DL-atoms exist).

An obvious example of a contradictory DL-atoms is $\mathrm{DL}[\,;\top \sqsubseteq \bot]()$, where $\bot$ and $\top$ are the customary empty and full concept, respectively. Indeed, the DL-query $\bot \sqsubseteq \top$ is false in every interpretation, i.e., a logical contradiction. As it turns out, contradictory DL-atoms are characterized by such contradictions, and have a simple input signature.

We call a DL-query $Q(\boldsymbol{t})$ *satisfiable*, if there exists some satisfiable ontology L such that $L \models Q(\boldsymbol{t})$, and *unsatisfiable* otherwise. Then we have the following result.

Proposition 1. *A ground DL-atom $a = \mathrm{DL}[\lambda;\, Q](\boldsymbol{t})$ is contradictory if and only if $\lambda = \epsilon$ and $Q(\boldsymbol{t})$ is unsatisfiable.*

Proof. (if) If $\lambda = \epsilon$, then for every I, $I \models^{\Phi} a$ iff $\Phi \models Q(\boldsymbol{t})$; as $Q(\boldsymbol{t})$ is unsatisfiable, we have for every satisfiable L that $L \not\models Q(\boldsymbol{t})$. Thus a is contradictory.

(Only If). Suppose a is contradictory, i.e., $I \not\models^{\Phi} a$ for every satisfiable ontology Φ and every interpretation I, i.e., $L \cup \tau^{I}(a) \not\models Q(\boldsymbol{t})$. It follows that $Q(\boldsymbol{t})$ is unsatisfiable. To show $\lambda = \epsilon$, assume towards a contradiction that $\lambda \neq \epsilon$. Then there exists some interpretation I_0 such that $\tau^{I_0}(a) \neq \emptyset$, i.e., contains some assertion B. Consider an arbitrary satisfiable ontology L. As $L \cup \tau^{I_0}(a) \not\models Q(\boldsymbol{t})$, it follows that $L \not\models \neg.B$, where $\neg$.

$\overline{B}$ is the opposite of B. However, it is not difficult to see that satisfiable ontologies L_0 exist such that $L_0 \models \neg.B$.[4] This, however, raises a contradiction. Thus $\tau = \epsilon$. $\square$

By this result, contradictory DL-atoms have a simple form. As concept and role instance queries are always satisfiable, Q must be a (possibly negated) concept inclusion query and of the form $\neg(C \sqsubseteq C)$, $\neg(C \sqsubseteq \top)$, $\neg(\bot \sqsubseteq C)$, $\neg(\bot \sqsubseteq \top)$, or $\top \sqsubseteq \bot$.

3.2 Tautologic DL-atoms

For tautologic DL-atoms, the situation is more complex. First of all, clearly a DL-atom is tautologic if it has a tautologic query (i.e., it is satisfied by the empty ontology). This is, however, only possible for concept inclusion queries; instance queries $(\neg)C(\boldsymbol{t})$, resp. $(\neg)R(\mathbf{t_1}, \mathbf{t_2})$, are clearly not tautologic.

DL-atoms with tautologic queries are of the form $\mathrm{DL}[\lambda;\ C \sqsubseteq \top]()$, $\mathrm{DL}[\lambda;\ \bot \sqsubseteq C]()$, $\mathrm{DL}[\lambda;\ C \sqsubseteq C]()$, or $\mathrm{DL}[\lambda;\ \top \not\sqsubseteq \bot]()$, where λ is an arbitrary input signature.

However, there are also tautologic DL-atoms whose query is not tautologic.

Example 5. Consider in the fruit scenario the DL-atom

$$a = \mathrm{DL}[EF \cap\!\!\!\!- \, fr, S \uplus fr, S \,\underline{\uplus}\, fr;\ \neg EF](c),$$

where EF stands for exotic fruit, S for sweet, fr for fruit.

Intuitively, we restrict here the concept $\neg EF$ and extend the concepts S and $\neg S$ by the predicate fr. Then we ask whether c is not an exotic fruit. No matter which interpretation I of the DL-program we consider and irrespective of Φ, we will always get that $\Phi \cup \tau^I(a) \models \neg EF(c)$. Indeed, if $fr(c) \in I$, then $\tau^I(a)$ is unsatisfiable; otherwise $\neg EF(c)$ is explicitly present in $\tau^I(a)$. Hence in both cases, $\tau^I(a) \models \neg EF(c)$. This means that a is tautologic.

In the rest of this section, we identify for each query type those forms of the input signature for which the DL-atom is tautologic, or prove nonexistence of such forms. We first consider concept queries, i.e., queries $(\neg)C(\boldsymbol{t})$ and $(\neg)(C \sqsubseteq D)$, and then role queries, for which similar results hold.

Concept queries

Concept instance. To start with, let us consider the query $C(\boldsymbol{t})$. No matter what input signature is considered for this type of the DL-atom, it can never be tautologic.

Proposition 2. *For no input signature λ, a ground DL-atom a of the form* $\mathrm{DL}[\lambda;\ C](\boldsymbol{t})$ *is tautologic.*

[4] If B is a negative (resp., positive) assertion, then $\neg B$ is a positive (resp. negative) assertion and we can take $L_0 = \{\neg B\}$. If $\neg B$ is not an admissible assertion, we can effect $\neg B$ by a set of possible more restrictive axioms (e.g. we can enforce a negative role assertion $\neg R(a, b)$ in basic *DL-Lite* e.g. by $L_0 = \{\exists R \sqsubseteq C, \exists R \sqsubseteq \neg C\}$ and in $\mathcal{EL}^{++}$ by $L_0 = \{\exists R \sqsubseteq \bot\}$). Note that if negative assertions were not explicitly available in the DL and the operators $\uplus, \cap\!\!\!\!-$ disallowed in DL-atoms, still the above construction may be used as e.g. in case of *DL-Lite* and $\mathcal{EL}^{++}$, and thus the same characterization of contradictions holds.

Proof. Consider a ground DL-atom $a = \mathrm{DL}[\lambda;\, C](\boldsymbol{t})$. Towards a contradiction, suppose that λ is a signature such that a is tautologic. Thus by definition, for all ontologies Φ and for all interpretations I it holds that $\Phi \cup \tau^I(a) \models C(\boldsymbol{t})$. Thus in particular, for $L = \emptyset$ it holds that $\tau^I(a) \models C(\boldsymbol{t})$. We consider two cases, according to the satisfiability of $\tau^I(a)$. (1) Suppose $\tau^I(a)$ is unsatisfiable. Then there must exist some S, such that $S(\boldsymbol{t}) \in \tau^I(a)$ and $\neg S(\boldsymbol{t}) \in \tau^I(a)$. The presence of $S(\boldsymbol{t})$ in $\tau^I(a)$ can only be ensured if some $S \uplus p$ occurs in the input signature λ of a for some p. Now consider the interpretation $I = \emptyset$. As $p(\boldsymbol{t}) \notin I$, we can not get $S(\boldsymbol{t}) \in \tau^I(a)$, which leads to contradiction.

(2) Now suppose $\tau^I(a)$ is satisfiable. Then $C(\boldsymbol{t})$ must be in $\tau^I(a)$. Similar to the previous case, this requires that $C \uplus p$ occurs in λ for some p. Again $I = \emptyset$ does not allow us to obtain $C(\boldsymbol{t}) \in \tau^I(a)$, hence $\tau^I(a) \not\models C(\boldsymbol{t})$. This contradicts our assumption. □

Concept inclusion. For DL-atoms with concept queries of the form $C \sqsubseteq D$ and $C \not\sqsubseteq D$, where $C \neq D$ and neither concept is $\top$ or $\bot$, we get the same result as for positive instance queries.

Proposition 3. *For no input signature λ, a ground DL-atom of the form* $\mathrm{DL}[\lambda;\, C \sqsubseteq D]()$ *or* $\mathrm{DL}[\lambda;\, C \not\sqsubseteq D]()$, *where $C \neq D$ are different concept names, is tautologic.*

Proof. Consider a ground DL-atom $a = \mathrm{DL}[\lambda;\, C \sqsubseteq D]()$, and suppose a is tautologic. Then for every ontology Φ and interpretation I, it holds that $\Phi \cup \tau^I(a) \models C \sqsubseteq D$. Let $\Phi = \emptyset$ and $I = \emptyset$. Observe that $\tau^I(a)$ is satisfiable, as it contains only negative assertions. Let c be a fresh constant; then $\Phi' = \Phi \cup \tau^I(a) \cup \{C(c), \neg D(c)\}$ is satisfiable, and $\Phi' \not\models C \sqsubseteq D$. By monotonicity of $\models$, it follows $\Phi \cup \tau^I(a) \not\models C \sqsubseteq D$. Thus a is not tautologic, which is a contradiction.

The proof for $a = \mathrm{DL}[\lambda;\, C \not\sqsubseteq D]()$ is similar. □

Out of the remaining concept queries, only the following (straightforwardly) give rise to tautologic DL-atoms.

Proposition 4. *A ground DL-atom of the form* $\mathrm{DL}[\lambda;\, Q]()$ *is a tautology iff $Q = C \sqsubseteq C$, $Q = C \sqsubseteq \top$, or $Q = \top \not\sqsubseteq \bot$, for any $C \in \mathcal{P}_c \cup \{\bot, \top\}$.*

Negative concept instance. Finally, we investigate the forms of tautologic DL-atoms with a query $\neg C(\boldsymbol{t})$.

Proposition 5. *A ground DL-atom a with the query $\neg C(\boldsymbol{t})$ is tautologic if and only if it has one of the following forms:*

c1. $\mathrm{DL}[\lambda,\, C \capminus p, C \uminus p;\, \neg C](\boldsymbol{t})$,

c2. $\mathrm{DL}[\lambda,\, C \capminus p, D \uplus p, D \uminus p;\, \neg C](\boldsymbol{t})$,

c3. $\mathrm{DL}[\lambda, C \capminus p_0, C^0 \uplus p_0, C^0 \capminus p'_0,$
$C^1 \uplus p_1, C^1 \capminus p'_1, \ldots, C^n \uplus p_n, C^n \capminus p'_n, C \uminus p_{n+1};\, \neg C](\boldsymbol{t})$,

c4. $\mathrm{DL}[\lambda, C \capminus p_0, C^0 \uplus p_0, C^0 \capminus p'_0,$
$C^1 \uplus p_1, C^1 \capminus p'_1, \ldots, C^n \uplus p_n, C^n \capminus p'_n, D \uplus p_{n+1} D \uminus p'_{n+1};\, \neg C](\boldsymbol{t})$,

where for every $i = 0, \ldots, n+1$, $p_i = p'_j$ for some $j < i$ or $p_i = p_0$, and $p'_{n+1} = p'_{i_j}$ for some $j \leq n$ or $p'_{n+1} = p_0$.

Informally, the lists of $(c3)$ and $(c4)$ include a "chain" $p = p_0 \subseteq p_{j_1} \subseteq p_{j_2} \subseteq p_{j_k} = p_{n+1}$ resp. $p = p_0 \subseteq p_{j'_1} \subseteq p_{j'_2} \subseteq p_{j'_{k'}} = p'_{n+1}$. The proof of this proposition is given in the extended paper [5]; likewise for subsequent results that are stated without proof.

Example 6 (cont'd). The DL-atom $a = \mathrm{DL}[EF \cap\!\!\!\!\!\;- fr, S \uplus fr, S \uplus fr; \neg EF](c)$ is an example of the tautologic form (c2). However, the DL-atom in the program of Example 1 is not of any form (c1)–(c4), and thus in general not tautologic.

Role queries. A careful analysis reveals that the result for tautologic DL-atoms with concept instance queries carries over to the case when the query $Q(\boldsymbol{t})$ is a role instance query. The same holds for negative concept and role instance queries, when the concept names C, D are replaced with names R_1, R_2 (and the predicates p, q are binary). For the latter consider $a = \mathrm{DL}[\tau; \neg R](\boldsymbol{t})$ that is tautologic. Following the analysis in Proposition 5, which is generic in the arity of the tuple $\boldsymbol{t}$, necessarily the existence of roles R_1 and R_2 instead of C resp. D, and binary instead of unary input predicates p and q can be concluded, For example, the form (c3) above for the role query $\neg R_1$ results in $\mathrm{DL}[\gamma, R_1 \cap\!\!\!\!\!\;- p, R_2 \cap\!\!\!\!\!\;- q, R_2 \uplus p, R_2 \cup\!\!\!\!\!\;- q; \neg R_1](\boldsymbol{t})$, where R_1, R_2 are roles and p, q are binary predicates. More formally, the following is obtained.

Proposition 6. *Propositions 2 and 5 hold if C and D are replaced by role names R_1 and R_2, respectively (and p and q are binary instead of unary).*

Thus, as an interesting consequence, there is no interference of concept and role names in tautologic DL-atoms.

Axiomatization. Based on the results above, we obtain a calculus for the derivation of all tautologic DL-atoms as follows. The axioms are:

a0. $\mathrm{DL}[;\, Q]()$,
a1. $\mathrm{DL}[S \cap\!\!\!\!\!\;- p, S \cup\!\!\!\!\!\;- p; \neg S](\boldsymbol{t})$,
a2. $\mathrm{DL}[S \cap\!\!\!\!\!\;- p, S' \uplus p, S' \cup\!\!\!\!\!\;- p; \neg S](\boldsymbol{t})$,

where $Q = S \sqsubseteq S$, $Q = S \sqsubseteq \top$, or $Q = \top \not\sqsubseteq \bot$, S, S' are either distinct concepts or distinct roles, and p is a unary resp. binary predicate.

The first rule of inference is reflecting the monotonicity of DL-atoms wrt. increasing input signatures:

Expansion
$$\frac{\mathrm{DL}[\lambda;\, Q](\boldsymbol{t})}{\mathrm{DL}[\lambda, \lambda';\, Q](\boldsymbol{t})}\ (e).$$

The following rules state that an update predicate p may be replaced by q, if the latter has in case of consistent update $\tau^I(a)$ a larger value than p:

Increase
$$\frac{\mathrm{DL}[\lambda, S \uplus q, S' \uplus p, S' \cap\!\!\!\!\!\;- q;\, Q](\boldsymbol{t})}{\mathrm{DL}[\lambda, S \uplus p;\, Q](\boldsymbol{t})}\ (in_1), \qquad \frac{\mathrm{DL}[\lambda, S \cup\!\!\!\!\!\;- q, S' \uplus p, S' \cap\!\!\!\!\!\;- q;\, Q](\boldsymbol{t})}{\mathrm{DL}[\lambda, S \cup\!\!\!\!\!\;- p;\, Q](\boldsymbol{t})}\ (in_1).$$

Let $\mathcal{K}_{taut}$ denote the respective calculus.

Lemma 1. *Every ground DL-atom* a *of form* $(c1)$ – $(c4)$ *is derivable from axioms* **a1** *and* **a2** *using the rules* (e)*,* $(in_{\uplus})$*, and* $(in_{\sqcup})$*.*

Since also correctness of the rules is easily establish we have:

Theorem 1. *The calculus* $\mathcal{K}_{taut}$ *is sound and complete for the theory of tautologic ground DL-atoms.*

Proof. Soundness. It is easily seen that the rules (e), $(in_{\sqcup})$, and $(in_{\uplus})$ are sound. Indeed if a' results from a by rule (e), then $\tau^I(a') \supseteq \tau^I(a)$; if a' results from a by rule $(in_{\uplus})$ resp. $(in_{\sqcup})$, then either $\tau^I(a')$ is unsatisfiable or again $\tau^I(a') \supseteq \tau^I(a)$ (and in case of satisfiability $p^I \subseteq q^I$ must hold).

Completeness. The completeness of the theory follows, as regards concept queries, from Propositions 2–5, and Lemma 1, and as regards roles from Proposition 6. □

Notice that in fact $\mathcal{K}_{taut}$ is minimal, i.e., no axiom scheme or inference rule is redundant.

4 Independence under Inclusion

In the previous section, we considered the existence of contradictory and tautologic DL-atoms in DL-programs in the general case, assuming that the rules of the DL-program are arbitrary. However, by simple analysis or by assertions, we might have information about the relationship between rule predicates that must hold in any model or answer set.

For example, suppose that a DL-program contains the rule

$$q(X) \leftarrow p(X). \tag{3}$$

It imposes an inclusion constraint on the predicates p and q, i.e., for every model I of the program, $p^I \subseteq q^I$ must hold. If p and q are input predicates for DL-atoms, then the rule (3) might affect the independence behavior of a DL-atom in the program: relative to the inclusion constraint, it might be tautologic. Similar rules might state inclusions between binary input predicates, e.g.

$$q(X,Y) \leftarrow p(X,Y), \tag{4}$$

$$q(Y,X) \leftarrow p(X,Y); \tag{5}$$

also projections, e.g.

$$r(X) \leftarrow p(X,Y), \text{ or } r(Y) \leftarrow p(X,Y), \tag{6}$$

(of p on r) might occur. An interesting question is how the presence of such predicate constraints influences the independence behavior, which we address in this section.

We call any rule

$$q(Y_1,\dots,Y_n) \leftarrow p(X_1,\dots,X_m) \tag{7}$$

where $n \leq m$ and the Y_i are pairwise distinct variables from $X_1,\dots,X_m$ an *inclusion constraint (IC)*; if $n=m$, we also write $p \subseteq q$ if $Y_i = X_i$ and $p \subseteq q^-$ if $Y_i = X_{n-i+1}$, for all $i=1,\dots,n$. Moreover, for the calculus for tautologic DL-atoms under inclusion constraints as developed in this section, we consider an extended language including p^- as a name, representing for every $p \in \mathcal{P}_o$ its inverse as defined above. By $Cl(\mathcal{C})$ we denote the closure of $\mathcal{C}$, i.e., the set of all ICs which are satisfied by every interpretation I such that $I \models \mathcal{C}$. In particular note that $p \subseteq q^- \models p^- \subseteq q$.

Let us now consider the impact of a set $\mathcal{C}$ on independence. To this end, we consider independence *relative to* $\mathcal{C}$, i.e., the interpretations I, I' in Definition 1 must satisfy $\mathcal{C}$.

Example 7 (cont'd). Reconsider P in Example 1. We can include rule (2) (also written $ex \subseteq vi$) as an inclusion constraint to the set $\mathcal{C}$, and also rule (4). Moreover, as none of the fruits is known to be bitter in our context, we additionally include $ex \subseteq sw$ in $\mathcal{C}$. The closure $Cl(\mathcal{C})$ moreover contains the ICs $vi(X) \leftarrow so(X,Y)$ and $sw \leftarrow so(X,Y)$.

4.1 Contradictory DL-atoms

In what follows, we show that the presence of inclusion constraints $\mathcal{C}$ does not change the result regarding contradictory DL-atoms as obtained for the general case.

Proposition 7. *Let $a = \mathrm{DL}[\lambda;\ Q](t)$ be a ground DL-atom. Then a is contradictory relative to a set $\mathcal{C}$ of inclusion constraints iff $\lambda = \epsilon$ and $Q(t)$ is unsatisfiable.*

Proof. (If) Identical to the if-part of the proof of the Proposition 1.

(Only If) We use the same reasoning as in Proposition 1. If $\lambda \neq \epsilon$ then we can always find an interpretation I_0 such that $\lambda^{I_0}(a) \neq \emptyset$; indeed, we can use $I_0 = \emptyset$, if $\cap\!\!\!-$ occurs in λ, and use $I_0 = \mathcal{HB}_\Pi$, i.e., the set of all ground atoms, otherwise. □

4.2 Tautologic DL-atoms

Next we investigate how the list of tautologies is modified when inclusion constraints are put on the predicates involved in them.

As we have noted above, the minimal forms of tautologic DL-atoms with concept (resp., role) queries involve only concepts and unary input predicates (resp., roles and binary input predicates).

An inclusion constraint of the form (6) in $\mathcal{C}$ (or the DL-program) will not allow us to get any further tautologic forms. E.g., consider the tautologic DL-atom $\mathrm{DL}[R \cap\!\!\!- p, R \uplus p;\ \neg R](\boldsymbol{t})$ we intuitively should get that $\mathrm{DL}[R \cap\!\!\!- r, R \uplus p;\ \neg R](\boldsymbol{t})$ is also tautologic. However, this is not a legal DL-atom, as the role R is extended by the unary predicate r.

Dependencies of the form (5) do not allow us to obtain new tautologic DL-atoms either. For example, consider a ground DL-atom $\mathrm{DL}[R \uplus p, R \cap\!\!\!- p;\ \neg R](a,b)$, which has the form of axiom **a1**. If we replace the first occurrence of p by q, the resulting DL-atom $\mathrm{DL}[R \uplus q, R \cap\!\!\!- p;\ \neg R](a,b)$ is not tautologic. However, for a constraint (4), it is tautologic; it also would be in the former case if the query argument is (a,a).

The following can be shown. For any DL-atom $a = \mathrm{DL}[\lambda;\, Q](\boldsymbol{t})$ and set $\mathcal{C}$ of ICs, let $\mathit{inp}_a(\mathcal{C})$ denote the set of all $q(\boldsymbol{Y}) \leftarrow p(\boldsymbol{X})$ in $\mathcal{C}$ such that p and q occur in λ. We call $\mathcal{C}$ *separable* for a, if every $ic \in \mathit{inp}_a(\mathit{Cl}(\mathcal{C}))$ involves predicates of the same arity.

Proposition 8. *Let $a = \mathrm{DL}[\lambda;\, Q](\boldsymbol{t})$ be a ground DL-atom and $\mathcal{C}$ a separable set of ICs for a. Then a is tautologic relative to $\mathcal{C}$ iff it is tautologic relative to $\mathcal{C}'$ which contains, depending on the type of $Q(\boldsymbol{t})$, the following constraints: (1) $\mathcal{C}' = \emptyset$, in case of a (negated) concept inclusion; (2) every $p \subseteq q$ in $\mathit{inp}_a(\mathit{Cl}(\mathcal{C}))$ where p, q are unary, in case of a (negated) concept instance; (3) every $p \subseteq q$ and $p \subseteq q^-$ in $\mathit{inp}_a(\mathit{Cl}(\mathcal{C}))$ where p, q are binary, in case of a (negated) role instance.*

Proof (Sketch). Every model I of $\mathcal{C}$ is a model of $\mathcal{C}'$. On the other hand, by the form of the ICs, every model I' of $\mathcal{C}'$ can be extended to an interpretation I such that $I \models \mathcal{C}$. In general, the intersection I of all models $I'' \supseteq I'$, which is given by the answer set of $\mathcal{C} \cup I'$, fulfills the claim. Indeed, a fact $a = q(\boldsymbol{c})$ can be in I iff it is provable from I' using a sequence $r_1, r_2, \ldots, r_k$ of rules from $\mathcal{C}$. As all rules are unary, a can be proved from some fact $a' = p(\boldsymbol{c'})$ in $\mathcal{C}$; unfolding the rules, we obtain a rule r of the form $q(\boldsymbol{Y}) \leftarrow p(\boldsymbol{X})$, where $\boldsymbol{Y} = Y_1, \ldots, Y_m$ are distinct variables from $\boldsymbol{X} = X_1, \ldots, X_n$. As $\mathcal{C} \models r$ and $\mathcal{C}$ is separable for a, it follows that $m = n$ and thus $r \in \mathcal{C}'$, which implies $a' \in I'$. Consequently, I is an extension of I' as claimed. □

That is, for negative role queries we must in general take inverse predicate inclusions into account. To this end, we consider a language including for every $p \in \mathcal{P}_p$ a name p^- for its inverse (as defined in the paper). Such an inverse can be also effected by means of inclusions $q(Y, X) \leftarrow p(X, Y)$ and $p(Y, X) \leftarrow q(X, Y)$ in the set $\mathcal{C}$ of inclusion constraints (where q is then p^- and p is q^-).

Each rule $q(Y_1, Y_2) \leftarrow p(X_1, X_2)$ in $\mathcal{C}$ is then either an inclusion $p \subseteq q$ or an inclusion $p \subseteq q^-$. Note that $p \subseteq q$ iff $p^- \subseteq q^-$ and that for unary predicates, $p^- = p$ and is thus immaterial; furthermore, viewing $\cdot^-$ as an operator, $(p^-)^- = p$. We let $\mathcal{P}_p^{(-)} = \mathcal{P}_p \cup \{p^- \mid p \in \mathcal{P}_p\}$. To see some examples, consider the tautologic form (c1) in Proposition 5. Taking the inclusion constraint $p \subseteq q$ into account, we obtain the following new tautologic form:

- $\mathrm{DL}[\lambda, S \cap\!\!\!\!\cap\, p, S' \uplus q;\, \neg S](\boldsymbol{t})$.

The form (c2) yields

- $\mathrm{DL}[\lambda, S \cap\!\!\!\!\cap\, p, S' \uplus q, S' \,\dot{\uplus}\, p;\, \neg S](\boldsymbol{t})$,
- $\mathrm{DL}[\lambda, S \cap\!\!\!\!\cap\, p, S' \uplus p, S' \,\dot{\uplus}\, q;\, \neg S](\boldsymbol{t})$, and
- $\mathrm{DL}[\lambda, S \cap\!\!\!\!\cap\, p, S' \uplus q, S' \,\dot{\uplus}\, q;\, \neg S](\boldsymbol{t})$,

From the tautological DL-atom for $(c3)$, we get for $n = 0$ and $p_0 = p$, $C^0 = S'$, and $p'_0 = p_1 = r$:

- $\mathrm{DL}[\lambda, S \cap\!\!\!\!\cap\, p, S' \,\dot{\uplus}\, q, S' \cap\!\!\!\!\cap\, r, S \uplus r;\, \neg S](\boldsymbol{t})$.

etc. For the cases when the DL-query has any of the forms $S(\mathbf{c})$, $C \sqsubseteq D$ or $C \not\sqsubseteq D$, where S is either a concept or a role and C, D are concepts, there are no new tautologies.

4.3 Axiomatization for Tautologies

The results presented above allow us to define rules of inference for deriving tautologies when inclusion constraints are put on the input predicates of a DL-atom.

Inclusion

$$\frac{\mathrm{DL}[\lambda, S \uplus p;\, Q](\boldsymbol{t}) \quad p \subseteq q}{\mathrm{DL}[\lambda, S \uplus q;\, Q](\boldsymbol{t})}\ (i_1), \tag{8}$$

$$\frac{\mathrm{DL}[\lambda, S \,\cup\!\!\!\!\!-\, p;\, Q](\boldsymbol{t}) \quad p \subseteq q}{\mathrm{DL}[\lambda, S \,\cup\!\!\!\!\!-\, q;\, Q](\boldsymbol{t})}\ (i_2). \tag{9}$$

The "increase" rules are slightly adapted, in comparison to the general case by taking into account that $p \subseteq q$ iff $p^- \subseteq q^-$:

Increase

$$\frac{\mathrm{DL}[\lambda, S \,\cup\!\!\!\!\!-\, p;\, Q](\boldsymbol{t})}{\mathrm{DL}[\lambda, S \,\cup\!\!\!\!\!-\, q, S' \,\cup\!\!\!\!\!-\, p, S' \Cap q;\, Q](\boldsymbol{t})}\ (in_{\,\cup\!\!\!\!\!-}), \quad \frac{\mathrm{DL}[\lambda, S \uplus p;\, Q](\boldsymbol{t})}{\mathrm{DL}[\lambda, S \uplus q, S' \,\cup\!\!\!\!\!-\, p, S' \Cap q;\, Q](\boldsymbol{t})}\ (in_{\uplus}),$$

$$\frac{\mathrm{DL}[\lambda, S \,\cup\!\!\!\!\!-\, p;\, Q](\boldsymbol{t})}{\mathrm{DL}[\lambda, S \,\cup\!\!\!\!\!-\, q, S' \,\cup\!\!\!\!\!-\, p^-, S' \Cap q^-;\, Q](\boldsymbol{t})}\ (in^-_{\,\cup\!\!\!\!\!-}), \quad \frac{\mathrm{DL}[\lambda, S \uplus p;\, Q](\boldsymbol{t})}{\mathrm{DL}[\lambda, S \uplus q, S' \,\cup\!\!\!\!\!-\, p^-, S' \Cap q^-;\, Q](\boldsymbol{t})}\ (in^-_{\uplus}),$$

where $p, q \in \mathcal{P}_p^{(-)}$ are of the same arity.

We consider the following extended set of axioms compared to the case without inclusion constraints:

a0. $\mathrm{DL}[;\, Q]()$,
a1. $\mathrm{DL}[S \Cap p, S \uplus p;\, \neg S](\boldsymbol{t})$,
a2. $\mathrm{DL}[S \Cap p, S' \,\cup\!\!\!\!\!-\, q, S' \uplus q;\, \neg S](\boldsymbol{t})$, where $q \in \{p, p^-\}$,

and $Q = S \sqsubseteq S$, $Q = S \sqsubseteq \top$, or $Q = \top \not\sqsubseteq \bot$, S, S' are either distinct concepts or distinct roles; moreover, p is a unary or binary predicate.

The described axioms and rules together with the expansion rule defined above, form a calculus for the derivation of tautologic DL-atoms, which we denote by $\mathcal{K}^{\subseteq}_{taut}$. The main result of this section, following next, is its soundness and completeness.

Theorem 2. *The calculus $\mathcal{K}^{\subseteq}_{taut}$ is sound and complete for tautologic ground DL-atoms a relative to any closed set of inclusion constraints $\mathcal{C}$ (i.e., such that $\mathcal{C} = Cl(\mathcal{C})$) that is separable for a.*

We use our running example to illustrate the application of $\mathcal{K}^{\subseteq}_{taut}$.

Example 8 (cont'd). Reconsider the DL-program in Example 1, and recall that no ground instance of its DL-atom, in particular

$$a = \mathrm{DL}[H \,\cup\!\!\!\!\!-\, vi, H \uplus sw, A \Cap ex;\, \neg A](pineapple)$$

is tautologic. Now let us take the predicate constraints in P into account. Recall that essentially by the rules (2) and (3), we have that $\{ex \subseteq vi, ex \subseteq sw\} \subseteq Cl(\mathcal{C})$ (which is also separable for a). We thus can derive a in $\mathcal{K}^{\subseteq}_{taut}$ given $\mathcal{C}$ as follows:

$$\dfrac{\dfrac{\dfrac{\mathrm{DL}[H \uplus ex, H \uminus ex, A \capminus ex;\, \neg A](\mathit{pineapple})}{\mathrm{DL}[H \uplus ex, H \uminus ex, A \capminus ex;\, \neg A](\mathit{pineapple})} \quad ex \subseteq vi}{\mathrm{DL}[H \uplus vi, H \uminus ex, A \capminus ex;\, \neg A](\mathit{pineapple})}\,(i_2) \qquad ex \subseteq sw}{\mathrm{DL}[H \uplus vi, H \uminus sw, A \capminus ex;\, \neg A](\mathit{pineapple})}\,(i_1)$$

The leaf of the proof tree is a DL-atom $\mathrm{DL}[H \uplus ex, H \uminus ex, A \capminus ex;\, \neg A](\mathit{pineapple})$. It has the form of axiom **a2**. Hence the initial DL-atom a is, by virtue of Theorem 2, tautologic relative to $\mathcal{C}$.

The results of this section can be readily used for optimization or reasoning tasks on DL-programs that involve ground DL-atoms, e.g. in diagnosis and repair [16,8]. They can moreover be exploited for dealing with non-ground DL-atoms. We may call a such a DL-atom $a = \mathrm{DL}[\lambda;\, Q](\boldsymbol{t})$ independent (resp. contradictory, tautologic), if each of its ground instances has this property. From the results above, we obtain that there are no contradictory nonground DL-atoms, and that to prove a tautologic, it is sufficient to consider a single instance a (particular constants do not matter, and for role queries $(\neg)R(t_1, t_2)$, consider different constants if possible).

Example 9. In our running example, e.g., the instance of a for $X = \mathit{pineapple}$ is tautologic relative to the constraints; hence a is tautologic and can be removed from rule (5).

5 Complexity

Let us now consider the complexity of determining whether a DL-atom a is independent. To determine whether a is contradictory is trivial, given the simple forms of unsatisfiable DL-queries. For determining whether a is tautologic, we can use the calculus $\mathcal{K}^{\subseteq}_{taut}$ established above, and aim at a derivation of a. In the search, we need an oracle for deciding whether $ic \in Cl(\mathcal{C})$, for a given IC ic and $\mathcal{C}$, to see whether a rule is applicable.

The complexity of this oracle is in fact the dominating factor for the search. Indeed, the inclusion rules of $\mathcal{K}^{\subseteq}_{taut}$ work strictly local, in the sense that they only replace one occurrence of an input predicate by another one, and few independent rule applications are needed to arrive at an axiom (see below).

The complexity of deciding, given an IC ic and a set $\mathcal{C}$ of ICs, whether $ic \in Cl(\mathcal{C})$, depends on the form of the ICs. In general, the problem is decidable in polynomial space, and it is NLogSpace-complete if the arities of the predicates in $\mathcal{C}$ are bounded by a constant k. In particular, for $k = 2$ deciding $ic \in Cl(\mathcal{C})$ if all predicates in ic have the same arity, is possible using the following inference rules:

$$\frac{X \subseteq Y \quad Y \subseteq Z}{X \subseteq Z} \qquad \frac{X \subseteq Y}{X^- \subseteq Y^-} \qquad \frac{X^- \subseteq Y^-}{X \subseteq Y} \tag{10}$$

where X, Y, Z are meta variables which denote unary (binary) predicates. On the other hand, the problem is NLogSpace-hard for every $k \geq 1$ as it subsumes graph reachability.

We have the following result.

Theorem 3. *Given a DL-atom a and a separable set $\mathcal{C}$ of ICs for a, deciding whether a is tautologic relative to $\mathcal{C}$ is (i)* NLogSpace*-complete and* NLogSpace*-hard even if $\mathcal{C} = \emptyset$, and is (ii) in* LogSpace*, and in fact expressible by a fixed first-order formula (hence in* AC^0*), if the DL query Q of a is not a negative concept resp. role query.*

Proof (Sketch). By the above results on $\mathcal{K}^{\subseteq}_{taut}$, we need an oracle for $ic \in Cl(\mathcal{C})$, where ic involves only unary resp. binary predicates. Due to the special form of ICs, $ic \in Cl(\mathcal{C})$ iff $ic \in Cl([\mathcal{C}]_2)$, where $[\mathcal{C}]_2$ is the set of all ICs in $\mathcal{C}$ that involve only unary and/or binary predicates. Thus, by the observation above, an NLogSpace oracle is sufficient.

To prove that $a = \mathrm{DL}[\lambda;\, Q](\boldsymbol{t})$ is tautologic, we can guess an instance of an axiom $\mathbf{ai}$ from which we want to arrive at a by application of rules in $\mathcal{K}^{\subseteq}_{taut}$. Checking that $Q(\boldsymbol{t})$ matches the query of $\mathbf{ai}$ is easy, and we can check in case of $\mathbf{a1}$, $\mathbf{a2}$ that $S \cap\!\!\!\!- p$ occurs in λ; we then can check whether $S \uplus p$ resp. $S' \uplus p^{(-)}, S' \cup\!\!\!\!- p^{(-)}$ occur in λ, and if not, in case of $\mathbf{a1}$ build nondeterministically a "chain" $q_0 (= p) \subseteq q_1 \subseteq \cdots \subseteq q_k$ such that $S' \cup\!\!\!\!- q_k \in \lambda$ and in case of $\mathbf{a2}$ also a "chain" $r_0 (= p) \subseteq r_1 \subseteq \cdots \subseteq r_{k'}$ such that $S' \uplus q_{k'} \in \lambda$, where for every q_i, we have that either $q_{i-1} \subseteq q_i$ (which can be checked with the oracle), or some pair $S'' \uplus q_{i-1}^{(-)}, S'' \cap\!\!\!\!- q_i^{(-)}$ occurs in λ and similarly, for every r_j we have that either $r_{j-1} \subseteq r_j$ (an oracle check), or some pair $S'' \uplus r_{j-1}^{(-)}, S'' \cap\!\!\!\!- r_j^{(-)}$ occurs in λ; building a chain stops as soon as $S' \cup\!\!\!\!- q_i \in \lambda$ resp. $S' \uplus r_i \in \lambda$ is found (it may else stop after a certain number of steps, but this is irrelevant here).

A simple analysis reveals that this overall algorithm is feasible, relative to the oracle, in logarithmic space (one can cycle through the few guesses with constantly many variables, and building chains as above is feasible in nondeterministic logarithmic space, as we just need to memorize q_i, p_0, p_{n+1} resp. p'_{n+1}, and S'). It follows that in general, the problem is in NLogSpace.

The problem is shown to be NLogSpace-hard via a reduction from the canonical graph reachability problem. Let $G = (V, E)$ be a directed graph and let $s, t \in V$ be nodes. We view each node $v \in V$ as a unary predicate, and define the DL-atom $a = \mathrm{DL}[C \cap\!\!\!\!- s, \lambda, C \cup\!\!\!\!- t;\, \neg C](a)$ where λ contains for each edge $(v, w) \in E$ the elements $C^{(v,w)} \uplus v, C^{(v,w)} \cap\!\!\!\!- w$, where $C^{(v,w)}$ does not occur elsewhere. Then it holds that a is tautologic (wrt. $\mathcal{C} = \emptyset$) iff t is reachable from s in G. Indeed, note that by its form, a must be derived from an instance $\mathrm{DL}[C \cap\!\!\!\!- s, C \cup\!\!\!\!- t;\, \neg C](a)$ of $\mathbf{a1}$, and that for this a chain $q_0 = s \subseteq q_1 \subseteq \cdots \subseteq q_k = t$ must be built to obtain $C \cup\!\!\!\!- t$, and only the rule $(in_{\cup\!\!\!\!-})$ is applicable. This chain corresponds to a path in G from s to t. Conversely from any path $s = v_0, v_1, \ldots v_k = t$ in G, we can build a corresponding chain with elements $C^{(v,w)} \uplus v, C^{(v,w)} \cap\!\!\!\!- w$ in λ using the rule $(in_{\cup\!\!\!\!-})$.

Finally, if Q is not a negative concept resp. role query, then for a to be tautologic it must be an instance of $\mathbf{a0}$, which is checkable in logarithmic space and also expressible by a FOL formula ϕ over a relational structure (roughly, a plain SQL query over a database) that stores in suitable relations: all triples $S_i\ op_i\ p_i$ in λ, using S_i, op_i, and p_i as constants; the query $Q(\boldsymbol{t})$; and all inclusions $p \subseteq q^{(-)}$ from $Cl(\mathcal{C})$. In fact, ϕ can be fixed, and the relations are easily assembled from a and $\mathcal{C}$. As evaluating a fixed FOL formula over relational structures is in AC^0, we obtain the result. □

6 Conclusion and Future Work

To the best of our knowledge, the notion of independent DL-atom has not been considered before, which is of use in optimization and for reasoning tasks on DL-programs. We investigated the forms of tautologic and contradictory ground DL-atoms in the general case, as well as in the case when inclusion constraints on the input predicates are known. We showed that contradictory DL-atoms have a simple form, and we presented a sound and complete calculus for determining tautologic DL-atoms. Based on it, we determined the complexity of deciding this problem, and showed that the problem is very efficiently solvable in general, as well as relative to the predicate constraints. Furthermore, the results for ground DL-atoms can be easily lifted to deal with nonground DL-atoms, and an implementation of the calculus using logic programming is rather straightforward.

Outlook. Several issues remain for further investigation. A possible extension is to consider DL queries which allow for non-atomic concepts, respectively roles. Some of our results can be readily extended to such queries (e.g., to conjunctive concept/role queries), but to get a clear picture further work is needed.

As an alternative, or in addition to ICs, further information about the DL-program might be available relative to which independence of a DL-atom can be established.

Regarding predicate constraints, one issue is non-separable sets of inclusion constraints., i.e., to permit projections among input predicates of DL-atoms, for which the presented calculus is sound but not complete. One can also imagine more general inclusion constraints, by relaxing the conditions to allow e.g. repetition of arguments, or inclusion of intersections. Other possibilities are to consider exclusion constraints, or (non-)emptiness constraints on predicates. Adopting a technical view, we could consider arbitrary sets of constraints that describe an envelope of the set of answer sets of the underlying DL-program. The study of different forms of constraints remains to be done.

Orthogonal to rules, one may exploit information about the ontology Φ. So far, no information about the concepts (roles) in Φ was assumed to be available, viewing Φ as blackbox under full information hiding. However, information about Φ may lead to further independent DL-atoms. For example, knowing that $\Phi \models C \sqsubseteq D$ and that $\mathrm{DL}[\lambda, C \uplus p;\, Q](\boldsymbol{t})$ is tautologic, we can infer that $\mathrm{DL}[\lambda, D \uplus p;\, Q](\boldsymbol{t})$ is also tautologic. Incorporating such and further information into the calculus remains for future work.

References

1. Baader, F., Calvanese, D., McGuinness, D., Nardi, D., Patel-Schneider, P.F. (eds.): The Description Logic Handbook: Theory, Implementation and Applications. Cambridge University Press (2003)
2. Dao-Tran, M., Eiter, T., Krennwallner, T.: Realizing Default Logic over Description Logic Knowledge Bases. In: Sossai, C., Chemello, G. (eds.) ECSQARU 2009. LNCS, vol. 5590, pp. 602–613. Springer, Heidelberg (2009)
3. 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 (2011)

4. Eiter, T., Ianni, G., Lukasiewicz, T., Schindlauer, R., Tompits, H.: Combining answer set programming with description logics for the Semantic Web. AI 172, 1495–1539 (2008)
5. Eiter, T., Fink, M., Stepanova, D.: Semantic independence in DL-programs. Tech. Rep. INFSYS RR-1843-12-07, Institut für Informationssysteme, Technische Universität Wien, A-1040 Vienna, Austria (2012)
6. Eiter, T., Ianni, G., Krennwallner, T., Schindlauer, R.: Exploiting conjunctive queries in description logic programs. Ann. Math. Artif. Intell. 53(1-4), 115–152 (2008)
7. Eiter, T., Ianni, G., Schindlauer, R., Tompits, H.: Nonmonotonic Description Logic Programs: Implementation and Experiments. In: Baader, F., Voronkov, A. (eds.) LPAR 2004. LNCS (LNAI), vol. 3452, pp. 511–527. Springer, Heidelberg (2005)
8. Fink, M., El Ghali, A., Chniti, A., Korf, R., Schwichtenberg, A., Lévy, F., Pührer, J., Eiter, T.: D2.6 Consistency maintenance. Tech. Rep. 2.6, ONTORULE ICT-2009-231875 Project (2011), http://ontorule-project.eu/outcomes?func=fileinfo&id=92
9. Fink, M., Pearce, D.: A Logical Semantics for Description Logic Programs. In: Janhunen, T., Niemelä, I. (eds.) JELIA 2010. LNCS, vol. 6341, pp. 156–168. Springer, Heidelberg (2010)
10. Heymans, S., Korf, R., Erdmann, M., Pührer, J., Eiter, T.: Loosely coupling F-logic rules and ontologies. In: Int'l Conf. on Web Intelligence (WI 2010), pp. 248–255. IEEE CS (2010)
11. Horrocks, I., Patel-Schneider, P.F.: Reducing OWL entailment to description logic satisfiability. Journal of Web Semantics 1(4), 345–357 (2004)
12. Horrocks, I., Patel-Schneider, P.F., van Harmelen, F.: From $\mathcal{SHIQ}$ and RDF to OWL: The making of a Web ontology language. Journal of Web Semantics 1(1), 7–26 (2003)
13. 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)
14. Motik, B., Patel-Schneider, P.F., Parsia, B. (eds.): OWL 2 Web Ontology Language: Structural Specification and Functional-Style Syntax (2008), w3C Working Draft (April 2009)
15. Motik, B., Rosati, R.: Reconciling Description Logics and Rules. JACM 57(5), 1–62 (2010)
16. Pührer, J., Heymans, S., Eiter, T.: Dealing with Inconsistency When Combining Ontologies and Rules Using DL-Programs. In: Aroyo, L., Antoniou, G., Hyvönen, E., ten Teije, A., Stuckenschmidt, H., Cabral, L., Tudorache, T. (eds.) ESWC 2010. LNCS, vol. 6088, pp. 183–197. Springer, Heidelberg (2010)
17. Shen, Y.D.: Well-supported semantics for description logic programs. In: Proc. IJCAI 2011, pp. 1081–1086. AAAI Press (2011)
18. Wang, K., Billington, D., Blee, J., Antoniou, G.: Combining Description Logic and Defeasible Logic for the Semantic Web. In: Antoniou, G., Boley, H. (eds.) RuleML 2004. LNCS, vol. 3323, pp. 170–181. Springer, Heidelberg (2004)
19. Wang, Y., You, J.H., Yuan, L.Y., Shen, Y.D.: Loop formulas for description logic programs. Theory and Practice of Logic Progamming 10(4-6), 531–545 (2010)

An Update on Query Answering with Restricted Forms of Negation

Víctor Gutiérrez-Basulto[1], Yazmín Angélica Ibáñez-García[2], and Roman Kontchakov[3]

[1] Fachbereich Mathematik und Informatik, Universität Bremen, Germany
victor@informatik.uni-bremen.de
[2] KRDB Research Centre, Free University of Bozen-Bolzano, Italy
ibanezgarcia@inf.unibz.it
[3] Department of CS and Information Systems, Birkbeck College, London, UK
roman@dcs.bbk.ac.uk

Abstract. One of the most prominent applications of description logic ontologies is their use for accessing data. In this setting, ontologies provide an abstract conceptual layer of the data schema, and queries over the ontology are then used to access the data. In this paper we focus on extensions of conjunctive queries (CQs) and unions of conjunctive queries (UCQs) with restricted forms of negations such as inequality and safe negation. In particular, we consider ontologies based on members of the *DL-Lite* family. We show that by extending UCQs with any form of negated atoms, the problem of query answering becomes undecidable even when considering ontologies expressed in the core fragment of *DL-Lite*. On the other hand, we show that answering CQs with inequalities is decidable for ontologies expressed in $\textit{DL-Lite}^{\mathcal{H}}_{\textit{core}}$. To this end, we provide an algorithm matching the known coNP lower bound on data complexity. Furthermore, we identify a setting in which conjunctive query answering with inequalities is tractable. We regain tractability by means of syntactic restrictions on the queries, but keeping the expressiveness of the ontology.

1 Introduction

In recent years, the use of ontologies for accessing data has been recognised as one of the most prominent applications of description logics (DLs) in the Semantic Web (SW) and relational databases. The characteristic feature of *ontology-based data access* (OBDA) is the use of ontologies to enrich instance data with background knowledge, thus providing users with an interface for querying potentially incomplete data. The importance of OBDA as a key technology for the SW has been acknowledged by the introduction of Web Ontology Languages (OWL) and its profiles based on tractable DLs. In the OBDA paradigm the study of query answering has mainly been focused on answering (unions of) conjunctive queries (CQs). In particular, a fairly clear landscape of the computational complexity of CQ answering has emerged, and specific algorithmic approaches have already been developed. Recently, some investigations on query answering using query languages beyond CQs have been initiated [19,6]. In particular, a desirable way to extend CQs, which belong to the positive existential fragment of first-order logic, is with some form of *negation*. Following the large body of literature on relational databases, we consider two ways of adding restricted forms of negation to CQs:

inequalities as atomic formulas ($\mathrm{CQ}^{\neq}$) and *safe negation* ($\mathrm{CQ}^{\neg s}$). In the OBDA setting, besides the class of queries, the DL for representing the ontology needs to be specified. Of special interest are those DLs allowing OBDA to scale to large amounts of data by answering queries in relational database management systems (RDBMSs). This is the case for the members of the *DL-Lite* family of DLs: $\textit{DL-Lite}_{core}$ and $\textit{DL-Lite}^{\mathcal{H}}_{core}$ [4]. Remarkably, CQ answering in these logics is AC^0 in *data complexity*, which is an important measure of complexity when large amounts of data are considered.

The aim of this paper is to continue the study initiated by Rosati [19] on answering (U)$\mathrm{CQs}^{\neq}$ and (U)$\mathrm{CQs}^{\neg s}$ in DLs of the *DL-Lite* family. In particular, we provide undecidability and complexity results for answering (U)$\mathrm{CQs}^{\neq}$, along with algorithmic approaches. Moreover, inspired by recent works we introduce syntactical restrictions to obtain tractable $\mathrm{CQ}^{\neq}$ answering.

Related Work

In recent years, extensions of CQs with some form of negation have been studied in different areas of computer science related to management of incomplete information. The main research done in this respect focuses on establishing decidability boundaries, complexity results and algorithms for query answering. We outline relevant results in some of these areas below.

CQs with Inequalities and Negation in Description Logics. Calvanese *et al.* [9] showed that in contrast to CQs, answering $\mathrm{CQs}^{\neq}$ in highly expressive DL $\mathcal{DLR}$ is undecidable. Later on, Rosati [18,19] presented a deeper study on query answering with restricted forms of negation in several DLs by considering not only inequalities but also safe negation. Rosati shows undecidability of answering CQs with any form of negation in the DL $\mathcal{AL}$. Furthermore, Rosati shows that also answering UCQs with any form of negation in fairly inexpressive DLs $\mathcal{EL}$ and $\textit{DL-Lite}^{\mathcal{H}}_{core}$ (called $\textit{DL-Lite}_{\mathcal{R}}$ in the paper) is undecidable. For the case of answering $\mathrm{CQs}^{\neq}$ and $\mathrm{CQs}^{\neg s}$ in $\textit{DL-Lite}^{\mathcal{H}}_{core}$ Rosati provides a coNP-hardness result in data complexity, leaving the exact complexity of the problem (and even decidability) open.

CQs with Inequalities in Data Exchange (DE). In their seminal work, Fagin *et al.* [12] showed that in the DE setting answering $\mathrm{UCQs}^{\neq}$ with target constraints given by weakly acyclic TGDs is coNP-complete. To provide an upper complexity bound they presented a procedure based on a variant of the *disjunctive chase* introduced by Deutsch *et al.* [11]. A remarkable contribution of this work is a PTime algorithm for computing certain answers of UCQs with at *most* one inequality per disjunct. The lower bound for an arbitrary number of inequalities follows from a result previously established by Abiteboul *et al.* [1].

$\mathrm{CQs}^{\neq}$ with Bounded Number of Inequalities. It is known that the complexity of answering $\mathrm{UCQs}^{\neq}$ can be affected by the number of inequalities allowed per query [15]. In settings dealing with incomplete information, Abiteboul *et al.* [1,2] showed in their work on answering queries via views that answering $\mathrm{UCQs}^{\neq}$ is coNP-complete. In particular, their coNP-hardness proof (also utilized as a lower bound in the DE setting) requires six inequalities. However, later on Madry [17] closed the gap in the DE setting by showing that even the case of two inequalities is intractable.

CQs$^{\neq}$ with Other Syntactic Restrictions. An orthogonal restriction to that on the number of inequalities has recently been proposed and investigated in the DE setting by Arenas *et al.* [3] on extensions of Datalog with negated atoms. Their approach is to define syntactic restrictions over the variables that can occur in inequalities. In particular, under such conditions one can have more than one inequality per disjunct without losing tractability.

Our paper is organized as follows. In Section 2, we provide Description Logic definitions. Section 3 is dedicated to the presentation of lower complexity bounds. Section 4 investigates the establishment of matching upper bounds. Section 5 studies syntactic restrictions over conjunctive queries with inequalities. Finally, in Section 6 we conclude with an outlook of the contribution and future research lines.

2 Preliminaries

In this section we recall some basics on description logics (DLs) and extensions of conjunctive queries (CQs) with negated atoms.

The Description Logic *DL-Lite*$^{\mathcal{H}}_{core}$: Syntax and Semantics

The language of *DL-Lite*$^{\mathcal{H}}_{core}$ [4] contains *individual names* $a_0, a_1, \ldots$, *concept names* $A_0, A_1, \ldots$, and *role names* $P_0, P_1, \ldots$. We define *complex roles* R and *basic concepts* B using the following grammar:

$$\begin{array}{lcl} R & ::= & P_i \mid P_i^-, \\ B & ::= & \bot \mid A_i \mid \exists R. \end{array}$$

A *DL-Lite*$^{\mathcal{H}}_{core}$ *TBox* $\mathcal{T}$ is a finite set of *concept* and *role inclusion axioms* of the form:

$$B_1 \sqsubseteq B_2, \qquad B_1 \sqsubseteq \neg B_2, \qquad R_1 \sqsubseteq R_2, \qquad R_1 \sqsubseteq \neg R_2.$$

Whenever we find it convenient we might use $B_1 \sqcap B_2 \sqsubseteq \bot$ instead of the equivalent $B_1 \sqsubseteq \neg B_2$. An *ABox* $\mathcal{A}$ is a finite set of *assertions* of the form:

$$A_k(a_i), \qquad P_k(a_i, a_j).$$

A *DL-Lite*$^{\mathcal{H}}_{core}$ *knowledge base* (*KB*) $\mathcal{K}$ is a pair $(\mathcal{T}, \mathcal{A})$ with $\mathcal{T}$ a TBox and $\mathcal{A}$ an ABox. In the following, we denote by $ind(\mathcal{A})$ the set of individual names occurring in $\mathcal{A}$, and by $role^{\pm}(\mathcal{K})$ the set of roles that consists of P_k and P_k^-, for each role name P_k in $\mathcal{K}$. *DL-Lite*$_{core}$ is the fragment of *DL-Lite*$^{\mathcal{H}}_{core}$ without role inclusion axioms in the TBox.

An *interpretation* $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ consists of a nonempty *domain* $\Delta^{\mathcal{I}}$ and an interpretation function $\cdot^{\mathcal{I}}$ that assigns an element $a_i^{\mathcal{I}} \in \Delta^{\mathcal{I}}$ to each object name a_i, a subset $A_k^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$ to each concept name A_k, and a binary relation $P_k^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$ to each role name P_k. As usual for *DL-Lite*, we adopt the *unique name assumption* (UNA): $a_i^{\mathcal{I}} \neq a_j^{\mathcal{I}}$, for all distinct individuals a_i, a_j.

The interpretation function $\cdot^{\mathcal{I}}$ is then extended to basic concepts and complex roles:

$$\begin{aligned}
(P_k^-)^{\mathcal{I}} &= \{(y,x) \in \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}} \mid (x,y) \in P_k^{\mathcal{I}}\}, && \text{(inverse role)}\\
\bot^{\mathcal{I}} &= \emptyset, && \text{(empty set)}\\
(\exists R)^{\mathcal{I}} &= \big\{x \in \Delta^{\mathcal{I}} \mid \text{there is } y \in \Delta^{\mathcal{I}} \text{ with } (x,y) \in R^{\mathcal{I}}\big\}. && \text{(domain/range constraints)}
\end{aligned}$$

We define the *satisfaction relation* $\models$ in a standard way:

$$\begin{aligned}
\mathcal{I} \models B_1 \sqsubseteq B_2 \quad &\text{iff} \quad B_1^{\mathcal{I}} \subseteq B_2^{\mathcal{I}}, & \mathcal{I} \models R_1 \sqsubseteq R_2 \quad &\text{iff} \quad R_1^{\mathcal{I}} \subseteq R_2^{\mathcal{I}},\\
\mathcal{I} \models B_1 \sqsubseteq \neg B_2 \quad &\text{iff} \quad B_1^{\mathcal{I}} \cap B_2^{\mathcal{I}} = \emptyset, & \mathcal{I} \models R_1 \sqsubseteq \neg R_2 \quad &\text{iff} \quad R_1^{\mathcal{I}} \cap R_2^{\mathcal{I}} = \emptyset,\\
\mathcal{I} \models A_k(a_i) \quad &\text{iff} \quad a_i^{\mathcal{I}} \in A_k^{\mathcal{I}}, & \mathcal{I} \models P_k(a_i, a_j) \quad &\text{iff} \quad (a_i^{\mathcal{I}}, a_j^{\mathcal{I}}) \in P_k^{\mathcal{I}}.
\end{aligned}$$

A KB $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ is *satisfiable* if there is an interpretation $\mathcal{I}$ satisfying all members of $\mathcal{T}$ and $\mathcal{A}$. In this case we write $\mathcal{I} \models \mathcal{K}$ (as well as $\mathcal{I} \models \mathcal{T}$ and $\mathcal{I} \models \mathcal{A}$) and say that $\mathcal{I}$ is a *model of* $\mathcal{K}$ (and of $\mathcal{T}$ and $\mathcal{A}$).

Conjunctive Queries with Restricted Forms of Negation

A *conjunctive query* (CQ) is an expression of the form

$$q(\boldsymbol{x}) \;=\; \exists \boldsymbol{y}\, \varphi(\boldsymbol{x}, \boldsymbol{y}), \tag{1}$$

where $\boldsymbol{x}$ and $\boldsymbol{y}$ denote sequences of variables from a set of variables, and φ is conjunction of concept atoms $A(t)$ and role atoms $P(t, t')$ with t, t' *terms*, i.e., individual names or variables from $\boldsymbol{x}, \boldsymbol{y}$. We call variables in $\boldsymbol{x}$ *answer* variables and those in $\boldsymbol{y}$ (existentially) *quantified* variables. We denote by $\mathsf{var}(q)$ the set of variables, by $\mathsf{avar}(q)$ the set of answer variables $\boldsymbol{x}$, by $\mathsf{qvar}(q)$ the set of quantified variables $\boldsymbol{y}$ and by $\mathsf{term}(q)$ the set of terms in q. A *conjunctive query with inequalities* (CQ$^{\neq}$) is an expression of the form (1) with each conjunct of $\varphi(\boldsymbol{x}, \boldsymbol{y})$ being either a concept or role atom, or an expression of the form $t \neq t'$, where t and t' are terms. A *conjunctive query with safe negation* (CQ$^{\neg s}$) is an expression of the form (1) where $\varphi(\boldsymbol{x}, \boldsymbol{y})$ is formed by literals, i.e., atoms or negated atoms, and such that each variable of each literal occurs in at least one positive atom. A *union of conjunctive queries* (UCQ) is a disjunction of conjunctive queries. UCQ$^{\neq}$ and UCQ$^{\neg s}$ are defined accordingly.

Query Answering over *DL-Lite* KBs. Let $\mathcal{I}$ be an interpretation and $q(\boldsymbol{x})$ a query with $\boldsymbol{x} = x_1, \dots, x_k$. A map $\pi\colon \mathsf{term}(q) \to \Delta^{\mathcal{I}}$ with $\pi(a) = a^{\mathcal{I}}$, for a an individual name in $\mathsf{term}(q)$, is called a *match* for q in $\mathcal{I}$ if $\mathcal{I}$ satisfies q under the variable assignment that maps each answer variable x_i to $\pi(x_i)$. For a k-tuple of individual names $\boldsymbol{a} = a_1, \dots, a_k$, a match π for q in $\mathcal{I}$ is called an $\boldsymbol{a}$*-match* if $\pi(x_i) = a_i^{\mathcal{I}}$. We say that $\boldsymbol{a}$ is an *answer* to q in an interpretation $\mathcal{I}$ if there is an $\boldsymbol{a}$-match for q in $\mathcal{I}$. We denote by $\mathsf{ans}(q, \mathcal{I})$ the set of all answers to q in $\mathcal{I}$. We say that $\boldsymbol{a} \subseteq \mathit{ind}(\mathcal{A})$ is a *certain answer* to q over a KB $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ if $\mathcal{I} \models q[\boldsymbol{a}]$ for all models $\mathcal{I}$ of $\mathcal{K}$. The set of all *certain answers* to q over $\mathcal{K}$ is denoted by $\mathsf{cert}(q, \mathcal{K})$. We consider the following query answering problem:

INPUT:	A query q, a *DL-Lite*$^{\mathcal{H}}_{core}$ KB $\mathcal{K}$ and a tuple of individuals $\boldsymbol{a}$.
QUESTION:	Is $\boldsymbol{a}$ in $\mathsf{cert}(q, \mathcal{K})$?

3 Lower Complexity Bounds

First, we analyse the case of unions of conjunctive queries and show that query answering with inequalities is undecidable even in the simplest of *DL-Lite* languages. In fact, the following proof will demonstrate that even though the ontology language is quite inexpressive, undecidable problems can still be encoded by means of (mostly) UCQs. In a nutshell, the proof uses the existential quantifiers of the TBox concept inclusion axioms to create an unbounded supply of elements, whereas the UCQ$^{\neq}$ allows one to express universal constraints in the following sense: the query has a positive answer iff there is no model of the KB satisfying the negated UCQ$^{\neq}$, which is a conjunction of universal sentences. This result is claimed in Theorem 8 [19], but no proof is given.

Theorem 1. *Answering UCQs$^{\neq}$ is undecidable over DL-Lite$_{core}$ KBs.*

Proof. The proof is by reduction of (the complement of) the $\mathbb{N} \times \mathbb{N}$-tiling problem, which is known to be undecidable [14]. The $\mathbb{N} \times \mathbb{N}$ tiling problem is formulated as follows: given a set $\mathfrak{T}$ of square tile types with the four sides of each tile type t in $\mathfrak{T}$ coloured by $\mathit{top}(t), \mathit{right}(t), \mathit{bottom}(t), \mathit{left}(t)$, respectively, and a tile type $t_0 \in \mathfrak{T}$, decide whether $\mathbb{N}\times\mathbb{N}$ can be tiled by $\mathfrak{T}$ with t_0 placed at the origin, i.e., whether there is a function $\tau\colon \mathbb{N}\times\mathbb{N} \to \mathfrak{T}$ such that $\tau(0,0) = t_0$ and $\mathit{top}(\tau(i,j)) = \mathit{bottom}(\tau(i,j+1))$ and $\mathit{left}(\tau(i,j)) = \mathit{right}(\tau(i+1,j))$, for all $(i,j) \in \mathbb{N}\times\mathbb{N}$.

Given an instance of the $\mathbb{N}\times\mathbb{N}$-tiling problem, we construct a *DL-Lite$_{core}$* KB $(\mathcal{T}, \mathcal{A})$ that encodes the tiling problem by placing tiles over objects in its model. The top and right neighbours of a tile are referred to by roles H and V, respectively (from the horizontal and vertical successor). To represent the type of a tile we take ABox individuals t_i, for $t_i \in \mathfrak{T}$, and a role T that connects a tile to its type. So, the TBox $\mathcal{T}$ contains the following concept inclusions:

$$\exists T \sqsubseteq \exists H, \qquad \exists H^- \sqsubseteq \exists T, \qquad \exists T \sqsubseteq \exists V, \qquad \exists V^- \sqsubseteq \exists T.$$

We also require two roles, N_H and H_V, that define impossible horizontal and vertical tile neighbours: let $\mathcal{A}_{\mathfrak{T}}$ contain

$$\begin{aligned} &N_H(t_i,t_j), &&\text{for each } t_i,t_j \in \mathfrak{T} \text{ with } \mathit{right}(t_i) \neq \mathit{left}(t_j),\\ &N_V(t_i,t_j), &&\text{for each } t_i,t_j \in \mathfrak{T} \text{ with } \mathit{top}(t_i) \neq \mathit{bottom}(t_j). \end{aligned}$$

Consider now the UCQ$^{\neq}$ q (without answer variables) which consists of the negations of the following sentences:

$$\begin{aligned} &\forall x,y\,\big(T(x,y) \to \textstyle\bigvee_i (y = t_i)\big),\\ &\forall x,y,z,v,u\,\big(H(x,y)\wedge V(y,v)\wedge V(x,z)\wedge H(z,u) \to (u = v)\big),\\ &\forall x,y,x',y'\,\big(H(x,y)\wedge T(x,x')\wedge T(y,y')\wedge N_H(x',y') \to \bot\big),\\ &\forall x,y,x',y'\,\big(V(x,y)\wedge T(x,x')\wedge T(y,y')\wedge N_V(x',y') \to \bot\big). \end{aligned}$$

It can be shown that q has a negative answer over $(\mathcal{T}, \mathcal{A}_{\mathfrak{T}} \cup \{T(a,t_0)\})$ iff $\mathfrak{T}$ tiles $\mathbb{N}\times\mathbb{N}$ with t_0 placed at the origin. Indeed, if q has a negative answer then the above formulas guarantee that each tile object is related to one of the t_i, that the H- and V-successors from a proper $\mathbb{N}\times\mathbb{N}$-grid and, finally, that the adjacent colours match.

We remark in passing that answering UCQs with safe negation is undecidable even over extremely simple ontology languages [19], including *DL-Lite*$_{core}$. Although the proof of Theorem 15 [19] does not directly apply to *DL-Lite*$_{core}$, it can easily be adapted for our language:

Theorem 2 ([19]). *Answering UCQs$^{\neg s}$ is undecidable over DL-Lite$_{core}$ KBs.*

We show that even in the case of conjunctive queries adding inequalities makes the query answering problem in *DL-Lite*$_{core}$ harder. In particular, we show that answering CQs$^{\neq}$ is coNP-hard in data complexity in contrast to answering CQs, which is in AC^0. Our result strengthens Theorem 15 [19], which claims coNP-hardness for *DL-Lite*$^{\mathcal{H}}_{core}$ and refers to Abiteboul and Duschka [2] for the proof. That proof, however, is for a different first-order setting and, if translated to the language of DL, would require role inclusions and a sort of a counting quantifier in the CQ, which of course, can be expressed using inequality. We mention in passing that this proof would also imply coNP-hardness (in data complexity) of the satisfiability problem for the extension of *DL-Lite*$_{core}$ with arbitrary number restriction (cf. Theorem 8.4 [4]). Although the proof we present below is inspired by Theorem 3.4 [2], it does not use role inclusions and requires a more sophisticated query instead.

Theorem 3. *Answering CQs$^{\neq}$ over DL-Lite$_{core}$ KBs is* coNP*-hard in data complexity.*

Proof. The proof is by reduction of the complement of 3CNF. Suppose we are given a 3CNF φ with n clauses and m variables. We construct a KB $(\mathcal{T}, \mathcal{A}_\varphi)$ and a query q such that both $\mathcal{T}$ and q are fixed (i.e., do not depend on φ) and φ is satisfiable iff q has a negative answer over $(\mathcal{T}, \mathcal{A}_\varphi)$. We present the construction in two steps. To aid our explanations, we consider a model of $(\mathcal{T}, \mathcal{A}_\varphi)$ in which q is false.

First, we take a concept name V to stand for the set of variables of φ and three individuals v_j^0, v_j^1, x_j, for each of the m variables x_j of φ: one may think that v_j^0 represents the literal $\neg x_j$ and v_j^1 represents the literal x_j. The ABox $\mathcal{A}_\varphi$ contains, for each x_j, the following assertions:

$$V(x_j), \quad P_0(x_j, v_j^0), \quad P_1(x_j, v_j^1), \quad P_2(x_j, x_j) \qquad \text{and} \qquad R_1(x_j, t),$$

where t is a fresh individual (t stands for *true*) and P_0, P_1, P_2 and R_1 are role names. So, every x_j has a P_i-successor, for each $0 \leq i \leq 2$, and an R_1-successor; moreover, by the UNA, the R_1-successor is distinct from the P_i-successors. Then, the TBox $\mathcal{T}$ contains the following two concept inclusions

$$V \sqsubseteq \exists R_0 \quad \text{and} \quad V \sqcap \exists R_0^- \sqsubseteq \bot,$$

where R_0 is a fresh role name, to ensure that every x_j also has an R_0-successor and that R_0-successor is not x_j itself; see Fig. 1 (a).

Consider now a CQ$^{\neq}$ q (without answer variables) which is equivalent to the following sentence:

$$\forall x, y_1, y_2, y_3, z_0, z_1 \Big(\bigwedge_{i=0}^{2} P_i(x, y_i) \ \wedge \bigwedge_{k=0,1} R_k(x, z_k) \ \rightarrow \bigvee_{k=0,1} \bigvee_{i=0}^{2} (y_i = z_k) \Big).$$

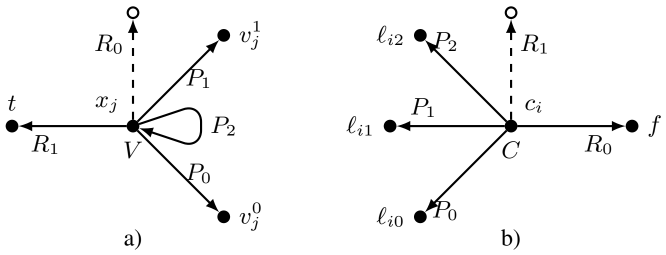


Fig. 1. Constellations of points in the proof of Theorem 3

If q has a negative answer then, for any point with P_i- and R_k-successors, either its R_0- or its R_1-successor coincides with one of the P_i-successors. In particular, when applied to individuals x_j, this means that the R_0-successor must coincide either with v_j^0 or v_j^1—in the latter case the literal $\neg x_j$ (represented by v_j^0) is chosen false by R_0 (and so, we say the variable x_j takes value *true*) and in the former case the literal x_j (represented by v_j^1) is chosen false by R_0 (and we say x_j takes value value *false*).

Second, we encode clauses in a similar way: we take a concept name C to stand for the set of clauses of φ and an individual c_i, for each of the n clauses of φ. Then the ABox $\mathcal{A}_\varphi$ contains the following assertions, for each clause $c_i = L_{i0} \vee L_{i1} \vee L_{i2}$:

$$C(c_i), \quad P_0(c_i, \ell_{i0}), \quad P_1(c_i, \ell_{i1}), \quad P_2(c_i, \ell_{i2}) \quad \text{and} \quad R_0(c_i, f),$$

where $\ell_{ik} = v_j^0$ if $L_{ik} = \neg x_j$ and $\ell_{ik} = v_j^1$ if $L_{ik} = x_j$, for each $0 \leq k \leq 2$, and f is a fresh individual (f stands for *false*). Similarly to the case of variables, we need the following concept inclusion in TBox $\mathcal{T}$:

$$C \sqsubseteq \exists R_1,$$

which, together with the ABox, ensures that every c_i has P_i-successors, for $0 \leq i \leq 2$, an R_0-successor (distinct from the P_i-successors) and an R_1-successor; see Fig. 1 (b). But then, if q has a negative answer, the R_1-successor must coincide with one of the P_i-successors. This choice of the P_i determines the literal of the clause that is required to be true—if the R_1-successor of c_i is v_j^0 then the variable x_j needs to be *false* and if it is v_j^1 then x_j needs to be *true*.

To sum up, the R_1-successors of the c_i identify the literals v_j^0/v_j^1 required to be true, while the R_0-successors of the x_j choose the literals v_j^0/v_j^1 required to be false. So, the last concept inclusion of $\mathcal{T}$ ensures the choices are consistent:

$$\exists R_0^- \sqcap \exists R_1^- \sqsubseteq \bot.$$

It should be clear that q has a negative answer over $(\mathcal{T}, \mathcal{A}_\varphi)$ iff the 3CNF φ is satisfiable.

4 Answering CQs$^{\neq}$ in *DL-Lite*$_{core}^{\mathcal{H}}$: Upper Complexity Bound

In this section, a coNP in data complexity algorithm to decide CQ$^{\neq}$ answering in *DL-Lite*$_{core}^{\mathcal{H}}$ is provided. We begin by recalling some important notions and properties of canonical models.

Canonical Model

The notion of the *canonical model* in DLs [8,4,16] is related to those of the *chase*, *universal model* and *universal solution* present in data exchange and data integration settings [11]. A key characteristic of *DL-Lite*$_{core}^{\mathcal{H}}$ ontologies is that they can be regarded as sets of Horn clauses, and so, for every satisfiable *DL-Lite*$_{core}^{\mathcal{H}}$ KB $\mathcal{K}$, there is a *universal model* $\mathcal{U}$ that can be homomorphically embedded into every other model $\mathcal{J}$ of $\mathcal{K}$. Since, the positive classes of queries such as CQs are preserved under homomorphisms, universal models clearly become handy in tackling the query answering problem in Horn DLs such as *DL-Lite*$_{core}^{\mathcal{H}}$ [8,16]. Next, we recall the definition of universal and canonical models, as well as, some properties to be used in the rest of the paper.

Definition 1. *Given two interpretations* $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ *and* $\mathcal{J} = (\Delta^{\mathcal{J}}, \cdot^{\mathcal{J}})$, *a* homomorphism from $\mathcal{I}$ to $\mathcal{J}$ *is a mapping* $h\colon \Delta^{\mathcal{I}} \to \Delta^{\mathcal{J}}$ *satisfying the following conditions*:
1. $h(a^{\mathcal{I}}) = a^{\mathcal{J}}$, *for each individual name* a,
2. $h(d) \in A^{\mathcal{J}}$, *for every* $d \in A^{\mathcal{I}}$ *and each concept name* A,
3. $(h(d), h(e)) \in P^{\mathcal{J}}$, *for every* $(d, e) \in P^{\mathcal{I}}$ *and each role name* P.

An interpretation $\mathcal{U}$ is said to be a *universal model* of a KB $\mathcal{K}$ if, for every interpretation $\mathcal{J}$ with $\mathcal{J} \models \mathcal{K}$ there exist a homomorphism from $\mathcal{U}$ to $\mathcal{J}$.

Since CQs, and more generally UCQs, are positive existential formulas, they are preserved under homomorphisms and so, a standard way of computing certain answers to a given UCQ q over a KB $\mathcal{K}$ is evaluating q in a universal model $\mathcal{U}$ of $\mathcal{K}$:

Lemma 1. *Let* $\mathcal{K}$ *be a satisfiable DL-Lite KB and let* $\mathcal{U}$ *be a universal model of* $\mathcal{K}$. *Then* $\mathsf{cert}(q, \mathcal{K}) = \mathsf{ans}(q, \mathcal{U})$, *for each UCQ* q.

Kontchakov *et al.* [16] present a way of constructing a universal model of a given a KB $\mathcal{K} = (\mathcal{T}, \mathcal{A})$. The constructed model is called the *canonical model* of $\mathcal{K}$ and is denoted $\mathcal{U}_{\mathcal{K}}$.

(*i*) First, the ABox $\mathcal{A}$ is saturated by applying the concept and role inclusions of $\mathcal{T}$ in a bottom-up fashion: e.g., if $A(a) \in \mathcal{A}$ and $\mathcal{T} \models A \sqsubseteq A'$ then the ABox is extended by $A'(a)$. Note that at this stage existential quantifiers do not create any new individuals. We denote the resulting ABox by $\mathcal{A}^+$.

(*ii*) On a second stage, new individuals d_R for roles R are created to witness all existential quantifiers that are not witnessed in the ABox $\mathcal{A}^+$: e.g., if $A(a) \in \mathcal{A}^+$ and $\mathcal{T} \models A \sqsubseteq \exists R$ but $\mathcal{A}^+$ does not contain $R(a, b)$, for any b, then the ABox is extended by all $S(a, d_R)$[1] for all $\mathcal{T} \models R \sqsubseteq S$ and all $A(d_R)$ with $\mathcal{T} \models \exists R^- \sqsubseteq A$; the *generating relation* $\leadsto$ is extended by (a, d_R); note that a here is not necessarily an individual from the ABox $\mathcal{A}$ and can also be another d_S.

The ABox resulting from applying (*i*) and (*ii*) is clearly finite; the interpretation determined by this ABox is called the *generating interpretation* and is denoted by $\mathcal{I}_{\mathcal{K}}$. However, $\mathcal{I}_{\mathcal{K}}$ is not necessarily a universal model. A standard way to construct a universal model from $\mathcal{I}_{\mathcal{K}}$ is to *unravel* it into a forest-shaped interpretation. A *path* in $\mathcal{I}_{\mathcal{K}}$ is a finite sequence $ad_{R_1} \cdots d_{R_k}$ $k \geq 0$, where $a \in \mathit{ind}(\mathcal{A})$, $a \leadsto d_{R_1}$ and $d_{R_i} \leadsto d_{R_{i+1}}$.

[1] We write $R(a, b) \in \mathcal{A}$ for $P(a, b) \in \mathcal{A}$ if $R = P$ and $P(b, a) \in \mathcal{A}$ if $R = P^-$.

We use $\mathsf{paths}(\mathcal{I}_\mathcal{K})$ to denote the set of all paths in $\mathcal{I}_\mathcal{K}$ and $\mathsf{tail}(\sigma)$ to denote the last element of a path $\sigma \in \mathsf{paths}(\mathcal{I}_\mathcal{K})$. The *canonical model* $\mathcal{U}_\mathcal{K}$ of $\mathcal{K}$ is then defined as follows:

$$\begin{aligned}
\Delta^{\mathcal{U}_\mathcal{K}} &= \mathsf{paths}(\mathcal{I}_\mathcal{K}),\\
a^{\mathcal{U}_\mathcal{K}} &= a, \text{ for all } a \in \mathit{ind}(\mathcal{A}),\\
A^{\mathcal{U}_\mathcal{K}} &= \{\sigma \in \Delta^{\mathcal{U}_\mathcal{K}} \mid \mathsf{tail}(\sigma) \in A^{\mathcal{U}_\mathcal{K}}\},\\
P^{\mathcal{U}_\mathcal{K}} &= \{(a,b) \in \mathit{ind}(\mathcal{A}) \times \mathit{ind}(\mathcal{A}) \mid P(a,b) \in \mathcal{A}\} \cup\\
&\quad\ \{(\sigma, \sigma \cdot d_R) \in \Delta^{\mathcal{U}_\mathcal{K}} \times \Delta^{\mathcal{U}_\mathcal{K}} \mid \mathcal{T} \models R \sqsubseteq P\} \cup\\
&\quad\ \{(\sigma \cdot d_R, \sigma) \in \Delta^{\mathcal{U}_\mathcal{K}} \times \Delta^{\mathcal{U}_\mathcal{K}} \mid \mathcal{T} \models R \sqsubseteq P^-\}.
\end{aligned}$$

The canonical model $\mathcal{U}_\mathcal{K}$ of $\mathcal{K}$ enjoys the following structural properties:

(abox) $(a_i, a_j) \in R^{\mathcal{U}_\mathcal{K}}$ iff $R(a_i, a_j) \in \mathcal{A}^+$, for all individuals a_i, a_j and roles R;

(forest) the graph $G = (\Delta^{\mathcal{U}_\mathcal{K}}, E)$ with $E = \{(\sigma, \sigma \cdot d_R) \mid \sigma \cdot d_R \in \Delta^{\mathcal{U}_\mathcal{K}}\}$ is a forest; moreover, each ABox individual a induces a partitioning of the graph into disjoint labelled trees $\mathfrak{T}_a = (T_a, E_a, \ell_a)$ with nodes $T_a = \{\sigma \in \Delta^{\mathcal{U}_\mathcal{K}} \mid \sigma = a \cdot \sigma'\}$, edges $E_a = E \cap (T_a \times T_a)$ and labelling function $\ell_a \colon E_a \to \mathit{role}^\pm(\mathcal{K})$ such that, for every $\sigma, \sigma' \in T_a$, we have $(\sigma, \sigma') \in P^{\mathcal{U}_\mathcal{K}}$ iff

$$\text{either } \ell_a(\sigma, \sigma') = R \text{ and } \mathcal{T} \models R \sqsubseteq P \quad \text{or} \quad \ell_a(\sigma', \sigma) = R \text{ and } \mathcal{T} \models R \sqsubseteq P^-;$$

(iso) for each role R, all labelled subtrees generated by $\sigma \cdot d_R \in \Delta^{\mathcal{U}_\mathcal{K}}$ are isomorphic.

The following lemma is a consequence of the results by Kontchakov *et al.* [16]:

Lemma 2. *A DL-Lite$^\mathcal{H}_{core}$ KB $\mathcal{K}$ is satisfiable iff $\mathcal{U}_\mathcal{K} \models \mathcal{K}$.*

In contrast to the classical CQ answering problem, certain answers to CQ$^{\neq}$s over a KB $\mathcal{K}$ cannot be obtained by evaluating queries in the canonical model $\mathcal{U}_\mathcal{K}$. The main reason for this is that CQ$^{\neq}$s are not preserved under homomorphisms. Hence, the fact that $\boldsymbol{a} \in \mathsf{ans}(q, \mathcal{U}_\mathcal{K})$ does not necessarily imply that $\boldsymbol{a} \in \mathsf{ans}(q, \mathcal{I})$, for every model $\mathcal{I}$ of $\mathcal{K}$. We illustrate this situation by the following example, which is adapted from [11].

Example 1. Let $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ be a KB and q a CQ$^{\neq}$ with

$$\begin{aligned}
&\mathcal{T} = \{\exists R_1 \sqsubseteq \exists R_2,\ \exists R_1^- \sqsubseteq \exists R_3^-,\ \exists R_2^- \sqsubseteq \exists R_3\},\\
&\mathcal{A} = \{R_1(a_1, b_1)\},\\
&q(x,z) = \exists y, y' \big(R_1(x,z) \wedge R_2(x,y) \wedge R_3(y',z) \wedge (y \neq y')\big).
\end{aligned}$$

The canonical model $\mathcal{U}_\mathcal{K}$ is depicted in Fig. 2 on the left; it can be seen that $\mathsf{ans}(q, \mathcal{U}_\mathcal{K}) = \{(a_1, b_1)\}$. However, there is a model $\mathcal{J}$ of $\mathcal{K}$, depicted in Fig. 2 on the right, where $\mathsf{ans}(q, \mathcal{J}) = \emptyset$. Therefore, $\mathsf{cert}(q, \mathcal{K}) = \emptyset$.

The Decision Procedure

We proceed to show that the coNP lower complexity bound from Theorem 3 is in fact tight for answering CQs$^{\neq}$ over *DL-Lite$^\mathcal{H}_{core}$* KBs and provide an algorithm for deciding

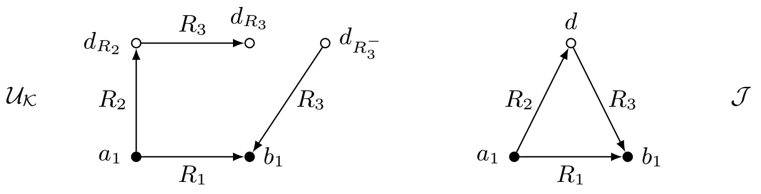


Fig. 2. The canonical model $\mathcal{U}_\mathcal{K}$ of $\mathcal{K}$ and another model $\mathcal{J}$ of $\mathcal{K}$

CQ$^{\neq}$ answering: this non-determinist algorithm will require time polynomial in the size of the given ABox. We observe that the problem of deciding, given a CQ$^{\neq}$ q and a KB $\mathcal{K}$, whether $\boldsymbol{a} \in \mathsf{cert}(q, \mathcal{K})$ can be reduced to the problem of deciding whether $\mathsf{cert}(q(\boldsymbol{a}), \mathcal{K}) \neq \emptyset$, where $q(\boldsymbol{a})$ is the query obtained by substituting $\boldsymbol{x}$ in q by $\boldsymbol{a}$; thus, $q(\boldsymbol{a})$ has no answer variables and is usually called a Boolean query. Furthermore, by the certain answer semantics, we can consider the problem of answering Boolean queries as a logical entailment problem, i.e., $\mathsf{cert}(q, \mathcal{K}) \neq \emptyset$ iff $\mathcal{K} \models q$, i.e., $\mathcal{I} \models q$ in every model $\mathcal{I}$ of $\mathcal{K}$. So, $\mathsf{cert}(q, \mathcal{K}) = \emptyset$ iff

$$\mathcal{K} \cup \neg q \text{ is satisfiable, i.e., there is a model } \mathcal{I} \text{ of } \mathcal{K} \text{ such that } \mathcal{I} \models \neg q. \tag{2}$$

It is not hard to see that there is a correspondence between negated CQ$^{\neq}$s and so-called disjunctive EGDs.

We remind the reader that an *equality-generating dependency* (*EGD*) [5] is a formula of the form $\forall \boldsymbol{x}\,\big(\phi(\boldsymbol{x}) \to (x_1 = x_2)\big)$, where x_1, x_2 are among the variables in $\boldsymbol{x}$. A *disjunctive EGD* [12] is a formula of the form

$$\forall \boldsymbol{x}\,\big(\phi(\boldsymbol{x}) \to \bigvee_{i=1}^{n} (x_i^1 = x_i^2)\big). \tag{3}$$

Note that an EGD is a disjunctive EGD whose right-hand side has only one equality.

Given a Boolean CQ$^{\neq}$ $q = \exists \boldsymbol{x}\,(\phi(\boldsymbol{x}) \land \bigwedge_i (x_i^1 \neq x_i^2))$, where $\phi(\boldsymbol{x})$ is a conjunction of concept and role atoms, it should be clear that $\neg q$ is logically equivalent to a disjunctive EGD of the form (3). Disjucntive EGDs are clearly able to express concept inclusion axioms with arbitrary number restrictions (in particular, functionality of roles), which is known to increase the complexity of reasoning in *DL-Lite* [7].

The previous discussion suggests the following algorithm to check condition (2): non-deterministically guess a model $\mathcal{J}$ of $\mathcal{K}$ and then check in polynomial time whether $\mathcal{J}$ satisfies $\neg q$. In order to obtain the coNP result, $\mathcal{J}$ needs not only to be finite but also small enough—at most polynomial in the size of the ABox of $\mathcal{K}$. Unfortunately, this straightforward approach is too naive. Indeed, since disjunctive EGDs allow to express *global functionality* of roles, and the extension of *DL-Lite*$_{core}^{\mathcal{H}}$ with functional roles does not enjoy the finite model property then we cannot guess a finite model $\mathcal{J}$ and check whether $\mathcal{J} \models \neg q$, as shown by the following example.

Example 2. Let $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ with $\mathcal{T} = \{\exists P^- \sqsubseteq \exists P,\ A \sqsubseteq \neg\exists P^-,\ A \sqsubseteq \exists P\}$, $\mathcal{A} = \{A(a)\}$ and $q = \exists x, y_1, y_2\,\big(P(y_1, x) \land P(y_2, x) \land (y_1 \neq y_2)\big)$, which 'says' that

P^- is functional. The canonical model $\mathcal{U}_\mathcal{K}$ is an infinite P-chain starting at a, whence $\mathcal{U}_\mathcal{K} \models \neg q$. However, there is no finite model of $\mathcal{K}$ satisfying $\neg q$. In fact, for every model $\mathcal{J}$ of $\mathcal{K}$ with $\mathcal{J} \models \neg q$ there is an *injective* homomorphism from $\mathcal{U}_\mathcal{K}$ to $\mathcal{J}$.

In order to have an effective algorithm we need then to find a way to simulate the possibly infinite model of a KB $\mathcal{K}$ in a small finite initial fragment of the canonical model $\mathcal{U}_\mathcal{K}$ of $\mathcal{K}$. We start by recalling that for answering CQs only a linear number (in the size of the TBox of $\mathcal{K}$) of existential witnesses are need to be considered [4,16]. Next, we show that for answering CQs$^{\neq}$ in *DL-Lite*$^{\mathcal{H}}_{core}$ it is also enough to consider only a linear number of existential witnesses for falsifying the inequalities in q. The main difference is that for answering CQs$^{\neq}$ we need to try all possible configurations of identifying objects in the initial fragment of the model, and hence the increase in complexity.

Let us fix a *DL-Lite*$^{\mathcal{H}}_{core}$ KB $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ and a CQ$^{\neq}$ q for the rest of this section. An *expansion* $\mathcal{A}'$ *of* $\mathcal{A}$ is a (possibly infinite) set of assertions (in the signature of $\mathcal{T}$) that contains $\mathcal{A}$ and whose individuals are taken from the domain of the canonical model $\mathcal{U}_\mathcal{K}$ of $\mathcal{K}$. In other words, an expansion is a description of a part of the canonical model $\mathcal{U}_\mathcal{K}$. Consider now a disjunctive EGD of the form (3) which is equivalent to $\neg q$. We associate with it the following set $\mathbf{E}$ of individual EGDs:

$$\mathbf{E} \;=\; \{\, \underbrace{\forall \boldsymbol{x}\,\big(\phi(\boldsymbol{x}) \to (x_1^1 = x_1^2)\big)}_{e_1},\; \ldots,\; \underbrace{\forall \boldsymbol{x}\,\big(\phi(\boldsymbol{x}) \to (x_n^1 = x_n^2)\big)}_{e_n} \,\}.$$

Definition 2. *Let $\mathcal{A}'$ be an ABox expansion of $\mathcal{A}$ and h a homomorphism from $\phi(\boldsymbol{x})$ to $\mathcal{A}'$. We say that e_i is* applicable to $\mathcal{A}'$ with h *if $h(x_i^1) \neq h(x_i^2)$ and the result of applying e_i to $\mathcal{A}'$ with h is one of the following*:

(fail) *a* failure*, in which case we write $\mathcal{A}' \xrightarrow{h,e_i} \bot$, if either*
  1. *$h(x_i^1), h(x_i^2) \in ind(\mathcal{A})$, or*
  2. *$B_1(h(x_i^1)), B_2(h(x_i^2)) \in \mathcal{A}'$, for some $\mathcal{T} \models B_1 \sqcap B_2 \sqsubseteq \bot$, or*
  3. *$R_1(a, h(x_i^1)), R_2(a, h(x_i^2)) \in \mathcal{A}'$, for $a \in ind(\mathcal{A}')$ and $\mathcal{T} \models R_1 \sqsubseteq \neg R_2$;*

(id) *an ABox expansion $\mathcal{A}''$ (written $\mathcal{A}' \xrightarrow{h,e_i} \mathcal{A}''$) obtained by identifying $h(x_i^1)$ and $h(x_i^2)$: every occurrence of x_i^1 and x_i^2 is replaced by x_i^k, if $h(x_i^k) \in ind(\mathcal{A})$ for $k = 1$ or 2, and by x_i^1, otherwise (the choice of x_i^1 here is arbitrary as neither of them is in the ABox).*

We say that $\mathbf{E}$ is *applicable to* $\mathcal{A}'$ *with* h if e_i is applicable to $\mathcal{A}'$ with h for every $1 \leq i \leq n$. The result of applying $\mathbf{E}$ to $\mathcal{A}'$ with h is the set $\{\mathcal{A}'_1, \ldots, \mathcal{A}'_n\}$, where each $\mathcal{A}'_i$ is the result of applying e_i to $\mathcal{A}'$ with h; we write $\mathcal{A}' \xrightarrow{h,\mathbf{E}} \{\mathcal{A}'_1, \ldots \mathcal{A}'_n\}$ in this case. If every $\mathcal{A}'_i = \bot$ we say that the application of $\mathbf{E}$ to $\mathcal{A}$ with h *fails*, and write $\mathcal{A}' \xrightarrow{h,\mathbf{E}} \bot$; otherwise we say it is *non-failing*.

We first show some technical results on disjucntive EGD applications. The following is an easy consequence of the definition of (non-failing) application of $\mathbf{E}$ to an ABox expansion $\mathcal{A}'$:

Proposition 1. *Let $\mathcal{A}'$ be a finite ABox expansion of $\mathcal{A}$ and $\mathcal{A}' \xrightarrow{h,\mathbf{E}} \{\mathcal{A}'_1, \dots \mathcal{A}'_n\}$ a non-failing application of* $\mathbf{E}$ *to $\mathcal{A}'$. Then $dom(\mathcal{A}') \supseteq dom(\mathcal{A}'_i)$, for all $1 \leq i \leq n$ with $\mathcal{A}'_i \neq \bot$.*

In fact, given any model $\mathcal{J}$ of $\mathcal{K}$, every homomorphism from an ABox expansion to $\mathcal{J}$ can be extended to a homomorphism from a non-failng application of $\mathbf{E}$:

Lemma 3. *Let $\mathcal{A}'$ be a finite ABox expansion of $\mathcal{A}$ and $\mathcal{A}' \xrightarrow{h,\mathbf{E}} \{\mathcal{A}'_1, \dots \mathcal{A}'_n\}$ a non-failing application of* $\mathbf{E}$ *to $\mathcal{A}'$ with h and $\mathcal{J}$ a model of $\mathcal{K}$ such that $\mathcal{J} \models \neg q$ and there is a homomorphism g from $\mathcal{A}'$ into $\mathcal{J}$. Then there exists a homomorphism g_j from $\mathcal{A}'_j$ into $\mathcal{J}$ for some $1 \leq j \leq n$.*

Now we consider *non-failing sequence of application of* $\mathbf{E}$, which are sequence of the form $\mathcal{A}' \xrightarrow{h_1,e_{i_1}} \mathcal{A}'_1 \xrightarrow{h_2,e_{i_2}} \dots \xrightarrow{h_k,e_{i_k}} \mathcal{A}'_k$ with $1 \leq i_j \leq n$ and $\mathcal{A}'_j \neq \bot$, for every $1 \leq i \leq k$. It turns out that after applying a non-failing application of EGD in some part of the ABox expansion the successive applications of the disjunctive EGD do not “use the same match”, and therefore, the process will eventually either succeed on the application or fail:

Proposition 2. *For every non-failing sequence $\mathcal{A}' \xrightarrow{h_1,e_{i_1}} \mathcal{A}_1 \xrightarrow{h_2,e_{i_2}} \dots \xrightarrow{h_k,e_{i_k}} \mathcal{A}_k$ of applications of* $\mathbf{E}$*, we have $h_j(x) \neq h_{j'}(x)$, for all $1 \leq j < j' \leq k$ and all $x \in \boldsymbol{x}$.*

Next, we show that for checking whether there is a model $\mathcal{I}$ of $\mathcal{K}$ with $\mathcal{I} \models \neg q$ it suffices to apply $\mathbf{E}$ to an ABox expansion $\widehat{\mathcal{A}}$ that corresponds to the canonical model ‘truncated’ to points of depth up to $N = |role^{\pm}(\mathcal{K})| + |q|$. More formally, given a natural number N, the *truncation* $\mathcal{U}_{\mathcal{K}}^N$ of the canonical model $\mathcal{U}_{\mathcal{K}}$ to depth N is the restriction of $\mathcal{U}_{\mathcal{K}}$ to the following domain:

$$\Delta^{\mathcal{U}_{\mathcal{K}}^N} = \{\sigma \in \Delta^{\mathcal{U}_{\mathcal{K}}} \mid \|\sigma\| \leq N\},$$

where $\|\sigma\|$ is the length of a path σ. By Proposition 2, there is a bound on the number of possible applications of $\mathbf{E}$ to $\widehat{\mathcal{A}}$. More precisely, the length of every application sequence of $\mathbf{E}$ to $\widehat{\mathcal{A}}$ is bounded by a polynomial in the size of $\mathcal{A}$.

Finally, we show the following:

Lemma 4. *Let $\widehat{\mathcal{A}}$ the ABox expansion of $\mathcal{A}$ induced the truncation $\mathcal{U}_{\mathcal{K}}^N$ of the canonical model $\mathcal{U}_{\mathcal{K}}$ of $\mathcal{K}$ to depth $N = |role^{\pm}(\mathcal{K})| + |q|$. The following statements are equivalent*:

1. *there exists a model $\mathcal{J}$ of $\mathcal{K}$ such that $\mathcal{J} \models \neg q$;*
2. *there is a sequence $e_1, \dots, e_k$ of elements of* $\mathbf{E}$ *such that $\widehat{\mathcal{A}} \xrightarrow{h_1,e_1} \mathcal{A}_1 \xrightarrow{h_2,e_2} \dots \xrightarrow{h_k,e_k} \mathcal{A}_k$ is non-failing and $\mathcal{A}_k$ satisfies $\neg q$.*

Now, given a *DL-Lite*$_{core}^{\mathcal{H}}$ KB $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ and a CQ$^{\neq}$ q, our algorithm for checking condition (2) works as follows:

1. It constructs the ABox expansion $\widehat{\mathcal{A}}$ of $\mathcal{A}$ induced by $\mathcal{U}_{\mathcal{K}}^N$, for $N = |role^{\pm}(\mathcal{K})| + |q|$.
2. Guesses a sequence Σ of elements of $\mathbf{E}$.
3. Checks whether $\mathbf{E}$ is satisfied after the application of Σ to $\widehat{\mathcal{A}}$.

It is not hard to see that this non-deterministic algorithm runs in polynomial time in the size of the ABox. So, by (2) and Lemma 4, we obtain a matching upper bound for Theorem 3, which results in the following:

Theorem 4. *Answering CQs$^{\neq}$ over DL-Lite$^{\mathcal{H}}_{core}$ KBs is* CONP*-complete in data complexity.*

5 Tractable Cases

In this section, we define syntactic restrictions on the class of CQs$^{\neq}$ in order to achieve tractability of CQ$^{\neq}$ answering in *DL-Lite*$^{\mathcal{H}}_{core}$. In the data exchange setting (DE) it has been shown that answering CQs$^{\neq}$ with *at most* two inequalities in the presence of target constraints expressed by *weakly acyclic* TGDs is CONP-complete in data complexity [17]. In the case of DLs, we note that even very simple *DL-Lite*$^{\mathcal{H}}_{core}$ TBoxes are not weakly acyclic. On the other hand, the reductions used for proving CONP-hardness of the CQ$^{\neq}$ answering problem in the DE setting make use of ternary relations, which is outside the expressive power of *DL-Lite*$^{\mathcal{H}}_{core}$. Up to this point, we can only conjecture that answering CQs$^{\neq}$ in *DL-Lite*$^{\mathcal{H}}_{core}$ that contain at least two inequalities is CONP-hard. Therefore, we based our syntactic restrictions on the latter assumption.

We shall consider CQs with at most two inequalities. Roughly, in order to have a polynomial algorithm in data complexity for checking that $\mathcal{K} \models q$ or alternatively that $\mathcal{K} \cup \neg q$ is unsatisfiable, we need to be able 1) to simulate the infinite chase in a finite search space and 2) perform the evaluation using a small amount of space (e.g., constant in the size of the ABox). In order to have a correct and complete algorithm fulfilling these conditions, we impose syntactic restrictions enforcing that, for every possible match π for a query q in a given interpretation $\mathcal{I}$ and for every inequality $x_i^1 \neq x_i^2$ in q, either $\pi(x_i^1) = a$ or $\pi(x_i^2) = a$, for some individual name a. This condition is enough to ensure polynomial-time query evaluation because, although this kind of inequalities are not preserved under homomorphisms, they induce only few possible models.

We adopt and adapt the notions of *constant joins* and *almost constant inequalities* introduced by Arenas *et al.* [3]. For defining these notions in the DL setting, it is convenient to identify the concepts that may need to be 'realised outside' the ABox in every model of a KB.

Definition 3. *Let $\mathcal{T}$ be a DL-Lite$^{\mathcal{H}}_{core}$ TBox. A concept $\exists R$ is called* affected in $\mathcal{T}$ *if either*

1. *$\mathcal{T} \models A \sqsubseteq \exists R^-$, for some concept name A, or*
2. *$\mathcal{T} \models \exists S \sqsubseteq \exists R^-$, for some role S with $\mathcal{T} \not\models S \sqsubseteq R^-$.*

We say an inequality $(x_1 \neq x_2)$ in q is *almost constant for $\mathcal{T}$* if q contains either some $R(t, x_1)$ or some $R(t, x_2)$ such that $\exists R^-$ is not affected in $\mathcal{T}$. Intuitively, queries with almost constant inequalities ensure that at least one variable in each inequality is forced to be an ABox individual. A query q is said to have *constant joins for $\mathcal{T}$* if either $\exists R_1^-$ or $\exists R_2^-$ is not affected in $\mathcal{T}$, for every join $R_1(t_1, t), R_2(t_2, t)$ in q. This means that t has to be mapped to an ABox individual by every possible match for q in any model of the KB.

Definition 4. *A CQ$^{\neq}$ q is said to be* safe *if one the following conditions holds*:

1. *q has no inequalities,*
2. *q has exactly one inequality, which is almost constant,*
3. *q has exactly two inequalities, which are almost constant, and constant joins.*

Intuitively, to falsify the inequalities in a safe CQ$^{\neq}$ it suffices to consider only inequalities of the form $a_1 \neq d$ and $a_1 \neq a_2$, where $a_1, a_2 \in ind(\mathcal{A})$ and d is an anonymous individual in the canonical model, i.e., a path in $\Delta^{\mathcal{U}_\mathcal{K}}$ of the form $\sigma \cdot d_R$. This means, that q can actually be evaluated in the ABox expansion corresponding to the generating interpretation $\mathcal{I}_\mathcal{K}$.

Given a safe CQ$^{\neq}$, we need to provide an algorithm for deciding whether $\mathcal{K} \models q$. The algorithm presented in Section 4 considers a truncation of the canonical model $\mathcal{U}_\mathcal{K}$ of $\mathcal{K}$ for evaluating $\neg q$. However, in this case—as we argued above—by the syntactic restrictions on q, the evaluation requires only to consider the generating interpretation $\mathcal{I}_\mathcal{K}$ as in the case for queries without inequalities and suggests that we can adapt the combined approach for query answering [16] by making minor changes to the rewriting of q. Thus, we obtain the following result:

Theorem 5. *Answering safe CQ$^{\neq}$ over DL-Lite$^{\mathcal{H}}_{core}$ KBs is in AC^0 in data complexity.*

6 Conclusions

The known and obtained complexity results on answering CQs and UCQs with safe negation and inequalities are presented in the table below:

	CQ$^{\neq}$	UCQ$^{\neq}$	CQ$^{\neg s}$	UCQ$^{\neg s}$
DL-Lite$_{core}$	coNP Thms. 3, 4	undec. Thm. 1	coNP-hard [19, Thm. 13]	undec. Thm. 2
DL-Lite$^{\mathcal{H}}_{core}$	coNP [19, Thm. 6] Thm. 4	undec. [19, Thm. 8]	coNP-hard [19, Thm. 13]	undec. [19, Thm. 15]

We have presented some further steps towards a systematic study of query answering in DLs when extensions of CQs with negated atoms are considered. In particular, we build on previous work by Rosati [19], and extend this investigation by adapting techniques from such areas as data exchange to identify tractable cases of CQ$^{\neq}$ answering in logics from the *DL-Lite* family. Clearly, more investigations needs to be done to construct a complete picture of the computational complexity and to develop algorithmic approaches. We outline below some research questions we will address in the future:

1. Investigating query answering with inequalities in other logics of *DL-Lite* family and $\mathcal{EL}$ family.
2. Closing the gap on the number of inequalities needed to make CQ$^{\neq}$ answering intractable in DLs of the *DL-Lite* family.
3. An interesting and challenging problem is the development of a decision procedures for answering CQs$^{\neg s}$. We also plan to consider other types of negation such as Boolean combinations of CQs (BCCQs) advocated in areas related to the management of incomplete information [10,13].

References

1. Abiteboul, S., Duschka, O.M.: Complexity of answering queries using materialized views. In: Proc. of PODS, pp. 254–263. ACM Press (1998)

2. Abiteboul, S., Duschka, O.M.: Complexity of answering queries using materialized views. Tech. Rep. Gemo Report 383, INRIA Saclay (1999)
3. Arenas, M., Barceló, P., Reutter, J.L.: Query languages for data exchange: Beyond unions of conjunctive queries. Theory Comput. Syst. 49(2), 489–564 (2011)
4. Artale, A., Calvanese, D., Kontchakov, R., Zakharyaschev, M.: The DL-Lite family and relations. J. Artif. Intell. Res. (JAIR) 36, 1–69 (2009)
5. Beeri, C., Vardi, M.Y.: A proof procedure for data dependencies. J. ACM 31(4), 718–741 (1984)
6. Bienvenu, M., Ortiz, M., Simkus, M.: Answering expressive path queries over lightweight DL knowledge bases. In: Proc. of DL. CEUR Workshop Proceedings, vol. 846 (2012)
7. Calvanese, D., De Giacomo, G., Lembo, D., Lenzerini, M., Rosati, R.: Data complexity of query answering in description logics. In: Doherty, P., Mylopoulos, J., Welty, C.A. (eds.) Proc. of KR 2006, pp. 260–270. AAAI Press (2006)
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. Calvanese, D., De Giacomo, G., Lenzerini, M.: On the decidability of query containment under constraints. In: Mendelzon, A.O., Paredaens, J. (eds.) Proc. of PODS, pp. 149–158. ACM Press (1998)
10. ten Cate, B., Chiticariu, L., Kolaitis, P.G., Tan, W.C.: Laconic schema mappings: Computing the core with SQL queries. PVLDB 2(1), 1006–1017 (2009)
11. Deutsch, A., Nash, A., Remmel, J.B.: The chase revisited. In: Lenzerini, M., Lembo, D. (eds.) Proc. of PODS. pp. 149–158. ACM Press (2008)
12. Fagin, R., Kolaitis, P.G., Miller, R.J., Popa, L.: Data exchange: semantics and query answering. Theor. Comput. Sci. 336(1), 89–124 (2005)
13. Gheerbrant, A., Libkin, L., Tan, T.: On the complexity of query answering over incomplete XML documents. In: Deutsch, A. (ed.) Proc. of ICDT, pp. 169–181. ACM Press (2012)
14. Harel, D.: Effective transformations on infinite trees, with applications to high undecidability, dominoes, and fairness. J. ACM 33(1), 224–248 (1986)
15. Klug, A.: On conjunctive queries containing inequalities. J. ACM 35(1), 146–160 (1988)
16. Kontchakov, R., Lutz, C., Toman, D., Wolter, F., Zakharyaschev, M.: The combined approach to query answering in DL-Lite. In: Proc. of KR 2010. AAAI Press (2010)
17. Madry, A.: Data exchange: On the complexity of answering queries with inequalities. Inf. Process. Lett. 94(6), 253–257 (2005)
18. Rosati, R.: On the decidability and finite controllability of query processing in databases with incomplete information. In: Vansummeren, S. (ed.) Proc. of PODS, pp. 356–365. ACM (2006)
19. Rosati, R.: The Limits of Querying Ontologies. In: Schwentick, T., Suciu, D. (eds.) ICDT 2007. LNCS, vol. 4353, pp. 164–178. Springer, Heidelberg (2006)

Towards a Unifying Approach to Representing and Querying Temporal Data in Description Logics

Víctor Gutiérrez-Basulto[1] and Szymon Klarman[2]

[1] Department of Computer Science, Universität Bremen
victor@informatik.uni-bremen.de
[2] Department of Computer Science, Vrije Universiteit Amsterdam
s.klarman@vu.nl

Abstract. Establishing a generic approach to representing and querying temporal data in the context of Description Logics (DLs) is an important, and still open challenge. The difficulty lies in that a proposed approach should reconcile a number of valuable contributions coming from diverse, yet relevant research lines, such as temporal databases and query answering in DLs, but also temporal DLs and Semantic Web practices involving rich temporal vocabularies. Within such a variety of influences, it is critical to carefully balance theoretical foundations with good prospects for reusing existing techniques, tools and methodologies. In this paper, we attempt to make first steps towards this goal. After providing a comprehensive overview of the background research and identifying the core requirements, we propose a general mechanism of defining temporal query languages for time-stamped data in DLs, based on combinations of linear temporal logics with first-order queries. Further, we advocate a controlled use of epistemic semantics in order to warrant practical query answering. We systematically motivate our proposal and highlight its basic theoretical and practical implications. Finally, we outline open problems and key directions for future research.

1 Introduction

The use of Description Logic (DL) ontologies for describing and interpreting data is acknowledged by now as a self-standing paradigm of data management in different areas of computer science — most prominently on the Semantic Web (SW), where DL-based ontology languages play a key architectural role. One big and yet unresolved challenge in this context, called for by numerous applications, is to formally incorporate and operationalize the notion of data's *validity time*, i.e. the explicitly declared time span within which the data is known to be true.

Problem: In this paper, we study the problem of managing temporal data in the framework of DLs. Our goal is to make first steps towards establishing a unifying approach to representing and querying such data under DL ontologies. Given the multifaceted nature of the problem and the scope of expected applications, one of main challenges which must be faced lies in reconciling a number of valuable contributions developed within diverse research areas. In particular:

– *temporal databases*: for ensuring commensurability with the commonly adopted temporal data models for representing validity time and with standard query languages based on temporal first-order logic,
– *query answering* in DLs: for enabling transfer of known query answering techniques, complexity results, and facilitating reuse of existing tools,
– *temporal DLs*: for enabling the possibility of managing temporal data under DL ontologies which capture temporal constraints on the intensional level,
– SW *temporal vocabularies*: for supporting typical SW practices involving OWL-based time ontologies, which provide rich temporal vocabularies employed on the level of queries and data annotations.

Clearly, under such a variety of influences, it is critical to carefully balance theoretical foundations of a proposed approach with good prospects for reusing existing techniques, tools and methodologies.

Contributions: We introduce a basic framework for representing temporal data in arbitrary DLs, where the data takes the form of time-stamped ABox assertions $[t_1, t_2] : \alpha$, stating validity of the assertion α during the interval $[t_1, t_2]$. Then we propose a general mechanism of defining corresponding temporal query languages, based on combinations of linear temporal logics with classes of first-order queries — specifically, with well-known conjunctive queries. In particular:

– we systematically motivate the proposed mechanism, present the syntax and certain answer semantics for the query languages that the mechanism generates, and the relationship of those languages to temporal first-order logic.
– we advocate a controlled use of epistemic semantics in order to warrant practical query answering in the defined setting. Under this restriction, we deliver a $\textsc{PSpace}^{QA(\mathcal{L})}$-completeness bound for the combined complexity of answering temporal queries in an arbitrary DL $\mathcal{L}$, where $QA(\mathcal{L})$ is an oracle answering conjunctive queries in $\mathcal{L}$. We highlight some essential theoretical and practical implications of this result.
– we discuss the possibility of pushing the approach further towards integration with temporal DLs and SW temporal vocabularies.

Structure of the Paper: In Section 2, we provide a comprehensive overview of the background research and identify the core requirements for the proposed approach. Next, we discuss DL preliminaries in Section 3 and introduce the temporal data model in Section 4. In Section 5, we present and study the proposed mechanism of defining temporal query languages. In Section 6 we discuss similarities to existing approaches and outline some future research directions. We conclude the paper in Section 7.

2 Overview and Background

Extending information systems with capabilities for managing temporal information has been deeply studied and advocated in many areas of computer science, particularly, in those concerned with relational databases and knowledge representation. Surprisingly, despite the successful use of the ontology-based data access (OBDA) paradigm as an application of DL technologies in databases, the development of mechanisms for extending the OBDA approach towards accessing temporal data have not been yet investigated. A proposed mechanism should naturally take into account the already well-founded research lines on representing and querying temporal information, as well as valuable contributions in related areas, which we outline in the following paragraphs.

Ontology-Based Data Access. The *ontology-based data access* is a paradigm of managing data in presence of background knowledge, represented as a formal ontology, enabling convenient query answering over *incomplete* data. In recent years, special attention has been given to ontologies based on DL languages. A considerable amount of research has been devoted to the problem of query answering in DLs, focusing predominantly on *conjunctive queries* (CQs). This has lead to establishing a clear picture of the computational complexity of CQ answering, and to the development of algorithmic approaches. The study has been focused on two major lines: 1) utilization of classical DLs with high expressive power, where the complexity of query answering turns out typically too high for practical applications [1]; 2) development of DLs allowing efficient query answering over large amounts of data. Calvanese et.al. [2] introduced the DL-Lite family of DLs, for which efficient OBDA can be achieved by reduction to query answering in relational database management systems (RDBMSs). One of the key motivations behind the design of the temporal query languages presented in this paper is to enable easy, modular reuse of the known techniques and results on query answering in DLs in the context of temporal data querying.

Temporal Databases. During the 90s, the database community conducted an exhaustive study on temporal extensions of the standard relational data models, supporting management of temporal information. The common way of constructing *temporal relational databases* (TDBs) is to enrich traditional data models with *time-stamps* representing data's validity time, i.e. the time span within which the data is known to be true. As one of the crucial requirements for our approach we pose formal compatibility with the TDB paradigm of representing temporal data. Inspired by the notion of *concrete temporal database* [3], we construct a *temporal ABox* by time-stamping every ABox assertion with a weak-interval of the form $[t_1, t_2]$, compactly representing a set of time points in which the assertion is valid. The semantics of a temporal ABox, by analogy to TDBs case [3], is given by mapping each time point in the underlying *time domain* to the non-temporal (standard) ABox — a so-called *snapshot* — containing exactly the assertions valid in that point. Eventually, the OBDA paradigm is applied within the scope of respective snapshots.

Regarding the choice of the time domain, the TDB literature reports on a number of possible representations, each one having far-reaching philosophical, logical and computational consequences [4]. The available degrees of freedom concern, among others: the nature of the atomic time entities (points *vs.* intervals), their ordering relationships (linear *vs.* branching *vs.* partial orders), the density (discrete *vs.* continuous), the boundaries (finite *vs.* infinite). Although strict commitment to any representation is always arbitrary to some extent, arguably one of the most natural and commonly used setups in TDBs, which we also adopt here, is the one capturing the intuition of a point-based time line [4].

A temporal data model is complemented by an adequate *temporal query language* for querying temporal data. In this aspect, we ground our proposal in two well-known research lines. 1) Following the research on TDBs, we consider languages based on fragments of *temporal first-order logic*, which has been advocated as a suitable high-level formalism for querying TDBs [3]. It has been shown, that queries expressed in temporal first-order logic can be translated directly to TSQL2 [5] — a temporal extension of the standard database query language SQL — and thus efficiently handled using existing TDB systems. 2) Given the known landscape of complexity results and developed techniques for query answering in DLs, we pay special attention to the expressiveness of the first-order component within the intended fragments of temporal first-order logic. As explained in detail in Section 5, our motivation is to provide a mechanism for defining such fragments in a controlled, modular manner, by selecting particular sets of temporal operators and particular classes of first-order queries to be combined. By specifying those two parameters one should effectively obtain a ready query language of a well-characterized computational behavior. To this end we make use of the methodology of temporalizing logic systems [6].

Semantic Web and Temporal DLs. In recent years, the problem of managing time-varying knowledge has gained a lot of interest also in the Semantic Web research community. Particularly, the need for describing temporal information on the Web gave rise to various *time ontologies* [7], which formalize common temporal notions, such as temporal instants, temporal intervals and calendar terms, and offer standardized formats for representing different types of temporal information. Although such ontologies succeed in facilitating exchange of time-oriented data among Web agents, they are not accompanied by any formally grounded methodologies of processing such information. Specifically, they offer no inference mechanisms to support genuinely temporal reasoning. This lack of rigorous logical foundations, is in practical scenarios partially remedied by the use of programming tools and ad-hoc hybrid architectures [8,9].

Some alternative approaches, building more systematically on the TDB philosophy, were also proposed for representing and querying temporal data in RDFs [10,11] and OWL [12]. Although employing the same or similar temporal data models as in our case, these frameworks are mostly technology-driven and do not consider the design of query languages in sufficient generality.

A somewhat orthogonal research effort has gone into designing a family of *temporal description logics* (TDLs) [13] tailored for representing and reasoning with inherently temporal terminologies. As proper combinations of temporal logics with DLs, TDLs count with a well-defined temporal semantics, which makes them very appealing from the theoretical perspective. Nevertheless, most of the contributions in this area focus on traditional reasoning tasks such as satisfiability and subsumption, related mostly to conceptual modeling rather than querying temporal data, with very few, limited exceptions [14]. In general, a potential transfer of the known query answering techniques for DLs to the TDL setting seems highly non-trivial.

Although in this paper we do not address the problems related to querying temporal data with support of SW temporal vocabularies or in presence of TDL ontologies, we do acknowledge them as worthwhile challenges for future research, and we briefly reconsider them in Section 6.

3 Description Logic Preliminaries

We use the standard nomenclature and notation for the syntax and semantics of DLs (see [15] for full details). A DL language $\mathcal{L}$ is defined over a vocabulary $\Sigma = (\mathsf{N_C}, \mathsf{N_R}, \mathsf{N_I})$, where $\mathsf{N_C}, \mathsf{N_R}, \mathsf{N_I}$ are countably infinite sets of concept names, role names and individual names, respectively. By convention, we use letters A, B to denote concept names, r, s for role names and a, b for individual names. The grammar for complex concepts, roles and axioms is defined relative to a given DL dialect. For instance, the DL $\mathcal{ALC}$ provides the following concept constructors:

$$\neg C \mid C \sqcap D \mid C \sqcup D \mid \exists r.C \mid \forall r.C$$

where C, D are (possibly complex) concepts. A TBox $\mathcal{T}$ is a finite set of concept inclusions of the form $C \sqsubseteq D$, whereas an ABox $\mathcal{A}$ is a finite set of assertions of types $A(a)$ and $r(a, b)$. By following the DL-based OBDA paradigm, we consider a TBox to be the ontology through which one accesses the data represented as an ABox.

The semantics of $\mathcal{L}$ is given through interpretations $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$, where $\Delta^{\mathcal{I}}$ is a non-empty domain of individuals and $\cdot^{\mathcal{I}}$ is an interpretation function, which maps $A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$, for every $A \in \mathsf{N_C}$, $r^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$, for every $r \in \mathsf{N_R}$, and $a^{\mathcal{I}} \in \Delta^{\mathcal{I}}$, for every $a \in \mathsf{N_I}$, and is inductively extended over complex expressions according to the fixed conditions associated with each constructor [15]. An interpretation $\mathcal{I}$ is a model of a TBox $\mathcal{T}$ (resp. ABox $\mathcal{A}$), written $\mathcal{I} \models \mathcal{T}$ (resp. $\mathcal{I} \models \mathcal{A}$) *iff* it satisfies all the concept inclusions in $\mathcal{T}$ (resp. assertions in $\mathcal{A}$), where the satisfaction relation for concept inclusions and assertions is defined in the usual way. An ABox $\mathcal{A}$ is consistent w.r.t. a TBox $\mathcal{T}$ *iff* there exists an interpretation $\mathcal{I}$ which is a model of both $\mathcal{A}$ and $\mathcal{T}$, written as $\mathcal{I} \models \mathcal{T}, \mathcal{A}$.

Next, we recall the standard notion of conjunctive queries — the most commonly studied query formalism in the context of DLs [1]. Let $\mathsf{N_V}$ be a countably infinite set of variables. A *conjunctive query* (CQ) over a DL vocabulary Σ is a first-order formula $\exists \vec{y}.\varphi(\vec{x}, \vec{y})$, where $\vec{x}, \vec{y}$ are sequences of variables. The sequence $\vec{x}$ denotes the free, *answer variables* in the query, while $\vec{y}$ the quantified ones. The formula φ is a conjunction of atoms over Σ of the form $A(u), r(u, v)$, where $u, v \in \mathsf{N_V} \cup \mathsf{N_I}$ are called terms.

Let $\mathcal{I}$ be an interpretation and $q(\vec{x})$ a CQ with the answer variables $\vec{x} = x_1, \ldots, x_k$. By $\text{term}(q)$ we denote the set of all terms occurring in q. For a sequence $\vec{a} = a_1, \ldots, a_k \in \mathsf{N_I}$, an $\vec{a}$-match to a query $q(\vec{x})$ in $\mathcal{I}$ is a mapping

$\mu : \mathrm{term}(q) \mapsto \Delta^{\mathcal{I}}$, such that $\mu(x_i) = a_i{}^{\mathcal{I}}$, for every $1 \leq i \leq k$, $\mu(a) = a^{\mathcal{I}}$, for every $a \in \mathsf{N_I} \cap \mathrm{term}(\mathsf{q})$, for every $A(u)$ in q it is the case that $\mu(u) \in A^{\mathcal{I}}$ and for every $r(u, v)$ in q it is the case that $(\mu(u), \mu(v)) \in r^{\mathcal{I}}$. We write $\mathcal{I} \models q[\vec{a}]$ whenever there exists an $\vec{a}$-match to q in $\mathcal{I}$, and $\mathcal{T}, \mathcal{A} \models q[\vec{a}]$ whenever there exists an $\vec{a}$-match to q in every model of $\mathcal{T}$ and $\mathcal{A}$. In the latter case $\vec{a}$ is called a *certain answer* to q *w.r.t.* $\mathcal{T}$ and $\mathcal{A}$.

In what follows we implicitly assume that all considered TBoxes, ABoxes and CQs are expressed over the same DL vocabulary Σ.

4 Temporal Data Model

A temporal data model is formally specified by two basic characteristics: the choice of the underlying time domain and the syntax and semantics of temporal annotations linking data to a time domain. As outlined in Section 2, a time domain permitted in our scenario is a structure defined as a linear ordering of a set of time instants [4].

Definition 1 (Time domain). *A* time domain *is a tuple* $(T, <)$*, where* T *is a nonempty set of elements called* time instants *and* $<$ *is an irreflexive, linear ordering on* T*.*

Some popularly considered structures satisfying this definition are based on sets of numbers, e.g. naturals $(\mathbb{N}, <)$, integers $(\mathbb{Z}, <)$, reals $(\mathbb{R}, <)$, with the ordering $<$ being interpreted as the usual *smaller-than* relation. By convention, we write $\leq$ to denote the reflexive closure of $<$.

Temporal annotations are based on the weak-interval time-stamping mechanism. Intuitively, a time-stamped ABox assertion $[t_1, t_2] : \alpha$ states that the axiom α is valid in all time instants within the interval $[t_1, t_2]$. Additionally, we allow special symbols $-\infty$ and $+\infty$ to represent possibly unbounded intervals.

Definition 2 (Temporal ABox). *Let* $(T, <)$ *be a time domain. A* temporal assertion *is an expression in one of the following forms:*

$$[t_1, t_2] : \alpha \ \mid \ [-\infty, t_1] : \alpha \ \mid \ [t_1, +\infty] : \alpha \ \mid \ [-\infty, +\infty] : \alpha$$

where α *is an ABox assertion and* $t_1, t_2 \in T$*. A* temporal ABox $\mathcal{A}_T$ *is a finite set of temporal ABox assertions. A* t-snapshot *of* $\mathcal{A}_T$*, for* $t \in T$*, is the smallest ABox* $\mathcal{A}_T(t)$ *containing all assertions* α*, for which any of the following conditions hold:*

$$\begin{aligned}
[t_1, t_2] : \alpha \in \mathcal{A} \quad & \textit{and} \quad t_1 \leq t \leq t_2, \\
[-\infty, t_1] : \alpha \in \mathcal{A} \quad & \textit{and} \quad t \leq t_1, \\
[t_1, +\infty] : \alpha \in \mathcal{A} \quad & \textit{and} \quad t_1 \leq t, \\
[-\infty, +\infty] : \alpha \in \mathcal{A}. &
\end{aligned}$$

The standard DL semantics is extended in a natural way by adding the temporal dimension and assigning a single DL interpretation to every time instant.

Definition 3 (Snapshot semantics). *Let* $(T, <)$ *be a time domain,* $\mathcal{T}$ *a TBox and* $\mathcal{A}_T$ *a temporal ABox. A* temporal interpretation *of* $\mathcal{T}$ *and* $\mathcal{A}_T$ *is a tuple* $\mathfrak{M} = (T, <, \mathcal{I})$*, where* $\mathcal{I}$ *is a function assigning to every* $t \in T$ *a DL interpretation* $\mathcal{I}(t) = (\Delta(t), \cdot^{\mathcal{I}(t)})$*. We say that* $\mathfrak{M}$ *is a* model *of* $\mathcal{T}$ *and* $\mathcal{A}_T$*, whenever* $\mathcal{I}(t)$ *is a model of* $\mathcal{T}$ *and* $\mathcal{A}_T(t)$*, for every* $t \in T$*.*

5 Temporal Query Language

In this section, we define and study a novel temporal query language $\mathcal{TQL}$, or strictly speaking, a family of such languages for querying temporal ABoxes *w.r.t.* standard TBoxes. At its core, our contribution should be seen as a general mechanism for constructing practical query formalisms, based on combinations of temporal logics with certain classes of first-order queries over DLs. This mechanism can be shortly described as follows. Consider a temporal logic $\mathcal{TL}$ and a class of queries $\mathcal{Q}$. We aim at identifying a fragment of *temporal first-order logic*, based on the operators of $\mathcal{TL}$, whose first-order component coincides with the class $\mathcal{Q}$. To this end, we follow the well-studied methodology of *temporalization of logic systems*, introduced by Finger and Gabbay [6]. Essentially, $\mathcal{TQL}$ is defined as the set of all $\mathcal{TL}$-formulas whose atomic subformulas are substituted with $\mathcal{Q}$-queries. The central motivation behind such a construction is to enable decoupling the problem of answering embedded $\mathcal{Q}$-queries from reasoning in $\mathcal{TL}$, which can be both efficiently addressed by existing, specialized tools. As it turns out, some potential interactions between the Boolean operators of $\mathcal{TL}$-formulas with those of $\mathcal{Q}$-queries make such decoupling still impossible in general. Hence, as a solution, we advocate a controlled use of epistemic semantics for interpreting the embedded $\mathcal{Q}$-queries, along the lines proposed by Calvanese et al. [16]. This, as we argue in Section 5.2, leads to a desirable theoretical and practical characterization of $\mathcal{TQL}$.

In our scenario, we focus on the class of conjunctive queries, as the most popular type of queries studied in the context of DLs. As the baseline temporal language we consider *first-order monadic logic of orders* (FOMLO), which is known to be expressively complete w.r.t. all linear orders [17], and thus subsumes a number of most popular linear temporal logics, including the prominent Propositional Linear Temporal Logic (PLTL).

5.1 Syntax and Semantics

To keep the design of $\mathcal{TQL}$ possibly modular, and yet maximally generic, we first introduce a mechanism of abbreviating the temporal components of the queries into customary *temporal connectives.* Those connectives, defined analogical to Chomicki and Toman [18], are used as templates to be instantiated with particular CQs and further combined by means of Boolean operators. The syntax of FOMLO is based on a countably infinite set $\mathsf{T_V}$ of time instant variables, such that $\mathsf{T_V} \cap \mathsf{N_V} = \emptyset$, one binary predicate $<$ and a countably infinite set $\mathsf{P_V}$ of monadic predicate variables.

Definition 4 (Temporal connectives: syntax). *A ϕ-formula is an expression constructed according to the grammar:*

$$\phi \ ::= \ u < v \ \mid \ \neg\phi \ \mid \ \phi \wedge \phi \ \mid \ \forall x.\phi \ \mid \ X(u)$$

where $u, v \in T \cup \mathsf{T_V}$, $x \in \mathsf{T_V}$ and $X \in \mathsf{P_V}$. An n-ary temporal connective *is a ϕ-formula containing $k \geq 0$ free variables $x_1, \ldots, x_k \in \mathsf{T_V}$, called the* temporal answer variables, *and $n \geq 0$ predicate variables $X_1, \ldots, X_n \in \mathsf{P_V}$. We define Ω to be a finite set of temporal connectives, where each connective $\omega \in \Omega$ is given via a definition consisting of a name $\omega(\vec{x})(\vec{X})$, with $\vec{x} = x_1 \ldots, x_k$ and $\vec{X} = X_1, \ldots, X_n$, and a (definitional) ϕ-formula ω^*.*

Intuitively, the predicate variables are place holders for CQs, which we add in the next step. The temporal answer variables range over time instants, which are explicitly represented in the answers to temporal queries.[1] A small sample of possible temporal connectives is given in Figure 1. Note, that we use some common abbreviations, such as $\exists, \rightarrow, \leq$, as well as compositions $x_1 < x_2 < x_3$, whose meaning is as expected.

$$\begin{array}{rcl}
\textbf{always}(X_1) & \triangleq & \forall x_1.X_1(x_1) \\
\textbf{sometime}(X_1) & \triangleq & \exists x_1.X_1(x_1) \\
\textbf{in}(x_1)(X_1) & \triangleq & X_1(x_1) \\
\textbf{after}(x_1, x_2) & \triangleq & x_2 < x_1 \\
\textbf{during-interval}(x_1, x_2)(X_1) & \triangleq & x_1 \leq x_2 \wedge \forall x_3.(x_1 \leq x_3 \leq x_2 \rightarrow X_1(x_3)) \\
\textbf{in-since}(x_1)(X_1, X_2) & \triangleq & \exists x_2.(x_2 < x_1 \wedge X_2(x_2) \wedge \forall x_3.(x_2 < x_3 \leq \\
 & & x_1 \rightarrow X_1(x_3))) \\
\textbf{in-until}(x_1)(X_1, X_2) & \triangleq & \exists x_2.(x_1 < x_2 \wedge X_2(x_2) \wedge \forall x_3.(x_1 \leq x_3 < \\
 & & x_2 \rightarrow X_1(x_3)))
\end{array}$$

Fig. 1. Examples of temporal connectives

The satisfaction relation is defined in terms of the standard first-order semantics, *modulo* an extra condition warranting the satisfaction of predicate variables.

Definition 5 (T-substitution). *For a time domain $(T, <)$, a T-substitution is a mapping $\pi : T \cup \mathsf{T_V} \mapsto T$ such that $\pi(t) = t$ for every $t \in T$.*

Definition 6 (Temporal connectives: satisfaction relation). *Let $\mathfrak{M} = (T, <, \mathcal{I})$ be a temporal model and $\omega \in \Omega$ a temporal connective with the definitional formula ω^*. For a T-substitution π, the satisfaction relation $\mathfrak{M}, \pi \models \omega^*$ is defined inductively as follows:*

[1] In practice, the range of answer variables might need to be further restricted in order to finitize the number of possible answers. In the context of temporal databases, it is common to consider only time instants that are explicitly mentioned in the data (in our case: temporal ABox). This, however, might require certain normalization of the used time-stamps — a problem which we do not address here.

– $\mathfrak{M}, \pi \models u < v$ iff $\pi(u) < \pi(v)$,
– $\mathfrak{M}, \pi \models \neg\phi$ iff $\mathfrak{M}, \pi \not\models \neg\phi$,
– $\mathfrak{M}, \pi \models \phi \wedge \psi$ iff $\mathfrak{M}, \pi \models \phi$ *and* $\mathfrak{M}, \pi \models \psi$,
– $\mathfrak{M}, \pi \models \forall x.\phi$ iff *for every* $t \in T$ *it is the case that* $\mathfrak{M}, \pi[x \mapsto t] \models \phi$,
– $\mathfrak{M}, \pi \models X_i(u)$ iff *the CQ associated with* X_i *is satisfied in* $\mathfrak{M}, \pi(u)$ *(see Definition 8).*

where $\pi[x \mapsto t]$ *denotes a* T*-substitution exactly as* π *except for that we fix* $\pi(x) = t$.

Finally, we define the syntax and semantics of the temporal query language.

Definition 7 (Temporal query language: syntax). *The* temporal query language $\mathcal{TQL}$ *is induced by the following grammar:*

$$\psi \ ::= \ \omega(\vec{x})(q_1(\vec{y}_1), \ldots, q_n(\vec{y}_n)) \ \mid \ \neg\psi \ \mid \ \psi \wedge \psi$$

where $\omega \in \Omega$ *is an* n*-ary temporal connective with temporal answer variables* $\vec{x}$*, and every* $q_i(\vec{y}_i)$ *is a CQ with answer variables* $\vec{y}_i$*, for* $1 \leq i \leq n$*. We write* $\psi(\vec{x}, \vec{y})$*, to denote a* $\mathcal{TQL}$ *query* ψ *with temporal answer variables* $\vec{x}$ *and CQ answer variables* $\vec{y}$*.*

An answer to a $\mathcal{TQL}$ query is a pair of sequences of time instants from T and individual names from $\mathsf{N_I}$, which substituted for the respective temporal and CQ answer variables must satisfy the query. The answer variables of both types can be shared among different CQs and temporal connectives occurring in the query, thus facilitating descriptions of complex dependencies between temporal data (*cf.* Example 1). To formally introduce the semantics of $\mathcal{TQL}$ queries, we first fix useful notation for handling subsequences of CQ answers. Let $\vec{y} = y_1, \ldots, y_k$ be a sequence of answer variables and $\vec{a} = a_1, \ldots, a_k$ a corresponding sequence of individual names. For an arbitrary subsequence $\vec{y}' \subseteq \vec{y}$, i.e. a subset of $\vec{y}$ preserving the ordering, we write $\vec{a}|_{\vec{y}'}$ to denote the subsequence of $\vec{a}$ such that for every $1 \leq i \leq k$, a_i occurs in $\vec{a}|_{\vec{y}'}$ *iff* y_i occurs in $\vec{y}'$.

Definition 8 (Temporal query language: semantics). *Let* $\psi(\vec{x}, \vec{y})$ *be a* $\mathcal{TQL}$ *query, with* $\vec{x} = x_1, \ldots, x_k$ *and* $\vec{y} = y_1, \ldots, y_l$*. For a pair of sequences* $(\vec{t}, \vec{a})$*, where* $\vec{t} = t_1, \ldots, t_k \in T$ *and* $\vec{a} = a_1, \ldots, a_l \in \mathsf{N_I}$*, a* $(\vec{t}, \vec{a})$*-match to* ψ *in a model* $\mathfrak{M} = (T, <, \mathcal{I})$ *is a* T*-substitution* π*, such that* $\pi(x_i) = t_i$*, for every* $1 \leq i \leq k$*, and* $\mathfrak{M}, \pi \models_{\vec{a}} \psi$*, where the satisfaction relation* $\models_{\vec{a}}$ *is defined inductively as follows:*

– $\mathfrak{M}, \pi \models_{\vec{a}} \omega(\vec{x_i})(q_1(\vec{y}_1), \ldots, q_n(\vec{y}_n))$ iff $\mathfrak{M}, \pi \models \omega^*$ *(see Definition 6), where for every* $1 \leq i \leq n$ *and any* T*-substitution* π' *we set:*

$$\mathfrak{M}, \pi' \models X_i(\pi'(u)) \text{ iff } \mathcal{I}(\pi'(u)) \models q_i[\vec{a}|_{\vec{y}_i}], \qquad (\dagger)$$

– $\mathfrak{M}, \pi \models_{\vec{a}} \neg\varphi$ iff $\mathfrak{M}, \pi \not\models_{\vec{a}} \varphi$,
– $\mathfrak{M}, \pi \models_{\vec{a}} \varphi \wedge \psi$ iff $\mathfrak{M}, \pi \models_{\vec{a}} \varphi$ *and* $\mathfrak{M}, \pi \models_{\vec{a}} \psi$.

We write $\mathfrak{M} \models \psi[\vec{t}, \vec{a}]$ *whenever there exists a* $(\vec{t}, \vec{a})$*-match to* ψ *in* $\mathfrak{M}$ *and* $\mathcal{T}, \mathcal{A}_T \models \psi[\vec{t}, \vec{a}]$*, whenever there exists a* $(\vec{t}, \vec{a})$*-match to* ψ *in every model of* $\mathcal{T}$ *and* $\mathcal{A}_T$*. In the latter case* $\vec{t}, \vec{a}$ *is called a* certain answer *to* ψ *w.r.t.* $\mathcal{T}, \mathcal{A}_T$*.*

Example 1. We formulate a $\mathcal{TQL}$ query $\psi(x_1, x_2, y)$ requesting all patients y who have been ever diagnosed with some allergy, at some point x_1 were administered a new drug, and at some point x_2, after x_1, had symptoms of an allergic reaction. The precise meaning of the temporal connectives used in the query is as defined in Figure 1.

$$\begin{aligned}\psi(x_1, x_2, y) \ ::= \ & \mathbf{sometime}(\exists x.(\mathsf{Patient}(y) \land \mathsf{diagnosedWith}(y, x) \land \mathsf{Allergy}(x))) \\ & \land \mathbf{in}(x_1)(\exists x.(\mathsf{administered}(y, x) \land \mathsf{NewDrug}(x))) \land \\ & \land \mathbf{after}(x_2, x_1) \\ & \land \mathbf{in}(x_2)(\exists x.(\mathsf{hasSymptom}(y, x) \land \mathsf{AllergicReaction}(x)))\end{aligned}$$

Consider the TBox $\mathcal{T}$ containing axioms:

$$\begin{aligned}&\mathsf{AllergicPatient} \sqsubseteq \mathsf{Patient} \sqcap \exists \mathsf{diagnosedWith}.\mathsf{Allergy} \\ &\mathsf{TestPatient} \sqsubseteq \mathsf{Patient} \sqcap \exists \mathsf{administered}.\mathsf{NewDrug}\end{aligned}$$

and the temporal ABox $\mathcal{A}$ containing time-stamped assertions:

$[1, 5] : \mathsf{AllergicPatient}(\mathit{john})$	$[2, 3] : \mathsf{Patient}(\mathsf{carl})$
$[1, 2] : \mathsf{hasSymptom}(\mathit{john}, \mathsf{id1})$	$[1, 2] : \mathsf{hasSymptom}(\mathit{carl}, \mathsf{id3})$
$[2, 2] : \mathsf{AllergicReaction}(\mathsf{id1})$	$[2, 2] : \mathsf{AllergicReaction}(\mathsf{id3})$
$[4, 5] : \mathsf{TestPatient}(\mathsf{john})$	$[2, 3] : \mathsf{diagnosedWith}(\mathsf{carl}, \mathsf{id4})$
$[6, 6] : \mathsf{hasSymptom}(\mathsf{john}, \mathsf{id2})$	$[2, 3] : \mathsf{Allergy}(\mathsf{id4})$
$[6, 6] : \mathsf{AllergicReaction}(\mathsf{id2})$	$[5, 5] : \mathsf{TestPatient}(\mathsf{carl})$

Given the time domain of natural numbers $(\mathbb{N}, <)$ there are two certain answers to the query $\psi(x_1, x_2, y)$, namely: $(4, 6, \mathsf{john})$ and $(5, 6, \mathsf{john})$.

5.2 Practical Query Answering

As it turns out, under the introduced semantics the expressive power of $\mathcal{TQL}$ is still too high to provide reasonable guarantees for the worst-case complexity of temporal query answering, and for the possibility of reusing the existing query answering techniques and tools. The level of interaction between the Boolean operators of the temporal language with CQs is sufficient to enable encoding Boolean combinations of conjunctive queries (BCCQs) over DLs, i.e. formulas induced by the grammar:

$$\varphi \ ::= \ q \ | \ \neg\varphi \ | \ \varphi \land \psi.$$

where q is a CQ. The decidability of BCCQs answering over DLs is, to the best of our knowledge, an open problem. Some of the largest classes of queries whose decidability has been studied so far are in fact unions (disjunctions) of CQs [1] and their syntactic generalization — positive existential queries [19]. In order to render query answering in $\mathcal{TQL}$ practical, we therefore need to employ some means of constraining the language. Quite a trivial fix is to tame the expressiveness of CQs, for instance by considering only CQs without existentially bounded variables — thus a variation of instance queries. In such case, the query answering can be reduced to reasoning with temporalized ABox axioms w.r.t. global TBox. As explained in [13], for a temporal logic coinciding with PLTL and an arbitrary DL with at least PSPACE-hard satisfiability problem, the complexity of the task remains the same as for the satisfiability in the underlying DL.

A much more interesting way of alleviating the problem of handling BCCQs, however, is to restrict the level of interaction between the operators of the temporal language with those of the embedded CQs, without reducing the expressiveness of the queries. To this end we propose to apply a limited form of the Closed World Assumption (CWA). Although essentially incompatible with the open-world semantics of DLs, a controlled use of CWA is claimed to be justified and beneficial in various application scenarios related to OBDA and Semantic Web reasoning [20,16,21]. In our case, we are interested in restricting $\mathcal{TQL}$ in a way that would enable answering individual CQs under the standard semantics, but at the same time, interpreting negation in front of CQs as Negation-As-Failure, and reducing the problem of answering BCCQs to Boolean operations over the certain answers to CQs. A clean and straightforward method of achieving this effect, advocated and studied in depth in [16], is to bind every occurrence of a CQ in a $\mathcal{TQL}$ query with the *autoepistemic* **K**-*operator*. Essentially, the operator **K** enforces that a bounded CQ is satisfied in a model, for a given answer, only if this answer is *known* to be certain, or formally:

$$\mathcal{I} \models \mathbf{K}q[\vec{a}] \textit{ iff } \mathcal{T}, \mathcal{A} \models q[\vec{a}]$$

where $\mathcal{I}$ is a model of $\mathcal{T}$ and $\mathcal{A}$. This immediately entails the requested reductions of a limited, closed-world flavor:

$$\begin{gathered}\mathcal{T}, \mathcal{A} \models \mathbf{K}q[\vec{a}] \textit{ iff } \mathcal{T}, \mathcal{A} \models q[\vec{a}] \\ \mathcal{T}, \mathcal{A} \models \neg\mathbf{K}q[\vec{a}] \textit{ iff } \mathcal{T}, \mathcal{A} \not\models q[\vec{a}] \\ \mathcal{T}, \mathcal{A} \models \mathbf{K}q_1[\vec{a}] \vee \mathbf{K}q_2[\vec{b}] \textit{ iff } \mathcal{T}, \mathcal{A} \models q_1[\vec{a}] \text{ or } \mathcal{T}, \mathcal{A} \models q_2[\vec{b}]\end{gathered}$$

Observe that the set of certain answers to a single CQ is invariant to the possible application of the **K**-operator in front of the query. Thus, the closed-world reasoning, emerging only on the level of Boolean combinations of CQs, does not affect the basic assumption of possible incompleteness of data, inherent to the OBDA paradigm.

Eventually, by replacing every q in $\mathcal{TQL}$ queries with $\mathbf{K}q$, or simply by interpreting it *as if it was bounded* by **K** (as we do below), we obtain the desired, well-behaved temporal query language.

Definition 9 ($\mathcal{TQL}$ semantics with epistemic interpretation of CQs). *The semantics of $\mathcal{TQL}$ with* epistemic interpretation *of embedded CQs is exactly the same as in Definition 8, modulo the replacement of the condition* (†) *with the following one:*

$$\mathfrak{M}, \pi' \models X_i(\pi'(u)) \text{ iff } \mathcal{T}, \mathcal{A}_T(\pi'(u)) \models q_i[\vec{a}|_{\vec{y}_i}]. \qquad (‡)$$

To witness the difference between evaluating $\mathcal{TQL}$ queries under the two compared semantics, consider an example involving TBox $\mathcal{T} = \{\mathsf{A} \sqsubseteq \neg\mathsf{D},\ \mathsf{B}\sqcap\mathsf{C} \sqsubseteq \mathsf{D}\}$, temporal ABox $\mathcal{A} = \{[1,1] : \mathsf{A}(\mathsf{a}),\ [1,2] : \mathsf{B}(\mathsf{a}),\ [2,3] : \mathsf{C}(\mathsf{a})\}$ and query $\psi(x,y)\ ::= \neg\mathbf{in}(x)(\mathsf{D}(y))$. Under the original semantics, presented in Definition 8, the query returns a unique certain answer $(1,\mathsf{a})$. On the other hand, by enforcing the epistemic interpretation of the embedded CQ $q(y) ::= \mathsf{D}(y)$, as argued above, and setting the time domain of natural numbers, we obtain an infinite set of certain answers $\{(t,\mathsf{a}) \mid 2 \neq t \in \mathbb{N}\}$.

Notably, the condition $\mathcal{T}, \mathcal{A}_T(\pi'(u)) \models q_i[\vec{a}|_{\vec{y}_i}]$ in (‡) is an instance of the standard CQ answering problem. Moreover, it is the only point in the revised semantics where DL reasoning is intertwined with reasoning over the temporal language. What follows, is that the most natural algorithm answering $\mathcal{TQL}$ queries can be constructed by augmenting any standard decision procedure for the satisfiability in the temporal language with an oracle answering CQs over the designated snapshots of the ABox *w.r.t.* the TBox. As the decision problem in FOMLO is known to be PSpace-complete [17], we thus obtain a result on the combined complexity of answering $\mathcal{TQL}$ queries.

Theorem 1 (Combined complexity of $\mathcal{TQL}$ query answering). *Let ψ be a $\mathcal{TQL}$ query over a temporal ABox $\mathcal{A}_T$ w.r.t. TBox $\mathcal{T}$, where ABox and TBox axioms are expressed in a DL language $\mathcal{L}$. The combined complexity of deciding $\mathcal{T}, \mathcal{A}_T \models \psi[\vec{t}, \vec{a}]$, for a pair of sequences $\vec{t}, \vec{a}$, under the epistemic interpretation of the CQs embedded in ψ, is* $\text{PSpace}^{QA(\mathcal{L})}$*-complete, where $QA(\mathcal{L})$ is an oracle answering CQs in $\mathcal{L}$.*

This seemingly unsurprising result has some significant theoretical and practical implications. On the theoretical side, it guarantees that answering $\mathcal{TQL}$ queries under the epistemic interpretation of CQs remains decidable, as long as answering CQs over the respective DLs is decidable. Moreover, it establishes a bridge for an immediate transfer of the combined complexity results. For instance, when $\mathcal{L} = \mathcal{ALC}$, answering $\mathcal{TQL}$ queries is $\text{PSpace}^{\text{ExpTime}}$-complete, thus effectively ExpTime-complete, as the combined complexity of CQ answering in $\mathcal{ALC}$ is ExpTime-complete [22]. Analogically, for $\mathcal{L} = \mathcal{SHIQ}$, the problem is $\text{PSpace}^{\text{2ExpTime}}$-complete, and effectively 2ExpTime-complete. In general, the combined complexity of answering $\mathcal{TQL}$ queries for an arbitrary DL $\mathcal{L}$ is equal to the complexity of answering CQs in $\mathcal{L}$, provided that the latter problem is at least PSpace-hard. This observation naturally generalizes over query languages based on combinations of FOMLO with arbitrary classes of first-order queries. Whenever the complexity of answering $\mathcal{Q}$-queries over $\mathcal{L}$, for a given query class $\mathcal{Q}$ and a DL $\mathcal{L}$, is at least PSpace-hard then answering the resulting temporal queries over $\mathcal{L}$ remains in the same complexity class. Otherwise it is PSpace-complete. This demonstrates that the temporalization technique employed here yields computationally cheap, yet expressive, temporal query languages over temporal ABoxes. In fact, temporalization of query languages for expressive DLs, subsuming $\mathcal{ALC}$, comes for free.

From the practical perspective, the restricted interaction between the temporal component and CQs, reflected in Theorem 1, promises relatively straightforward implementations of $\mathcal{TQL}$ query engines based on existing technologies, e.g.: temporal theorem provers and CQ answering tools. Roughly, to determine whether a candidate answer to a query ψ is certain for $\mathcal{T}, \mathcal{A}_T$, it suffices to check whether the direct rendering of ψ into FOMLO is satisfiable, where every CQ embedded in ψ is seen as a predicate variable, whose truth-value in a given time instant is determined by a call to an external CQ answering tool over the respective snapshot of $\mathcal{A}_T$ *w.r.t.* $\mathcal{T}$.[2]

Some further interesting prospects concern answering $\mathcal{TQL}$ queries over the DL-Lite family of DLs, enjoying the FO-rewritability property [2]. It is known that CQ answering in DL-Lites can be carried out efficiently using highly optimized RDBMSs. In a nutshell, for TBox $\mathcal{T}$, ABox $\mathcal{A}$ and a CQ q, one can always find a first-order query $q^{\mathcal{T}}$, such that for every $\vec{a}$ it is the case that $\mathcal{T}, \mathcal{A} \models q[\vec{a}]$ *iff* $\mathcal{A} \models q^{\mathcal{T}}[\vec{a}]$, where the latter problem can be solved directly in an RDBMS. Clearly, an analogical approach should enable rewriting a $\mathcal{TQL}$ query over $\mathcal{T}, \mathcal{A}_T$ into a temporal first-order formula, which could be then efficiently encoded and evaluated as a TSQL2 query [5] over a temporal database $\mathcal{A}_T$. Although providing precise definition of such a translation and proving its correctness is left as future work, we expect it to be straightforward given that every $\mathcal{TQL}$ query corresponds to a temporal formula with embedded CQs, where each CQ q can be replaced with the corresponding first-order formula $q^{\mathcal{T}}$ obtained by means of established FO-rewritability techniques.

6 Discussion and Outlook

The design of the language $\mathcal{TQL}$ follows closely the principles of query languages for temporal databases, as outlined in e.g. [18]. In the general TDB setup, the query component $\mathcal{Q}$ is based on the full first-order logic, while temporal operators defined in FOMLO can be nested within each other. Hence, the resulting language is expressively equivalent to the temporal first-order logic. In $\mathcal{TQL}$, we are deliberately constraining the query component and disallow nesting of operators in order to enable practical decoupling of the DL-level from the temporal-level reasoning. In the context of the SW, similar approaches have been proposed to deal with time-stamped RDF data [10,12] and OWL axioms [12]. Both contributions, however, lack the generality of our proposal. The temporal component of the query languages is in both cases highly restricted in order to ensure finite answer sets. In particular, Motik [12] introduces a specially fixed number of most practical temporal operators that can be combined with the data-level queries. All these can be easily restated in FOMLO, and so the language of [12] can be easily defined using the $\mathcal{TQL}$ mechanism.

An interesting open challenge is a potential integration of the framework with two orthogonal approaches to managing temporal information in the context of DLs, mentioned in Section 2, i.e. SW temporal vocabularies and temporal DLs.

[2] For time domains based on natural numbers and integers, FOMLO formulas can be translated into PLTL [17], and thus decided using off-shelf PLTL provers, such as listed in `http://www.cs.man.ac.uk/~schmidt/tools/`. CQ answering in selected DLs is supported by such systems as QuOnto, REQUIEM, Presto.

As argued below, our choice of standard temporal semantics and logic-based query formalism, renders such prospects quite realistic. A remarkable feature, shared by the two potential extensions, is that unlike the current framework, they aim to support reasoning with incomplete data, where the incompleteness occurs in the temporal dimension. Such characteristic might be conceptually attractive considering the open-world philosophy of DLs.

Supporting Temporal Vocabularies and Semantic Annotations. The practice of representing and reasoning with temporal information on the Semantic Web, for instance in the field of health care support [23], suggests that the presented data model and query language might not be sufficiently flexible for real-life applications. In particular, we are incapable of:

– directly expressing typical temporal patterns occurring in temporal queries and constraints used in clinical applications, such as:
  - Visit 17 must occur *at least 1 week but no later than 4 weeks after the end of 2003 ragweed season.*
  - Administer Rapamune *1 week from* Visit 0 *daily for 84 days.*

– supporting semantic annotations whose meaning could be described in terms of rich temporal vocabularies, e.g. for a temporal assertion $\tau : \alpha$:
  - τ is a time interval from 15.05.2005 until some day on 06.2005,
  - τ is a periodic interval, of duration 5 hours, recurring 5 times every 10 hours, starting some time on 12.05.2008.

In practice, such functionalities are supported by ad-hoc architectures, which retrieve temporal information encoded in OWL-based time ontologies and process it with application-specific tools, thus sacrificing some theoretical rigor and formal transparency of provided inferences [9]. Within our framework, a natural solution to this problem is to extend the temporal language with additional predicates (e.g. involving periodicity constraints [24]) to enable ontological-style axiomatization of the background knowledge about the underlying time domain *per se*, e.g. the Gregorian calendar. Such predicates could be then meaningfully used and reasoned with on the query- and annotation-level. Such a philosophy motivates to a great extent the framework proposed by Zimmerman et al. [11]. There, however, the annotation language is a non-standard, task-specific formalism, which cannot be directly translated into temporal logics or OWL.

Integration with TDLs. The framework studied in this paper is focused on querying temporal data with respect to a fixed, time-invariant terminology. A natural extension to this approach is to introduce means of querying temporal ABoxes in presence of temporal constraints occurring on the intensional, terminological level. Temporal DLs are a family of two-dimensional DLs, developed intensively in the recent years [13], intended specifically for the representation of this kind of terminologies. By allowing operators of temporal logics to occur in DL concepts, TDLs enable, for instance, to express the following axiom:

$$\mathsf{Patient} \sqcap \exists \mathsf{diagnosedWith.Allergy} \sqsubseteq \mathsf{AllergicPatient}\ \mathcal{U}\ \forall \mathsf{diagnosedWith}.\neg \mathsf{Allergy}$$

The axiom states that whenever a patient is diagnosed with an allergy, she should be considered an allergic patient until ($\mathcal{U}$) she is diagnosed with no allergies. Interestingly, TDL TBoxes are interpreted over the same type of semantic structures as used in our framework, i.e. tuples $\mathfrak{M} = (T, <, \mathcal{I})$. This means, that from the formal perspective integration of $\mathcal{TQL}$ with TDLs can be achieved seamlessly. Obviously, query answering in such setting should likely be computationally more expensive, considering that already the satisfiability problem in TDLs is usually harder than in the underlying DLs. So far the only query language for TDLs has been proposed by Artale et al. [14]. Differently than here, the queries are defined as unions of CQs, where the atomic predicates can be possibly preceded by temporal operators. As a consequence, the reuse of existing CQ answering techniques is not directly possible within this approach.

7 Conclusions

We believe that the framework proposed in this paper marks a first promising step towards establishing a generic approach to representing and querying temporal data under DL ontologies. Naturally, a number of important problems, which we merely touched upon, are left open to future research. Among others, it is critical to conduct a systematic study of possible ways of restricting the $\mathcal{TQL}$-like languages, in order to turn the query answering problem feasible in practice. The use of epistemic semantics, suggested here, is only one of possible options. Other might involve more fine-grained constraints on the expressiveness of the temporal component, the first-order component or both. We also advocate a study of possible extensions of the framework towards integration with temporal DLs and rich SW temporal vocabularies.

References

1. Glimm, B., Horrocks, I., Lutz, C., Sattler, U.: Conjunctive query answering for the description logic SHIQ. Journal of Artificial Intelligence Research (2008)
2. 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. of Automated Reasoning 39, 385–429 (2007)
3. Chomicki, J., Toman, D.: Temporal Databases. In: Handbook of Temporal Reasoning in Artificial Intelligence (Foundations of Artificial Intelligence), pp. 429–468. Elsevier Science Inc. (2005)
4. Montanari, A., Chomicki, J.: Time domain. In: Encyclopedia of Database Systems, pp. 3103–3107 (2009)
5. Böhlen, M.H., Chomicki, J., Snodgrass, R.T., Toman, D.: Querying TSQL2 Databases with Temporal Logic. In: Apers, P.M.G., Bouzeghoub, M., Gardarin, G. (eds.) EDBT 1996. LNCS, vol. 1057, pp. 325–341. Springer, Heidelberg (1996)
6. Finger, M., Gabbay, D.M.: Adding a temporal dimension to a logic system. Journal of Logic, Language and Information 1(3), 203–233 (1992)
7. Hobbs, J.R., Pan, F.: An ontology of time for the semantic web. ACM Trans. Asian Lang. Inf. Process. 3(1), 66–85 (2004)

8. Batsakis, S., Stravoskoufos, K., Petrakis, E.G.M.: Temporal Reasoning for Supporting Temporal Queries in OWL 2.0. In: König, A., Dengel, A., Hinkelmann, K., Kise, K., Howlett, R.J., Jain, L.C. (eds.) KES 2011, Part I. LNCS, vol. 6881, pp. 558–567. Springer, Heidelberg (2011)
9. O'Connor, M.J., Das, A.K.: A method for representing and querying temporal information in OWL. Biomedical Engineering Systems and Technologies (Selected Papers), 97–110 (2011)
10. Gutierrez, C., Hurtado, C.A., Vaisman, A.A.: Introducing Time into RDF. IEEE Trans. Knowl. Data Eng. 19(2), 207–218 (2007)
11. Zimmermann, A., Lopes, N., Polleres, A., Straccia, U.: A general framework for representing, reasoning and querying with annotated Semantic Web data. Web Semantics: Science, Services and Agents on the World Wide Web 11, 72–95 (2012)
12. Motik, B.: Representing and querying validity time in RDF and OWL: A logic-based approach. Web Semantics: Science, Services and Agents on the World Wide Web 12-13, 3–21 (2012)
13. Lutz, C., Wolter, F., Zakharyaschev, M.: Temporal description logics: A survey. In: Proc. of TIME 2008 (2008)
14. Artale, A., Franconi, E., Wolter, F., Zakharyaschev, M.: A Temporal Description Logic for Reasoning over Conceptual Schemas and Queries. In: Flesca, S., Greco, S., Leone, N., Ianni, G. (eds.) JELIA 2002. LNCS (LNAI), vol. 2424, Springer, Heidelberg (2002)
15. Baader, F., Calvanese, D., Mcguinness, D.L., Nardi, D., Patel-Schneider, P.F.: The description logic handbook: theory, implementation, and applications. Cambridge University Press (2003)
16. Calvanese, D., De Giacomo, G., Lembo, D., Lenzerini, M., Rosati, R.: Eql-lite: Effective first-order query processing in description logics. In: Proc. of IJCAI 2007 (2007)
17. Reynolds, M.: The complexity of decision problems for linear temporal logics. Journal of Studies in Logic 3, 1917 (2010)
18. Chomicki, J., Toman, D.: Temporal logic in information systems. In: Chomicki, J., Saake, G. (eds.) Logics for Databases and Information Systems, pp. 31–70. Kluwer (1998)
19. Ortiz, M., Calvanese, D., Eiter, T.: Data complexity of query answering in expressive description logics via tableaux. J. of Automated Reasoning 41, 61–98 (2008)
20. Grimm, S., Motik, B.: Closed world reasoning in the semantic web through epistemic operators. In: Second International Workshop on OWL: Experiences and Directions (OWLED 2006) (2005)
21. Lutz, C., Seylan, I., Wolter, F.: Mixing open and closed world assumption in ontology-based data access: Non-uniform data complexity. In: Proc. of the International Workshop on Description Logics, DL 2012 (2012)
22. Lutz, C.: The Complexity of Conjunctive Query Answering in Expressive Description Logics. In: Armando, A., Baumgartner, P., Dowek, G. (eds.) IJCAR 2008. LNCS (LNAI), vol. 5195, pp. 179–193. Springer, Heidelberg (2008)
23. Shankar, R.D., Martins, S.B., O'Connor, M.J., Das, A.K.: An ontological approach to representing and reasoning with temporal constraints in clinical trial protocols. In: HEALTHINF (1) (INSTICC - Institute for Systems and Technologies of Information, Control and Communication), pp. 87–93
24. Demri, S.: LTL over integer periodicity constraints. Theoretical Computer Science 360, 96–123 (2006)

Meta Programming with Answer Sets for Smart Spaces

Tomi Janhunen[1] and Vesa Luukkala[2]

[1] Aalto University
Department of Information and Computer Science
Tomi.Janhunen@aalto.fi
[2] Nokia Research Center
Vesa.Luukkala@nokia.com

Abstract. A smart space is an ecosystem of interacting computational objects embedded in some environment. The space seamlessly provides users with information and services using the best available resources. In this paper, the interoperability of heterogeneous objects participating in a smart space is enhanced by publishing their behavioral rules as RDF triples, i.e., in the same way as any other information in the space. This enables the use of answer-set programming (ASP) as the underlying paradigm for rule-based reasoning. The main idea of this paper is to apply meta programming techniques to reified ASP rules published in the smart space. Such techniques enable syntactic and semantic transformations of rules without essentially changing the underlying computational platform so that standard ASP tools can be used to implement inference over rules. These ideas are illustrated in several ways. In addition to basic meta evaluation tasks, we describe a meta grounder for ASP rules involving variables. Moreover, we demonstrate how the qualitative aspects of reasoning can be taken into account in our approach and how meta programming techniques are made available to users.

1 Introduction

The number of embedded communicating devices is growing constantly. Services and information provided by these devices benefit human users as well as other devices, all of which attempt to use the best resources available at a given time and a place. A *smart space* is an ecosystem of interacting computational objects embedded in some environment having mechanisms for the seamless use of information and resources. This can be seen as a realization of the *ubiquitous computing* vision [27].

To fully exploit the wide variety of heterogeneous devices existing today, the interoperability of the devices becomes an issue that must be addressed. In [16], four levels of interoperability are distinguished: machine-level, syntactic, semantic, and organizational interoperability. It is also argued that only the first two levels and a part of the third can be achieved by standardization activities. Semantic interoperability presumes the same understanding of the information by all participants whereas organizational interoperability insists that operations performed on the information produce consistent results understood by all participants. A partial solution for the semantic interoperability can be achieved by formalizing agreements [22] by domain-specific ontologies. The mechanisms for this are provided by Semantic Web [3] where the information is minimally described in terms of *resource description framework* (RDF)[1] triples. In addition, dedicated knowledge representation languages such as RDFS[2] and OWL[3] allow the specification of more complex structures on top of RDF triples.

One approach for implementing the combined vision of ubiquitous systems and Semantic Web is provided by Smart-M3,[4] an interoperability platform which allows devices to share and to access semantic information in RDF format through a single logical blackboard. The concept and the architecture of Smart-M3 are explained in [4] where two kinds of logical components are identified: a *semantic information broker* (SIB) and a number of independent *knowledge processors* (KP) exchanging RDF triples with the SIB by a dedicated protocol. There is no notion of a single application, but one is formed by the KPs which share an ontology and are able to produce and consume information. A new KP committed to the same ontology may contribute to the application by reusing the information in a new way or augmenting it with new information. This is partly enabled by the semistructured form of semantic web definitions and the self-describing nature of semantic information. There are several use case and application implementations on this architecture [9,14,15,18,23,24] which follow the approach described above and aim at semantic interoperability as described in [16].

Answer-set programming (ASP) [17,20,21] techniques have been applied when implementing a generic resource allocation and conflict resolution mechanism [19] for smart spaces. The use of ASP primitives provides a straightforward mapping of semantic concepts as first class components in rules. Furthermore, the required optimization, dynamic context constraint modeling, and extendibility are well-supported by the ASP methodology. Essentially, the approach of [19] defines an ontology for the required information and, in addition, rules specifying behavior. There is a good *impedance match* with handling of semantic information as already noted in [28]. In further work [2], the rule-based resource allocator is extended to work in fully scalable and distributed manner—allowing an arbitrary number of rule engines to be used.

Our vision is that the participants in a smart space publish information and concepts in a semantic format together with rules used to define new concepts. This enables the other participants to reason about the concepts and assumptions of others, thus increasing interoperability in the sense proposed in [13]. The new concepts described by the rules refer to information defined in ontologies which contain the agreement of the meaning of these concepts (semantic level interoperability). The rules contain the agreement of how new concepts and operations are defined using the existing semantic level concepts (organizational interoperability). Thus the meaning of information and intended operations over it become fully accessible in the smart space.

In this paper, we describe methodology to interpret and modify published rules. Eventually, rules are to be represented as any other information in a smart space, but for brevity we use a simple PROLOG-style notation, which can nevertheless easily be rendered using semantic web techniques. The rules are published in a reified format rather than plain text—allowing us to build ASP programs which reason and operate

[1] `http://www.w3.org/RDF/`
[2] `http://www.w3.org/TR/rdf-schema`
[3] `http://www.w3.org/2004/OWL/`
[4] `http://sourceforge.net/projects/smart-m3/`

on the rules themselves. We demonstrate this proof-of-concept firstly by a meta evaluation program which directly interprets the reified rules to produce the same answer sets as current tools would produce for the non-reified ones. Secondly, we explore the possibilities of modifying the semantics of reified rules by giving a syntactic translation as well as a meta-interpreter as alternative implementations to achieve the same effect. Furthermore, we examine possibilities of including qualitative elements, such as time stamps, to existing reified rules and show how to include them in the evaluation of rules. As there are several participants in the smart space, we may need this kind of mechanisms for managing the provenance and validity of rules. In addition, we explore the possibilities of isolating the interface between the grounding and solving phases of ASP in the smart space setting. For this reason, we present a meta-level grounder as a further application of the meta programming techniques addressed in this work.

The rest of this paper is organized as follows. First, in Section 2, we briefly review the main concepts of answer-set programming in the case of normal rules involving term variables. This forms the basic syntax for answer-set programs considered in this paper. The idea of reification is then presented in Section 3 and the respective representation of normal rules as syntax trees as well as sets of facts is then laid out. In Section 4, we illustrate the compatibility of our approach with the meta modelling framework of [10] but essentially in the variable-free case. Next, we demonstrate the potential of meta programming when it comes to changing the syntax and semantics of rules in Section 5. In Section 6, we turn our attention to rules with variables and show how meta programming techniques can be used to implement grounding—a basic step in ASP. Domain information plays an important role in this step and it is possible to query the smart space in question to determine variable domains. As further illustrated in Section 7, meta-level techniques can be used to implement reasoning when rules are annotated by tags which describe their qualitative properties. Such features may include time stamps (such as time of issue, date of expiry), provenance, and reliability. A brief account of related work is provided in Section 8 and Section 9 concludes the paper.

2 Answer-Set Programming

In this section, we present the syntax and answer-set semantics of normal programs with variables. This forms the basic fragment of answer-set programs supported by most ASP systems. The syntax is based on the primitives of first-order logic. Hence, *terms* are defined as either *constants*, denoted by a, b, $\ldots$, or *variables*, denoted by X, Y, etc. An *atom* is an expression of the form $p(t_1,\ldots,t_n)$ where p is a predicate symbol of arity n and $t_1,\ldots,t_n$ are terms. A *normal rule* is an expression of the form

$$\alpha \leftarrow \beta_1,\ldots,\beta_n, \mathbf{not}\ \gamma_1,\ldots,\mathbf{not}\ \gamma_m. \tag{1}$$

where α, $\beta_1,\ldots,\beta_n$, and $\gamma_1,\ldots,\gamma_m$ are atoms and "**not**" denotes *default negation*. The variables $X_1,\ldots,X_k$ appearing in a rule (1) are universally quantified so that the rule stands for any *instance* obtained by substituting $X_1,\ldots,X_k$ with constants $c_1,\ldots,c_k$. Given such a *substitution* $\sigma = [X_1/c_1,\ldots,X_k/c_k]$, we write $\alpha\sigma$ for an atom α where all occurrences of $X_1,\ldots,X_k$ in α have been substituted by the respective constants $c_1,\ldots,c_k$. The reading of (1) subject to σ is that the *head* $\alpha\sigma$ of the rule can be inferred if the *body* conditions of the rule are satisfied, i.e., the positive conditions $\beta_1\sigma, \dots, \beta_n\sigma$ can be inferred whereas none of the negative conditions $\gamma_1\sigma, \dots, \gamma_m\sigma$.

Example 1. Consider the following two rules which formalize an access policy:

$$\mathsf{denied}(C,S) \leftarrow \mathbf{not}\ \mathsf{granted}(C,S),\ \mathsf{customer}(C),\ \mathsf{service}(S). \tag{2}$$

$$\begin{aligned}\mathsf{granted}(C,S) \leftarrow\ &\mathsf{registered}(C),\ \mathsf{subscribed}(C,S),\\ &\mathsf{customer}(C),\ \mathsf{service}(S).\end{aligned} \tag{3}$$

The intuitive reading of the rules is as follows. The access to a service is denied by default and granted only for registered customers who have subscribed to the service. It is easy to add further reasons why the access should be granted. This nicely illustrates how default negation can be used to concisely encode exceptions. Given a particular customer and a service, denoted by constants j and g, the rule (2) instantiates to $\mathsf{denied}(j,g) \leftarrow \mathbf{not}\ \mathsf{granted}(j,g),\ \mathsf{customer}(j),\ \mathsf{service}(g)$ under $[C/j, S/g]$. ■

An answer-set *program* is a finite set of rules of the form (1). The *Herbrand universe* of P, denoted by $\mathrm{HU}(P)$, is a finite non-empty set of constants which contains all constants that appear in the rules of P. The *Herbrand base* of P, denoted by $\mathrm{HB}(P)$, consists all ground atoms of the form $p(c_1, \dots, c_n)$ where p is an n-ary predicate symbol appearing in P and $c_1, \dots, c_n$ are constants from $\mathrm{HU}(P)$. The ground instance of P, denoted by $\mathrm{Gnd}(P)$, consists of all ground instances of its rules (1) obtained by substitutions σ over $\mathrm{HU}(P)$. The *answer-set* semantics of rules is based on the idea of a *reduction*: Given a set $S \subseteq \mathrm{HB}(P)$, the reduced ground program $\mathrm{Gnd}(P)^S$ contains a *positive* rule $\alpha\sigma \leftarrow \beta_1\sigma, \dots, \beta_n\sigma$ for each rule (1) and for each substitution σ over $\mathrm{HU}(P)$ such that $\gamma_1\sigma \notin S, \dots, \gamma_m\sigma \notin S$. Given the set $R = \mathrm{Gnd}(P)^S$, we write $\mathrm{cl}(R)$ for the least subset of $\mathrm{HB}(P)$ *closed under* R, i.e., for each $\alpha\sigma \leftarrow \beta_1\sigma, \dots, \beta_n\sigma$ in R, if $\beta_1\sigma \in \mathrm{cl}(R), \dots, \beta_n\sigma \in \mathrm{cl}(R)$, then also $\alpha\sigma \in \mathrm{cl}(R)$.

Definition 1 (Answer sets). *Let P be a program. A set $S \subseteq \mathrm{HB}(P)$ is an answer set of P if and only if $S = \mathrm{cl}(\mathrm{Gnd}(P)^S)$.*

Example 2. Consider a program P based on the rules of Example 1 in the presence of *facts*[5] $\mathsf{customer}(j)$ and $\mathsf{service}(g)$. The unique answer set S of P is generated by the rule $\mathsf{denied}(j,g) \leftarrow \mathsf{customer}(j),\ \mathsf{service}(g)$ included in $\mathrm{cl}(\mathrm{Gnd}(P)^S)$ so that $S = \{\mathsf{customer}(j), \mathsf{service}(g), \mathsf{denied}(j,g)\}$. To infer $\mathsf{granted}(j)$, we would further need $\mathsf{registered}(j)$ and $\mathsf{subscribed}(j,g)$—excluding the preceding ground rule. ■

In a typical ASP system, the computation of answer sets for an input program P proceeds in two steps. The first is called *intelligent grounding* which tries to produce a reasonably small subset of $\mathrm{Gnd}(P)$ without affecting answer sets. The grounding step is typically implemented as a separate tool called a *grounder*. E.g., the instances of the rules (2) and (3) obtained by a substitution $\sigma = [C/j, S/j]$ are unnecessary because their positive body condition $\mathsf{service}(j)$ is false by default. [6] The second step is the actual *search* for answer sets natively implemented by an answer set *solver*.

[5] Facts are rules (1) which have an empty body ($n = m = 0$) and no variables.

[6] In practice, grounding presumes some notion of *safety* of a rule (1), e.g., all variables appearing in the head or in the negative body conditions must appear in the positive body conditions, too.

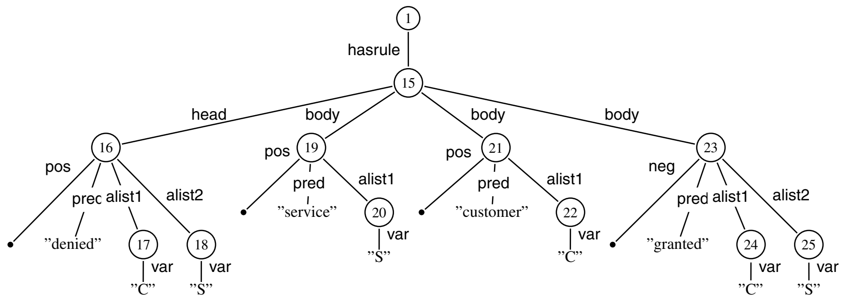


Fig. 1. Parse tree of a single rule

There are many syntactic extensions that have been proposed for normal rules (1). For instance, *cardinality* and *weight rules* [26] enable more concise encodings compared to normal rules. Moreover, the front-ends used to ground answer-set programs support further extensions some of which are also completely evaluated once the constants substituted for variables are known. For the purposes of this work, a useful extension is given by a *parameterized conjunction* $p(t_1, \ldots, t_n)\ :\ q(s_1, \ldots, s_m)$ which stands for the conjunction of all atoms $p(t_1, \ldots, t_n)$ subject to a *specifier* $q(s_1, \ldots, s_m)$ being true. The same notation extends to negative conditions and multiple specifiers. For instance, a rule $\mathsf{ok} \leftarrow \mathsf{colored}(X) : \mathsf{node}(X)$ states the everything is OK if all nodes (of a graph) have been colored. The nice feature of this construct is that the number of conjuncts in the rule body is determined dynamically at the time of grounding.

3 Reification

In this section, we describe the structure of abstract syntax trees which essentially provide the basis for the publication of rules. E.g., the rule (2) from Example 1 can be parsed into a syntax tree illustrated in Figure 1. Each node of the syntax tree has been assigned a unique identifier. For the purposes of this paper, it is sufficient to use integers as identifiers and, moreover, no similarity checks between nodes are performed when parsing a rule. This means, e.g., that nodes representing variable “C” within the same rule will get a different identifiers 17, 22, and 24 even though they refer to occurrences of the same variable in the rule. A root node, identified as the node 1 in the tree, collects all parsed rules under hasrule arcs which point out the individual rules of the program. Each rule has exactly one head-labeled subtree whereas arbitrarily many body-labeled subtrees are allowed. The latter represent the body conditions of the rule.

The declarative nature of ASP insists that the orders between rules and the body-labeled subtrees within rules are not significant. Thus we intentionally abstract on this and do not represent the original syntactic order in the tree. In contrast, the ordering information must be maintained for the arguments of predicates and explicitly encoded using an indexing scheme: the order of arguments is expressed using labels of the form $\mathsf{alist}n$ where n gives the position of an argument. The tree shown in Figure 2 can be *reified* as a set of facts. This step turns a rule into data accessible by meta-level rules—in terms of predicates whose names coincide with the labels used in syntax trees. The

```
hasrule(1, 15).          pos(19).                    neg(23).
rule(15).                body(15, 19).               body(15, 23).
pos(16).                 pred(19, "service").        pred(23, "granted").
head(15, 16).            var(20, "S").               var(24, "C").
pred(16, "denied").      alist(19, 1, 20).           alist(23, 1, 24).
var(17, "C").            pos(21).                    var(25, "S").
alist(16, 1, 17).        body(15, 21).               alist(23, 2, 25).
var(18, "S").            pred(21, "customer").
alist(16, 2, 18).        var(22, "C").
                         alist(21, 1, 22).
```

Fig. 2. Reified representation for a rule

predicates $\mathsf{head}(\cdot,\cdot)$ and $\mathsf{body}(\cdot,\cdot)$ associate the head and body conditions, respectively, with the rule. The predicate $\mathsf{pred}(\cdot,\cdot)$ is used to attach predicate symbols—treated as strings—for these conditions. The signs of these conditions are expressed using predicates $\mathsf{pos}(\cdot)$ and $\mathsf{neg}(\cdot)$. The predicate $\mathsf{alist}(\cdot,\cdot,\cdot)$ represents the argument list: the order of n arguments is encoded by selecting the second argument from the range $1 \ldots n$. The third argument refers to another node capturing either a constant or a variable expressed in terms of predicates $\mathsf{const}(\cdot,\cdot)$ and $\mathsf{var}(\cdot,\cdot)$.

As mentioned in the introduction, it is possible to create an ontology definition (in RDFS or OWL) corresponding to the concepts above in a straightforward way. This provides the basis for storing syntax trees, such as the one shown in Figure 1, in RDF.

4 Meta-level Evaluation of Rules

The goal of this section is to illustrate how the evaluation of rules can be formalized at meta level in ASP. For the ease of presentation, we start with the case of ground programs and predicates of arity 0, i.e., effectively *propositional* normal programs, and then address potential generalizations. For a propositional normal program P, the resulting reified program $\mathrm{Rfy}(P)$ does not contain any $\mathsf{var}(\cdot,\cdot)$ and $\mathsf{alist}(\cdot,\cdot,\cdot)$ entries.

The key idea is to introduce an auxiliary predicate $\mathsf{in}(H)$ for each rule head H. The intuitive reading of $\mathsf{in}(H)$ is that H belongs to an answer set being formalized. Then, according to the *meta evaluation* rule (4) below, the atom $\mathsf{in}(H)$ is derived if and only if $\mathsf{in}(P)$ is derivable for all positive body conditions P of the rule but $\mathsf{in}(N)$ is not derivable for any negative body conditions N. The two parameterized conjunctions in the rule body are expanded when grounding the program for a particular (reified) input program. In view of meta evaluation, this is a crucial feature of ASP grounders.

$$\begin{aligned} \mathsf{in}(H) \leftarrow\ & \mathsf{rule}(R),\ \mathsf{head}(R,H),\ \mathsf{in}(P) : \mathsf{body}(R,P) : \mathsf{pos}(P), \\ & \mathbf{not}\ \mathsf{in}(N) : \mathsf{body}(R,N) : \mathsf{neg}(N). \end{aligned} \tag{4}$$

In addition, we have to synchronize the (likely) multiple occurrences of atoms in the program. These are easily collected from the positive and negative occurrences recorded in the reified program using rules (5) and (6). Then any distinct occurrences which have the same predicate, or the *name* of the atom, can be identified as formalized by the rule (8). Finally, the rule (9) ensures that any two such atoms A_1 and A_2 equally hold.

$$\mathsf{atom}(A) \leftarrow \mathsf{pos}(A). \tag{5}$$
$$\mathsf{atom}(A) \leftarrow \mathsf{neg}(A). \tag{6}$$
$$\mathsf{equal}(A, A) \leftarrow \mathsf{atom}(A). \tag{7}$$
$$\begin{aligned}\mathsf{same}(A_1, A_2) \leftarrow\ & \mathsf{pred}(A_1, P),\ \mathsf{pred}(A_2, P),\\ & \mathsf{atom}(A_1),\ \mathsf{atom}(A_2),\ \mathbf{not}\ \mathsf{equal}(A_1, A_2).\end{aligned} \tag{8}$$
$$\mathsf{in}(A_1) \leftarrow \mathsf{in}(A_2),\ \mathsf{same}(A_1, A_2). \tag{9}$$

The downside of the rule (9) is that it leads to a quadratic blow-up upon grounding. Such an effect can be linearized by picking a unique representative amongst the multiple occurrences of a particular atom, or removed altogether by introducing unique node numbers in the parse tree created by the reifying program, thus making (5)–(9) obsolete.

Theorem 1. *Let P be a propositional normal logic program,* $\mathrm{Rfy}(P)$ *its reification, and Q the meta program consisting of rules (4)–(9) above.*

1. *If S is an answer set of P, then* $\mathrm{Rfy}(P) \cup Q$ *has an answer set T such that*

$$S = \{p \mid \exists n, m :\ \mathsf{head}(n, m) \in T,\ \mathsf{pred}(m, "p") \in T,\ \textit{and}\ \mathsf{in}(m) \in T\}. \tag{10}$$

2. *If T is an answer set of* $\mathrm{Rfy}(P) \cup Q$*, then S defined by (10) is an answer set of P.*

This result shows that our meta programming approach is compatible with that of [10] and hence provides means to program with reified rules that have been imported from somewhere else. However, to cover non-propositional programs which have predicates of arities greater than 0 a further refinement of (8) becomes necessary:

$$\mathsf{same}(C_1, C_2) \leftarrow \mathsf{const}(C_1, S),\ \mathsf{const}(C_2, S). \tag{11}$$
$$\begin{aligned}\mathsf{diffarg}(A_1, A_2) \leftarrow\ & \mathsf{alist}(A_1, N, C_1),\ \mathsf{alist}(A_2, N, C_2),\ \mathbf{not}\ \mathsf{same}(C_1, C_2),\\ & \mathsf{atom}(A_1),\ \mathsf{atom}(A_2).\end{aligned} \tag{12}$$
$$\begin{aligned}\mathsf{same}(A_1, A_2) \leftarrow\ & \mathsf{pred}(A_1, P),\ \mathsf{pred}(A_2, P),\ \mathbf{not}\ \mathsf{diffarg}(A_1, A_2),\\ & \mathsf{atom}(A_1),\ \mathsf{atom}(A_2),\ \mathbf{not}\ \mathsf{equal}(A_1, A_2).\end{aligned} \tag{13}$$

According to the revised definition (13), the (ground) atoms A_1 and A_2 based on a predicate P must not have differing arguments. This condition is formalized by the rule (12) for which we also define when two nodes C_1 and C_2 in a syntax tree refer to the same constant symbol S. Yet another possibility is to treat ground atoms $p(c_1, \ldots, c_n)$ as structural names for propositional atoms, and to apply reification and (8) as such. We defer the treatment of variables until Section 6 where meta-level grounding is addressed.

5 Changing the Syntax and Semantics of Rules

Having achieved a basic understanding how meta evaluation of reified rules is feasible in ASP, our next objective is to show how analogous meta programming techniques can be used to modify rules. Such modifications are usually syntactic by nature but they can also be motivated by the desire of tampering with the semantics of rules. In order to be able to address the semantic dimension, we first present an alternative semantics for normal logic programs with variables. A motivating example is provided as a starter.

Example 3. Consider the following rule capturing a social aspect of services:

$$\begin{aligned}\mathsf{uses}(U_1,S) \leftarrow\ & \mathsf{knows}(U_1,U_2),\ \mathsf{uses}(U_2,S),\\ & \mathsf{user}(U_1),\ \mathsf{user}(U_2),\ \mathsf{service}(S). \qquad (14)\end{aligned}$$

This oversimplified rule states that a user U_1 uses the services S used by other users U_2 known by U_1. The meaning of this rule deserves further attention in the presence of circularities demonstrated by the following set F of facts:

$$\mathsf{service}(f).\quad \mathsf{user}(a).\quad \mathsf{user}(b).\quad \mathsf{knows}(a,b).\quad \mathsf{knows}(b,a).$$

Actually, the set F forms the unique answer set of (14) subject to the facts listed above, i.e., neither a nor b would use the service. However, one could also argue for an alternative scenario where both a and b are using the service f since they know each other, hence capturing the social aspect of the service f. More formally, the symmetric instances of (14) provide circular support for $\mathsf{uses}(a,f)$ and $\mathsf{uses}(b,f)$ being true. ■

Example 3 essentially illustrates the difference between answer sets and *supported sets* [1]. Following the presentation from Section 2, a supported set $S \subseteq \mathrm{HB}(P)$ for a program P is closed under the rules of $\mathrm{Gnd}(P)^S$ and for each atom $p(c_1,\ldots,c_k) \in S$ there is a *supporting rule* (1) in P and a substitution σ over $\mathrm{HU}(P)$ such that the head $\alpha\sigma = p(c_1,\ldots,c_k)$, $\beta_1\sigma \in S,\ldots,\beta_n\sigma \in S$, and $\gamma_1\sigma \notin S,\ldots,\gamma_m\sigma \notin S$. The alternative semantics based on supported sets admits a further set $S = F \cup \{\mathsf{uses}(a,f),\mathsf{uses}(b,f)\}$ given the set of facts F. In this scenario, $\mathsf{uses}(a,f)$ is supported by (14) under the substitution $\sigma_a = [U_1/a, S/f, U_2/b]$: the atom $\mathsf{uses}(a,f)$ is the head and all body conditions are satisfied. The same can be stated about $\mathsf{uses}(b,f)$ under the substitution $\sigma_b = [U_1/b, S/f, U_2/a]$. The seminal paper [25] shows how the difference between answer sets and supported sets can be grasped within the answer-set semantics. Roughly speaking, the idea is to replace a ground positive body condition $p(c_1,\ldots,c_n)$ by a negative condition $\mathbf{not}\ \overline{p}(c_1,\ldots,c_n)$ where $\overline{p}$ is a new *complementary* predicate symbol defined by rules of the form $\overline{p}(c_1,\ldots,c_n) \leftarrow \mathbf{not}\ p(c_1,\ldots,c_n)$. In practice, it makes only sense to apply this transformation to particular predicates which may be limited by rule safety in the presence of variables.

Example 4. To modify the meaning of the predicate $\mathsf{uses}(\cdot,\cdot)$ in Example 3, the rule (14) can be rewritten using an auxiliary predicate $\mathsf{not\text{-}uses}(\cdot,\cdot)$ as follows:

$$\begin{aligned}\mathsf{uses}(U_1,S) \leftarrow\ & \mathsf{knows}(U_1,U_2),\ \mathbf{not}\ \mathsf{not\text{-}uses}(U_2,S),\\ & \mathsf{user}(U_1),\ \mathsf{user}(U_2),\ \mathsf{service}(S). \qquad (15)\\ \mathsf{not\text{-}uses}(U,S) \leftarrow\ & \mathbf{not}\ \mathsf{uses}(U,S),\ \mathsf{user}(U),\ \mathsf{service}(S). \qquad (16)\end{aligned}$$

These rules allow for an *answer set* with $\mathsf{uses}(a,f)$ and $\mathsf{uses}(b,f)$ for the set of facts F from Example 3. The predicates $\mathsf{knows}(\cdot,\cdot)$, $\mathsf{user}(\cdot)$, and $\mathsf{service}(\cdot)$ cannot be similarly transformed due to rule safety: the variables U_1, U_2, and S appearing in the head and negative body conditions of (14) would no longer be bound by positive conditions. ■

In what follows, we illustrate how meta programming techniques can be applied in order to change the semantics of rules if appropriate. For the sake of simplicity, we concentrate on the case of *propositional* normal programs and assume that all atoms are subject to Schlipf's transformation [25]. This contrasts with Example 4 but does not endanger rule safety since rules are variable-free. Given a reified propositional program as input, most of its structure can be copied as such by meta rules listed below:

$$\mathsf{hasrule}'(P,R) \leftarrow \mathsf{hasrule}(P,R). \qquad \mathsf{rule}'(R) \leftarrow \mathsf{rule}(R).$$
$$\mathsf{body}'(R,A) \leftarrow \mathsf{body}(R,A). \qquad \mathsf{head}'(R,A) \leftarrow \mathsf{head}(R,A).$$

The idea is that *primed* predicates represent the new structure after the syntactic modifications. E.g., by the first meta-level rule of the ones listed above, for each rule in the input program, there will be a corresponding rule in the translation.

$$\mathsf{neg}'(A) \leftarrow \mathsf{neg}(A). \tag{17}$$
$$\mathsf{pred}'(A,S) \leftarrow \mathsf{body}(R,A),\ \mathsf{neg}(A),\ \mathsf{pred}(A,S). \tag{18}$$
$$\mathsf{neg}'(A) \leftarrow \mathsf{pos}(A). \tag{19}$$
$$\mathsf{pred}'(A,S) \leftarrow \mathsf{head}(R,A),\ \mathsf{pred}(A,S). \tag{20}$$
$$\mathsf{pred}'(A,\mathsf{cat}(\text{"}\mathit{not}\text{-"},S)) \leftarrow \mathsf{body}(R,A),\ \mathsf{pos}(A),\ \mathsf{pred}(A,S). \tag{21}$$

By (17) and (18), negative body conditions remain untouched. The same applies to rule heads by (20) but, in contrast, positive body conditions are negated (19) and the predicate symbol in question is prefixed by "*not*-" (21). The function $\mathsf{cat}(\cdot,\cdot)$ is supposed to implement the concatenation of two strings in the grounder.[7] In addition to these, we need meta rules for creating new rule instances that define the complementary atoms.

$$\mathsf{create}(\mathsf{new}(A),A) \leftarrow \mathsf{body}(R,A),\ \mathsf{pos}(A). \tag{22}$$
$$\mathsf{head}'(R,\mathsf{new}(R)) \leftarrow \mathsf{create}(R,A). \tag{23}$$
$$\mathsf{pos}'(H) \leftarrow \mathsf{create}(R,A),\ \mathsf{head}'(R,H). \tag{24}$$
$$\mathsf{pred}'(H,\mathsf{cat}(\text{"}\mathit{not}\text{-"},S)) \leftarrow \mathsf{create}(R,A),\ \mathsf{head}'(R,H),\ \mathsf{pred}(A,S). \tag{25}$$
$$\mathsf{body}'(R,\mathsf{new}(R)) \leftarrow \mathsf{create}(R,A). \tag{26}$$
$$\mathsf{neg}'(B) \leftarrow \mathsf{create}(R,A),\ \mathsf{body}'(R,B). \tag{27}$$
$$\mathsf{pred}'(B,S) \leftarrow \mathsf{create}(R,A),\ \mathsf{body}'(R,B),\ \mathsf{pred}(A,S). \tag{28}$$

The rule (22) creates a new node corresponding to the rule being formed and (23) adds the new head node for it. The function $\mathsf{new}(\cdot)$ is evaluated by the grounder and it returns a new identity for a node. The rule (24) makes the head node positive whereas (25) defines its name. Similarly, the rules (26)–(28) create the syntax tree for the respective negative body condition. Finally, we have to link these new rules to the program.

$$\mathsf{hasrule}'(P,R) \leftarrow \mathsf{hasrule}(P,R'),\ \mathsf{create}(R,A). \tag{29}$$

As regards the use of rules listed above, the idea is that they are first joined with a reified input program. The syntactically transformed program can be extracted in the

[7] For instance, the GRINGO grounder has a Lua extension for implementing such functionality but in our preliminary experiments we implemented this via textual substitution.

reified form by looking at the instances of primed predicates in the resulting unique answer set. These facts can then be rendered back to a propositional rule which is the outcome of the overall syntactic transformation.

As the second objective of this section, we show how the semantics of reified rules can be changed at meta level. E.g., the shift from answer-set semantics to the supported one is feasible using a variant of the meta-evaluation rule (4).

$$\begin{aligned}\mathsf{in}(H) \leftarrow\ & \mathsf{rule}(R),\ \mathsf{head}(R,H),\ \mathbf{not}\ \mathsf{out}(P) : \mathsf{body}(R,P) : \mathsf{pos}(P),\\ & \mathbf{not}\ \mathsf{in}(N) : \mathsf{body}(R,N) : \mathsf{neg}(N). \qquad (30)\\ \mathsf{out}(P) \leftarrow\ & \mathbf{not}\ \mathsf{in}(P),\ \mathsf{pos}(P). \qquad (31)\end{aligned}$$

In contrast to (4), the idea is to express positive body conditions via double negation, i.e., $\mathbf{not}\ \mathsf{out}(P)$ aims to capture $\mathsf{in}(P)$. However, due to different treatment of positive and negative recursion under answer-set semantics, the meaning shifts to supported sets. The idea can be further restricted to positive conditions not crucial for rule safety.

Theorem 2. *Let P be a propositional normal logic program, $\mathrm{Rfy}(P)$ its reification, and Q the meta program consisting of rules (5)–(9), (30), and (31) above.*

1. *If S is a supported set of P, then $\mathrm{Rfy}(P) \cup Q$ has an answer set T such that (10).*
2. *If T is an answer set of $\mathrm{Rfy}(P) \cup Q$, then S defined by (10) is a supported set of P.*

Theorem 2 shows how meta rules can implement semantic shifts. Further examples in this direction can be found in [10] but for the class of disjunctive programs. We expect that such features can be very useful when putting together rules published by different participants of a smart space. This is even more realistic aspect given that knowledge processors are allowed to interact with several smart spaces simultaneously [4] meaning that different forms of meta interpretation may have to be applied to different spaces.

6 Meta-level Grounding of Rules

Rules containing variable occurrences can be published in a smart space using the abstract representation devised in Section 3. This leaves the grounding step of published rules open and, in order to use such rules in practice, raises two basic questions to be answered. The first concerns the domains of variables involved whereas the second is about how and when the substitution of variables by appropriate constants takes place. The goal of this section is to address these questions in our meta-level framework.

Due to the grounding phase, domains of variables play an important role in ASP modeling. Typically they are specified in terms of dedicated predicates called *domain predicates* which are either decided by the programmer beforehand or automatically detected by the grounder. E.g., in Example 3, the predicates $\mathsf{user}(\cdot)$ and $\mathsf{service}(\cdot)$ give the domains of variables U_1, U_2, and S. Once the respective extensions of these predicates are known, i.e., $\{f\}$ and $\{a,b\}$, the instances of the rule (14) can be intelligibly created. For the purposes of this paper, we assume that the domains of variables appearing in published rules can be determined from the context formed by the smart space(s) in question. We hypothesize that this forms a natural control point between the data residing in the space and any associated rules. The symbolic names of domain predicates or predicates in general can be presented as part of the underlying ontology. The same applies to the extensions of domain predicates involved, if appropriate for the use of rules, which thus can be recovered as sets of facts by posing queries on the ontology.

Let us then address the substitution of variables by constants. In principle, a rule (1) stands for its all Herbrand instances. But many of them are void because their positive body conditions remain false under answer-set semantics. Thus, for the purposes of this paper, we assume that the domains of variables are determined by domain predicates which are provided as a set of facts[8] amongst the rules subject to grounding and in the reified form. The following meta rules distinguish the structures of interest:

$$\mathsf{nonfact}(R) \leftarrow \mathsf{rule}(R),\ \mathsf{body}(R,A). \tag{32}$$

$$\mathsf{nonfact}(R) \leftarrow \mathsf{rule}(R),\ \mathsf{head}(R,H),\ \mathsf{alist}(H,N,V),\ \mathsf{var}(V,S). \tag{33}$$

$$\mathsf{poscond}(R,A) \leftarrow \mathsf{rule}(R),\ \mathsf{body}(R,A),\ \mathsf{pos}(A). \tag{34}$$

The rules (32) and (33) detect rules which are not facts. The positive body conditions of (other) rules are determined by the rule (34). To realize meta-level grounding, we aim at an encoding whose answer sets capture the rules of the resulting ground program. Thus, the first objective is to choose a rule which is not a fact for grounding:

$$\mathsf{ground}(R) \leftarrow \mathbf{not}\ \mathsf{other}(R),\ \mathsf{nonfact}(R). \tag{35}$$

$$\mathsf{other}(R_1) \leftarrow \mathsf{ground}(R_2),\ \mathsf{nonfact}(R_1),\ \mathsf{nonfact}(R_2),\ R_1 \neq R_2. \tag{36}$$

By (35), a rule R is subject to grounding if no *other* rule is. This concept is formalized by (36). Next, we match facts with the positive conditions of the rule being grounded.

$$\mathsf{fact}(R) \leftarrow \mathsf{rule}(R),\ \mathbf{not}\ \mathsf{nonfact}(R). \tag{37}$$

$$\mathsf{carg}(A,N,S) \leftarrow \mathsf{alist}(A,N,C),\ \mathsf{const}(C,S). \tag{38}$$

$$\mathsf{match}(A,F) \leftarrow \mathbf{not}\ \mathsf{differ}(A,F),\ \mathsf{poscond}(R,A),\ \mathsf{ground}(R),\ \mathsf{fact}(F). \tag{39}$$

$$\begin{aligned}\mathsf{differ}(A,F) \leftarrow\ & \mathsf{pred}(A,S_1),\ \mathsf{pred}(F,S_2),\ S_1 \neq S_2,\\ & \mathsf{poscond}(R,A),\ \mathsf{ground}(R),\ \mathsf{fact}(F).\end{aligned} \tag{40}$$

$$\begin{aligned}\mathsf{differ}(A,F) \leftarrow\ & \mathsf{carg}(A,N,S_1),\ \mathsf{carg}(F,N,S_2),\ S_1 \neq S_2,\\ & \mathsf{poscond}(R,A),\ \mathsf{ground}(R),\ \mathsf{fact}(F).\end{aligned} \tag{41}$$

$$\begin{aligned}\mathsf{differ}(A,F_1) \leftarrow\ & \mathsf{match}(A,F_2),\ \mathsf{poscond}(R,A),\ \mathsf{ground}(R),\\ & \mathsf{fact}(F_1),\ \mathsf{fact}(F_2).\end{aligned} \tag{42}$$

The rule (37) extracts facts into a distinguished relation whereas (37) detects the occurrences of constant symbols in atoms. The net effect of rules (39)–(42) is that each positive condition of the rule being grounded is matched against a unique fact having the same predicate symbol and constants in argument positions where the positive condition also has constants. What remains is to ensure that each variable gets a unique value when matched against constants appearing in the facts.

$$\mathsf{varg}(A,N,S) \leftarrow \mathsf{alist}(A,N,V),\ \mathsf{var}(V,S). \tag{43}$$

$$\begin{aligned}\mathsf{value}(R,S_1,S_2) \leftarrow\ & \mathsf{varg}(A,N,S_1),\ \mathsf{carg}(F,N,S_2),\ \mathsf{match}(A,F),\\ & \mathsf{poscond}(R,A),\ \mathsf{ground}(R),\ \mathsf{fact}(F).\end{aligned} \tag{44}$$

[8] Meta evaluation techniques from Section 4 can be used to infer such facts in an extra pass.

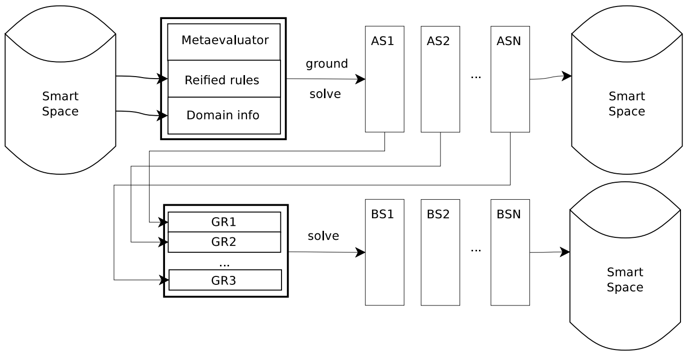


Fig. 3. An architecture for meta-level grounding

For space reasons, we omit rules analogous to (11)–(13) for checking the uniqueness of S_2 for each S_1 in R. Moreover, if some variable in the rule head or in the negative body conditions does not receive a value in the matching, a *safety* warning must be issued.

The overall architecture for meta-level grounding is illustrated in Figure 3. Grounding proceeds as follows. First, we read rules and extract domain information from the smart space and represent them as reified rules. Second, when combined with the meta-level grounding program, the resulting ground rules can be read off from the answer sets of the program and rendered as ASP rules. Third, the ground program is then fed as input to an ASP solver and the answer sets hereby obtained provide the basis for further reasoning. It is also possible to publish part of the information back to the smart space.

We also considered alternative formalizations of the grounding phase based on parameterized conjunctions discussed in the end of Section 2. Although such expressions are very convenient in the meta evaluation of propositional rules, as embodied in the rule (4), it is not easy to collect varying-length argument lists for predicate symbols. When grounding rules (1) with variables, the setting becomes very similar and it seems inherently difficult to specify the respective ground rules using a single meta-level rule whose ground instances would correspond to the ground instances of reified rules with variables. If this were possible, then even a single call to a grounder would be sufficient to ground rules at meta level—leading to an architecture different from Figure 3.

7 Qualitative Aspects of Reasoning

So far, we have concentrated on applying meta programming techniques to implement standard reasoning tasks in ASP. The objective of this section is to illustrate that such techniques enable us to address many qualitative aspects of rule-based reasoning. These needs easily arise in applications and, in particular, in the highly dynamic and heterogeneous environment provided by a smart space. In environments like this, rules may exist or may be available for limited periods of time and understanding their provenance and reliability becomes a major issue. If these aspects of rules matter, it is necessary to enrich reasoning mechanisms with appropriate pieces of meta information. This further suggests to use meta programming techniques when implementing actual inference.

In what follows, we assume that rules stored in a smart space are tagged with information regarding their validity, provenance, and integrity. For instance, each rule could have a time stamp indicating the time when it was published. It is straightforward to extend the reification procedure to yield facts of the form $\mathsf{time}(R, T)$ which annotates a rule R with its time T of publication. To incorporate this information to actual reasoning, the meta evaluation rule (4) could be refined with time stamps as follows:

$$\begin{aligned} \mathsf{in}(H) \leftarrow\ & \mathsf{rule}(R),\ \mathsf{time}(R, T),\ \mathsf{s} < T < \mathsf{e},\ \mathsf{head}(R, H), \\ & \mathsf{in}(P) : \mathsf{body}(R, P) : \mathsf{pos}(P), \\ & \mathbf{not}\ \mathsf{in}(N) : \mathsf{body}(R, N) : \mathsf{neg}(N). \end{aligned} \tag{45}$$

In the rule above, constants s and e specify a time interval which takes effect at the time of grounding. Consequently, only rules published within the given time interval are available for inference. In this way, answer sets become dependent on the interval and the technique can be used to analyze conclusions back in history if published rules are never removed from the smart space. Such considerations might involve both *brave* and *cautious* conclusions, i.e., whether an atom belongs to some or all answer sets of the selected rules. Having Example 1 in mind, one could analyze whether the user *potentially* or *necessarily* had access to a particular service given the provider's policies.

Assuming further that rules are tagged with their creators' identities it is possible to perform reasoning relative to who published the rules. E.g., by incorporating an additional argument to the predicate $\mathsf{in}(\cdot)$ above, the derivation of $\mathsf{in}(A, I)$ for an atom A and an identity I could mean that A can be derived by the rules published by I. Given this, a meta rule like $\mathsf{in}(A) \leftarrow \mathsf{in}(A, I) : \mathsf{id}(I)$ would express that A is an unanimous conclusion given the rules published by the participants of the space. With a little bit of elaboration, further concepts like *majority vote* can be formalized, e.g., using the extended rule types [26] supported by contemporary ASP systems. There are further objectives when reasoning about the provenance of rules. Since smart spaces form an open architecture, it may be desirable in the long run to endow rules with their publishers' signatures in order to prevent malicious users from forging rules. Similar cryptographic primitives can be introduced to ensure the integrity of rules in the first place. Since the verification of signatures and check sums involves non-trivial arithmetic operations, appropriate extensions are needed in grounders. To relieve the integration of rules, it is also possible to tag rules with syntactic and semantic descriptors. The latter, for instance, can be used to condition the meta rules used in their evaluation.

8 Related Work

In [6], a generalization of answer-set programs, viz. the class of HEX-*programs*, is introduced. The aim is to enable meta-level reasoning in the context of Semantic Web. The approach is based on syntactic extensions called *higher-order atoms* and *external atoms* which provide the means to extend answer-set programs in a systematic way and to interface an answer-set program with external sources. Thus ASP rules can be used at meta level to reason about information accessible through these kinds of special atoms. In particular, the goal is to provide an extended support for ontological languages. In contrast with HEX-programs, we explicitly focus on meta programming, i.e., having

ASP rules as input for ASP programs, and consider the applications of this feature in the context of smart spaces. The case of HEX-programs is further elaborated in [7] where the interdependencies of atoms, rule safety, and modularization are addressed.

A representation of answer-set programs in RuleML, i.e., the *markup language* for semantic web rules, is proposed in [8]. The syntax resembles that of XML and, in contrast with the RDF-based representation of syntax trees sketched in Section 3, it provides an alternative format to publish ASP rules in Semantic Web. Nevertheless, conversions between the formats are easy if the underlying ASP dialects coincide.

The framework described in this paper was sparked by the meta programming approach [10] where the reification of programs is performed analogously but partly using function symbols. The idea of meta evaluation (cf. Section 4) is already presented and further elaborated to change the semantics of disjunctive answer-set programs subject to *optimization statements* in order to grasp more complex optimization criteria. Our approach is quite similar as it counts on the meta-level interpretation of ASP rules. The application context, however, is different and we also highlight syntactic transformations and grounding as further tasks to be implemented via meta programming. The very same techniques provide a basis for *debugging* answer-set programs. E.g., the approach of [11] treats propositional disjunctive programs as object level programs whereas meta-level normal rules with variables are used to describe the central notions required to debug such programs. Analogous ideas can be exploited in smart spaces to reason about potential conclusions or to explain conclusions drawn by published rules.

Multi-context systems [5] formalize a reasoning mechanism for multiple contexts in each of which a different logic can be used to represent knowledge. The flow of information is modeled in terms of *bridge rules* which superficially resemble normal rules (1) but may refer to different contexts with context-specific conditions. The overall semantics is defined in terms of *equilibria* and the notion of a reduct in analogy to answer-set semantics. In contrast with multi-context systems, our meta programming framework aims at using a single formalism, i.e., logic programs subject to answer-set semantics, to represent the rules required for inference. Rules similar to bridge rules, if appropriate, can thus be used locally. The computational complexity results [5] suggest that the expressive power of our approach is lower than that of multi-context systems trading off computational costs. Moreover, since the semantics of rules is determined by answer-sets semantic shifts like the one illustrated in Section 5 may become necessary.

In database research, the notions of provenance [12] have been extensively studied. The idea is to annotate relations with algebraic structures and to deduce various kinds of justifications for queries over the relations. For the moment, we aim at a much simpler way to reason about provenance in our meta programming framework.

9 Conclusion

In this paper, we envision the potential of ASP in a setting where semantic rules arising in a smart space are represented as ASP rules and then published in the space. The idea is to reason with such rules by applying ASP techniques at meta level, i.e., when ASP rules are reified as facts and used as basis for meta programming. The encodings[9]

[9] Some are available under `http://research.ics.aalto.fi/software/asp/`

provided in Sections 4–7 illustrate the amenability of existing ASP technology to meta evaluation, syntactic and semantic transformations required in information integration, and addressing the qualitative aspects of underlying reasoning tasks. However, grounding at meta level is still challenging for existing grounders as discussed in the end of Section 6: it is difficult to reconstruct ground atoms with varying numbers of arguments out of logical conditions appearing in the meta rule. Hence the current architecture for grounding (recall Figure 3) counts on the use of an ASP solver for grounding purposes and a correspondence between answer sets and ground rules.

We anticipate that grounding is an essential step when rules published in a smart space are exploited by knowledge processors implemented using ASP technology. We view the grounding step as a potential control point where the domains of variables are dynamically determined and the applicability of particular public rules is partially decided. Since smart spaces are inherently dynamic and typically require context sensitive reasoning, we think that the primitives of ASP such as default negation and exceptions are well-suited for the needs arising in context-dependent knowledge processing.

In the future, we aim to trial the meta programming approach presented here in use cases similar to [2,23] paying particular attention to qualitative aspects (cf. Section 7) arising in the management of multi-user environments. The ideas of reification and meta evaluation are applicable in much wider sense than presented herein. When collecting rules from a number of sources there are safety risks that have to be dealt with, e.g., by introducing special mechanisms for rule selection. In this respect, the provenance of information is very central and we foresee use cases where tags, certificates, signatures, and check sums are widely applied. We also plan to pursue new solutions to meta grounding since it underlies the computationally challenging combinatorial problem of choosing appropriate values for variables. Moreover, we expect that an efficient implementation will require modularization of reasoning tasks as well as natural distribution amongst the entities present in a smart space.

Acknowledgments. This research has been supported by the European Institute of Technology (EIT) ICT Labs as part of its *Smart Spaces* thematic action line.

References

1. Apt, K., Blair, H., Walker, A.: Towards a theory of declarative knowledge. In: Foundations of Deductive Databases and Logic Programming, pp. 89–148. Morgan Kaufmann (1988)
2. Aziz, R.A., Janhunen, T., Luukkala, V.: Distributed Deadlock Handling for Resource Allocation in Smart Spaces. In: Balandin, S., Koucheryavy, Y., Hu, H. (eds.) NEW2AN 2011 and ruSMART 2011. LNCS, vol. 6869, pp. 87–98. Springer, Heidelberg (2011)
3. Berners-Lee, T., Hendler, J., Lassila, O.: The semantic web. Scientific American, 29–37 (2001)
4. Boldyrev, S., Oliver, I., Brown, R., Tuupola, J., Palin, A., Lappeteläinen, A.: Network and content aware information management. In: Proc. ICITST 2009, pp. 1–7. IEEE (2009)
5. Brewka, G., Eiter, T.: Equilibria in heterogeneous nonmonotonic multi-context systems. In: Proc. AAAI 2007, pp. 385–390. AAAI Press (2007)
6. Eiter, T., Ianni, G., Schindlauer, R., Tompits, H.: A uniform integration of higher-order reasoning and external evaluations in answer-set programming. In: Proc. IJCAI 2005, pp. 90–96. Professional Book Center (2005)

7. Eiter, T., Ianni, G., Schindlauer, R., Tompits, H.: Effective Integration of Declarative Rules with External Evaluations for Semantic-Web Reasoning. In: Sure, Y., Domingue, J. (eds.) ESWC 2006. LNCS, vol. 4011, pp. 273–287. Springer, Heidelberg (2006)
8. Eiter, T., Ianni, G., Schindlauer, R., Tompits, H.: A RuleML syntax for answer-set programming. In: Proc. ALPSWS 2006, pp. 107–108 (2006)
9. Främling, K., Oliver, I., Honkola, J., Nyman, J.: Smart spaces for ubiquitously smart buildings. In: Proc. UBICOMM 2009, pp. 295–300 (2009)
10. Gebser, M., Kaminski, R., Schaub, T.: Complex optimization in answer set programming. TPLP 11(4-5), 821–839 (2011)
11. Gebser, M., Pührer, J., Schaub, T., Tompits, H.: A meta-programming technique for debugging answer-set programs. In: Proc. AAAI 2008, pp. 448–453. AAAI Press (2008)
12. Green, T.J.: Containment of conjunctive queries on annotated relations. Theory Comput. Syst. 49(2), 429–459 (2011)
13. Hall, R.J.: Open Modeling in Multi-stakeholder Distributed Systems: Research and Tool Challenges. In: Hermenegildo, M.V., Puebla, G. (eds.) SAS 2002. LNCS, vol. 2477, p. 2. Springer, Heidelberg (2002)
14. Honkola, J., Laine, H., Brown, R., Oliver, I.: Cross-Domain Interoperability: A Case Study. In: Balandin, S., Moltchanov, D., Koucheryavy, Y. (eds.) ruSMART 2009. LNCS, vol. 5764, pp. 22–31. Springer, Heidelberg (2009)
15. Katasonov, A., Palviainen, M.: Towards ontology-driven development of applications for smart environments. In: Proc. PerCom 2010, pp. 696–701. IEEE (2010)
16. Lewis, G.A., Morris, E.J., Simanta, S., Wrage, L.: Why standards are not enough to guarantee end-to-end interoperability. In: Proc. ICCBSS 2008, pp. 164–173. IEEE (2008)
17. Lifschitz, V.: Answer set planning. In: Proc. ICLP 1999, pp. 25–37. MIT Press (1999)
18. Luukkala, V., Binnema, D., Börzsei, M., Corongiu, A., Hyttinen, P.: Experiences in implementing a cross-domain use case by combining semantic and service level platforms. In: Proc. ISCC 2010, pp. 1071–1076. IEEE (2010)
19. Luukkala, V., Niemelä, I.: Enhancing a Smart Space with Answer Set Programming. In: Dean, M., Hall, J., Rotolo, A., Tabet, S. (eds.) RuleML 2010. LNCS, vol. 6403, pp. 89–103. Springer, Heidelberg (2010)
20. Marek, V., Truszczyński, M.: Stable models and an alternative logic programming paradigm. In: The Logic Programming Paradigm: A 25-Year Perspective, pp. 375–398. Springer (1999)
21. Niemelä, I.: Logic programs with stable model semantics as a constraint programming paradigm. Ann. Math. Artif. Intell. 25(3-4), 241–273 (1999)
22. Ouksel, A.M., Sheth, A.P.: Semantic interoperability in global information systems: A brief introduction to the research area and the special section. SIGMOD Record 28(1), 5–12 (1999)
23. Pantsar-Syväniemi, S., Ovaska, E., Ferrari, S., Cinotti, T.S., Zamagni, G., Roffia, L., Mattarozzi, S., Nannini, V.: Case study: Context-aware supervision of a smart maintenance process. In: Proc. SAINT 2011, pp. 309–314. IEEE Computer Society (2011)
24. Peeters, J., Van Der Vlist, B., Niezen, G., Hu, J., Feijs, L.: Controlling smart home environments with semantic connections: a tangible and an AR approach. In: Proc. DeSForM (to appear, 2012)
25. Schlipf, J.: The expressive powers of the logic programming semantics. Journal of Computer and System Sciences 51(1), 64–86 (1995)
26. Simons, P., Niemelä, I., Soininen, T.: Extending and implementing the stable model semantics. Artificial Intelligence 138(1-2), 181–234 (2002)
27. Weiser, M.: The computer for the twenty-first century. Scientific American 265(3), 94–104 (1991)
28. Wielemaker, J., Hildebrand, M., van Ossenbruggen, J.: Prolog as the fundament for applications on the semantic web. In: Proc. ALPSWS 2007. CEUR-WS.org, vol. 287 (2007)

A Sound and Complete Backward Chaining Algorithm for Existential Rules

Mélanie König, Michel Leclère, Marie-Laure Mugnier, and Michaël Thomazo

University Montpellier 2, France

Abstract. We address the issue of Ontology-Based Data Access which consists of exploiting the semantics expressed in ontologies while querying data. Ontologies are represented in the framework of existential rules, also known as Datalog+/-. We focus on the backward chaining paradigm, which involves rewriting the query (assumed to be a conjunctive query, CQ) into a set of CQs (seen as a union of CQs). The proposed algorithm accepts any set of existential rules as input and stops for so-called finite unification sets of rules (fus). The rewriting step relies on a graph notion, called a piece, which allows to identify subsets of atoms from the query that must be processed together. We first show that our rewriting method computes a minimal set of CQs when this set is finite, i.e., the set of rules is a fus. We then focus on optimizing the rewriting step. First experiments are reported.

1 Introduction

In recent years, there has been growing interest in exploiting the semantics expressed in ontologies when querying data, an issue known as ontology-based data access (OBDA). To address this issue, several logic-based formalisms have been developed. The dominant approach is based on description logics (DLs), with the most studied DLs in this context being lightweight DLs, such as DL-Lite and $\mathcal{EL}$ families [Baa03, CGL$^+$07] and their Semantic Web counterparts, so callso-calleded tractable fragments of OWL2. A newer approach, to which this paper contributes, is based on existential rules. Existential rules have the ability of generating new unknown individuals, a feature that has been recognized as crucial in an open-world perspective, where it cannot be assumed that all individuals are known in advance. These rules are of the form *body* $\rightarrow$ *head*, where the body and the head are conjunctions of atoms (without functions), and variables that occur only in the head are *existentially* quantified, hence the name $\forall\exists$-rules in [BLMS09, BLM10] or existential rules in [BMRT11, KR11]. They are also known as Datalog +/-, a recent extension of plain Datalog to tuple-generating dependencies (expressive constraints that have long been studied in databases and have the same logical form as existential rules) [CGK08, CGL09].

In this paper, we consider knowledge bases composed of a set of facts -or data- and of existential rules. The basic problem, query answering, consists of computing the set of answers to a query in the knowledge base. We consider conjunctive queries (CQs), which are the standard basic queries. CQs can be seen as existentially quantified conjunctions of atoms. The fundamental decision problem associated with query answering can be expressed in several equivalent ways, in particular as a CQ entailment problem: is a given (Boolean) CQ logically entailed by a knowledge base?

CQ entailment is undecidable for general existential rules. There is currently an intense research effort aimed at finding decidable subsets of rules that provide good tradeoffs between expressivity and complexity of query answering (see [Mug11] for a synthesis). With respect to (lightweight) DLs, these decidable rule fragments are more powerful and flexible. However, the rule-based ODBA framework is rather new and it does not come yet with practically usable algorithms, with the exception of very simple classes of rules, which can be seen as slight generalizations of lightweight DLs. In this paper, we undertake a step in this direction.

There are two classical paradigms for processing rules, namely *forward chaining* and *backward chaining*, schematized in Figure 1. Both can be seen as ways of integrating the rules either into the facts or into the query (denoted by Q in the figure). Forward chaining uses the rules to enrich the facts and the query is entailed if it maps by homomorphism to the enriched facts. Backward chaining proceeds in the “reverse” manner: it uses the rules to rewrite the query in several ways and the initial query is entailed if a rewritten query maps to the initial facts.

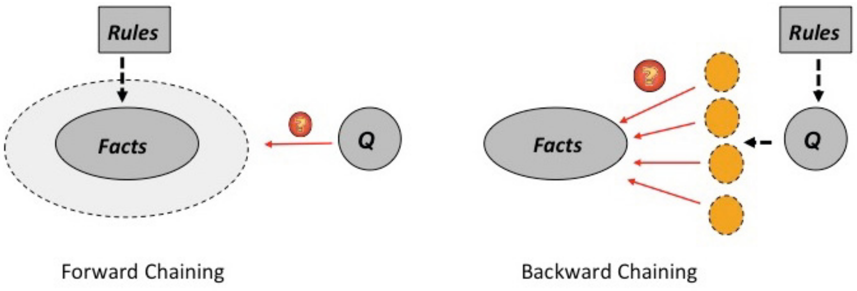


Fig. 1. Forward / Backward Chaining

In the context of large data, the obvious advantage of backward chaining is that it does not make the data grow. When the set of rewritten queries is finite, this set can be seen as a single query, which is the union of the conjunctive queries in the set. An approach initiated with DL-Lite consists of decomposing backward chaining into two steps: (1) rewrite the initial query as a union of CQs (2) use a database management system to answer this union query. This approach aims to benefit from the optimizations developed for classical database queries. Since the CQs are independent, their processing can be easily parallelized. This approach can be generalized to rewritings into first-order queries, which are the logical counterpart of SQL queries (with closed-world assumption). It is at the core of several systems, such as Nyaya [GOP11], QuOnto [CGL$^+$07] and Requiem [PUHM09]. Such rewritings are usually of exponential size with respect to the initial query (however [KKZ11] exhibits specific cases where the rewriting is of polynomial size). In [RA10] another method, also devoted to DL-Lite, is proposed: it consists of rewriting the query into a non-recursive Datalog program, which in turn can be translated into a first-order query of smaller size than the union of CQs that would be output. [GS12] defines such a rewriting with polynomial size in both Q and $\mathcal{R}$ for some specific classes of rules. However, distributed processing of non-recursive Datalog programs is not as easy as for UCQs.

While these works focus on specific rule sublanguages, in this paper we consider backward chaining with *general existential rules*, i.e., our algorithm accepts as input any set of existential rules, but of course is guaranteed to stop only for a subset of them (so-called "finite unification sets" of rules in [BLM10], which includes expressive classes of rules, see Section 3).

The originality of our method lies in the rewriting step, which is based on a graph notion, that of a *piece*. Briefly, a piece is a subset of atoms from the query that must be erased together during a rewriting step. The backward chaining mechanisms classically used in logic programming process rules and queries atom by atom: at each step, an atom a of a query Q is unified with the head of a rule R (which is composed of a single atom) and a new query is generated by replacing a in Q by the body of R (precisely: let u be the unifier, the new query is $u(\mathit{body}(R)) \cup u(Q \setminus \{a\})$. Here, existential variables in rule heads have to be taken into account, which prevents the use of atomic unification. Instead, subsets of atoms ("pieces") have to be considered at once. We present below a very simple example (in particular, the head of the rule is restricted to a single atom).

Example 1. Let the rule $R = q(x) \rightarrow p(x, y)$, which corresponds to the logical formula $\forall x\ (q(x) \rightarrow \exists y\ p(x, y))$, and the Boolean CQ $Q = p(u, v) \wedge p(w, v) \wedge p(w, t) \wedge r(u, w)$ (a closed existential formula), where all the terms are variables. Assume we want to unify $p(u, v)$, the first atom in Q, with $p(x, y)$ by a substitution $\{(u, x), (v, y)\}$. Since v is unified with the existential variable y, all other atoms containing v must also be considered: indeed, simply rewriting Q into $q(x) \wedge p(w, y) \wedge p(w, t) \wedge r(x, w)$ as would be done in a "classical" backward chaining step would be incorrect (intuitively, the fact that the atoms $p(u, v)$ and $p(w, v)$ in Q share a variable would be lost with $q(x)$ and $p(w, y)$). Thus, $p(u, v)$ and $p(w, v)$ are both unified with the head of R by means of the following substitution: $\{(u, x), (v, y), (w, x)\}$. Since w is associated with a non-existential variable, there is no need to include $p(w, t)$ in the set, although in this example it could be added. $\{p(u, v), p(w, v)\}$ is called a piece. The corresponding rewriting of Q is $q(x) \wedge p(x, t) \wedge r(x, x)$.

Pieces come from earlier work on conceptual graph rules, whose logical translation is exactly existential rules [SM96]. This notion has then been recast in the framework of existential rules in [BLMS09][BLMS11]. In this paper, we start from the definition of a piece-unifier, which unifies part of a rule head and part of the query, while respecting pieces: when it unifies an atom in the query, it must unify the whole piece to which this atom belongs. Backward chaining based on piece-unifiers is known to be sound and complete (e.g. [BLMS11], and basically [SM96] for conceptual graphs). An alternative method would be to consider the Skolem form of rules, i.e., to replace existential variables in the head by Skolem functions of variables occurring in the body, however we think it is simpler and more intuitive to keep the original rule language.

This framework established, we then posed ourselves the following questions:

1. Can we ensure that we produce a minimal set of rewritten conjunctive queries, in the sense that no sound and complete algorithm can produce a smaller set?

2. How to optimize the rewriting step? The problem of deciding whether there is a piece-unifier between a query and a rule head is NP-complete and the number of piece-unifiers can be exponential in the size of the query.

With respect to the first question, let us say that a set $\mathcal{Q}$ of rewritten CQs from a CQ Q and a set of rules $\mathcal{R}$ is *sound and complete* if the following holds: for any set of facts F, if Q is entailed by F and $\mathcal{R}$ then there is a query Q_i in $\mathcal{Q}$ such that Q_i is entailed by F (completeness), and reciprocally (soundness). We point out that any sound and complete set of CQs (w.r.t. the same Q and $\mathcal{R}$) remains sound and complete when it is restricted to its most general elements (w.r.t. the generalization relation induced by homomorphism). We then show that all sound and complete sets of CQs restricted to their most general CQs have the same cardinality, which is minimal w.r.t. the completeness property. It is easily checked that the algorithm we propose produces such a minimal set. If we moreover delete redundant atoms from the obtained CQs (which can be performed by a linear number of homomorphism tests for each query), we obtain a *unique* sound and complete set of CQs that has both minimal cardinality and elements of minimal size (unicity is of course up to a bijective variable renaming).

With respect to the second question, we consider rules with an atomic head. This is not a restriction in terms of expressivity, since any rule can be decomposed into an equivalent set of atomic-head rules by simply introducing a new predicate for each rule (e.g. [CGK08], [BLMS09]). Besides, many rules found in the literature have an atomic head. Restricting our focus to atomic head rules allows us to obtain nice properties. We first show that it is sufficient to consider piece-unifiers that (1) are most general unifiers, and (2) process a single piece at once.[1] We then show that the number of most general single-piece unifiers of a query Q with the (atomic) head of a rule R is bounded by the size of the query. Finally, we exploit the fact that each atom in Q belongs to at most one piece with respect to R (which is false for general existential rules) to efficiently compute a rewriting step, i.e., generate all queries obtained from R and Q by most general single-piece unifiers of Q with R. A backward chaining algorithm benefiting from these results has been implemented.

The paper is organized as follows. Section 2 introduces our framework. Sections 3 and 4 are respectively devoted to the first and to the second question. Finally, Section 5 reports first experiments and outlines further work. A long version of this paper with all proofs is available as a technical report [KLMT12].

2 Framework

An *atom* is of the form $p(t_1, \ldots, t_k)$ where p is a predicate with arity k, and the t_i are terms, i.e., variables or constants (we do not consider other function symbols). Given an atom or a set of atoms A, $\mathit{vars}(A)$, $\mathit{consts}(A)$ and $\mathit{terms}(A)$ denote its set of variables, of constants and of terms, respectively. In the following examples, all the terms are variables (denoted by x, y, z, etc.) unless otherwise specified. $\models$ denotes the classical logical consequence.

[1] Actually, this property should be extendable to rules with non-atomic head, but this would first involve defining a suitable comparison operation between piece-unifiers, operation which is simply defined with atomic-head rules.

Given atom sets A and B, a *homomorphism* h from A to B is a substitution of *vars*(A) by *terms*(B) such that $h(A) \subseteq B$. We say that A *maps* to B by h. If there is a homomorphism from A to B, we say that A is *more general* than B (or B is *more specific* than A), which is denoted $A \geq B$ (or $B \leq A$).

A *fact* is the existential closure of a conjunction of atoms.[2] A *conjunctive query* (CQ) is an existentially quantified conjunction of atoms. When it is a closed formula, it is called a *Boolean* CQ (BCQ). Note that facts and BCQs have the same logical form. In the following, we will see them as sets of atoms. It is well-known that, given a fact F and a BCQ Q, $F \models Q$ iff there is a homomorphism from Q to F.

The answer to a BCQ Q in a fact F is *yes* if there is a homomorphism from Q to F. Otherwise, let $x_1 \dots x_q$ be the free variables in Q: a tuple of constants $(a_1 \dots a_q)$ is an answer to Q in F if there is a homomorphism from Q to F that maps x_i to a_i for each i. In the following, we consider only Boolean queries for simplicity reasons. This is not a restriction, since a CQ with free variables $x_1 \dots x_q$ can be translated into a BCQ by adding the atom $ans(x_1 \dots x_q)$, where ans is a special predicate not occurring in the knowledge base. Since ans can never be erased by a rewriting step, it guarantees that the x_i can only be substituted and will not "disappear". Note that we could also consider unions of conjunctive queries, in this case each conjunctive subquery would be processed separately.

Definition 1 (Existential rule). *An* existential rule *(or simply* rule *when clear from the context) is a formula* $R = \forall \mathbf{x} \forall \mathbf{y} (B[\mathbf{x}, \mathbf{y}] \rightarrow (\exists \mathbf{z} H[\mathbf{y}, \mathbf{z}]))$ *where* $B =$ *body*(R) *and* $H =$ *head*(R) *are conjunctions of atoms, resp. called the* body *and the* head *of R. The* frontier *of R, noted* fr(R)*, is the set of variables* vars$(B) \cap$ vars$(H) = \mathbf{y}$. *The* existential variables *in R, noted* exist(R)*, is the set of variables* vars$(H) \setminus$ fr$(R) = \mathbf{z}$.

In the following, we will omit quantifiers in rules as there is no ambiguity.

A *knowledge base* (KB) $\mathcal{K} = (F, \mathcal{R})$ is composed of a finite set of facts (seen as a single fact) F and a finite set of existential rules $\mathcal{R}$. The (Boolean) CQ entailment problem is the following: given a KB $\mathcal{K} = (F, \mathcal{R})$ and a BCQ Q, does $F, \mathcal{R} \models Q$ hold?

This question can be solved with forward chaining: $F, \mathcal{R} \models Q$ iff there exists a finite sequence $(F_0 = F), \dots, F_k$, where each F_i for $i > 0$ is obtained by applying a rule from $\mathcal{R}$ to F_{i-1}, such that $F_k \models Q$ (see e.g. [BLMS11] for details).

As explained in the introduction, backward chaining relies on a unification operation between a query and a rule head. The following definition of piece-unifier is an alternative definition of the operation defined in [BLMS11].

Other Notations: Throughout the paper we note respectively R and Q the considered rule and query. We assume that R and Q have no variables in common. When needed, a "fresh copy" of R is obtained by bijectively renaming the variables in R into "fresh" variables. We note $\mathcal{C}$ the set of constants occurring in the set of rules $\mathcal{R}$ and in Q. Given $Q' \subseteq Q$, we note $\bar{Q}'$ the set $Q \setminus Q'$. The variables in *vars*$(Q') \cap$ *vars*$(\bar{Q}')$ are called *separating variables* and denoted *sep*(Q').

A piece-unifier is defined as a pair (Q', u), where Q' is a non-empty subset of Q, and u is a substitution that "unifies" Q' with a subset H' of *head*(R), in the sense

[2] We generalize the classical notion of a fact in order to take existential variables into account.

that $u(Q') = u(H')$; H' is the subset of *head*(R) composed of atoms a such that $u(a) = u(b)$ for some $b \in Q'$. The substitution u can be decomposed as follows: (1) it specializes the frontier of R, thus *head*(R), while leaving existential variables unchanged; (2) it maps Q' to $u($*head*$(R))$, while satisfying the following constraint: the separating variables in Q' are not mapped to existential variables, i.e., they are mapped to $u($*fr*$(R))$ or to constants.

Definition 2 (Piece-unifier). *Let Q be a CQ and R be a rule. A piece-unifier of Q with R is a pair $\mu = (Q', u)$ with $Q' \subseteq Q$, $Q' \neq \emptyset$, and u is a substitution of* fr$(R) \cup$ vars(Q') *by* terms$($head$(R)) \cup \mathcal{C}$ *such that:*

1. *for all $x \in$ fr(R), $u(x) \in$ fr$(R) \cup \mathcal{C}$ (for technical convenience, we allow $u(x) = x$);*
2. *for all $x \in$ sep(Q'), $u(x) \in$ fr$(R) \cup \mathcal{C}$;*
3. *$u(Q') \subseteq u($head$(R))$.*

u is divided into u^R with domain fr(R) *and $u^{Q'}$ with domain* vars(Q').

Note that instead of $\mathcal{C}$, we could consider *consts*$(Q') \cup$ *consts*$($*head*$(R))$, however $\mathcal{C}$ is convenient for proof purposes.

Example 2. Let us take again $R = q(x) \rightarrow p(x, y)$ and $Q = p(u, v) \wedge p(w, v) \wedge p(w, t) \wedge r(u, w)$. Here are three piece-unifiers of Q with R:
$\mu_1 = (Q'_1, u_1)$ with $Q'_1 = \{p(u, v), p(w, v)\}$ and $u_1 = \{(u, x), (v, y), (w, x)\}$
Note that we will omit identity pairs in all examples; f.i. u_1 contains (x, x)
$\mu_2 = (Q'_2, u_2)$ with $Q'_2 = \{p(w, t)\}$ and $u_2 = \{(w, x), (t, y)\}$
$\mu_3 = (Q'_3, u_3)$ with $Q'_3 = \{p(u, v), p(w, v), p(w, t)\}$ and $u_3 = \{(u, x), (v, y), (w, x), (t, y)\}$
These piece-unifiers will be called the "most general piece-unifiers" of Q with R in Section 4.

In the previous example, R has an atomic head, thus a piece-unifier of Q' with R actually unifies the atoms from Q' and the head of R into a single atom. In the general case, a piece-unifier unifies Q' and a subset H' of *head*(R) into a set of atoms, as shown by the next example.

Example 3. Let $R = q(x) \rightarrow p(x, y) \wedge p(y, z) \wedge p(z, t) \wedge r(y)$ and $Q = p(u, v) \wedge p(v, w) \wedge r(u)$. A piece-unifier of Q with R is (Q'_1, u_1) with $Q'_1 = \{p(u, v), p(v, w)\}$ and $u_1 = \{(u, x), (v, y), (w, z)\}$. $H' = \{p(x, y), p(y, z)\}$ and $u_1(Q') = u_1(H') = H'$. Another piece-unifier is (Q'_2, u_2) with $Q'_2 = Q$ and $u_2 = \{(u, y), (v, z), (w, t)\}$; in this case, $H' = \{p(y, z), p(z, t), r(y)\}$.

Finally, the next example illustrates the role of constants (in the query here, but constants may also occur in rules).

Example 4. Let $R = q(x, y) \rightarrow p(x, y, z)$ and $Q = p(u, a, v) \wedge p(a, w, v)$, where a is a constant. The variable v has to be mapped to the existential variable z. The unique piece-unifier is here $(Q, \{(x, a), (y, a), (u, a), (w, a), (v, z)\})$.

We are now able to formally define *pieces*. A piece of Q can be seen as a minimal subset Q' satisfying the above definition of a piece-unifier. Generally speaking, a set of atoms can be partitioned into subsets called pieces according to a set T of variables acting as 'cutpoints": two atoms are in the same piece if they are connected by a path of variables that do not belong to T [BLMS11]. Note that constants do not allow to connect atoms. Here, T is the set of variables from Q' that are not mapped to existential variables by u.

Definition 3 (Piece). *[BLMS11] Let A be a set of atoms and $T \subseteq (\mathsf{vars}(A))$. A piece of A according to T is a minimal non-empty subset P of A such that, for all a and a' in A, if $a \in P$ and $(\mathsf{vars}(a) \cap \mathsf{vars}(a')) \not\subseteq T$, then $a' \in P$.*

Definition 4 (Cutpoint, Piece of Q). *Given a piece-unifier $\mu = (Q', u)$ of Q with R, a variable $x \in Q'$ is a* cutpoint *if $u(x) \notin \mathsf{exist}(R)$ (equivalently: $u(x) \in \mathsf{fr}(R) \cup \mathcal{C}$). The set of cutpoints associated with μ is denoted by $T_Q(\mu)$. We call* piece *of Q (for μ) a piece of Q according to $T_Q(\mu)$.*

Example 3 (contd) Q'_1 and Q'_2 are pieces. Note that an atom may belong to different pieces according to different unifiers (it is the case here for $p(u, v)$ and $p(v, w)$).

The following property is easily checked and justifies the name "piece-unifier":

Property 1. For any piece-unifier $\mu = (Q', u)$, Q' is a set of pieces of Q. In particular, $\mathit{sep}(Q') \subseteq T_Q(\mu)$.

To summarize, a piece of Q is a minimal subset of atoms that must be considered together once cutpoints in Q have been defined. A piece-unifier may process several pieces. In Section 4, we will focus on unifiers processing a single piece. Finally, note that in rules without existential variables, such as in plain Datalog, each piece is restricted to a single atom. Concerning the next definitions, we recall the assumption that $vars(R) \cap vars(Q) = \emptyset$:

Definition 5 (Rewriting). *Given a CQ Q, a rule R and a piece-unifier $\mu = (Q', u)$ of Q with R, the rewriting of Q according to μ, denoted $\beta(Q, R, \mu)$ is $u^R(\mathsf{body}(R)) \cup u^{Q'}(\bar{Q}')$.*

Definition 6 ($\mathcal{R}$-rewriting of Q). *Let Q be a CQ and $\mathcal{R}$ be a set of rules. An $\mathcal{R}$-*rewriting of *Q is a CQ Q_k obtained by a finite sequence $(Q_0 = Q), Q_1, \ldots, Q_k$ such that for all $0 \le i < k$, there is $R_i \in \mathcal{R}$ and a piece-unifier μ of Q_i with R_i such that $Q_{i+1} = \beta(Q_i, R, \mu)$.*

Theorem 1 (Soundness and completeness of piece-based backward chaining). *(basically[SM96]) Let a KB $\mathcal{K} = (F, \mathcal{R})$ and a (Boolean) CQ Q. Then $F, \mathcal{R} \models Q$ iff there is an $\mathcal{R}$-rewriting of Q that maps to F.*

The soundness and completeness of the piece-based backward chaining mechanism can be proven via the following equivalence with forward chaining: there is an $\mathcal{R}$-rewriting from Q to Q' that maps to F iff there is a sequence of rule applications leading from F to F' such that Q maps to F'.

To evaluate the quality of rewriting sets produced by different mechanisms, we introduce the notions of soundness and completeness of a set of CQs with respect to Q and $\mathcal{R}$ (such a set is called a *rewriting set* hereafter):

Definition 7 (Sound and Complete (rewriting) set of CQs). *Let $\mathcal{R}$ be a set of existential rules and Q be a (Boolean) CQ. Let $\mathcal{Q}$ be a set of CQs. $\mathcal{Q}$ is said to be* sound *w.r.t. Q and $\mathcal{R}$ if for all facts F, for all $Q_i \in \mathcal{Q}$, if Q_i maps to F then $\mathcal{R}, F \models Q$. Reciprocally, $\mathcal{Q}$ is said to be* complete *w.r.t. Q and $\mathcal{R}$ if for all fact F, if $\mathcal{R}, F \models Q$ then there is $Q_i \in \mathcal{Q}$ such that Q_i maps to F.*

As expressed by Theorem 1, the set of $\mathcal{R}$-rewritings that can be produced with piece-unifiers is sound and complete. In the next section, we will address the issue of the size of a rewriting set.

3 Minimal Rewriting Sets

We first point out that only the *most general elements* of a rewriting set need to be considered. Indeed, let Q_1 and Q_2 be two elements of a rewriting set such that $Q_2 \leq Q_1$ and let F be any fact: if Q_1 maps to F, then Q_2 is useless; if Q_1 does not map to F, neither does Q_2; thus removing Q_2 will not undermine completeness (and it will not undermine soundness either). The output of a rewriting algorithm should thus be a minimal set of incomparable queries that "covers" all rewritings of the initial query:

Definition 8 (Cover). *Let $\mathcal{Q}$ be a set of BCQs. A* cover *of $\mathcal{Q}$ is a set of BCQs $\mathcal{Q}^c \subseteq \mathcal{Q}$ such that:*

1. *for any element $Q \in \mathcal{Q}$, there is $Q' \in \mathcal{Q}^c$ such that $Q \leq Q'$,*
2. *elements of $\mathcal{Q}^c$ are pairwise incomparable w.r.t. $\leq$.*

Note that a cover is inclusion-minimal. Moreover, it can be easily checked that all covers of $\mathcal{Q}$ have the same cardinality.

Example 5. Let $\mathcal{Q} = \{Q_1, \ldots, Q_6\}$ and the following preorder over $\mathcal{Q}$: $Q_6 \leq Q_5$; $Q_5 \leq Q_1, Q_2$; $Q_4 \leq Q_1, Q_2, Q_3$; $Q_1 \leq Q_2$ and $Q_2 \leq Q_1$ (Q_1 and Q_2 are thus equivalent). There are two covers of $\mathcal{Q}$, namely $\{Q_1, Q_3\}$ and $\{Q_2, Q_3\}$.

Note that the set of rewritings of Q can have a finite cover even when it is infinite, as illustrated by Example 6.

Example 6. Let $Q = t(u), R_1 = t(x) \wedge p(x, y) \rightarrow r(y), R_2 = r(x) \wedge p(x, y) \rightarrow t(y)$. The set of $\mathcal{R}$-rewritings of Q with $\{R_1, R_2\}$ is infinite. The first generated queries are the following (note that rule variables are renamed when needed):

$Q_0 = t(u)$
$Q_1 = r(x) \wedge p(x, y)$ // from Q_0 and R_2 with $\{(u, y)\}$
$Q_2 = t(x_0) \wedge p(x_0, y_0) \wedge p(y_0, y)$ // from Q_1 and R_1 with $\{(x, y_0)\}$
$Q_3 = r(x_1) \wedge p(x_1, y_1) \wedge p(y_1, y_0) \wedge p(y_0, y)$ // from Q_2 and R_2 with $\{(x_0, y_1)\}$
$Q_4 = t(x_2) \wedge p(x_2, y_2) \wedge p(y_2, y_1) \wedge p(y_1, y_0) \wedge p(y_0, y)$ // from Q_3 and R_1
and so on . . .

However, the set of the most general $\mathcal{R}$-rewritings is $\{Q_0, Q_1\}$ since any other query than can be obtained is more specific than Q_0 or Q_1.

A set of rules $\mathcal{R}$ for which it is ensured that the set of $\mathcal{R}$-rewritings of any query has a finite cover is called a finite unification set (*fus*). The *fus* property is not recognizable [BLMS11], but several *fus* recognizable classes have been exhibited in the literature: atomic-body [BLMS09], also known as linear TGDs [CGL09], domain-restricted [BLMS09], (join-)sticky [CGP10]. Following Algorithm 1 is a breadth-first algorithm that, given a fus $\mathcal{R}$ and a query Q, generates a cover of the set of $\mathcal{R}$-rewritings of Q. "Exploring" a query consists of computing the set of immediate rewritings of this query with all rules. Initially, Q is the only query to explore; at each step (while loop iteration), all queries generated at the preceding step and kept in the current cover are explored.

Algorithm 1. A BREADTH-FIRST REWRITING ALGORITHM

```
Data: A fus $\mathcal{R}$, a conjunctive query $Q$
Result: A cover of the set of $\mathcal{R}$-rewritings of $Q$
$\mathcal{Q}_F \leftarrow \{Q\}$; // resulting set
$\mathcal{Q}_E \leftarrow \{Q\}$; // queries to be explored
while $\mathcal{Q}_E \neq \emptyset$ do
    $\mathcal{Q}_t \leftarrow \emptyset$; // queries generated at this rewriting step
    for $Q_i \in \mathcal{Q}_E$ do
        for $R \in \mathcal{R}$ do
            for $\mu$ piece-unifier of $Q_i$ with $R$ do
                $\mathcal{Q}_t \leftarrow \mathcal{Q}_t \cup \beta(Q_i, R, \mu)$;
    $\mathcal{Q}^c \leftarrow$ ComputeCover$(\mathcal{Q}_F \cup \mathcal{Q}_t)$;
    $\mathcal{Q}_E \leftarrow \mathcal{Q}^c \backslash \mathcal{Q}_F$; // select unexplored queries of the cover
    $\mathcal{Q}_F \leftarrow \mathcal{Q}^c$;
return $\mathcal{Q}_F$
```

For any *fus*, CQ entailment is solvable in AC^0 for data complexity.[3] However, data complexity hides the complexity coming from the query: the size of the rewriting set can be exponential in the size of the original query. Most of the literature about rewriting techniques focuses on minimizing the *size* of the output rewritings. We will show that this size should not be a decisive criterion for comparing algorithms that output a union of CQs.

All covers of a given set have the same (minimal) cardinality. We now prove that this property can be extended to the covers of all sound and complete rewriting sets of Q, no matter of the rewriting technique used to compute these sets.

Theorem 2. *Let $\mathcal{R}$ be a fus, Q be a BCQ, and let $\mathcal{Q}$ be a sound and complete rewriting set of Q with $\mathcal{R}$. Any cover of $\mathcal{Q}$ is of minimal cardinality among sound and complete rewriting sets of Q with $\mathcal{R}$.*

Proof. Let $\mathcal{Q}_1$ and $\mathcal{Q}_2$ be two arbitrary sound and complete rewriting sets of Q with $\mathcal{R}$, and $\mathcal{Q}_1^c$ and $\mathcal{Q}_2^c$ be one of their respective covers. $\mathcal{Q}_1^c$ and $\mathcal{Q}_2^c$ are also sound and

[3] AC^0 is a subclass of LOGSPACE itself included in PTIME. Data complexity means that Q and $\mathcal{R}$ are fixed, thus the input is restricted to F.

complete, and are of smaller cardinality. We show that they have the same cardinality. Let $Q_1 \in \mathcal{Q}_1^c$. There exists $Q_2 \in \mathcal{Q}_2^c$ such that $Q_1 \leq Q_2$. If not, Q would be entailed by $F = Q_1$ and $\mathcal{R}$ since $\mathcal{Q}_1^c$ is a sound rewriting set of Q (and Q_1 maps to itself), but no elements of $\mathcal{Q}_2^c$ would map to F: thus, Q_2^c would not be complete. Similarly, there exists $Q_1' \in \mathcal{Q}_1^c$ such that $Q_2 \leq Q_1'$. Then $Q_1 \leq Q_1'$, which implies that $Q_1' = Q_1$ by assumption on Q_1^c. For all $Q_1 \in \mathcal{Q}_1^c$, there exists $Q_2 \in \mathcal{Q}_2^c$ such that $Q_1 \leq Q_2$ and $Q_2 \leq Q_1$. Such a Q_2 is unique: indeed, two such elements would be comparable for $\geq$, which is not possible by construction of $\mathcal{Q}_2^c$. The function associating Q_2 with Q_1 is thus a bijection from $\mathcal{Q}_1^c$ to $\mathcal{Q}_2^c$, which shows that these two sets have the same cardinality. □

From the previous observation, we conclude that any sound and complete rewriting algorithm can be "optimized" so that it outputs a set of rewritings of minimal cardinality. Please note that the algorithm presented in the sequel of this paper fullfils this property.

Furthermore, the proof of the preceding theorem shows that, given any two sound and complete rewriting sets of Q, there is a bijection from any cover of the first set to any cover of the second set such that two elements in relation are equivalent. However, these elements are not necessarily isomorphic (i.e., equal up to a variable renaming) because they may contain redundancies. It is well-known that the preorder induced by homomorphism on the set of all BCQs definable on some vocabulary is such that any equivalence class for this preorder possesses a unique element of minimal size (up to isomorphism), called its *core* (notion introduced for graphs, but easily transferable to queries). Every query can be transformed into its equivalent core by removing redundant atoms. From this remark and Theorem 2, we obtain:

Corollary 1. *Let $\mathcal{R}$ be a fus and Q be a BCQ. There is a unique sound and complete rewriting set of Q with $\mathcal{R}$ that has both minimal cardinality and elements of minimal size.*

4 Single-Piece Unification

We will now focus on rules with atomic head, which are often considered in the literature. Any rule can be decomposed into an equivalent set of rules with atomic head by introducing a new predicate gathering the variables of the original head, thus this restriction does not yield a loss in expressivity (e.g. [CGK08, BLMS09]).

What is simpler with these rules? The definition of a piece-unifier in itself does not change. The difference lies in the number of piece-unifiers that have to be considered in the backward chaining mechanism. We show it is sufficient to only keep most general single-piece unifiers. Moreover, the number of such unifiers is linear in the size of Q. Indeed, there is a unique way of associating any atom in Q with *head*(R).

4.1 Correctness of Rewriting Restricted to Most General Single-Piece Unifiers

We recall that, given substitutions s_1 and s_2, s_1 is said to be more general than s_2 if s_2 can be obtained from s_1 by composition with an additional substitution (i.e., there is s s.t. $s_2 = s \circ s_1$). Piece-unifiers can be compared via their substitutions, provided that they are defined on the *same* subset of Q.

Definition 9 (Most general piece-unifier). *Let Q be a CQ, R be a rule, and $\mu_1 = (Q', u_1)$, $\mu_2 = (Q', u_2)$ be two piece-unifiers of Q with R, defined on the same set of pieces $Q' \subseteq Q$. μ_1 is said to be* more general *than μ_2, noted $\mu_1 \geq \mu_2$, if u_1 is more general than u_2. Let μ be a piece-unifier of Q with R defined on $Q' \subseteq Q$, μ is called a most general piece-unifier if for all μ' piece-unifier of Q with R defined on Q', we have $\mu \geq \mu'$.*

Property 2. Let μ_1 and μ_2 be two piece-unifiers with $\mu_1 \geq \mu_2$. μ_1 and μ_2 have the same pieces.

Definition 10 (Single-piece unifier). *A piece-unifier $\mu = (Q', u)$ of a CQ Q with a rule R is a single-piece unifier if Q' is a piece of Q according to $T_Q(\mu)$.*

From Property 2, it follows that a single-piece unifier can be compared only with other single-piece unifiers. The next results show that it is sufficient to consider (1) most general piece-unifiers (Theorem 3) (2) single-piece unifiers, (Theorem 4) and finally most general single-piece unifiers (Theorem 5).

Property 3. Let $\mu_1 = (Q', u_1)$ and $\mu_2 = (Q', u_2)$ be two piece-unifiers such that $\mu_1 \geq \mu_2$. Then $\beta(Q, R, \mu_1) \geq \beta(Q, R, \mu_2)$.

Lemma 1. *If $Q_1 \geq Q_2$ then for all piece-unifiers μ_2 of Q_2 with R: either (i) $Q_1 \geq \beta(Q_2, R, \mu_2)$ or (ii) there is a piece-unifier μ_1 of Q_1 with R such that $\beta(Q_1, R, \mu_1) \geq \beta(Q_2, R, \mu_2)$.*

The following theorem follows from Property 3 and Lemma 1:

Theorem 3. *Given a BCQ Q and a set of rules $\mathcal{R}$, the set of $\mathcal{R}$-rewritings of Q obtained by considering exclusively most general piece-unifiers is sound and complete.*

Let $\mu = (Q', u)$ be a piece-unifier of Q with R. μ can be decomposed into several single-piece unifiers: for each piece P of Q according to $T_Q(\mu)$, there is a single-piece unifier (P, u_P) of Q with R where $u_P = u^R \cup u^{Q'}|_{vars(P)}$. However, applying successively each of these underlying single-piece unifiers may not lead to a CQ equivalent to $\beta(Q, R, \mu)$: the resulting query may be strictly more general than $\beta(Q, R, \mu)$, as the following example illustrates it.

Example 7. Let $R = p(x, y) \rightarrow q(x, y)$ and $Q = q(u, v) \wedge r(v, w) \wedge q(t, w)$. $\mu = (Q', u)$ with $Q' = \{q(u, v), q(t, w)\}$ and $u = \{(u, x), (v, y), (t, x), (w, y)\}$ is a piece-unifier of Q with R, which contains two pieces: $P_1 = \{q(u, v)\}$ and $P_2 = \{q(t, w)\}$. The rewriting of Q according to μ is $\beta(Q, R, \mu) = p(x, y) \wedge r(y, y)$. If we successively apply the two underlying single-piece unifiers, noted μ_{P_1} and μ_{P_2} (we note R' the fresh copy of R used for the second computation), we obtain $\beta(\beta(Q, R, \mu_{P_1}), R', \mu_{P_2}) = \beta(p(x, y) \wedge r(y, w) \wedge q(t, w), R', \mu_{P_2}) = p(x, y) \wedge r(y, y') \wedge p(x', y')$, which is strictly more general than $\beta(Q, R, \mu)$.

Property 4. For any piece-unifier μ of Q with R, there is a sequence of rewritings of Q with R using only single-piece unifiers and leading to a CQ Q^s such that $Q^s \geq \beta(Q, R, \mu)$.

From Lemma 1 and Property 4, it follows that:

Theorem 4. *Given a BCQ Q and a set of rules $\mathcal{R}$, the set of $\mathcal{R}$-rewritings of Q obtained by considering exclusively single-piece unifiers is sound and complete.*

Property 5. For any piece-unifier μ of Q with R, there is a sequence of rewritings of Q with R using only most general single-piece unifiers and leading to a CQ Q^s such that $Q^s \geq \beta(Q, R, \mu)$.

From Lemma 1 and Property 5, we obtain:

Theorem 5. *Given a BCQ Q and a set of rules $\mathcal{R}$, the set of $\mathcal{R}$-rewritings of Q obtained by considering exclusively most general single-piece unifiers is sound and complete.*

4.2 Computing all the Most General Single-Piece Unifiers

We first check that properties of most general unifiers in the classical logical meaning also hold for piece-unifiers (that operate on the same subset of Q): unicity of a most general piece-unifier up to a bijective variable renaming and existence of a most general piece-unifier.

Lemma 2. *If two piece-unifiers $\mu_1 = (Q', u_1)$ and $\mu_2 = (Q', u_2)$ are equivalent (i.e., $\mu_1 \geq \mu_2$ and $\mu_2 \geq \mu_1$), then μ_1 and μ_2 can be obtained from each other by a bijective variable renaming.*

Lemma 3. *If two piece-unifiers $\mu_1 = (Q', u_1)$ and $\mu_2 = (Q', u_2)$ are incomparable (i.e., $\mu_1 \not\geq \mu_2$ and $\mu_2 \not\geq \mu_1$), then there exists a piece-unifier $\mu = (Q', u)$ with $\mu \geq \mu_1$ and $\mu \geq \mu_2$.*

The next property follows from the two previous lemmas:

Property 6. Let Q be a CQ and R be a rule. For any $Q' \subseteq Q$, if Q' is a piece for a piece-unifier of Q with R, then Q' is part of a *unique* most general (single-piece) piece-unifier of Q with R (up to a bijective variable renaming).

Lemma 4. *Let Q be a CQ and R be a rule. For all atoms $a \in Q$, there is at most one $Q' \subseteq Q$ such that $a \in Q'$ and Q' is a piece for a piece-unifier of Q with R.*

Property 6 and the above lemma entail the following result:

Theorem 6. *Every atom in Q participates in at most one most general single-piece unifier of Q with R (up to a bijective variable renaming).*

It follows that the number of most general single-piece unifiers of Q with R is less or equal to the cardinality of Q.

To compute most general single-piece unifiers, we first introduce the notion of pre-(piece)-unifier of a set of atoms with the head of a rule. A pre-unifier is an adaptation of a classical logical unifier, that takes existential variables into account, and chooses to keep variables from the head of the rule in the resulting atom. To become a piece-unifier, a pre-unifier has to satisfy an additional constraint on *sep*(Q') (Condition 2 in piece-unifier definition).

Definition 11 (pre-unifier). *Let $Q' \subseteq Q$ and R be a rule. A pre-unifier u of Q' with R is a substitution of $\mathsf{fr}(R) \cup \mathsf{vars}(Q')$ by $\mathsf{terms}(\mathit{head}(R)) \cup \mathcal{C}$ such that:*

1. *for all $x \in \mathsf{fr}(R)$, $u(x) \in \mathsf{fr}(R) \cup \mathcal{C}$ (for technical convenience, we allow $u(x) = x$);*
2. *$u(Q') = u(\mathit{head}(R))$.*

Algorithm 2 computes a most general pre-unifier of a set of atoms, in a way similar to Robinson's algorithm.

Algorithm 2. MostGeneralPreUnifier

```
Data: $A$: a set of atoms with the same predicate $p$, $A \subseteq head(R) \cup Q$
Result: a most general pre-unifier of $A$ if it exists, otherwise Fail
$u \leftarrow \emptyset$;
foreach $i \in$ positions of $p$ do
    $E \leftarrow$ set of terms in position $i$ in $A$;
    if $E$ contains two constants or two existential variables or (a constant and an
    existential variable) or (a frontier variable and an existential variable) then
        return Fail
    if $E$ contains a constant or an existential variable then
        $t \leftarrow$ this term
    else
        // E contains at least one frontier variable
        $t \leftarrow$ one of these frontier variables
    $u' \leftarrow \{(v, t) \mid v$ is a variable in $E$ and $v \neq t\}$
    $u \leftarrow u' \circ u$;
    $A \leftarrow u'(A)$;
return $u$
```

The fact that an atom from Q participates in at most one most general single-piece unifier suggests an incremental method to compute these unifiers. Assume the head of R has predicate p. We start from each atom $a \in Q$ with predicate p and compute the subset of atoms from Q that would necessarily belong to the same piece as a; more precisely, we build Q' such that Q' and $\mathit{head}(R)$ can be pre-unified, then check if $\mathit{sep}(Q')$ satisfies the additional condition of a piece-unifier. If there is a piece-unifier of Q' built in this way with $\mathit{head}(R)$, all atoms in Q' can be removed from Q for the search of other single-piece unifiers; otherwise, a is removed from Q for the search of other single-piece unifiers but the other atoms in Q' still have to be taken into account.

Example 8. Let $R = q(x) \rightarrow p(x, y)$ and $Q = p(u, v) \wedge p(v, t)$. Let us start from $p(u, v)$: this atom is unifiable with $\mathit{head}(R)$ and $p(v, t)$ necessarily belongs to the same pre-unifier (if any) because v is mapped to the existential variable y; however, $\{p(u, v), p(v, t)\}$ is not unifiable with $\mathit{head}(R)$ because, since v occurs at the first and at the second position of a p atom, x and y should be unified, which is not possible since y is an existential variable; thus $p(u, v)$ does not belong to any pre-unifier with R. However, $p(v, t)$ still needs to be considered. Let us start from it: $p(v, t)$ is unifiable with $\mathit{head}(R)$ and forms its own piece because its single variable t mapped to an existential variable is not shared with another atom. There is thus one (most general) piece-unifier of Q with R, namely $(\{p(v, t)\}, \{(v, x), (t, y)\})$.

More precisely, Algorithm 3 first builds the subset A of atoms in Q with the same predicate as *head*(R). While A has not been emptied, it initializes a set Q' by picking an atom a in A, then repeats the following steps:

1. compute the most general pre-unifier of the current Q' with *head*(R) if it exists; if there is no pre-unifier, the attempt with a fails;
2. if the found pre-unifier satisfies the condition on *sep*(Q'), then it is a single-piece unifier, and all the atoms in Q' are removed from A;
3. otherwise, the algorithm tries to extend Q' with all atoms from Q containing a variable from *sep*(Q') that is mapped to an existential variable by the pre-unifier; if these atoms are in A, Q' can grow, otherwise the attempt with a fails.

Algorithm 3. Compute all most general single-piece unifiers

```
Data: a CQ $Q$ and an atomic-head rule $R$
Result: the set of most general single-piece unifiers of $Q$ with $R$
begin
    $U \leftarrow \emptyset$; // resulting set
    $A \leftarrow \{a \in Q \mid predicate(a) = predicate(head(R))\}$;
    while $A \neq \emptyset$ do
        $a \leftarrow$ choose an atom in $A$ ;
        $Q' \leftarrow \{a\}$ ;
        $u \leftarrow$MostGeneralPreUnifier($Q' \cup head(R)$) ;
        while $u \neq Fail$ and $sep(Q') \setminus T_Q(u) \neq \emptyset$ do
            $Q'' \leftarrow \{a' \in Q \mid a'$ contains a variable in $sep(Q') \setminus T_Q(u)\}$ ;
            if $Q'' \subseteq A$ then
                $Q' \leftarrow Q' \cup Q''$;
                $u \leftarrow$ MostGeneralPreUnifier($Q' \cup head(R)$)
            else
                $u \leftarrow Fail$
        if $u \neq Fail$ then
            $U \leftarrow U \cup \{u\}$ ;
            $A \leftarrow A \setminus Q'$
        else
            $A \leftarrow A \setminus \{a\}$
    return $U$
```

5 First Experiments and Perspectives

The global backward chaining algorithm (cf. Algorithm 1), based on most general single-piece unifiers (cf. Algorithm 3), has been implemented in Java. First experiments have been made with the same rules and queries as in [GOP11]. The considered sets of rules are translations from ontologies expressed in DL-Lite$_{\mathcal{R}}$ developed in several research projects, namely ADOLENA (A), STOCKEXCHANGE (S), UNIVERSITY (U) and VICODI (V). See [GOP11] for more details. The obtained rules have atomic head and body, which corresponds to the linear Datalog+/- fragment. Queries are canonical examples coming from projects in which the ontologies have been developed. For these first experiments, we compared our prototype to the NY* prototype, dedicated to linear Datalog+/- and part of the Nyaya system [GOP11]. The running time of both implementations are comparable. Concerning the sizes of the rewritings of the sample queries (*i.e.* the cardinalities of the output sets), they are equal for ontologies S, U and V, but not for ontology A (cf. Table 1, columns "final size"). Note that in [GOP11] the size of the rewritings output by NY* was already shown to be smaller than the one obtained with Requiem and QuOnto with substantial differences in some cases. Surprisingly, none of these systems computes a rewriting set of minimal size.

Table 1. Results with Nyaya and Piece-Based Rewriting

		NY*	Piece-Based Rewriting		
		final size	final size	# explorated	# generated
A	Q_1	249	27	457	1307
	Q_2	94	50	1598	4658
	Q_3	104	104	4477	13871
	Q_4	456	224	4611	15889
	Q_5	624	624	50508	231899
S	Q_1	6	6	6	9
	Q_2	2	2	48	256
	Q_3	4	4	64	536
	Q_4	4	4	240	1760
	Q_5	8	8	320	3320
U	Q_1	2	2	5	4
	Q_2	1	1	42	148
	Q_3	4	4	48	260
	Q_4	2	2	2196	9332
	Q_5	10	10	100	1280
V	Q_1	15	15	15	14
	Q_2	10	10	10	9
	Q_3	72	72	72	117
	Q_4	185	185	185	328
	Q_5	30	30	30	59

These first experimental results need to be extended by considering larger and more complex queries and rule bases, as well as comparing to other systems based on query rewriting. The size of the rewriting set should not be a decisive criterion (indeed, assuming that the systems are sound and complete, a minimal rewriting set is obtained by selecting most general elements, cf. Theorem 2). Therefore, other criteria have to be taken into account, such as the running time or the total number of CQs built during the rewriting process. As a first step in this direction, we indicate in Table 1 the number of explorated CQs (# explorated) and of generated CQs (# generated) with our system. The generated rewritings are all the rewritings built during the rewriting process (excluding the initial Q and possibly including some multi-occurrences of the same rewritings). Since we eliminate the subsumed rewritings at each step of the breadth-first algorithm, only some of the generated rewritings at a given step are explored at the next step.

Finally, our backward chaining mechanism is yet far from being optimized. Indeed, we have greatly simplified the unification operation —conceptually and algorithmically, which is important in itself— but in a way we have pushed the complexity into the composition of several rewritings. The question of whether it is worthwhile, when rules do not have atomic heads, to deal directly with them, still needs to be addressed.

Acknowledgements. We thank Giorgio Orsi for providing us with rule versions of the ontologies.

References

[Baa03] Baader, F.: Terminological cycles in a description logic with existential restrictions. In: IJCAI 2003, pp. 325–330 (2003)

[BLM10] Baget, J.-F., Leclère, M., Mugnier, M.-L.: Walking the decidability line for rules with existential variables. In: KR 2010, pp. 466–476. AAAI Press (2010)

[BLMS09] Baget, J.-F., Leclère, M., Mugnier, M.-L., Salvat, E.: Extending decidable cases for rules with existential variables. In: IJCAI 2009, pp. 677–682 (2009)

[BLMS11] Baget, J.-F., Leclère, M., Mugnier, M.-L., Salvat, E.: On rules with existential variables: Walking the decidability line. Artificial Intelligence 175(9-10), 1620–1654 (2011)

[BMRT11] Baget, J.-F., Mugnier, M.-L., Rudolph, S., Thomazo, M.: Walking the complexity lines for generalized guarded existential rules. In: IJCAI 2011, pp. 712–717 (2011)

[CGK08] Calì, A., Gottlob, G., Kifer, M.: Taming the infinite chase: Query answering under expressive relational constraints. In: KR 2008, pp. 70–80 (2008)

[CGL$^+$07] 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)

[CGL09] Calì, A., Gottlob, G., Lukasiewicz, T.: A general datalog-based framework for tractable query answering over ontologies. In: PODS 2009, pp. 77–86 (2009)

[CGP10] Calì, A., Gottlob, G., Pieris, A.: Query Answering under Non-guarded Rules in Datalog+/-. In: Hitzler, P., Lukasiewicz, T. (eds.) RR 2010. LNCS, vol. 6333, pp. 1–17. Springer, Heidelberg (2010)

[GOP11] Gottlob, G., Orsi, G., Pieris, A.: Ontological queries: Rewriting and optimization. In: ICDE 2011, pp. 2–13 (2011)

[GS12] Gottlob, G., Schwentick, T.: Rewriting ontological queries into small nonrecursive datalog programs. In: KR 2012 (2012)

[KKZ11] Kikot, S., Kontchakov, R., Zakharyaschev, M.: Polynomial Conjunctive Query Rewriting under Unary Inclusion Dependencies. In: Rudolph, S., Gutierrez, C. (eds.) RR 2011. LNCS, vol. 6902, pp. 124–138. Springer, Heidelberg (2011)

[KLMT12] König, M., Leclère, M., Mugnier, M.-L., Thomazo, M.: A Sound and Complete Backward Chaining Algorithm for Existential Rules. Technical Report RR-12016, LIRMM, GraphIK - INRIA Sophia Antipolis (2012)

[KR11] Krötzsch, M., Rudolph, S.: Extending decidable existential rules by joining acyclicity and guardedness. In: IJCAI 2011, pp. 963–968 (2011)

[Mug11] Mugnier, M.-L.: Ontological Query Answering with Existential Rules. In: Rudolph, S., Gutierrez, C. (eds.) RR 2011. LNCS, vol. 6902, pp. 2–23. Springer, Heidelberg (2011)

[PUHM09] Pérez-Urbina, H., Horrocks, I., Motik, B.: Efficient Query Answering for OWL 2. In: Bernstein, A., Karger, D.R., Heath, T., Feigenbaum, L., Maynard, D., Motta, E., Thirunarayan, K. (eds.) ISWC 2009. LNCS, vol. 5823, pp. 489–504. Springer, Heidelberg (2009)

[RA10] Rosati, R., Almatelli, A.: Improving query answering over DL-Lite ontologies. In: KR 2010 (2010)

[SM96] Salvat, E., Mugnier, M.-L.: Sound and Complete Forward and Backward Chainings of Graph Rules. In: Eklund, P., Mann, G.A., Ellis, G. (eds.) ICCS 1996. LNCS (LNAI), vol. 1115, pp. 248–262. Springer, Heidelberg (1996)

Deriving Predicate Statistics for Logic Rules

Senlin Liang and Michael Kifer

Department of Computer Science
Stony Brook University
Stony Brook, NY 11794, USA
{sliang,kifer}@cs.stonybrook.edu

Abstract. Database query optimizers rely on data statistics in selecting query execution plans and rule-based systems can greatly benefit from such optimizations as well. To this end, one first needs to collect data statistics for base and propagate them to derived predicates. However, there are two difficulties: dependencies among arguments and recursion. Earlier we developed an algorithm, called SDP, for estimating Datalog query sizes efficiently by estimating statistical dependency for both base and derived predicates [16]. Base predicate statistics were summarized as *dependency matrices*, while the statistics for derived predicate were estimated by abstract evaluation of rules over the dependency matrices. This previous work had several limitations. First, it only considered Datalog predicates. Second, only predicates of arity at most 2 were allowed—a very serious limitation of the approach. The present paper extends SDP to general rules and n-ary predicates. It also handles negation and mutual recursions as well as other operations. We also report on our experiments with SDP.

Keywords: derived predicate statistics, cost estimate, query optimization, rule-based systems, logic programming.

1 Introduction

Database query optimizers depend on accurate and fast algorithms for estimating predicate (relation) sizes, which in turn depend on *joint data distributions* of values in different arguments. Traditionally, query optimizers estimate these sizes using statistical summaries for base predicates and propagate them to relational expressions assuming *argument independence* [5]. Assuming that distributions of values in different arguments of a predicate are independent, joint data distribution can be derived relatively easily.

To make size estimates practical, data distributions must be summarized accurately and efficiently. Histograms is one such summarization technique that is in wide use in all major database systems (DB2, Oracle, Microsoft, etc.). Briefly, histograms groups values of *similar* frequencies into buckets and estimates the frequencies of values in each bucket in a uniform and efficient way. Different types of histograms have been proposed in the literature, which differ in their complexity, cost, and accuracy. As usual, accuracy comes at the expenses of time and space, and the problem of achieving the balance between these conflicting requirements was discussed in [13]. Comprehensive surveys and classification of various histograms can be found in [10,21].

There are several challenges in applying cost-based query optimization for rule-based systems. First, the argument independence assumption is rarely true for real world datasets, so size estimates based on histograms can be off by orders of magnitude [29]. It has been shown that estimation errors grow exponentially with the number of joins involved [12,16]. For rules, this problem is exacerbated by the presence of recursive predicates and it is the focus of this paper. Second, optimization algorithms have to take into account indexing and the basic operations performed by logic engines.

In [16], we addressed the problems associated with the argument independence assumption, but only for Datalog predicates. Further limiting the approach, only unary and binary predicates were allowed. In that work, we used dependency matrices to store predicate dependency statistics and an algorithm, called SDP, to compute dependency matrices for derived predicates. To address the problems with [16], the present paper extends both dependency matrix and SDP to permit n-ary predicates and more general types of rules, including mutual recursion, negation, selection, union, intersection and join. These extensions are essential in order for SDP to be useful for optimizing logic engines such as XSB [24,30], FLORA-2 [34], and SILK [33]. We also note that SDP-estimates of the distribution statistics are independent of optimization algorithms. Since virtually all cost-based optimizers use size estimates, our cost estimates may benefit most of such optimizers.

The rest of this paper is organized as follows. Section 2 extends dependency matrices to arbitrary n-ary predicates and defines related concepts. Section 3 extends SDP. Section 4 describes our experiments. Section 5 discusses related works and Section 6 concludes the paper.

2 Dependency Matrix

Dependency matrices were introduced in [16] to represent joint distribution of values in predicate arguments. To compute dependency matrices, we developed an algorithm, called SDP, which was limited to binary predicates and certain types of rules. This section extends dependency matrices to n-ary predicates.

Consider a n-ary predicate $p(x_1, \ldots, x_n)$. If p is a base predicate, then its associated set of facts will be denoted by $factset(p)$. The *value sequence*, $\bar{v}_d$ $(1 \leq d \leq n)$, is the *sorted* sequence of x_d-values that are present in $factset(p)$, and $\bar{v}_d^i$ is the i-th value of $\bar{v}_d$. The *frequency*, $\bar{f}_d^i$, of $\bar{v}_d^i$ is the number of facts in $factset(p)$ with $x_d = \bar{v}_d^i$. We will use $\bar{v}^i$ to denote the i-th element of a sequence $\bar{v}$, "$\bullet$" to denote sequence concatenation, and $\|...\|$ to denote the length of a sequence or the cardinality of a set. Since we are dealing with discrete values in finite relations, all argument values can be assumed to be integers. Without loss of generality, we adopt this assumption in the sequel, for simplicity.

Definition 1. *Given a fact-set for an n-ary predicate $p(x_1, \ldots, x_n)$, the* ***data distribution****, $\mathcal{T}_d$, for x_d is the sequence of value-frequency pairs $[(\bar{v}_d^1, \bar{f}_d^1), \ldots, (\bar{v}_d^m, \bar{f}_d^m)]$ where $m = \|\bar{v}_d\|$.* □

Data distribution is the basis for size estimation in all cost-based query optimizers, but this information is normally too large to store and use efficiently. One key step of all size estimation algorithms is to partition data distributions into distribution segments and summarize these segments in such a way that they can be approximated efficiently both in time and space. We proposed dependency matrices in [16] as one such summarization technique which keeps argument dependency information. There, data distributions are partitioned using one particularly popular partition rule called the *maxdiff* rule [10]. However, our size estimation algorithms do not depend on the choice of any particular partition rule and we can simply assume that distributions are partitioned according to *some* partition rule.

Example 1. Consider a predicate $p(x_1, x_2)$ with the following fact-set $factset(p)$:

```
p(2,2). p(3,7). p(3,8). p(4,4). p(5,5). p(5,7).
p(5,8). p(6,6). p(7,5). p(7,6). p(8,1). p(8,3).
```

Its data distributions are $\mathcal{T}_1 = [(2,1), (3,2), (4,1), (5,3), (6,1), (7,2), (8,2)]$ and $\mathcal{T}_2 = [(1,1), (2,1), (3,1), (4,1), (5,2), (6,2), (7,2), (8,2)]$. Their three partition segments using the maxdiff rule are $\mathcal{T}_1 = [(2,1), (3,2), (4,1)] \bullet [(5,3)] \bullet [(6,1), (7,2), (8,2)]$ and $\mathcal{T}_2 = [(1,1)] \bullet [(2,1), (3,1), (4,1)] \bullet [(5,2), (6,2), (7,2), (8,2)]$. □

Consider a distribution $\mathcal{T}$ and its partition segments $\mathcal{T}^1$, ..., $\mathcal{T}^n$, each $\mathcal{T}^i$ has three parameters: *floor, ceiling, size* which are the minimal argument value, the maximal argument value, and the number of argument values contained in $\mathcal{T}^i$. The set of values that are contained in $\mathcal{T}^i$ is approximated as $vals(\mathcal{T}^i) = \{v \in integers \mid \mathcal{T}^i.floor \leq v \leq \mathcal{T}^i.ceiling\}$. These three parameters constitute the summary of a partition segment and they are stored by dependency matrices.

Example 2. Consider the fact-set of Example 1, it can be represented by a 2-dimensional *fact-matrix* $\mathcal{F}$ shown in Figure 1(a), where $\mathcal{F}(x, y) = 1$ if and only if $p(x, y)$ is true, i.e., $p(x, y) \in factset(p)$. If we segment the rows of $\mathcal{F}$ according to the partition of the distribution $\mathcal{T}_1$, i.e., $|2, 3, 4|5|6, 7, 8|$, and the columns according to the partition of $\mathcal{T}_2$, i.e., $|1|2, 3, 4|5, 6, 7, 8|$, we obtain a partition of the

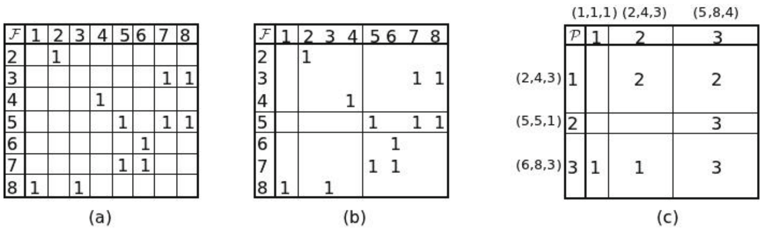


Fig. 1. A Dependency Matrix

matrix into rectangular regions as shown in Figure 1(b). The *dependency matrix* shown in Figure 1(c) is a *summary* of Figure 1(b): each region of Figure 1(b) is reduced to a single square in Figure 1(c) and the number in the square represents the number of 1's in the corresponding region of (b). In addition, the three parameters of each distribution segment are stored in (c) in the form of $(floor, ceiling, size)$. For instance, the floor, ceiling, and size of the last distribution segment of $\mathcal{T}_1$ are 6, 8, and 3 respectively. Therefore, the bottom coordinate on the vertical axis is annotated with (6,8,3). □

Definition 2. *Let $p(x_1, \ldots, x_n)$ be an n-ary predicate and let $\mathcal{T}_i$ $(1 \leq i \leq n)$ be the distribution for x_i. Suppose each $\mathcal{T}_i$ is partitioned into β_i segments $\mathcal{T}_i^{j_i}$, $1 \leq j_i \leq \beta_i$. The* ***dependency matrix*** *for* `p`*, denoted $\boldsymbol{M}\langle p\rangle$, is a matrix whose $(j_1, \ldots, j_n)$-th element, $\boldsymbol{M}\langle p\rangle(j_1, \ldots, j_n)$, is*

$$\|\{p(x_1, \ldots, x_n) \in factset(p) \mid \wedge_{1 \leq i \leq n}\, x_i \in vals(\mathcal{T}_i^{j_i})\}\|$$

In addition, the j_i-th coordinate on its i-th axis, denoted $\boldsymbol{M}\langle p\rangle_i^{j_i}$, is associated with three parameters: ***floor****,* ***ceiling****, and* ***size*** *whose values are the same as the corresponding values associated with the distribution segment $\mathcal{T}_i^{j_i}$.* □

As in Definition 2, we often use $\mathbf{M}\langle p\rangle_i$ to denote the i-th axis of a matrix $\mathbf{M}\langle p\rangle$ and $\mathbf{M}\langle p\rangle_i^{j_i}$ to denote the j_i-th coordinate on $\mathbf{M}\langle p\rangle_i$. For instance, if $\mathbf{M}\langle p\rangle$ is the 2-dimensional matrix in Figure 1(c), then $\mathbf{M}\langle p\rangle_1$ denotes the first axis of $\mathbf{M}\langle p\rangle$, i.e., the vertical axis, and $\mathbf{M}\langle p\rangle_1^3$ denotes the third coordinate on $\mathbf{M}\langle p\rangle_1$.

From Definition 2, we know that the matrix element $\mathbf{M}\langle p\rangle(j_1, \ldots, j_n)$ summarizes $vals(\mathcal{T}_1^{j_1}) \times \ldots \times vals(\mathcal{T}_n^{j_n})$, and $\mathbf{M}\langle p\rangle(j_1, \ldots, j_n)$ stores the number of $(x_1, \ldots, x_n)$-values that are in the fact-set and summarized by this matrix element. For instance of Example 2(c), $\mathbf{M}\langle p\rangle(1,3)$ summarizes $\{2,3,4\} \times \{5,6,7,8\}$, of which only $p(3,7)$ and $p(3,8)$ are in the fact-set. Thus, $\mathbf{M}\langle p\rangle(1,3) = 2$.

Given a $\beta_1 \times \ldots \times \beta_n$ dependency matrix $\mathbf{M}\langle p\rangle$, the size estimate of `p`, $size(p)$ or $size(\mathbf{M}\langle p\rangle)$, can be computed as the sum of all dependency matrix elements, i.e., $size(p) = \sum_{i_1, \ldots, i_n} \mathbf{M}\langle p\rangle(i_1, \ldots, i_n)$. Note that if `p` is a base predicate then $size(p)$ is its actual size. Therefore, one could estimate the size of a predicate by computing its dependency matrix.

3 Statistics for Derived Predicates

The dependency matrices for the base predicates can be computed according to some partition rule, such as the maxdiff rule [11] and Definition 2, as illustrated in Example 2. This section extends our earlier *SDP* algorithm [16], which estimated sizes for various types of *binary* derived predicates, to handle more general rule formats, including arbitrary n-ary predicates.

Consider a $\beta_1^p \times \ldots \times \beta_m^p$ dependency matrix $\mathbf{M}\langle p\rangle$, a $\beta_1^q \times \ldots \times \beta_n^q$ dependency matrix $\mathbf{M}\langle q\rangle$, and two integers $1 \leq d_p \leq m$ and $1 \leq d_q \leq n$. We say that the d_p-th axis of $\mathbf{M}\langle p\rangle$, $\mathbf{M}\langle p\rangle_{d_p}$, and the d_q-th axis of $\mathbf{M}\langle q\rangle$, $\mathbf{M}\langle q\rangle_{d_q}$, are **aligned** if $\beta_{d_p} = \beta_{d_q}$, $\mathbf{M}\langle p\rangle_{d_p}^i.floor = \mathbf{M}\langle q\rangle_{d_q}^i.floor$, and $\mathbf{M}\langle p\rangle_{d_p}^i.ceiling = \mathbf{M}\langle q\rangle_{d_q}^i.ceiling$

for all $1 \le i \le \beta_{d_p}$. That is, $\mathbf{M}\langle p\rangle_{d_p}$ and $\mathbf{M}\langle q\rangle_{d_q}$ are aligned if all their coordinates have the same floor and ceiling parameters. For any pair of dependency matrices $\mathbf{M}\langle p\rangle$ and $\mathbf{M}\langle q\rangle$, and a pair of axis indexes d_p and d_q, we can always make $\mathbf{M}\langle p\rangle_{d_p}$ and $\mathbf{M}\langle q\rangle_{d_q}$ aligned by *refining* the associated partition segments. Details of these operations can be found in [15].

Let $\mathbf{M}\langle p\rangle$ and $\mathbf{M}\langle q\rangle$ both be $\beta_1 \times \ldots \times \beta_n$ matrices. It follows directly from the definitions that $\mathbf{M}\langle p\rangle(j_1, ..., j_n)$ and $\mathbf{M}\langle q\rangle(j_1, ..., j_n)$ summarize the same values for all $j_1, ..., j_n$ if and only if $\mathbf{M}\langle p\rangle_d$ and $\mathbf{M}\langle q\rangle_d$ are aligned for all $d = 1, ..., n$. Since, as noted above, matrices can always be aligned, it follows that any pair matrices can be refined so that they will summarize exactly the same values.

The rest of this section presents the algorithms for computing $\mathbf{M}\langle q\rangle$ for different operations.

3.1 Dependency Matrix for Selection

Let $\mathbf{M}\langle r\rangle$ be the n-dimensional $\beta_1 \times ... \times \beta_n$ dependency matrix for $r(x_1, ..., x_n)$. We consider the following two types of selections: *selection with constant equalities* (`s1`) and *selection with range restrictions* (`s2`)

```
p(X1,...,Xn) :- r(X1,...,Xn), Xd = val.          (s1)
p(X1,...,Xn) :- r(X1,...,Xn), Xd >= l, Xd =< h.   (s2)
```

where $1 \le d \le n$. Observing that `s1` is a special case of `s2` when $l = h = val$, we can focus on computing $\mathbf{M}\langle p\rangle$ for `s2`. $\mathbf{M}\langle p\rangle$ can be computed by extracting the part of $\mathbf{M}\langle r\rangle$ whose x_d-values satisfy $l \le x_d \le h$, as *select*$(\mathbf{M}\langle r\rangle, d, l, h)$ in Algorithm 1.

Line 1 of the algorithm computes $perc(d, i_d)$ — the percentage of the i_d-th coordinate on d-th axis that satisfies the selection condition, and line 1 computes the dependency matrix values. Then, line 1 copies floor and ceiling parameters from $\mathbf{M}\langle r\rangle$ to $\mathbf{M}\langle p\rangle$, and line 1 computes the size parameters.

```
1 $perc(d, i_d) := \max\{\frac{\min\{h, \mathbf{M}\langle r\rangle_d^{i_d}.ceiling\} - \max\{l, \mathbf{M}\langle r\rangle_d^{i_d}.floor\} + 1}{\mathbf{M}\langle r\rangle_d^{i_d}.ceiling - \mathbf{M}\langle r\rangle_d^{i_d}.floor + 1}, 0\}$ for $1 \le i_d \le \beta_d$;
2 $\mathbf{M}\langle p\rangle(i_1, ..., i_n) := \mathbf{M}\langle r\rangle(i_1, ..., i_n) \times perc(d, i_d)$ for $1 \le i_1 \le \beta_1, ..., 1 \le i_n \le \beta_n$;
3 for all $1 \le j \le n$ and $1 \le i_j \le \beta_j$ do
4     $\mathbf{M}\langle p\rangle_j^{i_j}.\{floor, ceiling\} := \mathbf{M}\langle r\rangle_j^{i_j}.\{floor, ceiling\}$;
5     $perc(j, i_j) := \frac{\sum_{1 \le k \le n, k \ne j, 1 \le l_k \le \beta_k} \mathbf{M}\langle p\rangle(l_1, ..., l_{j-1}, i_j, l_{j+1}, ..., ln)}{\sum_{1 \le k \le n, k \ne j, 1 \le l_k \le \beta_k} \mathbf{M}\langle r\rangle(l_1, ..., l_{j-1}, i_j, l_{j+1}, ..., ln)}$;
6     $\mathbf{M}\langle p\rangle_j^{i_j}.size := \mathbf{M}\langle r\rangle_j^{i_j}.size \times perc(j, i_j)$;
7 end
8 return $M\langle p\rangle$;
```

Algorithm 1. *select*$(\mathbf{M}\langle r\rangle, d, l, h)$

3.2 Dependency Matrix for Union

Consider the predicate `p` defined as the union of `r` and `s`:

```
p(X1,...,Xn) :- r(X1,...,Xn).
p(X1,...,Xn) :- s(X1,...,Xn).
```

Before performing the union, we assume that $\mathbf{M}\langle r\rangle$ and $\mathbf{M}\langle s\rangle$ are both $\beta_1\times...\times\beta_n$ dependency matrices and they summarize exactly the same sets of values. As noted earlier, this means that $\mathbf{M}\langle r\rangle_d$ and $\mathbf{M}\langle s\rangle_d$ are aligned for $1\leq d\leq n$.

The dependency matrix for the union is also a n-dimensional $\beta_1\times ... \times\beta_n$ dependency matrix that can be computed by the operation *union*$(\mathbf{M}\langle r\rangle,\mathbf{M}\langle s\rangle)$ given in Algorithm 2. Lines 2 to 2 compute the floor, ceiling, and size parameters in a natural way, where line 2 is based on the *containment assumption* [25]. This assumption states that each individual value in a smaller set always matches some value from the larger set. It is a common assumption in the literature on size estimation. Then line 2 computes the values of $\mathbf{M}\langle p\rangle$ by integrating information from $\mathbf{M}\langle r\rangle$ and $\mathbf{M}\langle s\rangle$.

We should note that besides taking the maximum on line 2 of the algorithm, there are other ways of performing point-wise integration of summaries from dependency matrices [15].

```
1 for all $1\leq d\leq n$ and $1\leq i_d\leq\beta_d$ do                /* parameters */
2     $\mathbf{M}\langle p\rangle_d^{i_d}.\{floor, ceiling\} := \mathbf{M}\langle r\rangle_d^{i_d}.\{floor, ceiling\}$;
3     $\mathbf{M}\langle p\rangle_d^{i_d}.size := \max\{\mathbf{M}\langle r\rangle_d^{i_d}.size, \mathbf{M}\langle s\rangle_d^{i_d}.size\}$;
4 end
5 $\mathbf{M}\langle p\rangle(i_1,...,i_n) := \max\{\mathbf{M}\langle r\rangle(i_1,...,i_n), \mathbf{M}\langle s\rangle(i_1,...,i_n)\}$ for $1\leq i_1\leq\beta_1,...,1\leq$
  $i_n\leq\beta_n$;
6 return $M\langle p\rangle$;
```

Algorithm 2. *union*$(\mathbf{M}\langle r\rangle,\mathbf{M}\langle s\rangle)$

3.3 Dependency Matrix for Intersection

Consider the predicate `p` defined as the intersection of `r` and `s`:

```
p(X1,...,Xn) :- r(X1,...,Xn), s(X1,...,Xn).
```

Similar to the case of union, we assume that $\mathbf{M}\langle r\rangle$ and $\mathbf{M}\langle s\rangle$ are both $\beta_1\times...\times\beta_n$ dependency matrices and $\mathbf{M}\langle r\rangle_d$ and $\mathbf{M}\langle s\rangle_d$ are aligned for $1\leq d\leq n$.

Algorithm 3 computes the n-dimensional $\beta_1\times ... \times\beta_n$ dependency matrix for the above intersection, *intersect*$(\mathbf{M}\langle r\rangle,\mathbf{M}\langle s\rangle)$. The outline of the algorithm is the same as in the previous cases, but the details are simpler. Lines 3 and 3 are based on the containment assumption [25].

```
1 for all $1 \leq d \leq n$ and $1 \leq i_d \leq \beta_d$ do
2     $\mathbf{M}\langle p\rangle_d^{i_d}.\{floor, ceiling\} = \mathbf{M}\langle r\rangle_d^{i_d}.\{floor, ceiling\}$ ;        /* parameters */
3     $\mathbf{M}\langle p\rangle_d^{i_d}.size = \min\{\mathbf{M}\langle r\rangle_d^{i_d}.size, \mathbf{M}\langle r\rangle_d^{i_d}.size\}$;
4 end
5 $\mathbf{M}\langle p\rangle(i_1,...,i_n) = \min\{\mathbf{M}\langle r\rangle(i_1,...,i_n), \mathbf{M}\langle s\rangle(i_1,...,i_n)\}$ for $1 \leq i_1 \leq \beta_1,...,1 \leq i_n \leq \beta_n$;
6 return $M\langle p\rangle$;
```

Algorithm 3. $intersect(\mathbf{M}\langle r\rangle, \mathbf{M}\langle s\rangle)$

3.4 Dependency Matrix for Projection

Consider a projection on predicate $r(x_1,...,x_n)$ that projects out a subset of arguments $out\_args = \{x_{out_1},...,x_{out_k}\}$ and leaves $in\_args = \{x_{in_1},...,x_{in_m}\}$. Such a projection can be defined using a rule of the form

```
p(X_in1, ..., X_inm) :- r(X_1, ..., X_n).
```

Let $\mathbf{M}\langle r\rangle$ be a n-dimensional $\beta_1 \times ... \times \beta_n$ dependency matrix. The m-dimensional $\beta_{in_1} \times ... \times \beta_{in_m}$ dependency matrix, $project(\mathbf{M}\langle r\rangle, in\_args, out\_args)$, is computed via Algorithm 4, below. As in the case of union , alternative approaches to compute dependency matrix values can be founded in [15].

3.5 Dependency Matrix for Join

The most complicated part of any size estimation algorithm in a query optimizer is finding accurate estimates for the result of a join. Consider the following join:

```
p(Z1,...,Zk,Xk+1,...,Xm,Yk+1,...,Yn) :- r(Z1,...,Zk,Xk+1,...,Xm),
                                        s(Z1,...,Zk,Yk+1,...,Yn).
```

```
1 $\mathbf{M}\langle p\rangle_d^{i_d}.\{floor, ceiling, size\} := \mathbf{M}\langle r\rangle_{in_d}^{i_d}.\{floor, ceiling, size\}$ for $1 \leq d \leq m$ and $1 \leq i_d \leq \beta_{in_d}$;
2 $\mathbf{M}\langle p\rangle(i_1,...,i_m) := max\{\mathbf{M}\langle r\rangle(j_1,...,j_n) \mid i_1 = j_{in_1},...,i_m = j_{in_m}\}$ for $1 \leq i_1 \leq \beta_{in_1},...,1 \leq i_m \leq \beta_{in_m}$;
3 return $M\langle p\rangle$;
```

Algorithm 4. $project(\mathbf{M}\langle r\rangle, in\_args, out\_args)$

where r and s join on k arguments, $z_1,...,z_k$, which are listed first, for simplicity. Assume $\mathbf{M}\langle r\rangle$ and $\mathbf{M}\langle s\rangle$ are m-dimensional $\beta_1^r \times ... \times \beta_m^r$ and n-dimensional $\beta_1^s \times ... \times \beta_n^s$ dependency matrices respectively, and the first k axes of $\mathbf{M}\langle r\rangle$ and $\mathbf{M}\langle s\rangle$ are aligned. The $(m+n-k)$-dimensional dependency matrix for p can be computed as $join(\mathbf{M}\langle r\rangle, \mathbf{M}\langle s\rangle)$ by Algorithm 5. The abstract evaluation of the join is inspired by the *sort-merge-join* algorithm (e.g., [14]).

The overall outline of the algorithm is the same as before, but the details of computation of the values for the individual members of the matrix are somewhat involved.

```
1 $\beta_d^p := \beta_d^r$ for $1 \le d \le m$;   $\beta_d^p := \beta_{d-m+k}^s$ for $m < d \le m{+}n{-}k$;
2 $\mathbf{M}\langle p\rangle_d^{i_d}.\{floor, ceiling\} := \mathbf{M}\langle r\rangle_d^{i_d}.\{floor, ceiling\}$ and $\mathbf{M}\langle p\rangle_d^{i_d}.size :=$
  $\min\{\mathbf{M}\langle r\rangle_d^{i_d}.size, \mathbf{M}\langle s\rangle_d^{i_d}.size\}$ for all $1 \le d \le k$ and $1 \le i_d \le \beta_d^p$;
3 $\mathbf{M}\langle p\rangle_d^{i_d}.\{floor, ceiling, size\} := \mathbf{M}\langle r\rangle_d^{i_d}.\{floor, ceiling, size\}$ for all $k < d \le m$
  and $1 \le i_d \le \beta_d^p$;
4 $\mathbf{M}\langle p\rangle_d^{i_d}.\{floor, ceiling, size\} := \mathbf{M}\langle s\rangle_{d-m+k}^{i_d}.\{floor, ceiling, size\}$ for all
  $m < d \le m{+}n{-}k$ and $1 \le i_d \le \beta_d^p$;
5 $\mathbf{M}\langle r\rangle' := project(\mathbf{M}\langle r\rangle, \{z_1, ..., z_k\}, \{x_{k+1}, ..., x_m\})$;
6 $\mathbf{M}\langle s\rangle' := project(\mathbf{M}\langle s\rangle, \{z_1, ..., z_k\}, \{y_{k+1}, ..., y_n\})$;
7 $\mathbf{M}\langle p\rangle(i_1, ..., i_{m+n-k}) := \min\{\mathbf{M}\langle r\rangle'(i_1, ..., i_k), \mathbf{M}\langle s\rangle'(i_1, ..., i_k)\} \times \frac{\mathbf{M}\langle r\rangle(i_1,...,i_m)}{\mathbf{M}\langle r\rangle'(i_1,...,i_k)} \times$
  $\frac{\mathbf{M}\langle s\rangle(i_1,...,i_k,i_{m+1},...,i_{m+n-k})}{\mathbf{M}\langle s\rangle'(i_1,...,i_k)}$ for all $1 \le i_1 \le \beta_1^p, ..., 1 \le i_{m+n-k} \le \beta_{m+n-k}^p$;
8 return $M\langle p\rangle$;
```

Algorithm 5. $join(\mathbf{M}\langle r\rangle, \mathbf{M}\langle s\rangle)$

They are inspired by [4], and we briefly describe the computation of dependency matrix values below and details can be founed in [15].

Lines 5 to 5 compute $\mathbf{M}\langle r\rangle'$ and $\mathbf{M}\langle s\rangle'$ as projections of $\mathbf{M}\langle r\rangle$ and $\mathbf{M}\langle s\rangle$ which keep only join arguments. $\mathbf{M}\langle r\rangle'(i_1, ..., i_k)$ is the estimated number of $(z_1, ..., z_k)$-values such that $z_d \in vals(\mathbf{M}\langle r\rangle_d^{i_d})$ for $1 \le d \le k$, and $\mathbf{M}\langle s\rangle'(i_1, ..., i_k)$ is the estimated number of $(z_1, ..., z_k)$-values such that $z_d \in vals(\mathbf{M}\langle s\rangle_d^{i_d})$ for $1 \le d \le k$. Now, consider each fixed $(z_1, ..., z_k)$-value. There are, on average, $\frac{\mathbf{M}\langle r\rangle(i_1,...,i_m)}{\mathbf{M}\langle r\rangle'(i_1,...,i_k)}$ facts of the form $r(z_1, ..., z_k, x_{k+1}, ..., x_m)$ that are summarized by $\mathbf{M}\langle r\rangle(i_1, ..., i_m)$, and $\frac{\mathbf{M}\langle s\rangle(i_1,...,i_k,i_{m+1},...,i_{m+n\text{-}k})}{\mathbf{M}\langle s\rangle'(i_1,...,i_k)}$ facts of the form $s(z_1, ..., z_k, y_{k+1}, ..., y_n)$ that are summarized by $\mathbf{M}\langle s\rangle(i_1, ..., i_k, i_{m+1}, ..., i_{m+n\text{-}k})$. Thus, the number of resulting $p(z_1, ..., z_k, x_{k+1}, ..., x_m, y_{k+1}, ..., y_n)$ from the join on each $(z_1, ..., z_k)$-values can be estimated as $\frac{\mathbf{M}\langle r\rangle(i_1,...,i_m)}{\mathbf{M}\langle r\rangle'(i_1,...,i_k)} \times \frac{\mathbf{M}\langle s\rangle(i_1,...,i_k,i_{m+1},...,i_{m+n-k})}{\mathbf{M}\langle s\rangle'(i_1,...,i_k)}$. The containment assumption [25] says the number of such $(z_1, ..., z_k)$-values can be estimated as $\min\{\mathbf{M}\langle r\rangle'(i_1, ..., i_k), \mathbf{M}\langle s\rangle'(i_1, ..., i_k)\}$, which gives the formula at line 5.

3.6 Dependency Matrix for Cross Product

Consider the following cross product

```
p(X1,...,Xm,Y1,...,Yn) :- r(X1,...,Xm), s(Yk,...,Yn).
```

```
1 $\beta_d^p := \beta_d^r$ for $1 \le d \le m$;   $\beta_{d+m}^p := \beta_d^s$ for $1 \le d \le n$;
2 $\mathbf{M}\langle p\rangle_d^{i_d}.\{floor, ceiling, size\} := \mathbf{M}\langle r\rangle_d^{i_d}.\{floor, ceiling, size\}$ for all $1 \le d \le m$
  and $1 \le i_d \le \beta_d^p$ ;                                   /* parameters */
3 $\mathbf{M}\langle p\rangle_d^{i_d}.\{floor, ceiling, size\} := \mathbf{M}\langle s\rangle_{d-m}^{i_d}.\{floor, ceiling, size\}$ for all
  $m < d \le m{+}n$ and $1 \le i_d \le \beta_d^p$;
4 $\mathbf{M}\langle p\rangle(i_1, ..., i_{m+n}) := \mathbf{M}\langle r\rangle(i_1, ..., i_m) \times \mathbf{M}\langle s\rangle(i_{m+1}, ..., i_{m+n})$ for all $1 \le i_1 \le \beta_1^p$,
  $..., 1 \le i_{m+n} \le \beta_{m+n}^p$ ;                              /* values */
5 return $M\langle p\rangle$;
```

Algorithm 6. $product(\mathbf{M}\langle r\rangle, \mathbf{M}\langle s\rangle)$

where $\mathbf{M}\langle r\rangle$ and $\mathbf{M}\langle s\rangle$ are m-dimensional $\beta_1^r \times ... \times \beta_m^r$ and n-dimensional $\beta_1^s \times ... \times \beta_n^s$ dependency matrices respectively. The $(m+n)$-dimensional $\beta_1^p \times ... \times \beta_{m+n}^p$ dependency matrix for p can be estimated via the operation *product*$(\mathbf{M}\langle r\rangle, \mathbf{M}\langle s\rangle)$ defined in Algorithm 6.

3.7 Dependency Matrix for Negation

Let $\mathbf{M}\langle r\rangle$ be a n-dimensional $\beta_1 \times ... \times \beta_n$ dependency matrix for r and $\mathbf{M}\langle s\rangle$ be a m-dimensional $\beta_1 \times ... \times \beta_m$ dependency matrix for s. We consider the following negation

```
p(X1,...,Xn) :- r(X1,...,Xn), not s(X1,...,Xm).
```

where $m \leq n$ and the first m arguments are chosen to be common to r and s, for simplicity.

We assume that the first m axes of $\mathbf{M}\langle r\rangle$ and $\mathbf{M}\langle s\rangle$ are aligned. Algorithm 7 computes the operation *minus*$(\mathbf{M}\langle r\rangle, \mathbf{M}\langle s\rangle)$. Detailed explanations and alternative methods to compute matrix values at line 7 are available in [15].

```
1 $\mathbf{M}\langle p\rangle_d^{i_d}.\{floor, ceiling, size\} := \mathbf{M}\langle r\rangle_d^{i_d}.\{floor, ceiling, size\}$ for all $1 \leq d \leq n$
  and $1 \leq i_d \leq \beta_d$;                                   /* parameters */
2 $\mathbf{M}\langle r\rangle' := project(\mathbf{M}\langle r\rangle, \{x_1, ..., x_m\}, \{x_{m+1}, ..., x_n\})$;
3 $\mathbf{M}\langle p\rangle(i_1, ..., i_n) := \max\{1\text{-}\frac{\mathbf{M}\langle s\rangle(i_1,...,i_m)}{\mathbf{M}\langle r\rangle'(i_1,...,i_m)}, 0\} \times \mathbf{M}\langle r\rangle(i_1, ..., i_n)$ for all $1 \leq i_1 \leq \beta_1$,
  $..., 1 \leq i_n \leq \beta_n$;
4 return $\boldsymbol{M}\langle p\rangle$;
```

Algorithm 7. *minus*$(\mathbf{M}\langle r\rangle, \mathbf{M}\langle s\rangle)$

3.8 Recursive Predicates

The matrix for recursive predicates are computed iteratively until their size estimates reach *approximate* fixed points. For a set of mutually recursive predicates, the computation stops when all size estimates reach approximate fixed points.

Definition 3. *Consider a recursive predicate* p. *According to the definitions in Sections 3.1 – 3.7, the dependency matrix for* p *can be defined by the following recurrent equation:* $\boldsymbol{M}\langle p\rangle = expr(\boldsymbol{M}\langle p\rangle, other)$, *where expr is an expression in the algebra of estimation operators defined in earlier subsections. This equation is recursive in* $\boldsymbol{M}\langle p\rangle$ *but expr may take other arguments as well. We say that* $\boldsymbol{I}^{n+1}$ *is an* ***α-approximation*** *of* $\boldsymbol{M}\langle p\rangle$, *where* $0 \leq \alpha < 1$, *if* $\frac{|size(\boldsymbol{I}^{n+1}) - size(\boldsymbol{I}^n)|}{size(\boldsymbol{I}^n)} \leq \alpha$, *where* $\boldsymbol{I}^0 = \emptyset$ *and* $\boldsymbol{I}^{i+1} = expr(\boldsymbol{I}^i, other)$. *We will use α-approximations for estimating the size of* p. □

Example 3. Consider the following recursive program, which defines two mutually recursive predicates tcp and tcq:

```
tcp(X1,X2,X3) :- p(X1,X2,X3).
tcq(X1,X2,X3) :- q(X1,X2,X3).
tcp(X1,X2,X4) :- tcq(X1,X2,X3), p(X2,X3,X4).   (recp)
tcq(X1,X2,X4) :- tcp(X1,X2,X3), q(X2,X3,X4).   (recq)
```

Initially, the dependency matrices $\mathbf{M}\langle p\rangle$ and $\mathbf{M}\langle q\rangle$ are computed and propagated to $\mathbf{M}\langle tcp\rangle$ and $\mathbf{M}\langle tcq\rangle$ using the first two rules. Then, the following iterative steps are performed.

1. Compute $\mathbf{M}\langle tcp\rangle$ using the rule `recp` as in the case of a join followed by a projection, and then union with current dependency matrix $\mathbf{M}\langle tcp\rangle$, i.e., $\mathbf{M}\langle tcp\rangle = union(\mathbf{M}\langle tcp\rangle, project(join(\mathbf{M}\langle tcq\rangle, \mathbf{M}\langle p\rangle), \{x_1, x_2, x_4\}, \{x_3\}))$.
   Thus, in this case, the expression used in Definition 3 is $expr(\mathbf{M}\langle tcp\rangle, \mathbf{M}\langle tcq\rangle, \mathbf{M}\langle p\rangle) = union(\mathbf{M}\langle tcp\rangle, project(join(\mathbf{M}\langle tcq\rangle, \mathbf{M}\langle p\rangle), \{x_1, x_2, x_4\}, \{x_3\}))$.
2. Similarly, $\mathbf{M}\langle tcq\rangle = union(\mathbf{M}\langle tcq\rangle, project(join(\mathbf{M}\langle tcp\rangle, \mathbf{M}\langle q\rangle), \{x_1, x_2, x_4\}, \{x_3\}))$ and the expression in this case is similar: $expr(\mathbf{M}\langle tcq\rangle, \mathbf{M}\langle tcp\rangle, \mathbf{M}\langle q\rangle) = union(\mathbf{M}\langle tcq\rangle, project(join(\mathbf{M}\langle tcp\rangle, \mathbf{M}\langle q\rangle), \{x_1, x_2, x_4\}, \{x_3\}))$.
3. If the iteration reaches α-approximation for $\mathbf{M}\langle tcp\rangle$ and $\mathbf{M}\langle tcq\rangle$ (as defined in Definition 3), the computation stops. Otherwise, we keep iterating. □

In Example 3, one can also first compute $\mathbf{M}\langle tcq\rangle$ and then $\mathbf{M}\langle tcp\rangle$, i.e., switch the first two steps, during each iteration. That is to say, there exist many abstraction evaluation orders at each iteration step if multiple predicates are mutually recursive. We choose the order in which these predicates are first defined by rules in our current implementation. Since we are computing α-approximations, we assume that this evaluation order is trivial with respect to final estimates. However, more experimental studies are needed to validate this assumption.

The parameter α can be selected in various ways. Larger values make computation reach α-approximation sooner, while smaller α's cause longer computations, but produce better estimates. Note, however, that the evaluation is *not* guaranteed to reach an approximate fixed point for a chosen α, since $size(\mathbf{I}^n)$ in Definition 3 may oscillate. In this case, we can stop the iteration over $\mathbf{I}^n$ once oscillation of $size(\mathbf{I}^n)$ is detected and, as a practical measure, we can do with a coarser approximation.

3.9 Dependency Matrices for All Predicates

This section presents an algorithm for computing size estimates for all predicates in a bottom-up fashion using the algebra over matrices defined earlier.

Definition 4. *Given a knowledge base $\mathcal{K}$, its* ***predicate dependency graph*** *is a directed graph $G_{pdg}(\mathcal{K}) = (\mathcal{N}, \mathcal{E})$ where the set of nodes, $\mathcal{N}$, consists of all predicates contained in $\mathcal{K}$ and $(p_1, p_2) \in \mathcal{E}$ if and only if there is a rule in $\mathcal{K}$ such that p_1 is the rule's head predicate and p_2 is one of the body predicates.* □

A graph is *strongly connected* if there is a path from one node to every other node. The *strongly connected components* (SCC) of a directed graph are its *maximal* strongly connected subgraphs. An SCC in a predicate dependency graph contains a maximal subset of *recursive* predicates that mutually depend on one another. Thus, the dependency matrices for all predicates in the same SCC should be iteratively computed until their size estimates all become α-approximate for some chosen α.

The *condensation* of a predicate dependency graph $G_{pdg}(\mathcal{K}) = (\mathcal{N}, \mathcal{E})$, written as $G^c_{pdg}(\mathcal{K}) = (\mathcal{N}^c, \mathcal{E}^c)$, is a directed *acyclic* graph (a forest of trees) where $\mathcal{N}^c$ consists of all the SCC's of $G_{pdg}(\mathcal{K})$. Let $scc_1 = (\mathcal{N}_1, \mathcal{E}_1)$ and $scc_2 = (\mathcal{N}_2, \mathcal{E}_2)$ be two SCC's of $G_{pdg}(\mathcal{K})$, there is an edge $(scc_1, scc_2) \in \mathcal{E}^c$ if and only if there exists $p_1 \in \mathcal{N}_1$, $p_2 \in \mathcal{N}_2$ such that $(p_1, p_2) \in \mathcal{E}$.

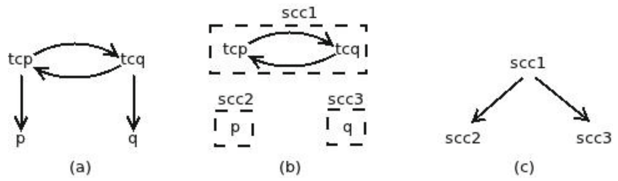


Fig. 2. Predicate Dependency Graph of Example 3

Example 4. Let $\mathcal{K}$ be the set of rules of Example 3, then its predicate dependency graph $G_{pdg}(\mathcal{K}) = (\mathcal{N}, \mathcal{E})$ is given in Figure 2(a). There are three SCC's: $scc_1 = (\{tcp, tcq\}, \{(tcp, tcq), (tcq, tcp)\})$, $scc_2 = (\{p\}, \emptyset)$, and $scc_3 = (\{q\}, \emptyset)$; they are given in Figure 2(b). The condensation of $G_{pdg}(\mathcal{K})$ is shown in Figure 2(c). □

Given a knowledge base $\mathcal{K}$, let $G^c_{pdg}(\mathcal{K})$ be the condensation of its predicate dependency graph. The dependency matrices for the set of predicates in each tree in the condensation can be computed by a bottom-up traversal of the tree, as shown in Algorithm 8. The algorithm resembles the usual naive bottom-up procedure for evaluating Horn rules except that here we employ abstract computation over the size estimation algebra. For instance, let $G^c_{pdg}(\mathcal{K})$ be the graph in Figure 2(c) and T be its (only) tree. Initially, $sccs = \{scc_2, scc_3\}$ at line 8 of the algorithm. During the first iteration of the while-loop, the dependency matrices for predicates contained in scc_2 and scc_3, i.e., $\mathbf{M}\langle p\rangle$ and $\mathbf{M}\langle q\rangle$, are computed, and then, on line 8, we follow the edges of T upwards and set $sccs = \{scc_1\}$. After the second iteration of the while-loop, all dependency matrices are computed.

```
1 Let $G^c_{pdg}(\mathcal{K}) = (\mathcal{N}^c, \mathcal{E}^c)$ be the predicate dependency condensation graph of $\mathcal{K}$;
2 for each tree $T \in G^c_{pdg}(\mathcal{K})$ do
3     Let $sccs$ be set of leaf SCC's of $T$;                    /* start with leaves */
4     while $sccs \neq \emptyset$ do                           /* bottom-up traversal of $T$ */
5         Compute dependency matrices for predicates in each SCC in $sccs$;
6         $sccs = \{scc_1 \mid (scc_1, scc_2) \in \mathcal{E}^c, scc_2 \in sccs\}$;
7     end
8 end
```

Algorithm 8. Compute Dependency Matrices for All Predicates in $\mathcal{K}$

Algorithm 8 computes dependency matrices for all predicates in the knowledge base in a bottom-up manner, without considering the query. If a query is given then we only need to evaluate one tree in the condensation graph, the one that contains the query predicate. Moreover, we need to perform the computation only for the subtree rooted in the clique that contains the query predicate.

Note that, given a set of rules for a predicate p, we can apply estimation operators for computing $\mathbf{M}\langle p\rangle$ in different orders, and this may yield different estimates.

3.10 Complexity Analysis

To estimate the complexity of our algorithms, assume that all predicates are n-ary, fact-set sizes of base predicates are bounded by fs, domains of the arguments sizes are bounded by ds, and dependency matrices are all n-dimensional *sparse* matrices of sizes $\beta \times \ldots \times \beta$. The complexity of SDP operations that are defined in earlier subsections is summarized in Table 1, and detailed analysis can be found in [15]. It is worth mentioning that typical applications on the semantic Web mostly use triple stores and the arities are bounded by 4, which makes the overhead for computing predicate sizes affordable.

Table 1. Complexity of SDP Operations

Operation	Complexity
construct $\mathbf{M}\langle p\rangle$	$O(n \times (ds \times \lg(ds) + fs))$
$project(\mathbf{M}\langle r\rangle, in\_args, out\_args)$	$O(\beta^n)$
$union(\mathbf{M}\langle r\rangle, \mathbf{M}\langle s\rangle)$	$O(n \times \beta^n)$
$intersect(\mathbf{M}\langle r\rangle, \mathbf{M}\langle s\rangle)$	$O(n \times \beta^n)$
$minus(\mathbf{M}\langle r\rangle, \mathbf{M}\langle s\rangle)$	$O(n \times \beta^n)$
$join(\mathbf{M}\langle r\rangle, \mathbf{M}\langle s\rangle)$	$O(\beta^{m+n-k})$
$product(\mathbf{M}\langle r\rangle, \mathbf{M}\langle s\rangle)$	$O(\beta^{m+n})$

Consider a knowledge base with m predicates whose maximal arity is n. The number of space units needed to store their dependency matrices is $m \times (\beta^n + 3 \times n \times \beta)$, where *unit* is the amount of bytes to store one dependency matrix value or coordinate parameter. If $n = 4$ and $\beta = 30$ (a reasonably high predicate arity bound and very accurate segmentation of distributions), these storage requirements translate into less than 4Mb per matrix.

4 Experiments

We experimented with SDP using several rule sets. Due to space limitations, we only include the results for the mutual recursive transitive closure rules of Example 3. More extensive experiments and analysis are available at [15]. All tests were performed on a dual core 2.4GHz Lenovo X200 with 3GB of main memory. The machine was running Ubuntu 11.04 with Linux kernel 2.6.38. Our implementation is based on XSB-Prolog version 3.3.5 (Pignoletto).

Test Datasets. Datasets for predicates `p` and `q` are generated from *homogeneous Poisson process* using system R [2]. Each dataset has one million facts for `p` and `q` each, with argument values in the range $[1, 1000]$. We generated 6 data sets with different data dependencies. In Figure 3, the lower value of the *box* parameter means that the data set has high dependency among the different arguments of the predicates, while higher values of *box* mean that the data is more evenly distributed.

Test Parameters. In all tests, we used $\beta \times \ldots \times \beta$ dependency matrices for $\beta = 5, 10, 20, 30, 40$, and tested different ways to compute the dependency matrix for *union* and *project*—some more accurate (and more complex) than the one given in this paper. Here we show the results using only one of these methods. The interested reader is referred to [15] for further details.

Results. Figure 3 compares size estimates to the real sizes of `tcp` and `tcq` for different datasets and dependency matrix sizes. First, observe that increasing the sizes of dependency matrices from $\beta = 5$ to $\beta = 40$ makes size estimates closer to the real sizes. Second, increasing *box* values from $box = 6$ to $box = 16$ makes dependency among the predicate arguments lower, so the size of the dependency matrix becomes less significant for the accuracy of the estimates, as expected. Third, for reasonably large dependency matrix sizes ($\beta = 30, 40$), our size estimates preserve the relative order of the real sizes. Since many query optimization algorithms use relative sizes to decide on the proper join orderings, this means that our size estimates are sufficiently accurate to be used for that purpose. Fourth, all base datasets have the same size in our experiments, so it is data dependencies that determine the real result sizes. It further validates the importance of considering argument dependencies in doing size estimates.

We also compare the running times of SDP (in seconds) compared to the real times needed to compute the actual queries (the transitive closures). Overall SDP's overhead is within 3% of the actual query time, which is acceptable. This overhead could be probably further reduced by improving our implementation.

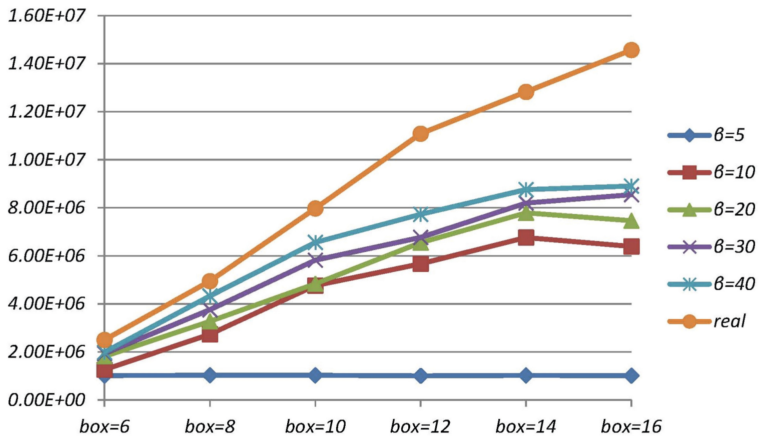


Fig. 3. Size Estimates Against Real Sizes

5 Related Work

We classify the work related to this paper into traditional approaches, approaches based on dependency graphs, work on multidimensional histograms, and cost estimation for recursion.

Traditional Estimation Techniques. Traditional size estimates use base table statistics and propagate them through derived predicates by assuming independence among arguments. However such estimates could be off by orders of magnitude [29], which was confirmed by our implementation of the histogram algorithms [16] proposed in [4]. In contrast, SDP maintains argument dependencies of both base and derived predicates, providing more accurate estimates.

Graph-Based Approaches. The present paper was inspired by the size estimation techniques based on *dependency graphs* [26] and on the *graph-based selectivity estimate approach* [28]. In [26], statistics for predicates (both base and derived) were kept in dependency graphs, which were essentially $1 \times \ldots \times 1$ dependency matrices. Recursive predicates were unfolded up to three steps without considering data distribution. Our dependency matrices keep more fine-grained dependency information and SDP computes the statistics for recursive predicates via an incremental evaluation that approximates fixpoints and takes base dataset distributions into account. It is unclear how to do this using the framework of [26] and whether such a generalization is possible at all.

Graph based selectivity estimates were proposed in [28], where relational data sets were modeled as graphs and join queries as graph traversals. Fixing a specific set of binary joins, the task in [28] was to summarize base data distributions and joins within the given storage allowance. There are two obvious points that differentiate our work. First, [28] required a priori knowledge of all the joins of interest, while SDP does not and thus is more flexible. Second, SDP handles recursion, while [28] dealt only with relational queries.

Multi-dimensional histograms. Multidimensional histograms have been proposed to keep argument dependency information [19,22,8,7]. For instance, [22] proposed several definitions of multi-dimensional histograms and compared their performance with traditional histograms. Multi-dimensional histograms are able to accurately capture argument dependency since they approximate joint data distributions directly by heuristically grouping similar values. They have been used to estimate the selectivity of spacial data in Geographic Information Systems [1]. Dynamic multi-dimensional histograms for continuous data streams were also studied in [31]. However, multi-dimensional histograms are quite expensive to compute and for an n-ary predicate, there is an exponential number of multi-dimensional histograms to compute [22]. It is also unclear how to efficiently compute multi-dimensional histograms for joins and recursions.

Size Estimation for Recursive Predicates. Computing the expected sizes for recursive predicates was studied in [17,27]. [17] proposed an adaptive sampling algorithm to estimate the sizes of transitive closures, which provided estimates within certain confidence intervals of the real sizes. [27] studied the expected sizes of transitive closure, same generation, and canonically factorable recursion of uniformly distributed base datasets. There, they proved many asymptomatic expressions about the expected sizes. Our method, SDP, also performs size estimates for recursive predicates, but it is different in that SDP summarizes base relations using dependency matrices and does not assume any specific data distribution although our experimental datasets were generated from homogeneous Poisson distributions. Second, [27] focused on deriving theoretical asymptotic expressions for the expected sizes, while SDP computes size estimates by maintaining data statistics for both base and derived predicates.

Finally, we need to place our work relative to a host of works on query optimization in rule systems [23,20,18,32,6,9,3,25]. In this regard, our work is about predicate size estimation, which is a major data point used by most optimizers that aim to reorder predicates in rule bodies. Thus our work is orthogonal to most of those works and could enhance these other approaches if combined with them. This, in fact, is our next goal.

6 Conclusions

In this paper, we extended dependency matrices and the SDP algorithm proposed in [16] to arbitrary n-ary predicates and more general rule formats including mutual recursion, negation, selection, union, intersection and join. These extensions are essential in order for SDP to be useful for optimizing declarative knowledge base engines such as XSB [24], FLORA-2 [34], and SILK [33]. Experimental studies were performed on a variety of rule sets to demonstrate the efficacy of our approach, which takes argument dependency for estimating predicate sizes. These experiments also show that our approach produces estimates that are in keeping with the relative order of *real* predicate sizes. This property is crucial for cost based optimization. To further validate the efficacy of SDP, we plan to add a cost-based optimization module to the XSB engine and explore other cost models and optimization algorithms.

Acknowledgments. This work is part of the SILK (Semantic Inference on Large Knowledge) project sponsored by Vulcan, Inc. We would like to thank reviewers for their very helpful comments.

References

1. Acharya, S., Poosala, V., Ramaswamy, S.: Selectivity estimation in spatial databases. In: SIGMOD 1999: Proceedings of the 1999 ACM SIGMOD International Conference on Management of Data, pp. 13–24. ACM, New York (1999)
2. Baddeley, A., Turner, R.: Spatstat: an R package for analyzing spatial point patterns. Journal of Statistical Software 12(6), 1–42 (2005), http://www.jstatsoft.org
3. Bowman, I.T., Paulley, G.N.: Join enumeration in a memory-constrained environment. In: Proceedings of the 16th International Conference on Data Engineering, pp. 645–654. IEEE Computer Society, Washington, DC (2000)
4. Nicolas, B., Surajit, C.: Exploiting statistics on query expressions for optimization. In: SIGMOD 2002: Proceedings of the 2002 ACM SIGMOD International Conference on Management of Data, pp. 263–274. ACM, New York (2002)
5. Christodoulakis, S.: Implications of certain assumptions in database performance evauation. ACM Trans. Database Syst. 9(2), 163–186 (1984)
6. DeHaan, D., Tompa, F.W.: Optimal top-down join enumeration. In: Proceedings of the 2007 ACM SIGMOD International Conference on Management of Data, SIGMOD 2007, pp. 785–796. ACM, New York (2007)
7. Amol, D., Minos, G., Rajeev, R.: Independence is good: dependency-based histogram synopses for high-dimensional data. SIGMOD Rec. 30(2), 199–210 (2001)
8. Furtado, P., Madeira, H.: Summary grids: Building accurate multidimensional histograms (1999)
9. Gassner, P., Lohman, G.M., Schiefer, K.B., Wang, Y.: Query optimization in the ibm db2 family. IEEE Data Eng. Bull. 16(4), 4–18 (1993)
10. Ioannidis, Y.: The history of histograms (abridged). In: Proc. of VLDB Conference. Morgan Kaufmann, Berlin (2003)
11. Ioannidis, Y.E.: Universality of serial histograms. In: VLDB 1993: Proceedings of the 19th International Conference on Very Large Data Bases, pp. 256–267. Morgan Kaufmann Publishers Inc., San Francisco (1993)
12. Ioannidis Yannis, E., Christodoulakis, S.: On the propagation of errors in the size of join results. SIGMOD Rec. 20(2), 268–277 (1991)
13. Ioannidis Yannis, E., Poosala, V.: Balancing histogram optimality and practicality for query result size estimation. In: SIGMOD 1995: Proceedings of the 1995 ACM SIGMOD International Conference on Management of Data, pp. 233–244. ACM, New York (1995)

14. Kifer, M., Bernstein, A., Lewis, P.M.: Database Systems: An Application Oriented Approach, Compete Version. Addison-Wesley, Boston (2006)
15. Liang, S.: Non-termination analysis and cost-based query optimization of logic programs. Ph.D. Dissertation (2012), http://www.cs.stonybrook.edu/~sliang
16. Liang, S., Kifer, M.: Deriving predicate statistics in datalog. In: Kutsia, T., Schreiner, W., Fernández, M. (eds.) PPDP, pp. 45–56. ACM (2010)
17. Lipton, R.J., Naughton, J.F.: Estimating the size of generalized transitive closures. In: VLDB 1989: Proceedings of the 15th International Conference on Very Large Data Bases, pp. 165–171. Morgan Kaufmann Publishers Inc., San Francisco (1989)
18. Moerkotte, G., Neumann, T.: Dynamic programming strikes back. In: SIGMOD 2008: Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data, pp. 539–552. ACM, New York (2008)
19. Muralikrishna, M., DeWitt, D.J.: Equi-depth histograms for estimating selectivity factors for multi-dimensional queries. In: Boral, H., Larson, P.Å. (eds.) Proceedings of the 1988 ACM SIGMOD International Conference on Management of Data, Chicago, Illinois, June 1-3, pp. 28–36. ACM Press (1988)
20. Ono, K., Lohman, G.M.: Measuring the complexity of join enumeration in query optimization. In: Proceedings of the Sixteenth International Conference on Very Large Databases, pp. 314–325. Morgan Kaufmann Publishers Inc., San Francisco (1990), http://portal.acm.org/citation.cfm?id=94362.94436
21. Poosala, V., Haas, P.J., Ioannidis, Y.E., Shekita, E.J.: Improved histograms for selectivity estimation of range predicates. In: SIGMOD 1996: Proceedings of the 1996 ACM SIGMOD International Conference on Management of Data, pp. 294–305. ACM, New York (1996)
22. Viswanath, P., Ioannidis, Y.E.: Selectivity estimation without the attribute value independence assumption. In: VLDB 1997: Proceedings of the 23rd International Conference on Very Large Data Bases, pp. 486–495. Morgan Kaufmann Publishers Inc., San Francisco (1997)
23. Ramakrishnan, R., Srivastava, D., Sudarshan, S., Seshadri, P.: The coral deductive system. VLDB J. 3(2), 161–210 (1994)
24. Sagonas, K.F., Swift, T., Warren, D.S.: An abstract machine for efficiently computing queries to well-founded models. J. Log. Program. 45(1-3), 1–41 (2000)
25. Selinger, P.G., Astrahan, M.M., Chamberlin, D.D., Lorie, R.A., Price, T.G.: Access path selection in a relational database management system. In: SIGMOD 1979: Proceedings of the 1979 ACM SIGMOD International Conference on Management of Data, pp. 23–34. ACM, New York (1979)
26. Sereni, D., Avgustinov, P., de Moor, O.: Adding magic to an optimising datalog compiler. In: SIGMOD 2008: Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data, pp. 553–566. ACM, New York (2008)
27. Seshadri, S., Naughton, J.F.: On the expected size of recursive datalog queries. In: PODS 1991: Proceedings of the tenth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, pp. 268–279. ACM, New York (1991)
28. Spiegel, J., Polyzotis, N.: Graph-based synopses for relational selectivity estimation. In: SIGMOD 2006: Proceedings of the 2006 ACM SIGMOD International Conference on Management of Data, pp. 205–216. ACM, New York (2006)
29. Stillger, M., Lohman, G.M., Markl, V., Kandil, M.: Leo - db2's learning optimizer. In: VLDB 2001: Proceedings of the 27th International Conference on Very Large Data Bases, pp. 19–28. Morgan Kaufmann Publishers Inc., San Francisco (2001)
30. Swift, T., Warren, D.S.: Xsb: Extending prolog with tabled logic programming. CoRR abs/1012.5123 (2010)

31. Thaper, N., Guha, S., Indyk, P., Koudas, N.: Dynamic multidimensional histograms. In: SIGMOD 2002: Proceedings of the 2002 ACM SIGMOD International Conference on Management of Data, pp. 428–439. ACM, New York (2002)
32. Vance, B., Maier, D.: Rapid bushy join-order optimization with cartesian products. In: Proceedings of the 1996 ACM SIGMOD International Conference on Management of Data, SIGMOD 1996, pp. 35–46. ACM, New York (1996)
33. The SILK project: Semantic Inferencing on Large Knowledge. The FLORA-2 Web Site, http://silk.semwebcentral.org/
34. Yang, G., Kifer, M., Zhao, C.: $\mathcal{F}$lora-2: A Rule-Based Knowledge Representation and Inference Infrastructure for the Semantic Web. In: Meersman, R., Schmidt, D.C. (eds.) CoopIS 2003, DOA 2003, and ODBASE 2003. LNCS, vol. 2888, pp. 671–688. Springer, Heidelberg (2003)

Consistent Answers in Probabilistic Datalog+/– Ontologies

Thomas Lukasiewicz, Maria Vanina Martinez, and Gerardo I. Simari

Department of Computer Science, University of Oxford, UK
{thomas.lukasiewicz,vanina.martinez,gerardo.simari}@cs.ox.ac.uk

Abstract. The Datalog+/– family of ontology languages is especially useful for representing and reasoning over lightweight ontologies, and has many applications in the context of query answering and information extraction for the Semantic Web. It is widely accepted that it is necessary to develop a principled way to handle uncertainty in this domain. In addition to uncertainty as an inherent aspect of the Web, one must also deal with forms of uncertainty due to inconsistency. In this paper, we take an important step in this direction by developing inconsistency-tolerant semantics for query answering in a probabilistic extension of Datalog+/–. The main contributions of this paper are: (i) extension and generalization to probabilistic ontologies of the well-known concepts of repairs and consistent answers to queries from databases; (ii) complexity analysis for the problems of consistency checking, repair identification, and consistent query answering; and (iii) adaptation of the intersection semantics (a sound heuristic for consistent answers) to probabilistic ontologies, yielding a subset of probabilistic Datalog+/– that is tractable modulo the cost of computing probabilities.

1 Introduction

It is widely acknowledged, both in the database and the Semantic Web community, that inconsistency is a central issue when dealing with knowledge bases. When integrating data from many different sources, either as a means to populate an ontology or simply to answer queries, integrity constraints are very likely to be violated in practice. In this paper, we address the problem of handling inconsistency in ontologies and the Semantic Web, where scalability is an important issue. We adopt a probabilistic extension of the recently developed Datalog+/– family of ontology languages [4] in which ontological axioms are annotated with constraints specifying the probabilistic events that must occur in order for them to hold. In this paper, the probabilistic model is left unspecified; in this sense, the language is a generalization of the one adopted in [7].

Our main goal is to develop inconsistency-tolerant semantics for query answering in this setting. We thus adapt and generalize the well-known concepts of *repair* and *consistent answers* from databases [1]. In the database community, repairs constitute consistent models of a set of constraints that are as close as possible to the original (possibly inconsistent) database. In our work, a repair consists of a subset of the original ontology, where only probabilistic atoms can be removed. Consistent query answering (CQA) is the most widely accepted semantics for querying a possibly inconsistent database. Instead of repairing an ontology, the result of a query returns *consistent answers*, the set of tuples (atoms) that appear in the answer to the query over *every* possible repair. That is, those are the answers that hold no matter what measures are taken to repair the database. CQA has already been adopted for various families of description logics (DLs) [9] and recently for guarded Datalog+/– ontologies [11]. In this work, we generalize CQA to the case of probabilistic Datalog+/– ontologies. The contributions of this paper are: (i) extension and generalization of the concepts of repairs and consistent answers to queries over probabilistic ontologies; (ii) complexity analysis for the problems of consistency checking, repair checking (deciding whether a probabilistic ontology is a repair of another one), and CQA; (iii) adaptation of the intersection semantics (a sound heuristic for consistent answers developed in DLs) to the setting of probabilistic ontologies; (iv) description how consistent answers under the intersection semantics can be obtained by means of traversing an annotated version of the classical chase graph for non-probabilistic Datalog+/–; and (v) identification of a fragment of probabilistic Datalog+/– for which CQA under the intersection semantics is tractable, modulo the cost of computing probabilities of arbitrary events.

2 Preliminaries

We briefly recall some basics on Datalog+/– [4], namely, on relational databases, (Boolean) conjunctive queries ((B)CQs), tuple- and equality-generating dependencies (TGDs and EGDs, respectively), negative constraints, and ontologies in Datalog+/–.

Databases and Queries. We assume (i) an infinite universe of *(data) constants* Δ (which constitute the "normal" domain of a database), (ii) an infinite set of *(labeled) nulls* Δ_N (used as "fresh" Skolem terms, which are placeholders for unknown values, and can thus be seen as variables), and (iii) an infinite set of variables $\mathcal{V}$ (used in queries, dependencies, and constraints). Different constants represent different values (*unique name assumption*), while different nulls may represent the same value. We assume a lexicographic order on $\Delta \cup \Delta_N$, with every symbol in Δ_N following all symbols in Δ. We denote by $\mathbf{X}$ sequences of variables $X_1, \ldots, X_k$ with $k \geqslant 0$.

We assume a *relational schema* $\mathcal{R}$, which is a finite set of *predicate symbols* (or simply *predicates*). A *term* t is a constant, null, or variable. An *atomic formula* (or *atom*) $\mathbf{a}$ has the form $P(t_1, ..., t_n)$, where P is an n-ary predicate, and $t_1, ..., t_n$ are terms. A conjunction of atoms is often identified with the set of all its atoms.

A *database (instance)* D for a relational schema $\mathcal{R}$ is a (possibly infinite) set of atoms with predicates from $\mathcal{R}$ and arguments from Δ. A *conjunctive query (CQ)* over $\mathcal{R}$ has the form $Q(\mathbf{X}) = \exists \mathbf{Y}\, \Upsilon(\mathbf{X}, \mathbf{Y})$, where $\Upsilon(\mathbf{X}, \mathbf{Y})$ is a conjunction of atoms (possibly equalities, but not inequalities) with the variables $\mathbf{X}$ and $\mathbf{Y}$, and possibly constants, but without nulls. A *Boolean CQ (BCQ)* over $\mathcal{R}$ is a CQ of the form $Q()$, often written as the set of all its atoms, without quantifiers. Answers are defined via *homomorphisms*, which are mappings $\mu\colon \Delta \cup \Delta_N \cup \mathcal{V} \to \Delta \cup \Delta_N \cup \mathcal{V}$ such that (i) $c \in \Delta$ implies $\mu(c) = c$, (ii) $c \in \Delta_N$ implies $\mu(c) \in \Delta \cup \Delta_N$, and (iii) μ is naturally extended to atoms, sets of atoms, and conjunctions of atoms. The set of all *answers* to a CQ $Q(\mathbf{X}) = \exists \mathbf{Y}\, \Upsilon(\mathbf{X}, \mathbf{Y})$ over a database D, denoted $Q(D)$, is the set of all tuples $\mathbf{t}$ over Δ for which there exists a homomorphism $\mu\colon \mathbf{X} \cup \mathbf{Y} \to \Delta \cup \Delta_N$ such that $\mu(\Upsilon(\mathbf{X}, \mathbf{Y})) \subseteq D$ and $\mu(\mathbf{X}) = \mathbf{t}$. The *answer* to a BCQ $Q()$ over a database D is *Yes*, denoted $D \models Q$, iff $Q(D) \neq \emptyset$.

Tuple-Generating Dependencies. Tuple-generating dependencies describe constraints on databases in the form of generalized Datalog rules with existentially quantified conjunctions of atoms in rule heads. Given a relational schema $\mathcal{R}$, a *tuple-generating dependency (TGD)* σ is a first-order formula of the form $\forall \mathbf{X} \forall \mathbf{Y}\, \Upsilon(\mathbf{X}, \mathbf{Y}) \rightarrow \exists \mathbf{Z}\, \Psi(\mathbf{X}, \mathbf{Z})$, where $\Upsilon(\mathbf{X}, \mathbf{Y})$ and $\Psi(\mathbf{X}, \mathbf{Z})$ are conjunctions of atoms over $\mathcal{R}$ (without nulls), called the *body* and the *head* of σ, denoted $body(\sigma)$ and $head(\sigma)$, respectively. Such σ is satisfied in a database D for $\mathcal{R}$ iff, whenever there exists a homomorphism h that maps the atoms of $\Upsilon(\mathbf{X}, \mathbf{Y})$ to atoms of D, there exists an extension h' of h that maps the atoms of $\Psi(\mathbf{X}, \mathbf{Z})$ to atoms of D. We usually omit the universal quantifiers in TGDs, and all sets of TGDs are finite here. Since TGDs can be reduced to TGDs with only single atoms in their heads, in the sequel, every TGD has w.l.o.g. a single atom in its head. A TGD σ is *guarded* iff it contains an atom in its body that contains all universally quantified variables of σ. The leftmost such atom is the *guard atom* (or *guard*) of σ. A TGD σ is *linear* iff it contains only a single atom in its body.

Query answering under TGDs, i.e., the evaluation of CQs and BCQs on databases under a set of TGDs, is defined as follows. For a database D for $\mathcal{R}$, and a set of TGDs Σ on $\mathcal{R}$, the set of *models* of D and Σ, denoted $mods(D, \Sigma)$, is the set of all (possibly infinite) databases B such that (i) $D \subseteq B$ and (ii) every $\sigma \in \Sigma$ is satisfied in B. The set of *answers* for a CQ Q to D and Σ, denoted $ans(Q, D, \Sigma)$, is the set of all tuples $\mathbf{a}$ such that $\mathbf{a} \in Q(B)$ for all $B \in mods(D, \Sigma)$. The *answer* for a BCQ Q to D and Σ is *Yes*, denoted $D \cup \Sigma \models Q$, iff $ans(Q, D, \Sigma) \neq \emptyset$. Note that query answering under general TGDs is undecidable [2], even when the schema and TGDs are fixed [3]. The two problems of CQ and BCQ evaluation under TGDs are LOGSPACE-equivalent [6,5]. Moreover, the query output tuple (QOT) problem (as a decision version of CQ evaluation) and BCQ evaluation are AC_0-reducible to each other. Henceforth, we thus focus only on BCQ evaluation, and any complexity results carry over to the other problems. Decidability of query answering for the guarded case follows from a bounded tree-width property. The data complexity of query answering in this case is P-complete [4].

The Chase. The *chase* was introduced to enable checking implication of dependencies [12], and later also for checking query containment [8]. It is a procedure for repairing a database relative to a set of dependencies, so that the result of the chase satisfies the dependencies. By "chase", we refer both to the chase procedure and to its output. The TGD chase works on a database through so-called TGD *chase rules* (an extended chase with also equality-generating dependencies is discussed below). The TGD chase rule comes in two flavors: *restricted* and *oblivious*, where the restricted one applies TGDs only when they are not satisfied (to repair them), while the oblivious one always applies TGDs (if they produce a new result). We focus on the oblivious one here; the *(oblivious) TGD chase rule* defined below is the building block of the chase.

TGD CHASE RULE. Consider a database D for a relational schema $\mathcal{R}$, and a TGD σ on $\mathcal{R}$ of the form $\Upsilon(\mathbf{X}, \mathbf{Y}) \rightarrow \exists \mathbf{Z}\, \Psi(\mathbf{X}, \mathbf{Z})$. Then, σ is *applicable* to D if there exists a homomorphism h that maps the atoms of $\Upsilon(\mathbf{X}, \mathbf{Y})$ to atoms of D. Let σ be applicable to D, and h_1 be a homomorphism that extends h as follows: for each $X_i \in \mathbf{X}$, $h_1(X_i) = h(X_i)$; for each $Z_j \in \mathbf{Z}$, $h_1(Z_j) = z_j$, where z_j is a "fresh" null, i.e., $z_j \in \Delta_N$, z_j does not occur in D, and z_j lexicographically follows all other nulls already introduced. The *application of* σ on D adds to D the atom $h_1(\Psi(\mathbf{X}, \mathbf{Z}))$ if not already in D. ■

The chase algorithm for a database D and a set of TGDs Σ consists of an exhaustive application of the TGD chase rule in a breadth-first (level-saturating) fashion, which leads to a (possibly infinite) chase for D and Σ. Formally, the *chase of level up to* 0 of D relative to Σ, denoted $chase^0(D,\Sigma)$, is defined as D, assigning to every atom in D the *(derivation) level* 0. For every $k \geqslant 1$, the *chase of level up to* k of D relative to Σ, denoted $chase^k(D,\Sigma)$, is constructed as follows: let $I_1,\ldots,I_n$ be all possible images of bodies of TGDs in Σ relative to some homomorphism such that (i) $I_1,\ldots,I_n \subseteq chase^{k-1}(D,\Sigma)$ and (ii) the highest level of an atom in every I_i is $k-1$; then, perform every corresponding TGD application on $chase^{k-1}(D,\Sigma)$, choosing the applied TGDs and homomorphisms in a (fixed) linear and lexicographic order, respectively, and assigning to every new atom the *(derivation) level* k. The *chase* of D relative to Σ, denoted $chase(D,\Sigma)$, is then defined as the limit of $chase^k(D,\Sigma)$ for $k \to \infty$.

The (possibly infinite) chase relative to TGDs is a *universal model*, i.e., there exists a homomorphism from $chase(D,\Sigma)$ onto every $B \in mods(D,\Sigma)$ [5,3]. This result implies that BCQs Q over D and Σ can be evaluated on the chase for D and Σ, i.e., $D \cup \Sigma \models Q$ is equivalent to $chase(D,\Sigma) \models Q$. In the case of guarded TGDs Σ, such BCQs Q can be evaluated on an initial fragment of $chase(D,\Sigma) \models Q$ of constant depth $k \cdot |Q|$, and thus can be done in polynomial time in the data complexity.

Negative Constraints. Another crucial ingredient of Datalog+/– for ontological modeling are *negative constraints* (or simply *constraints*) γ, which are first-order formulas $\forall \mathbf{X} \Upsilon(\mathbf{X}) \to \bot$, where $\Upsilon(\mathbf{X})$ (called the *body* of γ) is a conjunction of atoms (without nulls and not necessarily guarded). We usually omit the universal quantifiers, and all sets of constraints are finite here. Under the standard semantics of query answering of BCQs Q in Datalog+/– with TGDs, adding negative constraints is computationally easy, as for each $\forall \mathbf{X} \Upsilon(\mathbf{X}) \to \bot$, we only have to check that the BCQ $\Upsilon(\mathbf{X})$ evaluates to false in D under Σ; if one of these checks fails, then the answer to the original BCQ Q is true, otherwise the constraints can simply be ignored when answering the BCQ Q.

Equality-Generating Dependencies (EGDs). A further important ingredient of Datalog+/– for modeling ontologies are *equality-generating dependencies* (or *EGDs*) σ, which are first-order formulas $\forall \mathbf{X}\, \Upsilon(\mathbf{X}) \to X_i = X_j$, where $\Upsilon(\mathbf{X})$, called the *body* of σ, denoted $body(\sigma)$, is a conjunction of atoms (without nulls and not necessarily guarded), and X_i and X_j are variables from $\mathbf{X}$. Such σ is satisfied in a database D for $\mathcal{R}$ iff, whenever there exists a homomorphism h such that $h(\Upsilon(\mathbf{X},\mathbf{Y})) \subseteq D$, it holds that $h(X_i) = h(X_j)$. We usually omit the universal quantifiers in EGDs, and all sets of EGDs are finite here. Adding EGDs over databases with guarded TGDs along with negative constraints does not increase the complexity of BCQ query answering as long as they are *non-conflicting* (we will not introduce this concept formally here for reasons of space; cf. [4] for details). Intuitively, this ensures that, if the chase fails (due to strong violations of EGDs), then it already fails on the database D, and if it does not fail, then whenever "new" atoms (from the logical point of view) are created in the chase by the application of the EGD chase rule, atoms that are logically equivalent to the new ones are guaranteed to be generated also in the absence of the EGDs. This guarantees that EGDs do not have any impact on the chase with respect to query answering. Non-conflicting EGDs can be expressed as negative constraints of the form $\forall \mathbf{X}\, \Upsilon(\mathbf{X}), X_i \neq X_j \rightarrow \bot$. In the following, for ease of presentation, all non-conflicting EGDs are expressed as such special forms of negative constraints.

A *(linear) Datalog+/– ontology* consists of a (finite) database D, a finite set of (linear) TGDs Σ_T, and a finite set of negative constraints and separable EGDs Σ_{NC}. We use the notation (D, Σ), where $\Sigma = \Sigma_T \cup \Sigma_{NC}$.

Example 1. The following database D, set of TGDs Σ_T, and set of negative constraints Σ_{NC} form together a linear Datalog+/– ontology $KB = (D, \Sigma_T \cup \Sigma_{NC})$:

$$\begin{aligned} D &= \{a(x_1),\, b(x_1),\, b(x_2),\, d(x_1),\, e(x_1)\};\\ \Sigma_T &= \{\sigma_1 : d(X) \rightarrow i(X),\, \sigma_2 : e(X) \rightarrow i(X),\, \sigma_3 : a(X) \rightarrow m(X)\};\\ \Sigma_{NC} &= \{\upsilon_1 : a(X) \wedge b(Y) \wedge X \neq Y \rightarrow \bot, \upsilon_2 : d(X) \wedge m(X) \rightarrow \bot\}. \end{aligned}$$ ■

3 Probabilistic Guarded Datalog+/–

Recall that we have, as discussed in Section 2, a finite set of relation names $\mathcal{R}_{Ont}$, an infinite universe of (data) constants Δ_{Ont}, an infinite set of labeled nulls Δ_N, and an infinite set of variables $\mathcal{V}_{Ont}$.

We assume the existence of a probabilistic model M that represents a probability distribution P_M over some set $X = \{X_1, \ldots, X_n\}$ of Boolean variables. Note that we use the term *probabilistic model* in reference to a particular instance of a probabilistic formalism; this term should not be confused with other uses of *model* elsewhere in the paper. Relative to the specification of the probabilistic model, we assume the existence of a finite set of constants Δ_M, an infinite set of variables $\mathcal{V}_M$, and a finite set of predicate names $\mathcal{R}_M$ such that $\mathcal{R}_M \cap \mathcal{R}_{Ont} = \emptyset$. Finally, there is a 1-to-1 mapping from X to the set of all ground atoms over $\mathcal{R}_M$ and Δ_M.

Examples of the type of probabilistic models that we assume in this work are Markov logic networks and Bayesian networks. The work in [7] is a special case of the general model presented here, where Markov logic networks (MLNs) [13] are the probabilistic models underlying the probabilistic Datalog+/– ontology, and the model is assumed to be fixed with respect to data complexity (an assumption not made here).

As usual, a substitution maps variables to variables or constants. Two sets S and T *unify* via a substitution θ iff $\theta S = \theta T$, where θA denotes the application of θ to all variables in all elements of A (here, θ is a unifier). A *most general unifier* (mgu) is a unifier θ such that for all other unifiers ω there is a substitution σ such that $\omega = \sigma \circ \theta$.

Definition 1. Let M be a probabilistic model. Then, a *(probabilistic) annotation* λ relative to M is a (finite) set of expressions of the form $\langle A_i = x_i \rangle$, where: (i) A_i is an atom over $\mathcal{R}_M$, $\mathcal{V}_M$, and Δ_M; and (ii) $x_i \in \{0, 1\}$. A probabilistic annotation is *valid* iff for any two different expressions $\langle A = x \rangle, \langle B = y \rangle \in \lambda$, there does not exist a substitution that unifies A and B.

Intuitively, a probabilistic annotation is used to describe the class of events in which the random variables in a probabilistic model M are compatible with the settings of the random variables described by λ, i.e., each X_i has the value x_i. A *probabilistic scenario* is a valid probabilistic annotation λ for which $|\lambda| = |X|$ and all $\langle A = x_i \rangle \in \lambda$ are such that A is ground. We use *scn*(M) to denote the set of scenarios in M. A probabilistic annotation is equivalent to a formula that consists of the disjunction of all the scenarios that belong to the set denoted by the annotation. Therefore, with a slight abuse of notation, we sometimes combine annotations with logical operators. We use $\mathit{vars}(\lambda)$ to denote the set of variables $\mathcal{V}_M$ in λ (note that $\mathit{vars}(\lambda) = \emptyset$ for scenarios λ).

Example 2. Let the set of random variables in M be $X = \{p(x_1), p(x_2), r(x_1), r(x_2)\}$, i.e., the set of all ground atoms over $\mathcal{R}_M$ and Δ_M. The set $\{\langle p(X) = 1\rangle, \langle r(x_1) = 0\rangle\}$ is a probabilistic annotation, whereas $\{\langle p(x_1) = 1\rangle, \langle p(x_2) = 1\rangle, \langle r(x_1) = 0\rangle, \langle r(x_2) = 1\rangle\}$ is a probabilistic scenario (which is a special case of a probabilistic annotation). ■

Informally, a probabilistic Datalog+/– ontology consists of a finite set of probabilistic atoms, TGDs, negative constraints, and separable EGDs, along with a probabilistic model M over the set X.

Definition 2. If a is an atom, σ_T is a TGD, σ_{NC} is a negative constraint (or EGD), and λ is a valid probabilistic annotation, then: (i) $a : \lambda$ is a *probabilistic atom*; (ii) $\sigma_T : \lambda$ is a *probabilistic TGD (pTGD)*; (iii) $\sigma_{NC} : \lambda$ is a *probabilistic (negative) constraint* (pNC). Such probabilistic atoms, TGDs, and (negative) constraints are *annotated formulas*.

Intuitively, annotated formulas hold whenever the events associated with their probabilistic annotations occur. Note that whenever a random variable's value is left unspecified in an annotation, the variable is unconstrained; in particular, an empty annotation means that the formula holds in every possible scenario. Given a set S of annotated formulas, we use $\mathit{det}(S)$ to denote the set of formulas in S without the corresponding probabilistic annotations.

Definition 3. Let O be a set of probabilistic atoms, TGDs and (negative) constraints, and M be a probabilistic model. A *probabilistic Datalog+/– ontology* is of the form $\Phi = (O, M)$, where the annotations in formulas in O are relative to M.

In the following, we assume that probabilistic ontologies are of the form $\Phi = (O, M)$ with $O = D \cup \Sigma$, where D is a set of probabilistic atoms and Σ is a set of pTGDs and pNCs. Thus, to avoid repetition, these components are often left implicit.

Recall that random variables in the probabilistic model are Boolean and written in the form of atoms over $\mathcal{R}_M$, $\mathcal{V}_M$, and Δ_M; if a is such an atom, $a = 1$ (resp., $a = 0$) denotes that the variable is *true* (resp., *false*).

Definition 4. Let $\Phi = (O, M)$ be a *probabilistic Datalog+/– ontology*, and λ be a probabilistic scenario. The (non-probabilistic) Datalog+/– ontology *induced* from Φ by λ, denoted O_λ, is the set:

$$\{\theta_i F_i \mid \exists\, F_i{:}\lambda_i \in O \textit{ such that there exists an mgu } \theta_i \textit{ for } \lambda_i \textit{ and some } \lambda' \subseteq \lambda\}.$$

As in the case of non-probabilistic Datalog+/– ontologies, we assume that, given a scenario λ, the (uniquely determined) ontology O_λ contains non-conflicting key dependencies [4] to ensure the separability of the constraints. The annotation of formulas offers a clear modeling advantage by enabling a clear separation between the task of ontological modeling and of modeling the uncertainty in the ontology. More precisely, in our formalism, it is possible to express the fact that the probabilistic nature of a formula is determined by elements that are *outside of the domain modeled by the ontology*.

Example 3. Consider the linear Datalog+/– ontology from Example 1, and let the set of random variables in M be $X = \{p(x_1), p(x_2), r(x_1), r(x_2)\}$. We annotate formulas in the Datalog+/- ontology with classes of scenarios built from the random variables in X. Atoms are annotated in the following way:

$$a(x_1) : \{\langle p(x_1) = 1\rangle, \langle r(x_2) = 1\rangle\}, \quad b(x_1), b(x_2) : \{\langle p(X) = 1\rangle, \langle r(x_1) = 0\rangle\},$$
$$d(x_1), e(x_1) : \{\langle p(x_1) = 1\rangle, \langle r(x_2) = 1\rangle, \langle p(x_2) = 0\rangle\}.$$

The annotations for the rest of the atoms is $\{\}$; recall that formulas with annotation "$\{\}$" hold irrespective of the setting of the random variables in M. The annotations for the rest of the formulas are:

$$\sigma_1, \sigma_2 : \{\}, \quad \sigma_3 : \{\langle p(x_2) = 0\rangle\}, \quad \upsilon_1 : \{\langle r(x_1) = 1\rangle\}, \quad \upsilon_2 : \{\}.$$ ■

In the following, we use the concept of *decomposition* of probabilistic ontologies in their constituent (non-probabilistic) ontologies induced by the scenarios.

Definition 5. Let $\Phi = (O, M)$ is a probabilistic Datalog+/– ontology; the *decomposition* of Φ, denoted $\mathit{decomp}(\Phi)$, is the structure: $\mathit{decomp}(\Phi) = ([O_{\lambda_1}, \ldots, O_{\lambda_n}], M)$ where $\{\lambda_1, \ldots, \lambda_n\} = \mathit{scn}(M)$; we also refer to this structure as the *decomposed form* of Φ. To simplify notation, we assume that scenarios are ordered according to a lexicographical order of the values of the variables, and therefore the i-th ontology in a decomposition corresponds to the i-th scenario in this ordering.

Example 4. Consider again the ontology $\Phi = (O, M)$ from Example 3. There are 16 scenarios. Therefore, the decomposition of Φ has the form $\mathit{decomp}(\Phi) = ([O_{\lambda_1}, \ldots, O_{\lambda_{16}}], M)$. The scenario λ_5 is given by the set $\{\langle p(x_1) = 1\rangle, \langle p(x_2) = 0\rangle, \langle r(x_1) = 1\rangle, \langle r(x_2) = 1\rangle\}$, while λ_7 is given by $\{\langle p(x_1) = 1\rangle, \langle p(x_2) = 0\rangle, \langle r(x_1) = 0\rangle, \langle r(x_2) = 1\rangle\}$. It is easy to see that O_{λ_5} consists of the set of atoms $\{a(x_1), d(x_1), e(x_1)\}$, along with the constraints in Σ, and O_{λ_7} consists of the set of atoms $\{a(x_1), b(x_1), b(x_2), d(x_1), e(x_1)\}$ and the set of constraints $\{\sigma_1, \sigma_2, \sigma_3, \upsilon_2\}$. ■

3.1 Semantics

The semantics of probabilistic Datalog+/– ontologies is given relative to probabilistic distributions over *interpretations* of the form $\mathcal{I} = (D, v)$, where D is a database over $\Delta_{\mathit{Ont}} \cup \Delta_N$, and v is a scenario. In the following, we abbreviate "$\mathit{true} : \lambda$" with "λ".

Definition 6. An interpretation $\mathcal{I} = (D, v)$ *satisfies* an annotated formula $F : \lambda$, denoted $\mathcal{I} \models F : \lambda$, iff whenever there exists an mgu θ such that for all $\langle V_i = x_i\rangle \in \lambda$, it holds that $X_i = \theta V_i$ and $v[i] = x_i$, then $D \models \theta F$.

A *probabilistic interpretation* is then a probability distribution Pr over the set of all possible interpretations such that only a finite number of interpretations are mapped to a non-zero value. The probability of an annotated formula $F : \lambda$, denoted $\mathit{Pr}(F : \lambda)$, is the sum of all $\mathit{Pr}(\mathcal{I})$ such that $\mathcal{I}$ satisfies $F : \lambda$.

Definition 7. Let Pr be a probabilistic interpretation, and $F : \lambda$ be an annotated formula. We say that *Pr satisfies* (or is a *model* of) $F : \lambda$ iff $\mathit{Pr}(F : \lambda) = 1$. Furthermore, *Pr satisfies* (or is a *model* of) a probabilistic Datalog+/– ontology $\Phi = (O, M)$ iff: (i) Pr satisfies all annotated formulas in O, and (ii) $1 - \mathit{Pr}(\mathit{false} : \lambda) = \mathit{Pr}_M(\lambda)$ for all scenarios λ, where $\mathit{Pr}_M(\lambda)$ is defined as $\mathit{Pr}_M(\bigwedge_{\langle V_i, x_i\rangle \in \lambda} V_i = x_i)$ in M.

In Definition 7 above, condition (ii) is simply stating that the probability values that *Pr* assigns are in accordance with those of M (note that the equality in the definition implies that $Pr(true : \lambda) = Pr_M(\lambda)$) and that they are adequately distributed (since $Pr(true : \lambda) + Pr(false : \lambda) = 1$).

We now define the notion of a *probabilistic answer* for a conjunctive query to a probabilistic Datalog+/– ontology.

Definition 8. Let $\Phi = (O, M)$ be a probabilistic Datalog+/– ontology and $Q(\mathbf{X})$ be a CQ. A *probabilistic answer* for $Q(\mathbf{X})$ in Φ is a pair (θ, p) consisting of a ground substitution θ for the variables in $Q(\mathbf{X})$ and some real number $p \in [0, 1]$, such that $p = \sum_{\lambda \in scn(M),\, O_\lambda \models \theta Q(\mathbf{X})} Pr_M(\lambda)$.

Intuitively, (θ, p) is a probabilistic answer for $Q(\mathbf{X})$ iff p is the sum of the probabilities of the scenarios for which $\theta Q(\mathbf{X})$ holds in the corresponding induced ontology. Alternatively, we say that Φ *entails* $Q(\mathbf{X})$ *with probability* p, iff there exists some ground substitution θ such that (θ, p) is a probabilistic answer for $Q(\mathbf{X})$ to Φ.

Example 5. Consider again the ontology $\Phi = (O, M)$ from Example 1 and the query $Q(X) = a(X)$. A probabilistic answer for $Q(X)$ to Φ (the only one in this case) is the pair $([X/x_1], p)$, where $p = \sum_{\lambda_i \models \langle p(x_1)=1\rangle \wedge \langle r(x_2)=1\rangle} \Pr(\lambda_i) = \Pr(\lambda_1) + \Pr(\lambda_3) + \Pr(\lambda_5) + \Pr(\lambda_7)$. Note that $\lambda_i \models \langle p(x_1) = 1\rangle \wedge \langle r(x_2) = 1\rangle$ denotes the set of scenarios in M in which $p(x_1)$ and $r(x_2)$ are true. ■

3.2 Annotated Chase

We now modify the chase procedure for probabilistic Datalog+/– ontologies in order for it to account for the correct propagation of probabilistic events associated with the atomic consequences derived from the ontology.

pTGD Chase Rule. Consider a probabilistic Datalog+/– ontology $\Phi = (O, M)$ and a pTGD $\sigma \in O$ of the form $\Upsilon(\mathbf{X}, \mathbf{Y}) \rightarrow \exists \mathbf{Z}\, \Psi(\mathbf{X}, \mathbf{Z}) : e_\sigma$. Let D be the set of probabilistic atoms in O. Then, σ is *applicable* to D if there exists a homomorphism h that maps the atoms in $\Upsilon(\mathbf{X}, \mathbf{Y})$ to probabilistic atoms $\beta_1 : e_1, \ldots, \beta_k : e_k$ of D such that $h(e_\sigma \wedge \bigwedge_{i=1,\ldots,k} e_i)$ is a valid probabilistic annotation. Let σ be applicable to D, and h_1 be a homomorphism that extends h as follows: for each $X_i \in \mathbf{X}$, $h_1(X_i) = h(X_i)$; for each $Z_j \in \mathbf{Z}$, $h_1(Z_j) = z_j$, where z_j is a "fresh" null, i.e., $z_j \in \Delta_N$, z_j does not occur in O, and z_j lexicographically follows all other nulls already introduced. The *application of* σ on D adds to D the probabilistic atom $h_1(\Psi(\mathbf{X}, \mathbf{Z})) : h(e_\sigma \wedge \bigwedge_{i=1,\ldots,k} e_i)$, if not already in D. ■

The chase algorithm for a probabilistic ontology Φ consists of an exhaustive application of the pTGD chase rule in a breadth-first (level-saturating) fashion, which outputs a (possibly infinite) *annotated chase* for Φ. In the construction of the annotated chase for a probabilistic Datalog+/– ontology, we annotate each atom not only with its derivation level, but also the probabilistic event associated with its derivation. Note that there might be several ways of obtaining an atom through the application of different pTGDs and therefore associated with different probabilistic events; if the events are different, then different probabilistic atoms are created.

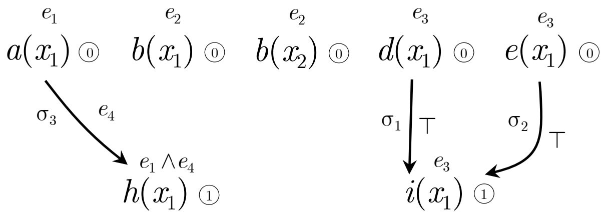


Fig. 1. Annotated chase graph for the probabilistic Datalog+/– ontology Φ from Example 1. The events shown are defined as: $e_1 = \{\langle p(x_1) = 1\rangle, \langle r(x_2) = 1\rangle\}$, $e_2 = \{\langle p(x_1) = 1\rangle, \langle p(x_2) = 1\rangle\}$, $e_3 = \{\langle p(x_1) = 1\rangle, \langle r(x_2) = 1\rangle, \langle p(x_2) = 0\rangle\}$, and $e_4 = \{\langle p(x_2) = 0\rangle\}$.

The *annotated chase of level up to* 0 of $\Phi = (O, M)$ relative to the set of pTGDs $\Sigma_T \subseteq O$, denoted $annChase^0(\Phi)$, is defined assigning to every probabilistic atom in O the (derivation) level 0 and the probabilistic annotation associated with it in the ontology. For every $k \geqslant 1$, the *annotated chase of level up to* k of O relative to Σ_T, denoted $annChase^k(\Phi)$, is constructed as follows: let $I_1, \ldots, I_n$ be all possible images of bodies of pTGDs in Σ_T relative to some homomorphism such that (i) $I_1, \ldots, I_n \subseteq annChase^{k-1}(\Phi)$ and (ii) the highest level of an atom in every I_i is $k-1$; then, perform every corresponding pTGD application on $annChase^{k-1}(\Phi)$, choosing the applied pTGDs and homomorphisms in a (fixed) linear and lexicographic order, respectively, and assigning to every new atom a probabilistic annotation equal to the conjunction of the probabilistic annotations associated with the atoms in I_i and that of the pTGD σ, and the (derivation) level k. The *annotated chase* of Φ, denoted *annChase*(Φ), is then defined as the limit of $annChase^k(D, \Sigma)$ for $k \to \infty$. The *chase graph* for a probabilistic (guarded) Datalog+/– ontology Φ is the directed graph consisting of *annChase*(Φ) as the set of nodes and having an arc from a to b iff b is obtained from a by a one-step application of a pTGD.

Example 6. Figure 1 shows the annotated chase graph for the probabilistic Datalog+/– ontology Φ from Example 1. The atom $m(x_1)$ is added to level 1 of the annotated chase with probabilistic annotation $e_1 \wedge e_4 = \{\langle p(x_1) = 1\rangle, \langle r(x_2) = 1\rangle, \langle p(x_2) = 0\rangle\}$. Note that the atom $i(x_1)$ is annotated with e_3, because it is derived either from $d(x_1)$ or $e(x_1)$, both annotated with e_3 in O, through the application of σ_1 or σ_2, respectively, whose annotations are both *true*. The pair $([X/x_1], p)$ is a probabilistic answer for the query $i(X)$ with $p = Pr_M(e_3) = Pr_M(\lambda_5) + Pr_M(\lambda_7)$. ■

The following result shows the relationship between the chase in non-probabilistic Datalog+/– ontologies and the annotated chase in the probabilistic case. Intuitively, the process of constructing the annotated chase is equivalent to constructing the classical chase in parallel for every non-probabilistic ontology induced by some scenario.

Proposition 1. *If* $\Phi = (O, M)$ *is a probabilistic Datalog+/– ontology and* $k \geq 0$*, then:*
(i) det$(annChase^k(\Phi)) = \bigcup_{\lambda \in scn(M)} chase^k(O_\lambda)$*,*
(ii) if $A = \{\alpha : e_i \mid \alpha : e_i \in annChase^k(\Phi)\}$ *for some atom* α *and* $e = \bigvee_{\alpha : e_i \in A} e_i$ *then for every* $\lambda \in scn(M)$ *such that* $\lambda \models e$*, it holds that* $\alpha \in chase^k(O_\lambda)$*, and*
(iii) there is no $\lambda \in scn(M)$ *such that* $\alpha \in chase^k(O_\lambda)$ *and* $\lambda \not\models e$*.*

Note that using Proposition [1], we can directly conclude that $\mathit{det}(\mathit{annChase}(\Phi)) = \bigcup_{\lambda \in \mathit{scn}(M)} \mathit{chase}(O_\lambda)$, and therefore, as in the case of (classical) guarded Datalog+/– ontologies, whenever (homomorphic images of) the query atoms are contained in the (annotated) chase, then the entire derivation of the query atoms are also contained in a finite, initial portion of the annotated guarded chase graph, whose size is determined only by the query and the schema.

The following proposition establishes the link between probabilistic answers for a conjunctive query to a probabilistic ontology and the corresponding annotated chase; it allows to compute probabilistic answers for a query to a probabilistic ontology by means of computing the annotated chase, thus avoiding the brute-force approach of computing every classical chase as described above.

Proposition 2. *Let Φ be a probabilistic guarded Datalog+/– ontology and Q be a CQ. Then, (θ, p), $p > 0$, is a probabilistic answer for Q to Φ iff $p = \mathit{Pr}_M(\bigvee_{S_i \in \mathcal{S}} \bigwedge_{\alpha : e_j \in S_i} e_j)$ where $\mathcal{S}$ is the set of all sets S_i such that S_i is a smallest subset of $\mathit{annChase}(\Phi)$ (in the set-theoretic inclusion sense) for which there is a homomorphism from the atoms in θQ to the atoms in $\mathit{det}(S_i)$.*

4 Inconsistent Probabilistic Datalog+/– Ontologies

Inconsistency in probabilistic Datalog+/– ontologies can be characterized as follows.

Definition 9. Let $\Phi = (O, M)$ be a probabilistic Datalog+/– ontology. Then, Φ is *inconsistent* iff there exists a scenario λ in M such that O_λ is inconsistent, i.e., $\mathit{chase}(O_\lambda) \models \mathit{body}(\nu)$, for some (negative) constraint $\nu \in \Sigma_\lambda$. Otherwise, Φ is *consistent*.

Example 7. Consider again the ontology from Example [3]. Both O_{λ_5} and O_{λ_7} are inconsistent, since in both we have the atom $a(x_1)$, and therefore, through σ_3, the atom $m(x_1)$, which, together with the atom $d(x_1)$, violates the negative constraint υ_2. ■

In this setting, the data complexity is defined as: the set of pTGDs and pNCs are considered to be fixed; on the other hand, the sets Δ_{Ont} and Δ_M are not, and therefore the set of ground atoms on the ontology side and the set of random variables on the probabilistic model side are not considered to be fixed, either. The following proposition shows that the problem of checking consistency is intractable in the data complexity. The result holds both for guarded and linear probabilistic Datalog+/– ontologies.

Theorem 1. *Given $\Phi = (O, M)$, a probabilistic guarded Datalog+/– ontology, the problem of deciding whether Φ is inconsistent is NP-complete in the data complexity.*

The treatment of inconsistency that we approach in this work is based on the concept of *repairs* and *consistent answers*. We first define the notion of repair of probabilistic Datalog+/– ontologies as an extension of the notion of repair from the relational database setting [1]. A *data repair* for a non-probabilistic Datalog+/– ontology (D, Σ) is a database instance D' such that the Datalog+/– ontology (D', Σ) is consistent and D' minimally differs from D in the set-inclusion sense. Clearly, there can be more than one maximal subset of D that is consistent relative to the sets of constraints. We denote the set of data repairs for $O = (D, \Sigma)$ by $\mathit{DRep}(O)$. The fact that an arbitrary data repair for a non-probabilistic Datalog+/– ontology can be computed in polynomial time in the data complexity follows almost directly from the results in [1].

Definition 10. Let $\Phi = (O, M)$ be a (possibly inconsistent) probabilistic Datalog+/– ontology such that $decomp(\Phi) = ([(D_1, \Sigma_1), \ldots, (D_n, \Sigma_n)], M)$. Then, a probabilistic Datalog+/– ontology $\Phi' = (O', M)$ is a *repair for* Φ iff Φ' is consistent and is such that $decomp(\Phi') = ([(r_1, \Sigma_1), \ldots, (r_n, \Sigma_n)], M)$ where $r_i \in \mathit{DRep}((D_i, \Sigma_i))$, $1 \leq i \leq n$. We denote with $\mathit{Rep}(\Phi)$ the set of repairs for Φ.

Intuitively, a repair for a probabilistic ontology $\Phi = (O, M)$ is a consistent probabilistic ontology that corresponds to choosing a data repair for each non-probabilistic Datalog+/– ontology induced by some scenario in M.

Example 8. Let Φ be the probabilistic ontology from Example 3 with the decomposition $decomp(\Phi) = ([(D_1, \Sigma_1), \ldots, (D_{16}, \Sigma_{16})], M)$. The ontology Φ has four repairs of the form $\Phi_i = ([(r^i_1, \Sigma^i_1), \ldots, (r^i_{16}, \Sigma^i_{16})], M)$, where we have:

$$\begin{aligned}
&r^1_5 = \{a(x_1), e(x_1)\}, r^1_7 = \{b(x_1), b(x_2), d(x_1), e(x_1)\},\\
&r^2_5 = \{a(x_1), e(x_1)\}, r^2_7 = \{a(x_1), b(x_1), b(x_2), e(x_1)\},\\
&r^3_5 = \{d(x_1), e(x_1)\}, r^3_7 = \{a(x_1), b(x_1), b(x_2), e(x_1)\},\\
&r^4_5 = \{d(x_1), e(x_1)\}, r^4_7 = \{d(x_1), b(x_1), b(x_2), e(x_1)\}.
\end{aligned}$$ ■

Repairs for a probabilistic Datalog+/– ontology are defined as the combination of data repairs for each Datalog+/– ontology induced by each scenario. Unfortunately, in contrast with the non-probabilistic case, checking if a probabilistic Datalog+/– ontology is a repair for another one is intractable in the data complexity if no restrictions are made over the number of scenarios in M.

Theorem 2. *Let $\Phi = (O, M)$ and $\Phi' = (O', M)$ be probabilistic guarded Datalog+/– ontologies. Deciding whether Φ' is a repair of Φ is coNP-complete in data complexity.*

4.1 Consistent Answers for Probabilistic Datalog+/– Ontologies

The area of *consistent query answering* (CQA) enforces consistency at query time. The work of [1], one of the seminal works in the area, provides a model-theoretic construct of a database repair. The most widely accepted semantics for querying a possibly inconsistent database is that of *consistent answers*, which yields the set of tuples (atoms) that appear in the answer to the query over *every* possible repair. CQA allows to focus on a smaller portion of the database and, in general, it is not necessary to materialize every possible repair. Here, we generalize the notion of consistent answers to the setting of probabilistic guarded Datalog+/– ontologies.

Definition 11. Let $\Phi = (O, M)$ be a probabilistic (guarded) Datalog+/– ontology, $Q(\mathbf{X})$ be a CQ, and θ be a ground substitution for the variables in $\mathbf{X}$. A pair $(\theta, [\ell, u])$ is a *consistent answer* for $Q(\mathbf{X})$ to Φ iff for every repair $\Phi' \in \mathit{Rep}(\Phi)$, we have that (θ, p) is a probabilistic answer for $Q(\mathbf{X})$ to Φ' and $\ell \leq p \leq u$. We say that $(\theta, [\ell, u])$ is a *tight consistent answer* iff (θ, ℓ) and (θ, u) are both probabilistic answers for Q to some (not necessarily the same) repair $\Phi' \in \mathit{Rep}(\Phi)$.

Intuitively, the fact that $(\theta, [\ell, u])$ is a tight consistent answer for Q means that every repair that derives θQ does it with a probability within $[\ell, u]$ and the endpoints of the interval are tight in the sense that they are the minimum and the maximum probabilities with which θQ is derived in every repair. That is, under every possible way of fixing the probabilistic ontology, we can derive θQ with a probability within that interval.

Example 9. Consider again the ontology Φ from Example 3 (which has four repairs as shown in Example 8) and the query $Q(X) = i(X)$. We can easily check that $([X/x_1], [p, p])$, with $p = \mathit{Pr}_M(\lambda_5) + \mathit{Pr}_M(\lambda_7)$, is a tight consistent answer for Q to Φ. On the other hand, suppose that $\mathit{Pr}_M(\lambda_1) = 0.1$, $\mathit{Pr}_M(\lambda_3) = 0.3$, $\mathit{Pr}_M(\lambda_5) = 0.2$, and $\mathit{Pr}_M(\lambda_7) = 0.2$; for $Q'(X) = a(X)$, the pair $([X/x_1], [\ell, u])$, with $\ell = \mathit{Pr}_M(\lambda_1) + \mathit{Pr}_M(\lambda_3) = 0.4$ and $u = \mathit{Pr}_M(\lambda_1) + \mathit{Pr}_M(\lambda_3) + \mathit{Pr}_M(\lambda_5) + \mathit{Pr}_M(\lambda_7) = 0.8$, is a tight consistent answer for Q'. ■

The next result shows that deciding whether $(\theta, [\ell, u])$ is a consistent answer is coNP-complete in the data complexity even when the number of scenarios in $\mathit{scn}(M)$ is restricted to be polynomial in the size of the data in the ontology.

Theorem 3. *Let $\Phi = (O, M)$ be a guarded probabilistic Datalog+/– ontology, $Q(\mathbf{X})$ be a CQ, and θ be a ground substitution for the variables in $Q(\mathbf{X})$. If $|\mathit{scn}(M)|$ is polynomial in the size of O, and $\mathit{Pr}_M(e)$ can be computed in polynomial time for an arbitrary event e, then deciding whether $(\theta, [\ell, u])$ is a consistent answer for Q to Φ is coNP-complete in the data complexity.*

The above result shows that it is intractable to compute consistent answers even if we restrict the probabilistic model M. This is unfortunate but expected, since even for non-probabilistic guarded Datalog+/– ontologies, the consistent query answering problem is coNP-complete [11]. In pursuit of tractable CQA, in the next section we adopt and generalize for the case of probabilistic guarded (and in particular linear) Datalog+/– ontologies the notion of consistent answers under the intersection semantics, previously studied for non-probabilistic guarded Datalog+/– ontologies [11] and originally introduced by [9] for the case of certain families of tractable description logics.

4.2 Computing Consistent Answers under the Intersection Semantics

In the case of non-probabilistic ontologies, the work in [9] proposes an alternative semantics that considers only the atoms that are in the *intersection* of all data repairs, which constitutes a sound approximation to consistent answers. We now study the generalization of this semantics to probabilistic guarded Datalog+/– ontologies.

Let $\Phi_1 = (O_1, M), \ldots, \Phi_k = (O_k, M)$ be probabilistic ontologies; the intersection of $\Phi_1, \ldots, \Phi_k$, denoted $\bigcap_{i=1,\ldots,k} \Phi_i$, is a probabilistic ontology $\Phi = (O, M)$, such that for each $\lambda \in$ *scn(M)*, $O_\lambda = (\bigcap_{i=1,\ldots,k} D_{i_\lambda}, \bigcap_{i=1,\ldots,k} \Sigma_{i_\lambda})$, where $O_{i_\lambda} = (D_{i_\lambda}, \Sigma_{i_\lambda})$.

Definition 12. Let $\Phi = (O, M)$ be a probabilistic Datalog+/– ontology, and Q be a CQ. Then, (θ, p) is a *consistent answer under the intersection semantics* for Q to Φ, iff (θ, p) is a probabilistic answer for Q to $\bigcap_{\Phi_i \in \mathit{Rep}(\Phi)} \Phi_i$.

As in the classical case, the intersection semantics is a sound approximation to consistent answers; it is easy to see that, if (θ, p) is a consistent answer for a query under the intersection semantics, then every repair derives the query with probability $p^* \geq p$.

Example 10. Consider Φ from Example 3. The intersection of $\Phi_i \in \mathit{Rep}(\Phi)$ yields $\Phi' = (O', M)$ where $O'_{\lambda_5} = (\{e(x_1)\}, \Sigma_{\lambda_5})$, $O'_{\lambda_7} = (\{b(x_1), b(x_2), e(x_1)\}, \Sigma_{\lambda_7})$, and $O'_\lambda = (O_\lambda, \Sigma_\lambda)$ for any other $\lambda \in \mathit{scn}(M)$. Example 9 shows that $([X/x_1], [p, p])$, with

$p = Pr_M(\lambda_5) + Pr_M(\lambda_7)$, is a consistent answer for $Q(X) = i(X)$ to Φ. We can see that $([X/x_1], p)$ is a consistent answer for Q to Φ under the intersection semantics since Φ' entails $i(x_1)$ with probability p through $e(x_1) : e_3$ and σ_2 in Φ'. ■

In [11], it was shown that computing consistent answers under the intersection semantics in a non-probabilistic Datalog+/– ontology (D, Σ), is the same as computing classical answers on the ontology $(D - C, \Sigma)$, where C corresponds to the union of minimal inconsistent subsets relative to the set of negative constrains in Σ. We obtain a similar result for the case of consistent answers under the intersection semantics for probabilistic ontologies. First, we define the notion of a *culprit* relative to a set of probabilistic negative constraints, which is informally a minimal inconsistent set (under set inclusion) of probabilistic atoms relative to the set of constraints.

Definition 13. Let $\Phi = (O, M)$ be a probabilistic Datalog+/– ontology, and D be the set of probabilistic atoms in O. A *culprit* in Φ is a pair $\langle e_c, c\rangle$, where e_c is a valid probabilistic annotation and $c \subseteq D$, such that for every scenario $\lambda \in \mathit{scn}(M)$, $\mathit{det}(c)$ is a minimal inconsistent subset in O_λ iff $\lambda \models e_c$.

Example 11. In our running example, $\{a(x_1), d(x_1)\}$ is a minimal inconsistent subset of O_{λ_5} and O_{λ_7}, and therefore $\langle e_3, c = \{a(x_1) : e_1, d(x_1) : e_3\}\rangle$ is a culprit (the only one) in Φ. Suppose now that we add to Φ a pNC $\upsilon_3 : i(X) \rightarrow \perp\ : \{\}$. Now, $\langle e_3, c\rangle$ is no longer a culprit, since there exists $c_1 = \{d(x_1) : e_3\} \subset c$ such that c_1 is an inconsistent subset of O_{λ_5} and O_{λ_7}. Furthermore, $\langle e_3, \{e(x_1) : e_3\}\rangle$ is a culprit for Φ, since from it and σ_2 we can derive $i(x_1)$ and therefore violate constraint υ_3 in O_{λ_5} and O_{λ_7}. ■

We now establish the relationship between culprits and consistent answers under the intersection semantics. Intuitively, (θ, p), $p > 0$, is a consistent answer under the intersection semantics for Q in Φ iff there is at least one way of deriving θQ from O such that there is no scenario in which the atoms involved in the derivation belong to a culprit. We now introduce the notion of root set for $Q(\mathbf{X})$ in Φ, which is, intuitively, a minimal set of probabilistic atoms from which a probabilistic answer for Q can be computed in O.

Definition 14. Let $\Phi = (D \cup \Sigma_T \cup \Sigma_{NC}, M)$ be a probabilistic guarded Datalog+/– ontology, and $Q(\mathbf{X})$ be a CQ. A *root set* for $Q(\mathbf{X})$ in Φ is a pair $\langle e_r, r\rangle$ such that: (i) $r \subseteq D$, (ii) there exists a ground substitution θ such that $\mathit{annChase}((r \cup \Sigma_T, M)) \models \theta Q$ with probability $Pr_M(e_r)$, and (iii) there is no $r' \subset r$ such that $\mathit{annChase}((r' \cup \Sigma_T, M)) \models \theta Q$ with probability $Pr_M(e'_r)$, where $e_r \wedge e'_r$ is a valid probabilistic annotation. We denote with $\mathit{roots}(Q(\mathbf{X}), \Phi)$ the set of root sets for $Q(\mathbf{X})$ in Φ.

Example 12. Consider Example 3; the set of root sets for $i(X)$ in Φ is $\mathit{roots}(i(X), \Phi) = \{\langle e_3, r_1 = \{d(x_1) : e_3\}\rangle, \langle e_3, r_2 = \{e(x_1) : e_3\}\rangle\}$. If we consider the query $Q'(X) = d(X) \wedge m(X)$, then $\mathit{roots}(Q'(X), \Phi) = \{\langle e_3, r'_1 = \{a(x_1) : e_1, d(x_1) : e_3\}\rangle\}$. ■

The probabilistic annotation of the root set represents the set of scenarios in which the probabilistic atoms in the root set, together with the applicable pTGDs, derive an answer for $Q(\mathbf{X})$. Note that if $Q(\mathbf{X})$ is atomic, then each root set for $Q(\mathbf{X})$ in Φ is a singleton whenever Φ is a probabilistic linear Datalog+/– ontology.

Root sets for a conjunctive query $Q(\mathbf{X})$ in a probabilistic guarded Datalog+/– ontology are simple to compute: the basic idea is to traverse the annotated chase graph back

```
Algorithm findCulprits($\Phi$, annChaseGraph $G$)
 1. $culp$:= $\emptyset$;
 2. for each pNC $\upsilon : e_\upsilon \in O$ do
 3.     $culp$:= $culp \cup$ findRootSets($\Sigma_T, G, body(\upsilon)$);
 4. let $G_C = (V, E)$:= computeContainmentGraph($culp$);
 5. for every node $n_i \in V$ (corresponding to $\langle e_i, c_i \rangle \in culp$) with in-degree zero do
 6.     for every $n_j \in V$ (corresponding to $\langle e_j, c_j \rangle \in culp$) s.t. $(n_i, n_j) \in E$ do
 7.         if $e_i \wedge e_j$ is a valid probabilistic annotation then
 8.             if $e_i \wedge \neg e_j$ is valid then change $e_i$ to $e_i \wedge \neg e_j$ in $\langle e_i, c_i \rangle \in culp$;
 9.             else remove $\langle e_i, c_i \rangle$ from $culp$;
10.     remove $n_i$ from $V$ and every edge $(n_i, n_j)$ from $E$;
11. return $culp$;
```

Fig. 2. Computing the set of culprits in a probabilistic guarded Datalog+/– ontology

from the last level up to level 0, following each possible derivation of $Q(\mathbf{X})$ until each necessary minimal subset of $annChase^0(\Phi)$ that produces an answer for $Q(\mathbf{X})$ is identified. The annotation in a root set corresponds to the event with which that particular answer is obtained in the last level of the annotated chase graph.

Such an algorithm can also be used to compute the set of culprits in a probabilistic (guarded) Datalog+/– ontology in the following way. Let $Q(\mathbf{X}) = body(\upsilon)$ for some pNC $\upsilon \in O$, where $\mathbf{X}$ are the universally quantified variables appearing in $body(\upsilon)$; we can compute the set of culprits in Φ *relative only to* υ by computing the root sets of $Q(\mathbf{X})$. For instance, in our running example, let $Q(X) = d(X) \wedge m(X)$ (which corresponds to the body of υ_2); the set of root sets is $roots(Q, \Phi) = \{\langle e_3, r'_1 = \{a(x_1), d(x_1)\}\rangle\}$, and actually $\{a(x_1) : e_3, d(x_1) : e_3\}$ is the unique culprit in Φ relative only to υ_2. The set of culprits in Φ is not necessarily the union of the culprits of individual pNCs, since they might violate the minimality requirement. Therefore, after computing the culprits for individual pNCs we must check and eliminate any root set for $body(\upsilon)$ that is a strict superset of a root set for another $body(\upsilon')$ (to check this we need to check containment of atoms and the overlapping of probabilistic annotations). Algorithm findCulprits in Figure 2 shows how to compute the set of culprits in a probabilistic Datalog+/– ontology assuming there is available a subroutine findRootSets that computes the root sets of a query, and computeContainmentGraph that produces a directed graph in which there is an edge between two culprits c_1 and c_2 iff the set of probabilistic atoms in c_1 is a superset of those in c_2.

Proposition 3. *Let Φ be a probabilistic (guarded) Datalog+/– ontology and G its annotated chase graph. Algorithm* findCulprits(Φ, G) *correctly computes the set culprits(Φ).*

Intuitively,$(\theta, p), p > 0$, is a consistent answer under the intersection semantics for Q in Φ iff there is at least one way of obtaining θQ from O with probability p such that there is no scenario in which the atoms involved in the derivation belong to a culprit.

Proposition 4. *Given a probabilistic Datalog+/– ontology Φ and a CQ $Q(\mathbf{X})$, (θ, p), $p > 0$, is a consistent answer for Q under the intersection semantics iff there is $rs = \langle e, r \rangle \in roots(Q, \Phi)$ and $p = \Pr(e')$, with $e' = e \wedge \neg \bigwedge_{\langle e_c, r_c \rangle \in culprits(\Phi), det(r_c) \cap det(r) \neq \emptyset} e_c$.*

Example 13. Consider the root sets for the query $Q(X) = i(X)$ shown in Example 12. Note that r_1 has a non-empty intersection with $c = \{a(x_1) : e_1, d(x_1) : e_3\}$, which is part of a culprit for Φ. That is not the case for the root set r_2, and therefore $([X/x_1], \Pr(e_3))$ is a consistent answer for Q to Φ under the intersection semantics. ■

Proposition 4 provides a direct way of computing the consistent answers for a query Q under the intersection semantics. That is, we can compute the root sets for Q (the set of all probabilistic answers for Q) to Φ and then verify which of them is a consistent answer under the intersection semantics by inspecting the culprits and checking for intersections between root sets and the culprits. The following theorem states that this procedure is tractable for probabilistic linear Datalog+/– ontologies, as long as the cost of probabilistic inference in the model remains tractable as well.

Theorem 4. *Let $\Phi = (D \cup \Sigma, M)$ be a linear probabilistic Datalog+/– ontology and $Q(\mathbf{X})$ be a CQ. If $|scn(M)|$ is polynomial in the size of D, then the consistent answers under the intersection semantics for $Q(\mathbf{X})$ to Φ can be computed in time $O(poly(|O|) + CP(M))$, where $poly(|O|)$ is the time needed to compute the culprits and the root sets for $Q(\mathbf{X})$ in Φ, and $CP(M)$ is the time needed to compute the probability of an arbitrary probabilistic annotation in M.*

5 Related Work

In databases, the fields of *database repairing* and *consistent query answering* (CQA) have gained much attention since the work of [1], which provided a model-theoretic construct of a database *repair*. The most widely accepted semantics for querying a possibly inconsistent database is that of *consistent answers*, which yields the set of tuples (atoms) that appear in the answer to the query over *every* possible repair. CQA enforces consistency at query time as an alternative to enforcing it at the instance level, as conventional data cleaning techniques do. More recently, several works have focused on inconsistency handling for several classes of DLs, adapting and specializing general techniques previously considered for traditional logics. The work in [9] studies the adaptation of CQA for *DL-Lite* ontologies, and presents the *intersection* semantics as a sound approximation of consistent answers, which is easier to compute for *DL-Lite*. In [11], an adaptation of this semantics is provided for guarded Datalog+/– ontologies.

The closest approach in the literature is perhaps the one in [10] for CQA over probabilistic databases, in which tuples are annotated with probability values. In contrast, the probabilistic Datalog+/– language adopted here allows the separation of the probabilistic model from the data model, yielding a tool that is more suitable for applications related to the Semantic Web. Finally, the extension of CQA proposed here adheres to the widely adopted criterion of minimal change, which is not the case in [10].

6 Conclusion

In this paper, we have extended consistent query answering and its approximation by the intersection semantics to the case of probabilistic Datalog+/– ontologies. We have analyzed the complexity of deciding the existence of repairs, consistent answers, and consistency checking, and we have identified a fragment of our formalism for which consistent query answering under the intersection semantics is tractable.

Acknowledgments. This work was supported by the Engineering and Physical Sciences Research Council (EPSRC) grant EP/J008346/1 "PrOQAW: Probabilistic Ontological Query Answering on the Web", the European Research Council (FP7/2007-2013)/ERC grant 246858 (DIADEM), by a Google Research Award, and by a Yahoo! Research Fellowship.

References

1. Arenas, M., Bertossi, L.E., Chomicki, J.: Consistent query answers in inconsistent databases. In: Proc. PODS 1999, pp. 68–79 (1999)
2. Beeri, C., Vardi, M.Y.: The implication problem for data dependencies. In: Proc. ICALP 1981, pp. 73–85 (1981)
3. Calì, A., Gottlob, G., Kifer, M.: Taming the infinite chase: Query answering under expressive relational constraints. In: Proc. KR 2008, pp. 70–80 (2008)
4. Calì, A., Gottlob, G., Lukasiewicz, T.: A general Datalog-based framework for tractable query answering over ontologies. Journal of Web Semantics 14, 57–83 (2012)
5. Deutsch, A., Nash, A., Remmel, J.B.: The chase revisited. In: Proc. PODS 2008, pp. 149–158 (2008)
6. Fagin, R., Kolaitis, P.G., Miller, R.J., Popa, L.: Data exchange: Semantics and query answering. Theoretical Computer Science 336(1), 89–124 (2005)
7. Gottlob, G., Lukasiewicz, T., Simari, G.I.: Conjunctive Query Answering in Probabilistic Datalog+/– Ontologies. In: Rudolph, S., Gutierrez, C. (eds.) RR 2011. LNCS, vol. 6902, pp. 77–92. Springer, Heidelberg (2011)
8. Johnson, D.S., Klug, A.C.: Testing containment of conjunctive queries under functional and inclusion dependencies. Journal of Computer and System Sciences 28(1), 167–189 (1984)
9. Lembo, D., Lenzerini, M., Rosati, R., Ruzzi, M., Savo, D.F.: Inconsistency-Tolerant Semantics for Description Logics. In: Hitzler, P., Lukasiewicz, T. (eds.) RR 2010. LNCS, vol. 6333, pp. 103–117. Springer, Heidelberg (2010)
10. Lian, X., Chen, L., Song, S.: Consistent query answers in inconsistent probabilistic databases. In: Proc. SIGMOD 2010, pp. 303–314 (2010)
11. Lukasiewicz, T., Martinez, M.V., Simari, G.I.: Inconsistency handling in Datalog+/– ontologies. In: Proc. ECAI 2012 (2012)
12. Maier, D., Mendelzon, A.O., Sagiv, Y.: Testing implications of data dependencies. ACM Transaction of Database Systems 4(4), 455–469 (1979)
13. Richardson, M., Domingos, P.: Markov logic networks. Machine Learning 62, 107–136 (2006)

Query Rewriting under Ontology Contraction

Eleni Tsalapati, Giorgos Stoilos, Giorgos Stamou, and George Koletsos

School of Electrical and Computer Engineering,
National Technical University of Athens,
Zographou Campus, 15780, Athens, Greece

Abstract. Conjunctive query (CQ) answering is a key reasoning service for ontology-based data access. One of the most prominent approaches to conjunctive query answering is query rewriting where a wide variety of systems has been proposed the last years. All of them accept as input a fixed CQ q and ontology $\mathcal{O}$ and produce a rewriting for $q, \mathcal{O}$. However, in many real world applications ontologies are very often dynamic—that is, new axioms can be added or existing ones removed frequently. In this paper we study the problem of computing a rewriting for a CQ over an ontology that has been *contracted* (i.e., some of its axioms have been removed) given a rewriting for the input CQ and ontology. Our goal is to compute a rewriting directly from the input rewriting and avoid computing one from scratch. We study the problem theoretically and provide sufficient conditions under which this process is possible. Moreover, we present a practical algorithm which we implemented and evaluated against other state-of-the-art systems obtaining encouraging results. Finally, axiom removal can also be relevant to ontology design. For each test ontology we study how much the removal of an axiom affects the size of the rewriting and the performance of systems. If the removal of a single axiom causes a significant decrease either in the size or in the computation time then this part of the ontology can be re-modelled.

Keywords: Ontologies, Query Rewriting, Ontology contraction, Axiom Removal.

1 Introduction

A recent application of ontologies that continuously gains momentum is *ontology-based data access* (OBDA) [17]. Ontologies aim to provide a formal semantically rich conceptualization of the (possibly distributed) data layer, thus simplifying numerous data management problems such as information integration [13], data exchange [9], data warehousing [23] and more. The main advantages of OBDA are that, firstly, the conceptual data description does not directly reflect the specific system/storage specifications and restrictions and, secondly, the data access can be performed by answering *conjunctive queries* (CQs) expressed in terms of the ontology [17], which is usually more intuitive for the user.

Unfortunately, the problem of answering conjunctive queries over ontologies expressed using expressive ontology languages (like those underpinning the Web

Ontology Language OWL 2) has been proved to be very difficult [14]. Consequently, less expressive languages (like those underpinning the OWL 2 QL and OWL 2 EL fragments) have been proposed in the literature. For these languages query answering is tractable [5,16,11] and thus efficient systems can be implemented. One of the widely used methods to query answering over such languages is *query rewriting*. Given a query q and an ontology $\mathcal{O}$, a rewriting r of $q, \mathcal{O}$ is a set of clauses (usually Datalog rules or unions of CQs) such that for any database the answers of q over the database and the ontology coincide with the answers of the rewriting over the database and discarding the ontology. Thus, r can be used for finding the answers by translating it into an (recursive) SQL query.

So far many algorithms and systems for computing the rewriting of a query over an ontology have been developed in the literature [5,16,19,11,6,15,22]. Several of them, like Presto, Quest,[1] Nyaya,[2] Rapid,[3] and IQAROS[4] employ sophisticated optimisations in order to reduce either the size of the computed rewriting or the computation time. Despite very encouraging results it is clear that the problem of query rewriting remains open since there are several problematic cases for which either the computation time or the computed rewriting are quite large. The latter is a significant problem as it is well-known that large rewritings are likely to cause a problem when evaluated over a database (a large rewriting implies a large SQL query with many unions and joins) [11].

All the aforementioned systems assume a fixed query and ontology and for this input they employ a set of 'rewriting' rules in order to compute the target rewriting. However, in many applications ontologies are very often dynamic and can change in time [7,3,18]. More precisely, an ontology can be extended by adding new axioms or be *contracted*—that is, some of its axioms might be removed because they no longer hold. For example, the NCI ontology (a well-known medical ontology) has been updated more than 85 times [10]. In such scenarios all aforementioned algorithms would compute a rewriting for the input (fixed) query over the updated ontology from scratch discarding any information previously computed, although it is expected that the new rewriting has a significant overlap with the initial one. In the current paper we study the following problem: Given a query q, an ontology $\mathcal{O}$, a rewriting r for $q, \mathcal{O}$ and a set of axioms A, compute a rewriting for $q, \mathcal{O} \setminus \mathsf{A}$ 'directly' from r and by avoiding using any of the known rewriting algorithms. Firstly, we study the problem theoretically to investigate its feasibility. We thus develop sufficient conditions that if satisfied by r then this process is possible. Subsequently, we present a practical algorithm for computing the rewriting of a query over a contracted ontology. Finally, we have implemented our algorithm and we have conducted an experimental evaluation using the evaluation framework proposed in [16]. We compared the performance of our system to the performance of cutting-edge query rewriting systems and we obtained encouraging results.

[1] http://obda.inf.unibz.it/protege-plugin/quest/quest.html

[2] http://mais.dia.uniroma3.it/Nyaya/Home.html

[3] http://www.image.ece.ntua.gr/~achort/rapid.zip

[4] http://code.google.com/p/iqaros/

To the best of our knowledge there is no previous study of the problem of query rewriting over contracted ontologies. We believe that such an algorithm can be helpful in cases where computing the rewriting for a large and complex ontology is time consuming. In such cases an initial rewriting can be computed once while then rewritings for contractions of the input ontology can be computed using a lightweight algorithm. Additionally, ontology contraction can also be interesting in designing ontologies for practical applications. More precisely, given an ontology $\mathcal{O}$ and query q the proposed method can be used to investigate which of the axioms of $\mathcal{O}$ affect the size of the rewriting for $q, \mathcal{O}$. More precisely, if r is a rewriting for $q, \mathcal{O}$ while for some $\alpha \in \mathcal{O}$ r' is a rewriting for $q, \mathcal{O} \setminus \{\alpha\}$ that is significantly smaller than r, then we can deduce that the presence of α makes rewriting particularly 'hard' and hence should be revised. The specific idea has lately gained attention in the area of terminological reasoning over expressive DLs with important theoretical and practical results [10]. However, as far as we know it has not been studied in the area of query rewriting. Finally, our techniques are also relevant to the problem of ontology repairing for incomplete reasoners [21], which provides an alternative and very promising way to scalable ontology-based data access. More precisely, our methods can be used to compute a repair for a contracted ontology by avoiding re-computing one from scratch.

2 Preliminaries

We use standard notions of first-order constants, variables, function symbols, terms, substitutions, predicates, atoms, (ground) formulae, sentences, and entailment ($\models$). A *fact* is a ground atom and an *instance* is a finite set of facts. A tuple (vector) of variables (constants) is denoted by $\vec{x}$ ($\vec{a}$). For ϕ a formula, with $\phi(\vec{x})$ we denote that $\vec{x}$ are the free variables of ϕ, while for σ a substitution, $\phi\sigma$ is the result of applying σ to ϕ. Satisfiability and entailment are defined as usual.

Existential Rules. An *existential rule* [2,4], often called *axiom*, is a sentence of the form

$$\forall\vec{x}.\forall\vec{z}.[\phi(\vec{x},\vec{z}) \rightarrow \exists\vec{y}.\psi(\vec{x},\vec{y})] \tag{1}$$

where $\phi(\vec{x},\vec{z})$ and $\psi(\vec{x},\vec{y})$ are conjunctions of atoms and $\vec{x}, \vec{y}$ and $\vec{z}$ are pair-wise disjoint. Formula ϕ is the *body*, formula ψ is the *head*, and universal quantifiers are often omitted. If $\vec{y}$ is empty, the rule is called *datalog*. An *ontology* $\mathcal{R}$ is a finite set of existential rules.

Many popular Description Logics, such as $\mathcal{ELHI}$ [16], as well as database constraint languages, such as *tuple generating dependencies* [1], can be captured by existential rules.

Queries. A *datalog query* $\mathcal{Q}$ is a tuple $\langle Q_P, P\rangle$, where Q_P is a query predicate and P is a set of datalog rules such that the body of each clause in P does not contain Q_P. A datalog query $\mathcal{Q} = \langle Q_P, P\rangle$ is called a *union of conjunctive queries* (UCQ) if Q_P is the only head predicate in the head of the rules in

P; furthermore, $\mathcal{Q}$ is a *conjunctive query* (CQ) if it is a union of conjunctive queries and P contains exactly one rule—that is, $\mathcal{Q} = \langle Q_P, \{Q_C\}\rangle$ and Q_C is a datalog rule with Q_P as a head predicate. We often abuse notation and write $\mathcal{Q} = Q_C$ if $\mathcal{Q}$ is a CQ in which case the head of Q_C is the query predicate. A tuple of constants $\vec{a}$ is a *certain answer* of a datalog query $\mathcal{Q} = \langle Q_P, P\rangle$ over an ontology $\mathcal{R}$ and an instance I if and only if $\mathcal{R} \cup I \cup P \models Q_P(\vec{a})$. We denote with $\mathsf{cert}(\mathcal{Q}, \mathcal{R} \cup I)$ the set of certain answers of the datalog query $\mathcal{Q}$ over $\mathcal{R} \cup I$.

Given two datalog rules r_1, r_2 we say that r_2 *subsumes* r_1 if there exists a substitution σ such that $(r_2)\sigma \subseteq r_1$.

Query Rewriting. Intuitively, a *rewriting* of $\mathcal{Q}$ w.r.t. an ontology $\mathcal{R}$ is another query that captures all the information from $\mathcal{R}$ relevant for answering $\mathcal{Q}$ over $\mathcal{R}$ and an arbitrary instance I. Today several query rewriting algorithms for many ontology languages such as DL-Lite, $\mathcal{ELHI}$, *linear*-TGDs and many more have been presented [5,16,11,15]. For all these works, UCQs and datalog are common target languages for computing a query rewriting.

Definition 1. *A* datalog rewriting *of a conjunctive query $\mathcal{Q} = \langle Q_P, \{Q_C\}\rangle$ w.r.t. an ontology $\mathcal{R}$ is a datalog query $\mathcal{Q}' = \langle Q_P, P\rangle$ such that the following properties hold:*

- *each $r \in P$ either does not mention Q_P or contains Q_P only in the head,*
- *for each $r \in P$ we have $\mathcal{R} \cup \mathcal{Q} \models r$,*
- *for each instance I using only predicates from $\mathcal{R}$ we have $\mathsf{cert}(\mathcal{Q}, \mathcal{R} \cup I) = \mathsf{cert}(\mathcal{Q}', I)$.*

If the rewriting $\mathcal{Q}'$ is a UCQ then it is called UCQ rewriting.

Many state-of-the-art systems often normalise the input ontology $\mathcal{R}$ by introducing *new* (*fresh*) predicates that do not appear in $\mathcal{R}$ and then compute a rewriting using the normalised ontology. For example, the ontology $\mathcal{R} = \{R(x, y) \wedge C(y) \wedge D(y) \rightarrow A(x)\}$ would usually be normalised to $\mathcal{R}' = \{R(x, y) \wedge A_0(y) \rightarrow A(x), C(x) \wedge D(x) \rightarrow A_0(x)\}$, where A_0 is a new predicate. For such systems the second condition of Definition 1 is likely not to hold. However, the rules of the rewriting that contain such fresh predicates can be eliminated by 'unfolding' the definition of the fresh symbols creating new rules for which the condition holds.

3 Rewriting Reduced TBoxes

In this section we study the problem of computing a rewriting for a conjunctive query $\mathcal{Q}$ over an ontology $\mathcal{R}'$ given a rewriting for $\mathcal{Q}$ and an ontology $\mathcal{R} \supseteq \mathcal{R}'$—that is, given a rewriting for $\mathcal{Q}$ over a superset of $\mathcal{R}'$. Since rewriting over large ontologies can be a rather time consuming process our motivation is to avoid computing the new rewriting from scratch using any of the standard algorithms, but instead to re-use the previously computed information as much as possible. In the following, we first study the problem at a theoretical level providing illustrative examples that highlight important technical points and motivate several assumptions that are required and, then, we present the algorithm in detail.

Example 1. Consider the ontology $\mathcal{R}_1 = \{\alpha_1, \alpha_2, \alpha_3\}$, where α_1, α_2, and α_3 are defined as follows:

$$\begin{aligned}\alpha_1 &= \mathsf{Painting}(x) \rightarrow \mathsf{ManMadeObject}(x),\\ \alpha_2 &= \mathsf{isSimilarTo}(x,y) \wedge \mathsf{Painting}(y) \rightarrow \mathsf{Painting}(x),\\ \alpha_3 &= \mathsf{isCopyOf}(x,y) \rightarrow \mathsf{isSimilarTo}(x,y)\end{aligned}$$

The ontology states that a painting is a man made object, that if some object is similar to a painting then it is also a painting and anything that is a copy of an entity is also similar to this entity. Consider now the CQ $\mathcal{Q} = Q(x) \leftarrow \mathsf{ManMadeObject}(x)$. The datalog query $\mathcal{Q}' = \langle Q, P\rangle$, where P is the program consisting of the rules defined below, is a datalog rewriting of $\mathcal{Q}$ over $\mathcal{R}_1$:

$$q = Q(x) \leftarrow \mathsf{ManMadeObject}(x) \tag{2}$$

$$q_1 = Q(x) \leftarrow \mathsf{Painting}(x) \tag{3}$$

$$r_1 = \mathsf{Painting}(x) \leftarrow \mathsf{isSimilarTo}(x,y) \wedge \mathsf{Painting}(y) \tag{4}$$

$$r_2 = \mathsf{Painting}(x) \leftarrow \mathsf{isCopyOf}(x,y) \wedge \mathsf{Painting}(y) \tag{5}$$

This rewriting can be computed by any state-of-the-art query rewriting system that at-least supports the DL language $\mathcal{ELHI}$.

Assume now that we remove axiom α_3 from $\mathcal{R}_1$ obtaining the new ontology $\mathcal{R}_1' = \{\alpha_1, \alpha_2\}$. A new rewriting for $\mathcal{Q}$ and $\mathcal{R}_1$ can be computed using again the same algorithm; the rewriting would consist of rules (2)–(4). ♢

Although the new rewriting can be computed using again our rewriting system we can see that when applied over $\mathcal{Q}$ and $\mathcal{R}_1'$ this system would re-compute the rules (2)–(4). Moreover, we can see that one can compute a rewriting directly from $\mathcal{Q}'$ simply by removing rule (5) from the program P. Intuitively, this rule is produced by resolving rule (4) with axiom α_3 which has been removed from the initial ontology. Hence, this rule cannot be produced using the axioms of $\mathcal{R}_1'$. This suggests that if one has additionally annotated the elements of a rewriting with the subset of the ontology that is required to generate them, then a new rewriting would be easily computable.

Definition 2. *Let $\mathcal{Q} = \langle Q_P, \{Q_C\}\rangle$ be a CQ, let $\mathcal{R}$ be an ontology, let $\mathcal{Q}_D = \langle Q_P, P\rangle$ be a datalog rewriting of $\mathcal{Q}$ w.r.t. $\mathcal{R}$ and let some $r \in P$. We say that $\mathcal{R}' \subseteq \mathcal{R}$ is* minimal *for r if the following conditions hold:*

1. *$r \in P'$ for some $P' \subseteq P$ s.t. $\langle Q_P, P'\rangle$ is a datalog rewriting for $\mathcal{Q}$ w.r.t. $\mathcal{R}'$.*
2. *For all $\mathcal{R}'' \subset \mathcal{R}'$ condition 1 does not hold.*

Intuitively, $\mathcal{R}'$ is minimal for some r if r occurs in some rewriting for $\mathcal{Q}$ and $\mathcal{R}'$, but if we remove any rule from $\mathcal{R}'$ then r no longer occurs in *any* rewriting for $\mathcal{Q}$ and the modified ontology.

In our running example (Example 1) we have the following minimal sets for each element of P:

$$\begin{array}{lll}\mathcal{R}_q & := \emptyset & \text{is minimal for } q\\ \mathcal{R}_{q_1} & := \mathcal{R}_q \cup \{\alpha_1\} & \text{is minimal for } q_1\\ \mathcal{R}_{r_1} & := \{\alpha_2\} & \text{is minimal for } r_1\\ \mathcal{R}_{r_2} & := \mathcal{R}_{r_1} \cup \{\alpha_3\} & \text{is minimal for } r_2\end{array}$$

Hence, since the minimal subset for r_2 contains the axiom α_3 that was previously removed from $\mathcal{R}_1$ to obtain $\mathcal{R}'_1$, we can deduce that r_2 cannot be part of a rewriting for $\mathcal{Q}, \mathcal{R}'_1$.

Note here that for a rule r of a rewriting for some query and ontology $\mathcal{R}$ there may be many minimal subsets. For example, for ontology $\mathcal{R} = \{A(x) \rightarrow B(x), C(x) \rightarrow B(x)\}$ and CQ $\mathcal{Q} = \langle Q(x), \{Q(x) \leftarrow A(x), B(x), C(x)\}\rangle$ the rewriting would contain the rule $r = Q(x) \leftarrow A(x), C(x)$ and both $\{A(x) \rightarrow B(x)\}$ and $\{C(x) \rightarrow B(x)\}$ are minimal for r. Hence, for each $r \in P$ the rewriting needs to contain all minimal subsets for a member of the rewriting.

Definition 3. *Let $\mathcal{Q} = \langle Q_P, \{Q_C\}\rangle$ be a CQ and let $\mathcal{R}$ be an ontology. A* labelled *datalog rewriting is a triple $\langle Q_P, P, \rho\rangle$ where $\langle Q_P, P\rangle$ is a datalog rewriting for $\mathcal{Q}$ w.r.t. $\mathcal{R}$ and ρ is a mapping from P to sets of subsets of $\mathcal{R}$ such that for each $r \in P$, $\rho(r)$ contains all $\mathcal{R}' \subseteq \mathcal{R}$ that are minimal for r.*

Note that to compute a labelled rewriting for a CQ $\mathcal{Q} = \langle Q, \{Q_C\}\rangle$ over an input set $\mathcal{R}$ one has to modify the internals of the used rewriting algorithm. This can be done easily by initialising the empty set $\emptyset$ as the minimal set for Q_C and the singleton set $\{\alpha\}$ for each axiom $\alpha \in \mathcal{R}$ and then track the axioms that are used to generate the elements of the output rewriting. For example, if r' is produced by resolving rule r with axiom α, then $\rho(r') = \rho(r') \cup \{\rho(r) \cup \rho(\alpha)\}$.

An important technical question at this point is whether we can compute a rewriting for a CQ over a reduced ontology $\mathcal{R}'$ given any rewriting for the input ontology $\mathcal{R} \supseteq \mathcal{R}'$. As the following example shows, this is not always possible.

Example 2. Consider the following ontology $\mathcal{R}_2$ and CQ:

$$\mathcal{R}_2 = \{\mathsf{Creator}(x) \rightarrow \mathsf{Agent}(x)\}$$
$$\mathcal{Q} = Q(x) \leftarrow \mathsf{Creator}(x), \mathsf{Agent}(x)$$

The tuple $\mathcal{Q}_1 = \langle \mathcal{Q}(x), \{r, r_1\}\rangle$, where $r = Q(x) \leftarrow \mathsf{Creator}(x), \mathsf{Agent}(x)$ and $r_1 = Q(x) \leftarrow \mathsf{Creator}(x)$ is a rewriting for $\mathcal{Q}$, $\mathcal{R}_2$. However, r_1 subsumes r, hence $\mathcal{Q}_2 = \langle \mathcal{Q}(x), \{r_1\}\rangle$, is also a rewriting for $\mathcal{Q}, \mathcal{R}_2$.

Assume now that axiom $\alpha_1 = \mathsf{Creator}(x) \rightarrow \mathsf{Agent}(x)$ is removed from $\mathcal{R}_2$ obtaining a new ontology $\mathcal{R}'_2$. A rewriting for $\mathcal{Q}, \mathcal{R}'_2$ consists of the query $\mathcal{Q}_3 = \langle Q(x), \{r\}\rangle$. However, it is quite clear that we cannot compute $\mathcal{Q}_3$ from the (non-redundant) rewriting $\mathcal{Q}_2$, as it does not contain the rule r at all. Instead, $\mathcal{Q}_3$ can be computed from $\mathcal{Q}_1$ that contains rule r simply by removing r_1. ◊

Intuitively, the issue in the previous example is that although r is redundant in $\mathcal{Q}_1$ the query that subsumes it (r_1) is not part of all rewritings for $\mathcal{Q}, \mathcal{R}'_2$ because the axiom that is used to generate it (i.e., α_1) has been removed. Hence, r is no longer redundant in rewritings of $\mathcal{Q}, \mathcal{R}'_2$.

As we will show next, the following condition that was first introduced in [8], provides a sufficient condition for computing a rewriting for an ontology $\mathcal{R}'$ given a rewriting for an ontology $\mathcal{R} \supseteq \mathcal{R}'$.

Definition 4. *A datalog rewriting $\langle Q_P, P\rangle$ of a CQ $\mathcal{Q} = \langle Q_P, \{Q_C\}\rangle$ w.r.t. an ontology $\mathcal{R}$ is* subset-closed *if for each $\mathcal{R}' \subseteq \mathcal{R}$ there exists $P' \subseteq P$ such that $\langle Q_P, P'\rangle$ is a datalog rewriting for $\mathcal{Q}$ w.r.t. $\mathcal{R}'$.*

Example 3. Consider the ontology $\mathcal{R}_2$ and CQ $\mathcal{Q}$ from Example 2. For $\mathcal{R}_2'' = \emptyset$ no subset of $\mathcal{Q}_2$ is a datalog rewriting for $\mathcal{Q}, \mathcal{R}_2''$. Instead, for the rewriting $\mathcal{Q}_1$ the query $\mathcal{Q}_1' = \langle Q(x), \{r\}\rangle$ is a datalog rewriting for $\mathcal{Q}, \mathcal{R}_2''$. Therefore, $\mathcal{Q}_1$ is subset-closed while $\mathcal{Q}_2$ is not. $\diamondsuit$

As noted in [8], however, from a practical point of view subset-closed rewritings are not straightforward to compute. As also illustrated by the above example, to compute such rewritings one would typically need to disable (at least partially) subsumption-based optimisations, whereas many rewriting systems are optimised hence their output is typically not subset-closed. However, on the one hand, there exist highly efficient algorithms and systems that compute subset-closed rewritings [22] and on the other hand, we argue that one can compute a subset-closed rewriting once as an off-line procedure and then a lightweight algorithm can be used to compute rewritings for the revised ontologies.

Concluding our presentation of the technical issues of the algorithm we show that there are certain kinds of dependencies between the elements of a labelled rewriting which the algorithm can exploit in order to compute the new rewriting more efficiently.

Example 4. Consider our running example (Example 1) and assume that instead of α_3 we remove axiom α_2 creating the new ontology $\mathcal{R}_1'' = \{\alpha_1, \alpha_3\}$. A rewriting for $\mathcal{Q}, \mathcal{R}_1''$ consists only of rules (2) and (3). The algorithm can compute this by checking whether for all $\mathcal{R}_{r_1} \in \rho(r_1)$ we have $\alpha_2 \in \mathcal{R}_{r_1}$, which holds hence r_1 is removed, and then the same for r_2, which again holds hence r_2 is also removed obtaining finally the correct rewriting.

However, the latter check can be avoided if we order the elements of the rewriting according to the order induced by their minimal sets in ρ. More precisely, in our running example the (only) minimal set for r_2 is a superset of the (only) minimal set for $\rho(r_1)$; hence if r_1 is removed because $\alpha_2 \in \mathcal{R}_{r_1}$ then all rules produced "after" r_1 (i.e., r_2) can be discarded from further processing. $\diamondsuit$

To exploit the above idea the algorithm introduced in the next section first orders the elements of a rewriting according to their minimal sets. This is performed using the function order that is defined next.

Definition 5. *Let $\mathcal{Q}_D = \langle Q_P, P, \rho\rangle$ be a labelled rewriting for a CQ $\mathcal{Q}$ w.r.t. an ontology $\mathcal{R}$. The function* $\mathsf{order}(\mathcal{Q}_D)$ *returns a directed graph $\mathcal{G} = \langle P, \mathcal{H}\rangle$ where $\langle r_1, r_2\rangle \in \mathcal{H}$ iff for all $\mathcal{R}_1 \in \rho(r_1)$ there exists $\mathcal{R}_2 \in \rho(r_2)$ such that $\mathcal{R}_1 \subset \mathcal{R}_2$ and no $r' \in P$ exists such that for some $\mathcal{R}' \in \rho(r')$ we have $\mathcal{R}_1 \subset \mathcal{R}' \subset \mathcal{R}_2$.*

In our running example the function $\mathsf{order}(\mathcal{Q}')$ would return $\mathcal{G} = \langle P, \mathcal{H}\rangle$ where $\mathcal{H} = \{\langle q, q_1\rangle, \langle r_1, r_2\rangle\}$.

Algorithm 1. $\mathsf{DELETE}(\mathsf{A}, \mathcal{Q}_D)$

```
    Input: $\mathcal{Q}_D = \langle Q_P, P, \rho\rangle$ is a labelled datalog rewriting and $\mathsf{A}$ a set of axioms.
 1: $\mathcal{G} := \mathsf{order}(\mathcal{Q}_D)$
 2: Initialise a triple $\mathcal{Q}'_D = \langle Q_P, P', \rho'\rangle$, where $P' = \emptyset$ and $\rho'$ is an empty mapping
 3: Initialise a stack $S$ to contain all vertices $r$ of $\mathcal{G}$ s.t. $\nexists r'.\langle r', r\rangle \in \mathcal{G}$
 4: while $S \neq \emptyset$ do
 5:     Pop an element $r$ from $S$
 6:     if $\mathcal{R}_i \in \rho(r)$ exists s.t. $\mathsf{A} \cap \mathcal{R}_i = \emptyset$ then
 7:         Add $r$ to $P'$
 8:         if $\rho'(r)$ is undefined then
 9:             Initialise $\rho'(r) := \emptyset$
10:         end if
11:         for all $\mathcal{R}_j \in \rho(r)$ do
12:             if $\mathsf{A} \cap \mathcal{R}_j = \emptyset$ then
13:                 $\rho'(r) = \rho'(r) \cup \{\mathcal{R}_j\}$
14:             end if
15:         end for
16:         Push all $r'$ such that $\langle r, r'\rangle \in \mathcal{G}$ to $S$
17:     end if
18: end while
19: return $Q'_D$
```

3.1 The Delete Algorithm

As described previously it is possible to compute a rewriting for a CQ $\mathcal{Q}$ and an ontology $\mathcal{R}'$ from some subset-closed labelled rewriting for $\mathcal{Q}$ and a superset of $\mathcal{R}'$ without relying at all on traditional rewriting algorithms. Such a detailed algorithm is depicted in Algorithm 1.

Algorithm Delete accepts as input a labelled datalog rewriting $\mathcal{Q}_D$ for a query $\mathcal{Q}$ and ontology $\mathcal{R}$ and a set $\mathsf{A} \subseteq \mathcal{R}$ of axioms to be removed from $\mathcal{R}$ and produces a new datalog rewriting for $\mathcal{Q}, \mathcal{R} \setminus \mathsf{A}$. First, the algorithm calls function order to sort the elements of $\mathcal{Q}_D$ and create a directed graph $\mathcal{G}$ (line 1) while then it initialises a new labelled datalog rewriting $\mathcal{Q}'_D$ which will be the output of the algorithm. Then $\mathcal{G}$ is traversed in a depth-first manner (using a stack S) and checks if for some element r of the graph there exists a minimal subset in $\rho(r)$ that does not contain any element of A. This implies that r can be generated by not using any of the removed axioms and hence should be in the output of the algorithm. Thus, r is added to the new rewriting (line 7) and $\rho'(r)$ is set to all minimal subsets of r that do not contain any axiom from A (lines 11–15). Finally, all successor nodes of r in the graph are added to the stack (line 16).

Example 5. Consider the ontology $\mathcal{R}_1$ of the running example (Example 1) extended by the set of axioms $\{\alpha'_1, \alpha'_2, \alpha'_3\}$, where α'_1, α'_2, α'_3 are defined as follows:

$$\begin{aligned}
\alpha'_1 &= \mathsf{Potrait}(x) \rightarrow \mathsf{Painting}(x) \\
\alpha'_2 &= \mathsf{Fossil}(x) \rightarrow \mathsf{ManMadeObject}(x), \\
\alpha'_3 &= \mathsf{ResinFossil}(x) \rightarrow \mathsf{Fossil}(x)
\end{aligned}$$

The new ontology $\mathcal{R}'_1 = \mathcal{R}_1 \cup \{\alpha'_1, \alpha'_2, \alpha'_3\}$ additionally states that a potrait is a painting, a fossil is a man made object and a resin fossil is a fossil. Consider again the CQ of Example 1. The query $\mathcal{Q}' = \langle Q, P', \rho \rangle$, where $P' = P \cup \{q'_1, q'_2, q'_3\}$ and q'_1, q'_2, q'_3 are defined as follows, is a labelled subset-closed datalog rewriting of $\mathcal{Q}, \mathcal{R}'_1$:

$$q'_1 = Q(x) \leftarrow \mathsf{Potrait}(x) \tag{6}$$

$$q'_2 = Q(x) \leftarrow \mathsf{Fossil}(x) \tag{7}$$

$$q'_3 = Q(x) \leftarrow \mathsf{ResinFossil}(x) \tag{8}$$

Assume now that we remove the axiom α'_2. We will show how Algorithm 1 will compute a rewriting for $\mathcal{Q}, \mathcal{R} \setminus \{\alpha'_2\}$.

The algorithm would first initialise a rewriting $\mathcal{Q}'_D = \langle Q_P, P', \rho' \rangle$, with $P' = \emptyset$ and ρ' an empty mapping. Then, it would execute the function $\mathsf{order}(\mathcal{Q}')$ which would return the directed graph $\mathcal{G} = \langle P, \mathcal{H} \rangle$, where

$$\mathcal{H} = \{\langle q, q_1 \rangle, \langle q_1, q'_1 \rangle, \langle q, q'_2 \rangle, \langle q'_2, q'_3 \rangle, \langle r_1, r_2 \rangle\}$$

Then, initially S would contain q and r_1. Suppose that q is popped. Since $\mathcal{R}_q = \emptyset$ CQ q would be added to P' and $\rho'(q)$ is set to $\emptyset$. Since $\langle q, q_1 \rangle$, $\langle q, q'_2 \rangle \in \mathcal{H}$, the CQs q_1, q'_2 are pushed in the stack. Suppose that q_1 is popped from the stack next. Since $\mathcal{R}_{q_1} = \{\alpha_1\}$ the condition in line 6 is satisfied and q_1 would also be added to P' while the algorithm sets $\rho'(q_1) = \{\alpha_1\}$. Similarly, q'_1 is added to P' and so far we have $P' = \{q'_1, q_1, q\}$.

Now since there is no q' s.t. $\langle q'_1, q' \rangle \in \mathcal{H}$ nothing is pushed in the stack. Next, q'_2 is popped from the stack. Since $\mathcal{R}_{q'_2} = \{\alpha'_2\}$ the condition of line 6 is not satisfied; therefore the algorithm continuous with the next element of the stack which is r_1. Following the same process as before the algorithm would add r_1 and r_2 to P', hence we will have $P' = \{r_1, r_2, q'_1, q_1, q\}$. It can be verified that the datalog rewriting $\langle Q(x), P' \rangle$ returned by the algorithm is a rewriting for $\mathcal{Q}, \mathcal{R} \setminus \{\alpha_2\}$. $\diamondsuit$

Next we show correctness of Algorithm 1.

Theorem 1. *Let $\mathcal{R}$ be an ontology, let $\mathcal{Q} = \langle Q, P_0 \rangle$ be a CQ and let $\mathcal{Q}_D = \langle Q, P, \rho \rangle$ be a labelled datalog rewriting for $\mathcal{Q}, \mathcal{R}$ that is subset-closed. Let also A be a subset of $\mathcal{R}$. When applied to A and $\mathcal{Q}_D$ Algorithm 1 terminates. Let $\mathcal{Q}'$ be the tuple produced by the algorithm; then, $\mathcal{Q}'$ is a rewriting for $\mathcal{Q}, \mathcal{R} \setminus \mathsf{A}$ that is subset-closed.*

Proof. First we show termination. Let $\mathcal{G}$ be the graph computed at line 1 of Algorithm 1. First, we show that $\mathcal{G}$ is a directed acyclic graph. Assume that there is a cycle in $\mathcal{G}$—that is, there exist vertices r_1, r_2 such that r_2 is reachable from r_2 and r_1 from r_2. By Definition 5 we have that for all $\mathcal{R}_1 \in \rho(r_1)$ there exists $\mathcal{R}_2 \in \rho(r_2)$ s.t. $\mathcal{R}_1 \subset \mathcal{R}_2$. Let an arbitrary $\mathcal{R}_1$ and $\mathcal{R}_2$. From the latter we also get that for this specific $\mathcal{R}_2$ there exists $\mathcal{R}'_1 \in \rho(r_1)$ s.t. $\mathcal{R}_2 \subset \mathcal{R}'_1$. Hence, we have that $\mathcal{R}_1 \subset \mathcal{R}'_1$ which contradicts the assumption that $\mathcal{R}'_1$ is minimal for r_1.

Now, since $\mathcal{G}$ is a directed acyclic graph and Algorithm 1 performs a standard depth-first traversal of $\mathcal{G}$ the algorithm clearly terminates.

Next we show that $\mathcal{Q}' = \langle Q, P' \rangle$ computed by the algorithm is a rewriting for $\mathcal{Q}, \mathcal{R} \setminus \mathsf{A}$. In order to show this it suffices to show that for all instances I we have $\mathsf{cert}(\mathcal{Q}, \mathcal{R} \setminus \mathsf{A} \cup I) = \mathsf{cert}(\mathcal{Q}', \mathcal{A})$.

First, we show that $\mathsf{cert}(\mathcal{Q}, \mathcal{R} \setminus \mathsf{A} \cup I) \supseteq \mathsf{cert}(\mathcal{Q}', I)$. By construction, a rule r is in P' only if there exists $\mathcal{R}' \in \rho(r)$ such that $\mathcal{R}' \subseteq \mathcal{R} \setminus \mathsf{A}$. This implies that $\mathcal{R} \setminus \mathsf{A} \models \mathcal{R}'$. Moreover, by definition of ρ we have that r belongs in some rewriting for $\mathcal{Q}, \mathcal{R}'$ hence we have that $\mathcal{R}' \cup \mathcal{Q} \models r$. From both conditions and monotonicity it follows that $\mathcal{R} \setminus \mathsf{A} \cup \mathcal{Q} \models r$. This holds for all members of P' hence we have $(\mathcal{R} \setminus \mathsf{A}) \cup \mathcal{Q} \models P'$ which implies that for any I the answers of $\mathcal{Q}'$ over I are also answers of $\mathcal{Q}$ over $(\mathcal{R} \setminus \mathsf{A}) \cup I$.

Second, we show that $\mathsf{cert}(\mathcal{Q}, \mathcal{R} \setminus \mathsf{A} \cup I) \subseteq \mathsf{cert}(\mathcal{Q}', I)$. Since $\mathcal{Q}_D = \langle Q, P \rangle$ is subset-closed there exists $P'' \subseteq P$ such that $\langle Q, P'' \rangle$ is a rewriting for $\mathcal{R} \setminus \mathsf{A}$. Clearly for each $r \in P''$ and for each $\mathcal{R}' \in \rho(r)$ we have $\mathcal{R}' \subseteq \mathcal{R} \setminus \mathsf{A}$. Hence, it follows easily by construction of P' that it contains r and thus we have $P'' \subseteq P'$; hence, $P' \models P''$ and it follows that any answer of $\mathcal{Q}$ over $(\mathcal{R} \setminus \mathsf{A}) \cup I$ is also an answer of $\mathcal{Q}'$ over I.

Finally, we show that $\mathcal{Q}' = \langle Q, P' \rangle$ is subset-closed. Consider some arbitrary subset $\mathcal{R}_s \subseteq \mathcal{R} \setminus \mathsf{A}$. Clearly, $\mathcal{R}_s \subseteq \mathcal{R}$ and since $\mathcal{Q} = \langle Q, P \rangle$ is subset-closed then there exists some $P_s \subseteq P$ s.t. $\langle Q, P_s \rangle$ is a rewriting for $\mathcal{Q}, \mathcal{R}_s$. By the latter we get that for all $r_s \in P_s$ we have $\mathcal{R}_s \cup \mathcal{Q} \models r_s$, hence there exists $\mathcal{R}'_s \in \rho(r_s)$ that is minimal for r_s. It follows easily that $r_s \in P'$. Since r_s is an arbitrary rule we have that $P_s \subseteq P'$. Moreover, also note that $\mathcal{R}_s$ is arbitrary. Hence, it follows that $\mathcal{Q}'$ is subset-closed. $\square$

The performance of Algorithm 1 can be further improved if one additionally has pre-computed and stored the subsumption relations between the elements of the input rewriting $\mathcal{Q}_D$. This can be accomplished by executing the standard subsumption checking algorithm over $\mathcal{Q}_D$ and creating an additional mapping λ such that for a clause r, $\lambda(r)$ contains the subsumers of r in $\mathcal{Q}_D$. Algorithm 1 uses λ as follows:

- When it selects a new clause r in line 5 it proceeds in processing r only if $\lambda(r) = \emptyset$ or none of the clauses in $\lambda(r)$ is already in P'; otherwise r and all the clauses that are "after" r in the graph can be discarded by continuing with the next element in the stack.
- In line 16 it checks whether for some clause r_k such that $\langle r, r_k \rangle \in \mathcal{H}$ we have $r_k \in \lambda(r)$. In such case, only r_k is pushed to the stack.

The correctness of this optimisation is a straightforward consequence of the correctness of subsumption for First-Order logic. More precisely, if a clause r subsumes a clause r', then any resolution inference using r' will produce clauses that are subsumed by clauses produced using resolution over r. Hence, if r is already in P', then both r' and all descendant rules are redundant and can be discarded from the output. Note, however, that the output of this optimised algorithm is clearly not guaranteed to be subset-closed.

Concluding this section we comment on the problem of query answering. Note that a subset-closed rewriting $\mathcal{Q}$ for a query $\mathcal{Q}_0$ and ontology $\mathcal{R}$ can typically be much larger than an equivalent non-redundant one. Hence, for an instance I it would not be very practical to use $\mathcal{Q}$ to compute the answers of $\mathcal{Q}_0$ over $\mathcal{R} \cup I$. In such setting one should use $\mathcal{Q}$ to compute a rewriting for further constructions of $\mathcal{R}$ while to evaluate the computed rewriting a non-redundant one should be computed from $\mathcal{Q}$. Note that given λ this is a fairly easy task.

4 Evaluation

We have developed a prototype tool for computing the rewriting of a conjunctive query w.r.t. a contracted ontology based on Algorithm 1. Our implementation is based on the query rewriting system ProgRes [20]—that is, we have modified ProgRes to extract subset-closed labelled rewritings.[5] Then, Algorithm 1 is executed over the subset-closed rewriting and a set of axioms. We have developed two versions of the algorithm; an unoptimised one, called Del, and one that uses the optimisations outlined at the end of Section 3, called DelOpt.

We have compared our implementations against the standard (non-modified) version of ProgRes and a recently developed highly-optimised query rewriting system IQAROS which has been shown to outperform many existing rewriting systems [12]. For the evaluation we used the framework proposed in [16]. It consists of nine test ontologies together with a set of five hand-crafted test queries for each of them. All experiments were conducted on a Intel(R) Core (TM) with a 3.20GHz processor and 4GB of RAM.

For each test ontology and query we compute a subset-closed labelled rewriting and then execute Del and DelOpt by selecting one axiom of the input ontology. Finally, we remove the subsumed (redundant) clauses. This process is repeated for all axioms of the ontology. For ProgRes and IQAROS we measure the time to compute the rewriting for the respective contracted ontology from scratch. Table 1 shows the average computation time for each ontology and query. Note that all four tools returned rewritings of the same size so for brevity reasons we do not present these numbers.

Comparing Del with DelOpt we see that DelOpt is in most cases faster than Del. This is due to the optimisations that have been implemented which prune the search space of Algorithm 1 significantly by discarding parts of the graph $\mathcal{G}$ that are redundant, e.g., because for some r in line 5 we have $\lambda(r) \neq \emptyset$. However, in ontology $P5X$ queries Q_4 and Q_5, Del performs better than DelOpt. We concluded that this is due to the overhead of the implemented optimisations of DelOpt. More precisely, DelOpt needs to perform several checks over potentially large sets in order to decide whether a selected clause can be skipped. However, as shown by the table this is noticeable only in these two queries.

Compared to ProgRes and IQAROS, both Del and DelOpt are faster in the vast majority of cases. Actually, in most ontologies and queries DelOpt can compute a new rewriting almost instantaneously in less than 10 milliseconds. A large

[5] However, we plan to use other systems as well in the future.

Table 1. Performance results for Del, DelOpt, ProgRes, and IQAROS

	V					S				
	Q1	Q2	Q3	Q4	Q5	Q1	Q2	Q3	Q4	Q5
Del	0	0	0	0	0	0	1	7	8	88
DelOpt	0	0	0	0	0	0	0	3	3	26
ProgRes	2	2	11	36	15	1	5	23	25	157
IQAROS	1	1	2	3	3	0	1	8	6	98

	P5					P5X				
	Q1	Q2	Q3	Q4	Q5	Q1	Q2	Q3	Q4	Q5
Del	0	0	0	0	0	1	1	8	152	3014
DelOpt	0	0	0	0	0	0	1	8	158	3384
ProgRes	3	9	4	5	6	3	11	131	2891	123094
IQAROS	0	0	0	2	12	0	3	13	280	8431

	U					UX				
	Q1	Q2	Q3	Q4	Q5	Q1	Q2	Q3	Q4	Q5
Del	0	0	1	9	39	0	0	3	30	95
DelOpt	0	0	0	3	10	0	0	1	13	25
ProgRes	1	4	7	73	46	1	4	30	223	141
IQAROS	1	2	4	9	10	1	2	7	13	14

	A					AX				
	Q1	Q2	Q3	Q4	Q5	Q1	Q2	Q3	Q4	Q5
Del	1	0	0	1	1	1	2	15	26	-
DelOpt	0	0	0	0	1	1	2	16	17	-
ProgRes	34	9	30	122	604	117	2587	44104	41154	-
IQAROS	3	4	32	11	204	8	59	559	405	31125

difference compared to these systems can be noticed in $P5X$ query Q_5 as well as in ontologies A and AX which are particularly hard for them. However, note that we were not able to obtain results for ontology AX query Q_5 as the ProgRes implementation that we based Del did not terminate after 9 hours. Moreover, Del was slower than IQAROS in ontologies U, UX. By investigating these cases we concluded that this is due to the large size of the rewriting $\mathcal{Q}_D$ that is given as an input to Algorithm 1 as well as that for α the removed axiom and for most CQs q in the rewriting we have $\alpha \not\in \rho(q)$; hence, the algorithm also produces a large output. For instance, for ontology UX and query Q_5 the graph contains on average 6622 queries most of which also belong in the rewriting $\mathcal{Q}'_D$ computed by Algorithm 1. But subsequently, most of them are redundant and need to be removed. In contrast due to the optimisations implemented in DelOpt the algorithm is able to identify on the fly many redundant CQs and avoid traversing this large graph that is given as an input.

The goal of our second experiment was to assess and interpret the extent to which an axiom of an ontology affects the size and computation time of a computed rewriting for fixed queries. If by removing an axiom the size of the rewriting or the computation time is significantly smaller than the original one then we can conclude that its existence in the specific ontology is particularly 'problematic' for the rewriting systems and hence in practical settings one would

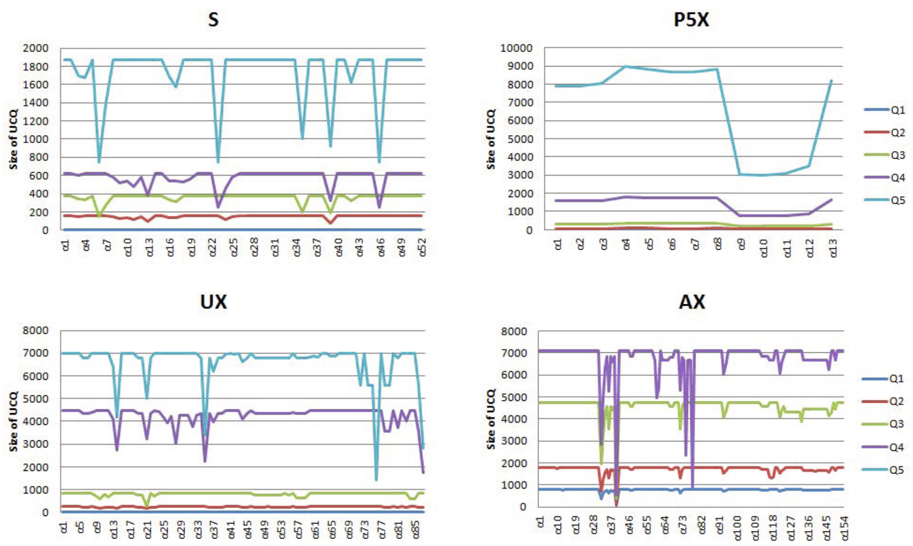


Fig. 1. Size of rewriting for ontologies $S, P5X, UX$, and AX for CQs Q_1–Q_5 when axiom α_i is removed

probably need to revise it. Note that even if the rewriting can be computed fast the database system would likely not be able to answer it as large rewritings imply large complex SQL queries.

For this experiment we proceeded as follows: for each ontology and query we removed iteratively each axiom and measured the size of the resulting UCQ using our system Del. We did not eliminate subsumed clauses in order to have a better picture of the number of clauses that could be produced during a rewriting process. Then, we drew plots of rewriting size vs. removed axiom in order to see for which and how many axioms there is a significant reduction in the size of the rewriting. Figure 1 presents the plots for ontologies $S, P5X, UX$, and AX which according to Table 1 are the ones that are the most difficult for the systems.

A first interesting observation is that indeed there are axioms that affect the size of the rewriting significantly. For example, in AX the removal of one of the axioms $\alpha_{32}, \alpha_{40}, \alpha_{72}$ and α_{78} causes the size of the rewriting to drop to less than half. Especially, if we remove axiom α_{40} then the rewriting of Q_4 drops from 7000 CQs to just 528 CQs. Similar observations can be made for the other ontologies as well. A second interesting observation is that for all ontologies the set of axioms that demonstrates the largest reduction is the same regardless of which query we examine. This shows that most queries are interrelated (i.e., they mention the same predicates) and that there are usually specific points in the ontology that are hard for a given query. A third interesting observation is that for all ontologies and queries the number of axioms that affect the size of the rewriting is usually small. More precisely, for each ontology there are usually less than five axioms which if removed the size of the rewriting drops significantly.

Subsequently, we wanted to investigate the reason why these axioms affect the size of the rewriting. For AX one such axiom is $\alpha_{40} = \mathsf{AssistiveDevice}(x) \to \mathsf{Device}(x)$ ($\mathsf{AssistiveDevice} \sqsubseteq \mathsf{Device}$ in DL notation). By inspecting manually the ontology we concluded that concept Device appears very high in the hierarchy of this ontology,[6] it has many descendant concepts (that is, there are many unary predicates A in the ontology such that $AX \models A(x) \to \mathsf{Devise}(x)$), and finally it appears in all test queries. In contrast, although axiom $\alpha_{47} = \mathsf{VisualDisability}(x) \to \mathsf{Disability}(x)$ also refers to Disability that is also high in the hierarchy it does not affect the size of the rewriting as the hierarchy below it is rather 'shallow'. After examining all ontologies we concluded that this is a main reason for hardness. However, note that in many cases this is not immediately obvious by inspecting the ontology. For example, in case an axiom involves binary predicates the interpretation is more difficult since these can participate in axioms with unary predicates (e.g., in axioms of the form $C(x) \to R(x, f(x))$ or $R(x, y) \to C(y)$) which are not reflected in the hierarchy.

Finally, we also wanted to check whether a large reduction in the size of a rewriting also implies a large reduction in computation time for each of the tested systems. Indeed the computation time decreases in a similar way as the size of the rewriting. An interesting case is the system ProgRes and query Q_5 of ontology AX. Although the system is not able to terminate when processing the original input ontology even after several hours, by removing axiom α_{40} (i.e., one of the problematic ones) it can compute a rewriting for $AX \setminus \{\alpha_{40}\}$ in 16 seconds. Hence, we see that this analysis can indeed be very helpful when designing an ontology for practical applications.

5 Conclusions

In the current paper we present and study a novel problem in the area of query rewriting. More precisely, we have studied query rewriting of fixed queries over contracted ontologies—that is, over ontologies for which one or more axioms have been removed. We presented a practical algorithm which, given a rewriting $\mathcal{Q}'$ for the input query $\mathcal{Q}$ and ontology $\mathcal{O}$ (that satisfies certain conditions) and a set of axioms A to be removed from $\mathcal{O}$ it computes a rewriting $\mathcal{Q}''$ for $\mathcal{Q}, \mathcal{O} \setminus \mathsf{A}$ directly from $\mathcal{Q}$. We have implemented and evaluated the algorithm over state-of-the-art rewriting systems and have obtained encouraging results. Moreover, we have used the algorithm to analyse the role that each axiom of an ontology plays to the 'complexity' of the final rewriting. More precisely, we have measured how much the size of a rewriting is reduced if one removes an axiom of the ontology making interesting observations.

Regarding future work we plan to investigate the same problem under ontology extensions—that is, when new axioms are added to the ontology, further optimise and evaluate our algorithm and also delve more into the role that each axiom plays in the rewriting.

[6] That is, there are few (if any) unary predicates A in the ontology such that $AX \models \mathsf{Devise}(x) \to A(x)$.

Acknowledgments. This work was partially supported by the European commission project Linked Heritage (contract number: ICT-PSP-270905). Giorgos Stoilos is supported by a Marie Curie Career Reintegration Grant within European Union's Seventh Framework Programme (FP7/2007-2013) under REA grant agreement 303914.

References

1. Abiteboul, S., Hull, R., Vianu, V.: Foundations of Databases. Addison-Wesley (1995)
2. Baget, J.F., Leclère, M., Mugnier, M.L., Salvat, E.: On rules with existential variables: Walking the decidability line. Artificial Intelligence 175(9-10), 1620–1654 (2011)
3. Booth, R., Meyer, T., Varzinczak, I.J.: First steps in EL contraction. In: Proceedings of the Workshop on Automated Reasoning about Context and Ontology Evolution, ARCOE 2009 (2009)
4. Calì, A., Gottlob, G., Lukasiewicz, T., Marnette, B., Pieris, A.: Datalog+/-: A family of logical knowledge representation and query languages for new applications. In: Proceedings of the 25th Annual IEEE Symposium on Logic in Computer Science, LICS 2010, pp. 228–242 (2010)
5. Calvanese, D., De Giacomo, G., Lembo, D., Lenzerini, M., Rosati, R.: Tractable reasoning and efficient query answering in description logics: The DL-Lite family. Journal of Automated Reasoning 39(3), 385–429 (2007)
6. Chortaras, A., Trivela, D., Stamou, G.: Optimized query rewriting in OWL 2 QL. In: Proceedings of the 23rd International Conference on Automated Deduction (CADE 23), Polland, pp. 192–206 (2011)
7. Cuenca Grau, B., Kharlamov, E., Zheleznyakov, D.: Ontology contraction: Beyond propositional paradise. In: Alberto Mendelzon International Workshop on Foundations of Data Management (AMW), Ouro Preto, Brazil (June 2012)
8. Cuenca Grau, B., Motik, B., Stoilos, G., Horrocks, I.: Completeness guarantees for incomplete ontology reasoners: Theory and practice. Journal of Artificial Intelligence Research 43, 419–476 (2012)
9. 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)
10. Gonçalves, R.S., Parsia, B., Sattler, U.: Categorising logical differences between OWL ontologies. In: Proceedings of the 20th ACM Conference on Information and Knowledge Management, CIKM 2011, pp. 1541–1546 (2011)
11. Gottlob, G., Orsi, G., Pieris, A.: Ontological queries: Rewriting and optimization. In: Proceedings of the 27th International Conference on Data Engineering, ICDE 2011 (2011)
12. Imprialou, M., Stoilos, G., Grau, B.C.: Benchmarking ontology-based query rewriting systems. In: Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence, AAAI 2012. AAAI Press (July 2012)
13. Lenzerini, M.: Data integration: a theoretical perspective. In: Proceedings of the Twenty-First ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, PODS 2002, pp. 233–246. ACM, New York (2002)
14. Lutz, C.: The Complexity of Conjunctive Query Answering in Expressive Description Logics. In: Armando, A., Baumgartner, P., Dowek, G. (eds.) IJCAR 2008. LNCS (LNAI), vol. 5195, pp. 179–193. Springer, Heidelberg (2008)

15. Orsi, G., Pieris, A.: Optimizing query answering under ontological constraints. Proceedings of the VLDB Endowment 4(11), 1004–1015 (2011)
16. Pérez-Urbina, H., Motik, B., Horrocks, I.: Tractable query answering and rewriting under description logic constraints. Journal of Applied Logic 8(2), 186–209 (2010)
17. Poggi, A., Calvanese, D., De Giacomo, G., Lembo, D., Lenzerini, M., Rosati, R.: Linking data to ontologies. JoDS X, pp. 133–173 (2008)
18. Ribeiro, M., Wassermann, R., Antoniou, G., Flouris, G., Pan, J.: Belief contraction in web-ontology languages. In: Proceedings of the 3rd International Workshop on Ontology Dynamics, IWOD 2009 (2009)
19. Rosati, R., Almatelli, A.: Improving query answering over DL-Lite ontologies. In: Proceedings of the Twelfth International Conference on Principles of Knowledge Representation and Reasoning, KR 2010 (2010)
20. Stamou, G., Trivela, D., Chortaras, A.: Progressive semantic query answering. In: Scalable Semantic Web Knowledge Base Systems Workshop. SSWS 2010, Shanghai, China, November 7-8 (2010)
21. Stoilos, G., Cuenca Grau, B., Motik, B., Horrocks, I.: Repairing Ontologies for Incomplete Reasoners. In: Aroyo, L., Welty, C., Alani, H., Taylor, J., Bernstein, A., Kagal, L., Noy, N., Blomqvist, E. (eds.) ISWC 2011, Part I. LNCS, vol. 7031, pp. 681–696. Springer, Heidelberg (2011)
22. Venetis, T., Stoilos, G., Stamou, G.: Incremental query rewriting for OWL 2 QL. In: Proceedings of the 25th International Workshop on Description Logics, DL 2012, Rome, Italy (2012)
23. Widom, J.: Research problems in data warehousing. In: Proceedings of International Conference on Information and Knowledge Management, pp. 25–30 (1995)

Improving the Recall of Live Linked Data Querying through Reasoning

Jürgen Umbrich[1], Aidan Hogan[1], Axel Polleres[2], and Stefan Decker[1]

[1] Digital Enterprise Research Institute, National University of Ireland, Galway
[2] Siemens AG Österreich, Siemensstrasse 90, 1210 Vienna, Austria
{Jürgen.Umbrich,Aidan.Hogan,Stefan.Decker}@deri.org,
axel.polleres@siemens.com

Abstract. Linked Data principles allow for processing SPARQL queries on-the-fly by dereferencing URIs. Link-traversal query approaches for Linked Data have the benefit of up-to-date results and decentralised execution, but operate only on explicit data from dereferenced documents, affecting recall. In this paper, we show how inferable knowledge—specifically that found through `owl:sameAs` and RDFS reasoning—can improve recall in this setting. We first analyse a corpus featuring 7 million Linked Data sources and 2.1 billion quadruples: we (1) measure expected recall by only considering dereferenceable information, (2) measure the improvement in recall given by considering `rdfs:seeAlso` links as previous proposals did. We further propose and measure the impact of additionally considering (3) `owl:sameAs` links, and (4) applying lightweight RDFS reasoning for finding more results, relying on static schema information. We evaluate different configurations for live queries covering different shapes and domains, generated from random walks over our corpus.

1 Introduction

Recently, a rich lode of RDF data has been published on the Web as *Linked Data* by governments, academia, industry, communities and individuals alike [15]. Publishing Linked Data is governed by four principles, here summarising [2]: (P1) *use URIs to name things*, such that (P2) those *URIs can be dereferenced via HTTP*, such that (P3) *dereferencing yields useful RDF content* about that which is named, such that (P4) the returned *content includes links* (mentions external URIs) for further discovery. Given that the URIs used to name resources map (through HTTP) to the physical location of structured information about them, information published as Linked Data can be viewed as forming a scale-free, decentralised database, consisting of millions of structured Web documents [13]. Further still, thanks to the provision of typed "RDF links" between such documents [15, § 4.5], agents can traverse and navigate the resulting *Web of Data* in a manner analogous to browsing through the *Web of Documents.*

Tangentially, SPARQL [22]—the W3C standardised RDF query language—provides the declarative means to formulate structured queries against these data. Traditional approaches for posing queries against Linked Data retrieve and cache data in local indexes; however, the dynamicity and scope of Web data implies that results are often stale or missing. Hartig et al. [12] propose using dereferenceable URIs in a SPARQL query—and recursively, in the intermediate results—to automatically determine a focussed set of sources that, by Linked Data principles, are likely to be *query relevant*, retrieving them live from the Web at query-time and processing them for answers. By operating over compliant Linked Data, their approach bypasses the need for source graphs to be explicitly named or pre-indexed, allowing for *ad hoc*, live discovery. Later work [10] calls this approach *Link Traversal Based Query Execution* (LTBQE). A core challenge for LTBQE is to identify, retrieve and process a minimal number of sources that yield maximal results, keeping response times low while maximising answers.

In this paper, we first reintroduce the LTBQE approach, and highlight the core assumptions under which it operates well. We then show to what extent these assumptions hold in practice—*i.e.*, measuring how much data is attainable from dereferencing—through empirical analysis of a corpus of $\sim$7.4 m RDF Web documents. We further propose extensions of LTBQE to (i) minimise the number of sources accessed by being more selective about links followed; (ii) increase recall by considering some lightweight semantics of Linked Data that allow for (ii.a) finding additional query-relevant sources and data through consideration of owl:sameAs links, and (ii.b) finding additional query-relevant data through rule-based materialisation with respect to a lightweight subset of RDFS (viz. ρDF [20]). We measure the expected effect on recall for each of these extensions through similar analysis of our data. Finally, we generate a diverse set of benchmark queries from our corpus and run them live over remote sources, comparing different LTBQE configurations and extensions.

2 Background and Related Work

Traditional approaches to query Linked Data locally replicate the content of remote Linked Data sources and execute SPARQL queries over the local copy. In previous years, we supported such a service powered by YARS2 [9] allowing for querying over millions of RDF Web documents (and their entailments), but discontinued the endpoint due to prohibitive running costs. Current centralised SPARQL endpoints harvesting Linked Data include "FactForge" [3][1] (powered by BigOWLIM [4]), OpenLink's LOD cache[2] and Sindice's "Semantic Web Index" [21][3] (both powered by Virtuoso [7]). The primary targets for these engines are (i) to have a broad coverage of the Web of Data, (ii) to keep results up to date, (iii) to have fast response times. These objectives are (partially) met using distribution techniques, replication, optimised indexes, compression techniques, data synchronisation, and so forth [4,7,9,21]. However, maintaining a broad, up-to-date and optimised local index is a Sisyphean task.

Federated SPARQL engines execute queries over a group of independent endpoints, dividing and routing sub-queries to individual endpoints [1,23,24]. Given

[1] http://factforge.net/sparql
[2] http://lod.openlinksw.com/sparql
[3] http://sparql.sindice.com/

the recent spread of SPARQL endpoints on the Web of Data, federation is a timely topic and enjoys increasing attention. However, our techniques operate over raw source documents, not SPARQL endpoints.

Recently, various authors have proposed methods for performing live querying, accessing remote data at runtime. Ladwig and Tran [17] categorise these approaches as follows: (i) top-down query evaluation, (ii) bottom-up query evaluation, and (iii) mixed strategy query evaluation. Top-down evaluation determines remote, query-relevant sources using a *source-selection index*: a local repository summarising information about sources that can vary from inverted-index structures [19,21], to query-routing indexes [26], schema-level indexes [25], or lightweight hash-based structures [27]. The bottom-up query evaluation strategy involves discovering relevant sources on-the-fly during the evaluation of queries by selectively and recursively following links starting from a "seed set" of URIs taken from the query [12]. The third strategy uses (in a top-down fashion) some knowledge about sources to generate the seed list, then discovering additional relevant sources using a bottom-up approach [17]. All such approaches rely on time-consuming remote lookups, but conversely offer fresh results.

An appealing use-case for live querying is to combine both centralised and live querying results: to complement the fast but potentially stale results of a centralised engine with slower but fresher live results. A number of works have tackled this combination on a variety of levels (see, *e.g.*, [13,18,28]).

3 Preliminaries

We now present some preliminaries. Before we continue, we introduce a motivating example that will be used to explain the concepts involved: Figure 1 illustrates an RDF (sub)graph taken from four (real) interlinked sources on the Web of Data. The graph contains structured information about one publication (`dblpPub:HartigBF09`), "four" people (`oh:olaf`, `cb:chris`, `dblpAuth:Olaf_Hartig`, `dblpAuth:Christain_Bizer`) and four dereferenceable documents.

Presented below Figure 1 are three example queries. For Query 1, the LTBQE approach dereferences `oh:olaf`, finds `cb:chris` as a binding, and dereferences it to look for depictions; however the URI does not dereference, and so the LTBQE resorts to following the `rdfs:seeAlso` link, as supported in the original proposal [12]. For Query 2, `oh:olaf` is again dereferenced, the FOAF file of `cb:chris` is found through a see-also link; however, to traverse further and find answers, the query-processor needs to traverse the `owl:sameAs` link and also support the semantics thereof. We propose and evaluate this extension later. Finally, the FOAF vocabulary defines `foaf:name` to be a sub-property of `rdfs:label`, where RDFS reasoning is required to answer Query 3; we also propose this extension.

We now formally define these concepts, covering preliminaries relating to RDF and Linked Data (§ 3.1), SPARQL (§ 3.2) and RDFS & OWL (§ 3.3).

3.1 RDF and Linked Data

We first provide some notation for dealing with RDF and Linked Data principles.

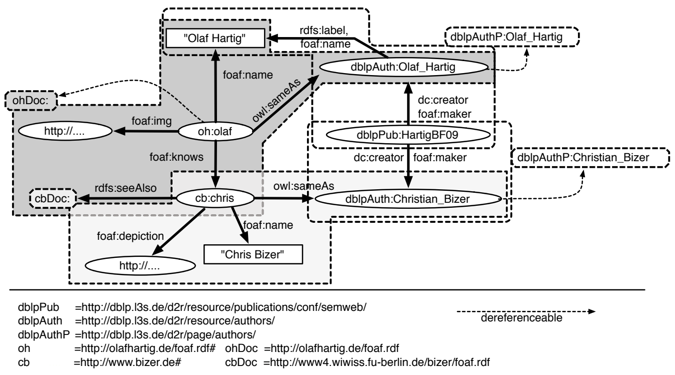


Fig. 1. Snapshot of a sub-graph from the Linked Open Data Web

```
SELECT ?f ?img WHERE {
   oh:olaf foaf:knows ?f .
   ?f foaf:depiction ?img }
```

Query 1. Friends' images

```
SELECT ?f WHERE {
   oh:olaf foaf:knows ?f .
   ?pub dc:creator ?f, oh:olaf }
```

Query 2. Coauthors

```
SELECT ?f ?l WHERE {
   oh:olaf foaf:knows ?f .
   ?f rdfs:label ?l }
```

Query 3. Friends' labels

Definition 1 (RDF Term, Triple and Graph).
The set of RDF terms *consists of the set of URIs* $\mathbf{U}$*, the set of blank-nodes* $\mathbf{B}$ *and the set of literals* $\mathbf{L}$*. An* RDF triple $t := (s, p, o)$ *is an element of the set* $\mathbf{G} := \mathbf{UB} \times \mathbf{U} \times \mathbf{UBL}$ *(where, e.g.,* $\mathbf{UB}$ *is a shortcut for set-union). A set of RDF triples* $G \subset \mathbf{G}$ *is called an* RDF graph*. We use the functions* $\mathsf{subj}(G)$, $\mathsf{pred}(G)$, $\mathsf{obj}(G)$, $\mathsf{terms}(G)$*, to denote the set of all terms projected from the resp. triple position (*terms *gives all positions).*

Definition 2 (Data Source and Linked Dataset).
We define the http-download *function* $\mathsf{get} : \mathbf{U} \to 2^{\mathbf{G}}$ *as the mapping from URIs to RDF graphs provided by means of HTTP lookups that directly return status code* 200 OK *and data in a suitable RDF format. We define the set of (RDF)* data sources $\mathbf{S} \subset \mathbf{U}$ *as the set of URIs* $\mathbf{S} := \{s \in \mathbf{U} : \mathsf{get}(s) \neq \emptyset\}$*. We define a Linked Dataset as* $\Gamma \subset \mathsf{get}$*; i.e., a finite set of pairs* $(u, \mathsf{get}(u))$*, and* $\mathsf{merge}(\Gamma) := \biguplus_{(u,G)\in\Gamma} G$ *as the RDF merge of graphs in* Γ*, which preserves the uniqueness of blank-node labels across graphs [14].*

Definition 3 (Dereferencing RDF). *A URI may issue a HTTP redirect to another URI with a* 30x *response code; we denote this function as* $\mathsf{redir} : \mathbf{U} \to \mathbf{U}$*, which strips the fragment identifier of a URI (if present) and which would (thereafter) map a URI to itself in the case of failure (e.g., where no redirect exists). We denote the fixpoint of* redir *as* redirs*, denoting traversal of a number of redirects (a limit may be imposed to avoid cycles). We denote* dereferencing *by the composition* $\mathsf{deref} := \mathsf{get} \circ \mathsf{redirs}$*, which maps a URI to an RDF graph retrieved with status code* 200 OK after *following redirects, or which maps a URI to the* *empty set in the case of failure. We denote the set of* dereferenceable URIs *as* $\mathbf{D} := \{d \in \mathbf{U} : \mathsf{deref}(d) \neq \emptyset\}$*; note that* $\mathbf{S} \subset \mathbf{D}$ *and we place no expectations on what* $\mathsf{deref}(d)$ *returns (as long as it returns* some *valid RDF).*

Taking Figure 1, *e.g.*, $\mathsf{redir}(\texttt{oh:olaf}) = \texttt{ohDoc:}$, $\mathsf{deref}(\texttt{oh:olaf}) = \mathsf{deref}(\texttt{ohDoc:}) = \{(\texttt{oh:olaf}, \texttt{foaf:name}, \texttt{"Olaf Hartig"}) \ldots\}$ and $\mathsf{deref}(\texttt{cb:chris}) = \emptyset$.

3.2 SPARQL

We now introduce some concepts relating to SPARQL [22]. Note that herein, we focus on evaluating simple, conjunctive, *basic graph patterns* (BGPs) .

Definition 4 (Variables, Triple Patterns and Queries (BGPs)).
Let $\mathbf{V}$ *be the set of variables ranging over* $\mathbf{UBL}$*. A* triple pattern $tp := (s, p, o)$ *is an element of the set* $\mathbf{Q} := \mathbf{VUL} \times \mathbf{VU} \times \mathbf{VUL}$*. For simplicity, we do not consider blank-nodes in triple patterns (they could be replaced with variables). A finite (herein, non-empty) set of triple patterns* $Q \subset \mathbf{Q}$ *is called a* Basic Graph Pattern*, or herein, simply a* query*. We use* $\mathsf{vars}(Q) \subset \mathbf{V}$ *to denote the set of variables in* Q*. Finally, we may overload graph notation where, e.g.,* $\mathsf{terms}(Q)$ *returns all elements of* $\mathbf{VUL}$ *in* Q*.*

Definition 5 (SPARQL solutions).
Call the partial function $\mu : \mathrm{dom}(\mu) \cup \mathbf{UL} \to \mathbf{UBL}$ *a* solution mapping*, which binds variables in* $\mathrm{dom}(\mu) \subset \mathbf{V}$ *to* $\mathbf{UBL}$ *and which is the identify function for* $\mathbf{UL}$*. Overloading notation, let* $\mu : \mathbf{Q} \to \mathbf{G}$ *and* $\mu : 2^{\mathbf{Q}} \to 2^{\mathbf{G}}$ *also resp. denote a solution mapping from triple patterns to RDF triples, and basic graph patterns to RDF graphs such that* $\mu(tp) := (\mu(s), \mu(p), \mu(o))$ *and* $\mu(Q) := \{\mu(tp) \mid tp \in Q\}$*. Now, we define the set of solutions for a query* Q *over a Linked Dataset* Γ *as* $\Omega(\Gamma, Q) := \{\mu \mid \mu(Q) \subseteq \mathsf{merge}(\Gamma) \wedge \mathrm{dom}(\mu) = \mathsf{vars}(Q)\}$*. Note that herein, and unlike SPARQL, solutions are given as sets (not multi-sets), implying a default* DISTINCT *semantics for queries.*

Taking an example, if we let Γ be Figure 1 and Q be Query 3, then $\Omega(\Gamma, Q) = \{(\texttt{?f}, \texttt{cb:chris}), (\texttt{?l}, \texttt{"Chris Bizer"})\}$.

3.3 RDFS and OWL

We define some preliminaries relating to RDFS and OWL. In particular, we support a miniature subset of OWL 2 RL/RDF rules for supporting owl:sameAs entailments, given in Table 1. Our RDFS rules are the subset of ρDF rules proposed by Muñoz et al. [20], which deal with instance data entailments.[4] Our subset of OWL rules are specifically chosen to support the semantics of equality (particularly replacement) for owl:sameAs. Note that these rules support the RDFS/OWL features originally recommended for use by Bizer et al. when publishing Linked Data [5, §4.2, §6]. The rules we consider are given in Table 1.

[4] We drop *implicit typing* [20] rules but allow generalised RDF in interim inferences.

Table 1. ρDF and owl:sameAs rules with OWL 2 RL/RDF naming

ID	Body	Head
PRP-SPO1	?p$_1$ rdfs:subPropertyOf ?p$_2$. ?s ?p$_1$?o .	?s ?p$_2$?o .
PRP-DOM	?p rdfs:domain ?c . ?s ?p ?o .	?p a ?c .
PRP-RNG	?p rdfs:range ?c . ?s ?p ?o .	?o a ?c .
CAX-SCO	?c$_1$ rdfs:subClassOf ?c$_2$. ?s a ?c$_1$.	?s a ?c$_2$.
EQ-SYM	?x owl:sameAs ?y .	?y owl:sameAs ?x .
EQ-TRANS	?x owl:sameAs ?y . ?y owl:sameAs ?z .	?x owl:sameAs ?z .
EQ-REP-S	?s owl:sameAs ?s$'$. ?s ?p ?o .	?s$'$?p ?o .
EQ-REP-P	?p owl:sameAs ?p$'$. ?s ?p ?o .	?s ?p$'$?o .
EQ-REP-O	?o owl:sameAs ?o$'$. ?s ?p ?o .	?s ?p ?o$'$.

Definition 6 (Entailment Rules and Closure). *Given a ruleset R and a Linked Dataset Γ, we denote by ${}^{R}\Gamma := \Gamma \cup (v, G)$ the closure of Γ wrt. R, where (abusing notation) G contains the materialised inferences from recursively applying the rules in R over* $\mathsf{merge}(\Gamma) \uplus G$ *up to a fixpoint, and where v is a built-in URI for naming the materialised graph.*

4 Link Traversal Based Query Execution

We first introduce the Link Traversal Based Query Execution (LTBQE) approach introduced by Hartig et al. [12] (§ 4.1), and then look at extensions of the approach (§ 4.2). Our formalisms are tailored for the purpose of this work; a comprehensive study of semantics and computability is presented in [11].

4.1 Baseline LTBQE

Definition 7 (LTBQE Query Relevant Sources and Answers). *Define* $\mathsf{derefs} : 2^{\mathbf{U}} \to \mathbf{U} \times 2^{\mathbf{G}}$*; $U \mapsto \{(\mathsf{redirs}(u), \mathsf{deref}(u)) \mid u \in U)\}$ as the mapping from a set of URIs to the Linked Dataset it represents by dereferencing all URIs. Given a BGP query Q as before, let $U_Q := \mathsf{terms}(Q) \cap \mathbf{U}$ denote the set of URIs appearing in Q. Let $\Gamma_0^Q := \mathsf{derefs}(U_Q)$ represent the dataset retrieved by dereferencing all query URIs.*[5] *Next let $\mathsf{uris}(\mu) := \{u \in \mathbf{U} \mid \exists v \text{ s.t. } (v, u) \in \mu\}$ denote the set of URIs in a solution mapping μ, and let $U_i := \{u \in \mathsf{uris}(\mu) \mid \exists \mu, \exists tp \in Q \text{ s.t. } \mu(\{tp\}) \subseteq \mathsf{merge}(\Gamma_{i-1}^Q)\}$ for $i \in \mathbb{N}$ be the set of URIs that appear as a solution mapping for a triple pattern in Q for the dataset Γ_{i-1}^Q, and let $\Gamma_i^Q := \mathsf{derefs}(U_i) \cup \Gamma_0^Q$.*[6] *The set of* LTBQE query relevant sources *for Q is given as the least n such that $\Gamma_n^Q = \Gamma_{n+1}^Q$, denoted simply Γ^Q. The set of* LTBQE query answers *for Q is given as $\Omega(\Gamma^Q, Q)$, or simply Ω^Q.*

With regards to completeness, let get denote the dataset (theoretically) represented by the entire Web of Data (note: $\mathsf{get} \subset \mathbf{U} \times 2^G$). One may then ask when Ω^Q is complete with respect to get. A trivial sufficient condition for completeness

[5] One could consider Γ_0^Q as also containing "seed" data [12].

[6] Or, equivalently (for static data) $\Gamma_i^Q := \Gamma_{i-1}^Q \cup \mathsf{derefs}(U_i \setminus U_{i-1})$.

```
SELECT * WHERE { cb:chris ?p ?o . }
```

Query 4. Not dereferenceable

```
SELECT ?olaf ?name WHERE {
   oh:olaf foaf:name ?name . ?olaf foaf:name ?name . }
```

Query 5. Connected by literal

```
SELECT ?s WHERE { ?s owl:sameAs dblpAuth:Olaf_Hartig . }
```

Query 6. Not in dereferenceable document

```
SELECT ?paper WHERE {
  cb:chris owl:sameAs ?dblpC .
  oh:olaf owl:sameAs ?dblpO .
  ?dblpC foaf:maker ?paper .
  ?dblpO foaf:maker ?paper .
}
```

Query 7. Query answer only reachable from the seed URI oh:olaf

is given by $\Gamma^Q = \mathsf{get}$; such a case is, however, infeasible. Otherwise the completeness condition is rather simple and entirely unverifiable: Γ^Q must contain all of the data relevant on the Web to answer the query. Of course, verifying that the condition does not hold is significantly easier in many cases. The implications are that: first, given a query with no dereferenceable URIs, LTBQE cannot return results (as per Query 4 where cb:chris does not dereference). Second, given a query with multiple URIs, different reachability conditions can occur from different starting points (as per Query 7 where the answer is only reachable starting from ol:olaf); thus, all query URIs must be initially retrieved [12]. Third, answers "connected" by literals or involving blank-nodes in unreachable documents will often affect completeness (as per Query 5 where the answer is connected by the literal "Olaf Hartig"', here not yet considering owl:sameAs). Fourth, in the general case, reachability is heavily dependent on the amount of data returned by the $\mathsf{deref}(u)$ function, which would ideally return all triples mentioning u on the Web of Data (*e.g.*, in Query 6, the owl:sameAs inlinks for dblpAuth:Olaf_Hartig are not in its dereferenced document and will not be found). The fourth assumption is clearly idealised; hence, in Section 5 we will empirically analyse how much the assumption holds in practice, giving insights into the recall of LTBQE. First, however, we propose our reasoning extensions.

4.2 Extending LTBQE

1. Following rdfs:seeAlso*:* The first extension to LTBQE is proposed by Hartig et al., and uses rdfs:seeAlso links to extend the set of query sources. Adapting Definition 7, let $\bar{\Gamma}^Q_0 := \Gamma^Q_0$ and let:

$$\bar{U}_i := U_i \cup \{u \in \mathbf{U} \mid \exists u' \in U_i \text{ s.t. } (u', \texttt{rdfs:seeAlso}, u) \in \mathsf{merge}(\bar{\Gamma}^Q_{i-1})\}\,,$$

let $\bar{\Gamma}^Q_i := \mathsf{derefs}(\bar{U}_i) \cup \Gamma^Q_0$, and finally let $\bar{\Gamma}^Q$ be the fixpoint as before and let $\bar{\Omega}^Q$ be the respective solutions. This extends LTBQE to find more sources through rdfs:seeAlso links. An example for this has been presented in Query 1.

2. Following and Reasoning Over owl:sameAs*:* We propose an extension of LTBQE to consider owl:sameAs inferences. Let R denote the set of rules of the form EQ-* in Table 1. Let now ${}^{e}\Gamma_0^Q := {}^{R}\Gamma_0^Q$ (recalling ${}^{R}\Gamma$ from Def. 6 and Γ_0^Q from Def. 7), and let:

$$U_i' := \{u \in \mathsf{uris}(\mu) \mid \exists \mu, \exists tp \in Q \text{ s.t. } \mu(\{tp\}) \subseteq \mathsf{merge}({}^{e}\Gamma_{i-1}^Q)\}\,,$$
$${}^{e}U_i := \{u \in \mathbf{U} \mid \exists u' \in U_i' \text{ s.t. } (u', \texttt{owl:sameAs}, u) \in \mathsf{merge}({}^{e}\Gamma_{i-1}^Q)\}\,,$$

where ${}^{e}\Gamma_i^Q := {}^{R}\mathsf{derefs}({}^{e}U_i) \cup {}^{e}\Gamma_0^Q$, and finally let ${}^{e}\Gamma^Q$ be the fixpoint as before and let ${}^{e}\Omega^Q$ be the respective solutions. Here, owl:sameAs links are used to expand the set of query relevant sources, and owl:sameAs rules are used to materialise inferable knowledge given by the OWL semantics, potentially generating additional answers. An example for this has been presented in Query 2.

3. Reasoning for ρDF: We propose a final novel extension of LTBQE to consider a subset of RDFS reasoning as per the PRP-* and CAX-SCO rules in Table 1, which we again denote here by R. We currently consider a static set of schema data representing vocabularies on the Web, which we denote by Γ^{voc}. This serves as input into the LTBQE algorithm. In future work, we plan to investigate dereferencing schema knowledge live from the Web of Data.

Now, adapting Definition 7, let ${}_{\rho}\Gamma_0^Q := \Gamma^{voc} \cup {}^{R}\Gamma_0^Q$ and let:

$${}_{\rho}U_i := \{u \in \mathsf{uris}(\mu) \mid \exists \mu, \exists tp \in Q \text{ s.t. } \mu(\{tp\}) \subseteq \mathsf{merge}({}_{\rho}\Gamma_{i-1}^Q)\}\,,$$

where ${}_{\rho}\Gamma_i^Q := {}^{R}\mathsf{derefs}({}_{\rho}U_i) \cup {}_{\rho}\Gamma_0^Q$, and finally let ${}_{\rho}\Gamma^Q$ be the fixpoint as before and let ${}_{\rho}\Omega^Q$ be the respective solutions. Here, RDFS rules and background schema knowledge (Γ^{voc}) are used to materialise inferable knowledge, potentially generating additional answers (and thus possibly finding new query relevant sources). An example for this has been presented in Query 3.

Combined. Of course, the above methods can be combined in a natural fashion, where, *e.g.*, for combining all extensions, the query relevant sources are denoted ${}^{e}_{\rho}\bar{\Gamma}^Q$ and the answers by ${}^{e}_{\rho}\bar{\Omega}^Q$.

5 Empirical Study

In Section 4.1, we mentioned that the recall of the LTBQE approach is—in the general case—dependent on the dereferenceability of data. Along those lines, we now present the results of our empirical study of a Linked Data corpus. We survey the ratio of all triples mentioning a URI in our corpus against those returned in the dereferenceable document of that URI; we do so for different triple positions. We also look at the comparative recall of data considering (1) explicit, dereferenceable information; (2) including rdfs:seeAlso links [12]; (3) including owl:sameAs links and inferable knowledge; (4) including RDFS reasoning.

Empirical Corpus. We use the Billion Triple Challenge 2011 dataset for our survey, which was crawled in mid-May 2011 from 7.4 million RDF/XML documents spanning 791 pay-level domains (data providers). The resulting corpus contains 2.15 g quadruples (1.97 g unique triples) mentioning 538 m RDF terms, of which 52 m (10%) are Literals, 382 m (71%) are blank nodes, and 103 m (19%) are URIs. We denote the corpus as $\Gamma_{\sim}$. It's important to note that this corpus is only a *sample* of the Web of Data; in particular, we only use the information about HTTP lookups provided by the dataset. We found that a total of 25.4 m lookups were performed (excluding `robots.txt`). As such, we only have knowledge of redir and deref functions for 18.65 m URIs; all of these URIs are HTTP and do not have non-RDF file-extensions. We denote these URIs by $U_{\sim}$. Of the 18.65 m, 8.37 m (44.8%) dereferenced to RDF; we denote these by $D_{\sim}$. Further note that, wrt. the Web of Data, our sample recall measures specify an upper bound.

RDFS Schema. From our corpus, we extract a static set of schema data for the RDFS reasoning. As argued in [6], schema data on the Web is often noisy, where third-party publishers "redefine" popular terms outside of their namespace; for example, one document defines nine *properties* as the domain of `rdf:type`, which would have a drastic effect on our reasoning.[7] Thus, we perform *authoritative reasoning*, which conservatively discards certain third-party schema axioms (cf. [6]). Our schema data only considers triples of the following form:

$$(s, \texttt{rdfs:subPropertyOf}, o) \in \mathsf{deref}(s), (s, \texttt{rdfs:subClassOf}, o) \in \mathsf{deref}(s)$$
$$(s, \texttt{rdfs : domain}, o) \in \mathsf{deref}(s), (s, \texttt{rdfs : range}, o) \in \mathsf{deref}(s)$$

We extracted a total of 397 thousand such authoritative RDFS triples from 98 PLDs as follows: 334 thousand `rdfs:subClassOf` (82 PLDs); 11 thousand `rdfs:subPropertyOf` (67 PLDs); 26 thousand `rdfs:domain` (79 PLDs); and 26 thousand `rdfs:range` (77 PLDs).

5.1 Recall for Baseline

We first measure the average dereferenceability of information in our sample. For a dereferenceable uri d, we compute the *sample dereferencing recall* $\mathsf{sdr}(d)$ as the ratio of the number of unique triples mentioning d in $\mathsf{deref}(d)$ vs. unique triples mentioning d across the entire sample. We denote by $\mathsf{sdr}_{\sim}$ the average $\mathsf{sdr}(d)$ for all $d \in D_{\sim}$. We also analyse the $\mathsf{sdr}(d)$ restricting the specific triple positions where d appears. We ignore d in the average if it does not appear in the relevant triple position in the sample. Table 2 presents the results for different triple positions. Column *type*-**object** considers d only when appearing as object in a triple with the predicate `rdf:type` (a class position).

The analysis provides some interesting initial insights into the LTBQE approach. Given a HTTP URI without a common *non*-RDF extension, we have a

[7] viz. http://www.eiao.net/rdf/1.0

Table 2. Dereferenceability results for different triple positions

Measure	**any**	**subject**	**predicate**	**object**	*type*-**object**
$\lvert U_{\sim}\rvert$	18.65 m	9.55 m	47.67 k	9.73 m	213.38 k
$\lvert D_{\sim}\rvert$	8.37 m	8.09 m	745	4.5 m	21.1 k
$\lvert U_{\sim}\rvert/\lvert D_{\sim}\rvert$	0.44	0.85	0.01	0.46	0.09
average $\mathsf{sdr}_{\sim}$	**0.51**	**0.95**	**0.00007**	**0.438**	**0.002**
std. dev. $\pm$ $\mathsf{sdr}_{\sim}$	0.5	±0.195	±0.008	±0.458	±0.05

44.8% success ratio to receive RDF/XML content regardless of the triple position. If such a URI dereferences to RDF, we receive on average (at most) 51% of all triples in which it appears on the Web. Given a pattern with a URI in the subject position, the dereferenceable ratio increases to 95%; for objects, the ratio drops to 43.8%; LTBQE would perform poorly for triple patterns with (only) a URI in the predicate position (0.007%); *etc.* High standard deviations imply that the dereferenceability is often "all or nothing". In summary, LTBQE performs well when URIs appear as the subject of triple patterns, moderately when URIs appear in the object, but will perform poorly when URIs appear in the predicate or object of an `rdf:type` triple. In practice, documents dereferenced by property and class terms do not host a high percentage of their extension.

5.2 Recall for Extensions

We now measure the sdr increase given by extending LTBQE to also consider rdfs:seeAlso and owl:sameAs links, as well as inferable knowledge given by owl:sameAs and RDFS reasoning.

Benefit of Following rdfs:seeAlso *Links.* We measured the percentage of dereferenceable URIs in $D_{\sim}$ which have at least one rdfs:seeAlso link in their dereferenced document to be 2% (201 k URIs). Where such links exist, following them increases the amount of unique triples by a factor of 1.006× vs. triples in the dereferenced document alone. We conclude that, in the general case, considering rdfs:seeAlso triples will only marginally affect the recall increase of LTBQE.

Benefit of Following owl:sameAs *Links & inferable Knowledge.* We measured the percentage of dereferenceable URIs in $D_{\sim}$ which have at least one owl:sameAs link in their dereferenced document to be 16% for our sample. Where such links exist, following them and applying the EQ-* entailment rules over the resulting information increases the amount of unique triples by a factor of 2.5× vs. the unique (explicit) triples in the dereferenced document alone. We conclude that, in the general case, owl:sameAs links are only sometimes found for deferenceable URIs, but where available, following them and applying entailment generates significantly more data for generating answers (albeit potentially producing "duplicate" answers under different URI aliases).

Benefit of Including ρDF Inferable Knowledge. We measured the percentage of dereferenceable URIs in $D_\sim$ whose dereferenced documents given non-empty unique entailments through authoritative ρDF reasoning with respect to Γ^{voc} as 81%. Where such entailments are non-empty, they increase the amount of unique triples by a factor of 1.78× vs. the unique (explicit) triples in the dereferenced document. We conclude that such reasoning often increases the amount of data available for LTBQE query answering, and by a significant amount.

6 Evaluation

In order to test our methods for queries covering a diverse set of sources and query types, we evaluate our proposed LTBQE extensions for a set of pseudo-randomly generated queries extracted from our Linked Data corpus. These queries are then applied live to determine real-world behaviour. Our aim is to compare and contrast different setups and assess our proposed extensions in a realistic scenario. To the best of our knowledge, we are the first work to evaluate the original LTBQE proposal in a live and diverse environment.

Implementation: We have (re-)implemented Hartig et al.'s iterator-based algorithm for LTBQE (which was shown to be complete) [12]. We use ARQ to parse and process input SPARQL queries.[8] We further use the LDSpider crawling framework for performing live Linked Data lookups; LDSpider respects the `robots.txt` policy, blacklists typical non-RDF URI patterns (*e.g.,* `.jpeg`) and enforces a half-second delay between two consequential lookups for URIs hosted at the same domain.[9] We use the SAOR engine to support the aforementioned rule-based reasoning extensions [6]. Note that we use the same input RDFS data as used in the empirical study of the previous section.

Optimised LTBQE: Inspired by our empirical analysis, we also implement and evaluate a variation of the LTBQE approach which does not dereference URIs appearing only in the predicate position of a (possibly partially bound) triple-pattern. Further, we add another optimisation to avoid dereferencing URIs that are only bound by non-distinguished variables not appearing elsewhere in the query (*i.e.*, variables whose value is not used elsewhere). Since these optimisation reduce the number of query-relevant sources, they may in theory lead to less results, though in practice (and as per our empirical survey), we would expect a minimal change in recall over the baseline.

Evaluation Queries: We benchmark queries of elemental graph shapes, *viz.,* entity, star and path queries.

Entity Queries. (*entity*-[s|o|so]) ask for all available triples for an entity. We generate three types of entity queries, asking for triples where a URI appears as

[8] http://jena.sourceforge.net/ARQ/
[9] http://code.google.com/p/ldspider/

the subject (*entity-s*); as the object (*entity-o*); as the subject *and* object (*entity-so*). An example for *entity-so* would be `{<d> ?p1 ?o . ?s ?p2 <d> .}`. These type of queries are very common in Linked Data browsers or display interfaces.

Star Queries. (*star*-[*s3*|*o3*|*s1-o1*|*s2-o1*|*s1-o2*]) contain three acyclic triple patterns which share exactly one URI (called the centre node) where predicate terms are constant. We generate four variations of such queries, differing in the number of triple patterns where the centre node appears as the subject (s) or object (o). An example for *star-s2-o1* would be `{<d> foaf:knows ?o ; foaf:name ?o1 . ?o3 dc:creator <d> .}`

Path Queries. ([s|o]-*path*-[*2*|*3*]) consist of 2 or 3 triple patterns that form a path where precisely two triple pattern share the same variable. Exactly one triple pattern has a URI at either the subject or object position and all predicate terms are constant. We generate four path sub-types: path shaped queries of length 2 and 3 where either the subject or object of one triple pattern is a constant. An example for *s-path-2* is Query 1.

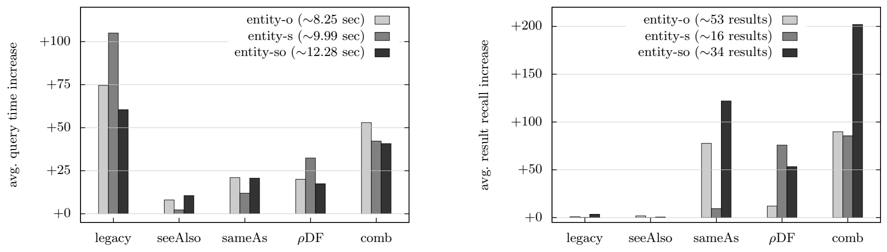
Query Generation: In total, we generate 100 `SELECT DISTINCT` queries for *each* of the above 11 query shapes using random walks in our corpus. To help ensure that queries return non-empty results (in case there are no HTTP connection errors or time outs) we consider dereferenceable information for the query generation which (1) picks randomly a pay-level-domain available in the dereferenceable URIs $D_\sim$, (2) selects randomly a URI from $D_\sim$ for that PLD and (3) generates appropriate triple patterns from the dereferenceable document of the selected URI. For path shaped queries, when performing steps (2) and (3), the URI for the next triple pattern is selected out of the URIs contained for the previous triple pattern, as per a random walk of dereferenceable URIs. Distinguished variables are picked by randomly choosing a single variable as distinguished and make further variables distinguished with a probability of 0.5.

Benchmark Stable Queries: We observed that the results for the same query varied over different runs and thus introduce the notion of "benchmark stable queries" which are queries for which the response codes for the baseline URIs are the same across all setups runs. The variation of results for different runs is caused by remote server or connection failures while dereferencing content, *e.g.*, we sometimes encountered `503 - Service unavailable` response codes possibly due to temporarily high loads on remote servers. Only considering stable queries improves the comparability of results across different setups. Table 3 shows that in average $\sim$94% ($\frac{1{,}029}{1{,}100}$) of the queries are stable from which $\sim$38% ($\frac{389}{1{,}029}$) returned empty results. We inspected the causes for the empty result set for all "stable" queries (right side of Table 3). The typewritten numbers correspond to HTTP server response codes.[10] The column "*mixed*" indicates that there are at least two URIs with different response codes and the column "*data*" indicates that the

[10] Aside from common response codes, a `498` code denotes robots.txt forbidden access, `499` denotes that a server returned a mime type different to `application/rdf+xml`, `602` denotes socket timeouts and `603` denotes unknown host exceptions.

Table 3. Statistics about stability of queries

Breakdown (1,100)			Causes for empty results (389)									
stable	*(empty)*	*unstable*	403	404	498	499	500	502	602	603	*mixed*	*data*
1,029	(389)	71	15	110	15	38	18	2	6	85	7	93



(a) Average *time* increase. (b) Average *recall* increase.

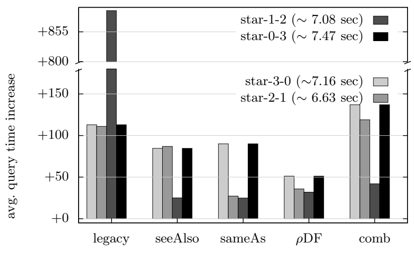
Fig. 2. Average *time* and *recall* increase relative to baseline for *entity queries*

missing results are not HTTP-related (*e.g.*, the data changed). The main reasons for empty "stable" queries are (i) either query relevant documents are not available any more (`404`), (ii) the IP address of a host could not be determined (indicated by `603`) or (iii) the underlying data changed (last column).

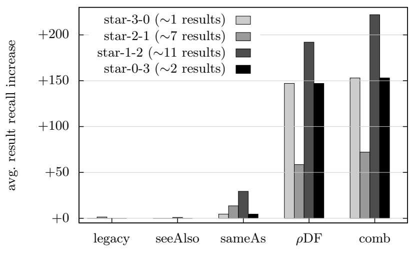
Results per Query Class: We execute each query with six different setups: `legacy` denotes the original LTBQE approach; `base` denotes optimised LTBQE; `select` denotes `seeAlso`, `sameAs`, and ρ`DF` which all extend `base`; `comb` denotes all extensions of `base` applied together. We discuss the results for each query-type, presented as bar plots showing the average recall and time increase per setup vs. `base` (represented as the x-axis). For ρ`DF`, we do not include the time needed to load schema data, which can be done prior to query time.

Entity Queries. Figure 2a shows that the original source selection approach requires between 60% to 110% additional time compared to our source selection optimisation and increases the recall by 3%, shown in Figure 2b. *All such figures include the absolute baseline results (represented by the x-axis) in the legend.* Our extensions increase the recall for all entity queries with a maximum increase of 200% for *entity-so* queries and the combination of all extensions. To enable such increases in recall, the query time increases by up to 55%.

Star Queries. The results for the star shaped queries, presented in Figure 3a and Figure 3b show similarities with the observations for the entity queries. The query times increase without an improvement of the recall if we dereference all appearing URIs and our extensions improve the recall by a maximum of 140% for 3 out of the 4 query classes. It is worth noting the small volume of

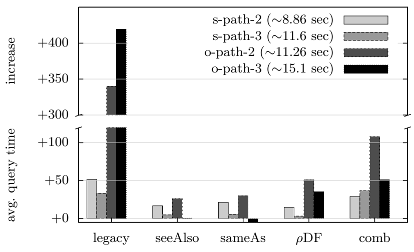


(a) Average *time* increase.

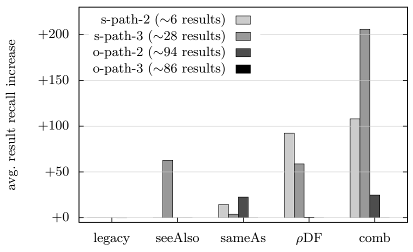


(b) Average *recall* increase.

Fig. 3. Average *time* and *recall* increase relative to baseline for *star queries*



(a) Average *time* increase.



(b) Average *recall* increase.

Fig. 4. Average *time* and *recall* increase relative to baseline for *path queries*

results for the baseline, which may skew average relative increases. Further, we observed some extreme outliers for the query class *star-1-2* due to one query which took around 1 hour to terminate because of a document download from the `ecowlim.tfri.gov.tw` provider (which did not even contribute to the results).

Path Queries. The results for the path shaped queries show an average time increase of up to 420% vs. `legacy` compared to our optimised selection for the two *o-path* query classes in Figure 4a. In contrast to the previous query classes, we see in Figure 4b that the `seeAlso` extension improves the recall by over 50% for the *s-path-3* query class, which is the highest measured improvement for that extension across all query classes. In addition, we observe a recall increase of at least 50% for *s-path-** queries with the ρ`DF` and `comb` setups, whereas we measured only a marginal increase for *o-path-** queries.

7 Conclusion

Proposed link-traversal query approaches for Linked Data have the benefit of up-to-date results and decentralised execution, but operate over incomplete knowledge available in dereferenced documents, thus affecting recall for results.

We empirically study this issue for a large sampling of the Web of Data, consisting of 7.4 million Linked Data documents and 2.1 billion quadruples. We further propose to improve recall by considering inferable knowledge, specifically that found through `owl:sameAs` and RDFS reasoning. We again validate our extensions by analysis of our corpus, where we show increases in data available to the LTBQE approach (1) of 1.006× considering `rdfs:seeAlso` information as proposed in [12], (2) of 2.5× considering `owl:sameAs` and (3) of 1.8× if we apply ρDF reasoning using static schema information. We generate and run queries (of eleven different shapes) live over the Web of Data, comparing six different setups, demonstrating the degree to which our extensions find additional results at the cost of accessing more sources and thus taking longer. In addition, our comprehensive experiment also highlights the problem of unreliable server behaviour, which affects query processing and is symptomised by outliers in results and unavailability of data for certain queries.

Future Work. We plan to extend our entailment rules to cover more of OWL 2 RL/RDF and to investigate changes in recall when considering dynamically dereferenced schema data vs. static schema data. We also plan to use `owl:sameAs` optimisations for canonicalising equivalent URIs as opposed to materialising all equivalent data. Another open issue relates to that of data quality, where, *e.g.*, Halpin et al. [8] suggest that `owl:sameAs` may often be unreliable; we have experiences of dealing with data-quality issues for reasoning in other works [16]. From such work, we herein borrowed the notion of authoritative RDFS reasoning; further measures to make results more "robust" are a subject for future work.

Given the relatively slow response times from the LTBQE approach (where 1% of all queries returned in less than a second, whereas 67% of the queries executed in less than 10 seconds), our ultimate goal is to use such live querying techniques, in a best-effort manner, to compliment centralised query services with fresh answers, particularly for query patterns that are determined to be dynamic and for which data should be retrieved directly from source. In this scenario, our optimisations should help to get faster live-query answers and our extensions to find further answers. In general, combining different Linked Data querying techniques to find a sweet-spot between fast response times and a high recall of (fresh) answers is an open question, the solution to which is likely to be user- and query-specific. Along such lines, in this paper we have formalised, proposed and evaluated the feasibility in a real-world setting of several novel live querying techniques, a selection of which also incorporate lightweight reasoning.

Acknowledgements. *This research has been supported by Science Foundation Ireland under Grant No. SFI/08/CE/I1380 (Lion-II).*

References

1. Buil-Aranda, C., Arenas, M., Corcho, O.: Semantics and Optimization of the SPARQL 1.1 Federation Extension. In: Antoniou, G., Grobelnik, M., Simperl, E., Parsia, B., Plexousakis, D., De Leenheer, P., Pan, J. (eds.) ESWC 2011, Part II. LNCS, vol. 6644, pp. 1–15. Springer, Heidelberg (2011)

2. Berners-Lee, T.: Linked Data. Design issues, W3C (2006)
3. Bishop, B., Kiryakov, A., Ognyanoff, D., Peikov, I., Tashev, Z., Velkov, R.: Factforge: A fast track to the web of data. Sem. Web J. (2011)
4. Bishop, B., Kiryakov, A., Ognyanoff, D., Peikov, I., Tashev, Z., Velkov, R.: Owlim: A family of scalable semantic repositories. SWJ (2011)
5. Bizer, C., Cyganiak, R., Heath, T.: How to publish Linked Data on the web, Tutorial (July 2008), linkeddata.org
6. Bonatti, P.A., Hogan, A., Polleres, A., Sauro, L.: Robust and scalable Linked Data reasoning incorporating provenance and trust annotations. JWS (2011)
7. Erling, O., Mikhailov, I.: RDF Support in the Virtuoso DBMS. In: Pellegrini, T., Auer, S., Tochtermann, K., Schaffert, S. (eds.) Networked Knowledge - Networked Media. Studies in Computational Intelligence, vol. 221, pp. 7–24. Springer, Heidelberg (2009)
8. Halpin, H., Hayes, P.J., McCusker, J.P., McGuinness, D.L., Thompson, H.S.: When owl:sameAs Isn't the Same: An Analysis of Identity in Linked Data. 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. 305–320. Springer, Heidelberg (2010)
9. Harth, A., Umbrich, J., Hogan, A., Decker, S.: YARS2: A Federated Repository for Querying Graph Structured Data from the Web. In: Aberer, K., Choi, K.-S., Noy, N., Allemang, D., Lee, K.-I., Nixon, L.J.B., Golbeck, J., Mika, P., Maynard, D., Mizoguchi, R., Schreiber, G., Cudré-Mauroux, P. (eds.) ASWC 2007 and ISWC 2007. LNCS, vol. 4825, pp. 211–224. Springer, Heidelberg (2007)
10. Hartig, O.: Zero-Knowledge Query Planning for an Iterator Implementation of Link Traversal Based Query Execution. In: Antoniou, G., Grobelnik, M., Simperl, E., Parsia, B., Plexousakis, D., De Leenheer, P., Pan, J. (eds.) ESWC 2011, Part I. LNCS, vol. 6643, pp. 154–169. Springer, Heidelberg (2011)
11. Hartig, O.: SPARQL for a Web of Linked Data: Semantics and Computability. In: Simperl, E., Cimiano, P., Polleres, A., Corcho, O., Presutti, V. (eds.) ESWC 2012. LNCS, vol. 7295, pp. 8–23. Springer, Heidelberg (2012)
12. Hartig, O., Bizer, C., Freytag, J.-C.: Executing SPARQL Queries over the Web of Linked Data. In: Bernstein, A., Karger, D.R., Heath, T., Feigenbaum, L., Maynard, D., Motta, E., Thirunarayan, K. (eds.) ISWC 2009. LNCS, vol. 5823, pp. 293–309. Springer, Heidelberg (2009)
13. Hartig, O., Langegger, A.: A database perspective on consuming Linked Data on the web. Datenbank-Spektrum (2010)
14. Hayes, P.: RDF semantics. W3C Recommendation (February 2004)
15. Heath, T., Bizer, C.: Linked Data: Evolving the Web into a Global Data Space. Morgan & Claypool (2011)
16. Hogan, A.: Exploiting RDFS and OWL for Integrating Heterogeneous, Large-Scale, Linked Data Corpora. PhD thesis, DERI, NUIG (2011)
17. Ladwig, G., Tran, T.: Linked Data Query Processing Strategies. 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. 453–469. Springer, Heidelberg (2010)
18. Ladwig, G., Tran, T.: SIHJoin: Querying Remote and Local Linked Data. In: Antoniou, G., Grobelnik, M., Simperl, E., Parsia, B., Plexousakis, D., De Leenheer, P., Pan, J. (eds.) ESWC 2011, Part I. LNCS, vol. 6643, pp. 139–153. Springer, Heidelberg (2011)

19. Li, Y., Heflin, J.: Using Reformulation Trees to Optimize Queries over Distributed Heterogeneous Sources. 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. 502–517. Springer, Heidelberg (2010)
20. Muñoz, S., Pérez, J., Gutierrez, C.: Simple and efficient minimal RDFS. JWS (2009)
21. Oren, E., Delbru, R., Catasta, M., Cyganiak, R., Stenzhorn, H., Tummarello, G.: Sindice.com: a document-oriented lookup index for open Linked Data. IJMSO (2008)
22. Prud'hommeaux, E., Seaborne, A.: SPARQL query language for RDF. W3C Recommendation (January 2008), http://www.w3.org/TR/rdf-sparql-query/
23. Quilitz, B., Leser, U.: Querying Distributed RDF Data Sources with SPARQL. In: Bechhofer, S., Hauswirth, M., Hoffmann, J., Koubarakis, M. (eds.) ESWC 2008. LNCS, vol. 5021, pp. 524–538. Springer, Heidelberg (2008)
24. Schwarte, A., Haase, P., Hose, K., Schenkel, R., Schmidt, M.: FedX: A Federation Layer for Distributed Query Processing on Linked Open Data. In: Antoniou, G., Grobelnik, M., Simperl, E., Parsia, B., Plexousakis, D., De Leenheer, P., Pan, J. (eds.) ESWC 2011, Part II. LNCS, vol. 6644, pp. 481–486. Springer, Heidelberg (2011)
25. Stuckenschmidt, H., Vdovjak, R., Houben, G.-J., Broekstra, J.: Index structures and algorithms for querying distributed RDF repositories. In: WWW (2004)
26. Tran, T., Zhang, L., Studer, R.: Summary Models for Routing Keywords to Linked Data Sources. 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. 781–797. Springer, Heidelberg (2010)
27. Umbrich, J., Hose, K., Karnstedt, M., Harth, A., Polleres, A.: Comparing data summaries for processing live queries over Linked Data. In: WWWJ (2011)
28. Umbrich, J., Karnstedt, M., Hogan, A., Parreira, J.X.: Freshening up while staying fast: Towards hybrid SPARQL queries. In: EKAW (2012)

Distributed Ontology Based Data Access via Logic Programming[⋆]

Rosamaria Barilaro, Nicola Leone, Francesco Ricca, and Giorgio Terracina

Department of Mathematics, University of Calabria, Italy
{barilaro,leone,ricca,terracina}@mat.unical.it

1 Introduction

In the area of data and knowledge management, ontology-based query answering (OB QA) is becoming more and more a relevant task [2,3]. In fact, many organizations and autonomous contributors are generating the so called "Web of Data", making publicly available Semantic Web repositories built either from scratch or by translation of existing data in ontological form. Contemporarily, database technology providers – such as Oracle, Ontotext, and Ontoprise – have started to build ontological reasoning modules on top of their existing software. Ontological reasoning is also the goal of several research-based systems, such as Quest, Owlgres, Owlim, and QuOnto, just to cite a few.

A main stream of research concerning OBQA aims at extending the expressiveness of tractable ontological theories (see, e.g., [5]). The results obtained on data complexity for query answering over the most significant fragments of the Ontology Web Language (OWL) revealed the practical unfeasibility of handling large amounts of data. To overcome this limitation, several restrictions were imposed on ontology languages, originating the so-called *lightweight* ontologies. These include, e.g., the *DL-Lite* family and the $\mathcal{EL}$ family, which ensure cheap (i.e., from LOGSPACE up to polynomial) data-complexity bounds for query answering.

Several approaches to OBQA rely on query reformulation, where the original query posed on the ontology is rewritten into an equivalent set of "cheap" queries that can be evaluated directly on the ontology instances. Many query rewriters exist, but they often miss the evaluation layer. In this paper we do not focus on a specific language/profile, but we provide a framework for evaluating queries over ontologies that can be Datalog-rewritable (see [4] for an overview). This setting is particularly interesting since the evaluation of the corresponding queries can be delegated to Datalog evaluators, which may possibly rely on DBMSs [8]. A further issue in this context is that existing approaches do not explicitly take into account distribution of data on different sites. Actually, the basic problem of linking data coming from different sources to a formal conceptualization of the domain of interest, has been addressed in [6], where data stored in data management systems like relational databases is linked to ontology theories by the notion of virtual Abox. However, [6] does not specifically deal with the efficient evaluation of queries on *distributed* data. A naïve solution to the distribution problem

[⋆] This work has been partially supported by the Calabrian Region under PIA (Pacchetti Integrati di Agevolazione industria, artigianato e servizi) project DLVSYSTEM approved in BURC n. 20 parte III del 15/05/2009 - DR n. 7373 del 06/05/2009.

clearly includes copying all distributed data on a centralized ontology storehouse; however, this kind of solution is clearly not well suited when the ontologies are numerous, large, frequently changing, or when the applications need to operate on "fresh" data.

From the considerations above it comes out that the capability to carry out efficient distributed OBQA is a crucial task. This paper provides a contribution in this setting. Specifically, it deals with the efficient management of (Datalog-rewritable) *queries* posed on *light ontologies* possibly *distributed* over physically-different sites. In particular, we propose a framework for the efficient *distributed* evaluation of ontological queries via Datalog programs, where we adopt an evaluation technique based on structural query-decomposition methods. Moreover, we report on the results of some preliminary experiments in the context of OBQA, based on a well-established ontology benchmark, which demonstrates the applicability of our approach.

2 Query Evaluation Framework for Distributed Ontology-Based Data Access

The notion of *Ontology-Based Data Access* (OBDA) was originally introduced in [6], Here we elaborate on it to allow for querying distributed data in a Datalog-based environment. In the original framework, a set of pre-existing data sources are assumed to form the Data Layer of an information system. The conceptual view of data contained in the system is expressed in terms of an ontology which is linked to the Data Layer by a set of mapping assertions (virtual ABox). In complex scenarios, it is natural to assume that this data is natively distributed among a set of autonomous and independent data sources, and that these do not necessarily reside on the same host computer. In the framework of [6] each mapping assertion is intended to be an SQL Query over a database DB. Here we extend the setting by allowing each mapping assertion to be expressed as a generic set of Datalog rules (possibly with stratified negation and recursion), where each rule may refer to relations possibly distributed on a set of relational databases $DB_1, \ldots, DB_n$ ($n > 0$).

In this framework, the ontology not only models the domain of interest, but also represents a global access point for clients to interact with the distributed system. In our approach, we assume that the ontology ABox is stored in different databases, and query rewritings are Datalog programs which include the mappings for virtual ABox. Moreover, each DBMS involved by the query rewriting can be used both as a source site of data and as a server site for computation. Clearly, if some site limits its visibility only to data accesses, it will not be considered among the possible sites for computation in the optimization process.

An efficient query evaluation layer can, thus, be obtained by developing a distributed Datalog evaluator. To this end, we employ an evaluator which is able to deal with DBMSs. This is repeatedly invoked by an external evaluation manager which, in turn, optimizes both the flow of data among available sites and the execution of rules.

The distributed Datalog evaluator that we propose to exploit was preliminarily introduced in [1] and is based on the DLV^{DB} system [8]. As far as the evaluation manager is concerned, it takes the input program and first applies query optimization methods. Then, it determines the parallel execution workflow of rules, according to the partial order induced by rule-head dependencies. Finally, it optimizes the distributed evaluation

Table 1. Query evaluation runtimes in seconds. **(N)** Naïve – **(D)** Distributed.

	Presto (N)	**Presto (D)**	**Requiem (N)**	**Requiem (D)**
Query-1	687.5	45.1	618.3	43.6
Query-2	469.4	86.3	447.4	179.1
Query-3	265.0	1.4	284.9	2.3
Query-4	348.1	3.3	324.6	2.8

of single rules by structural decomposition methods. As a part of the parallel execution workflow the system chooses where to store partial results of each rule evaluation. These choices, in turn, impact on the optimization of the rules depending on these intermediate results. Several techniques could be adopted for this task. One possibility is to chose the rule output sites globally, over all the program rules. While this approach could optimize data transfers for the evaluation of several rules, its main drawback is that it requires a quite precise estimation of the cardinality of intermediate relations. The latter is a difficult task (especially for recursive rules), and a wrong estimation at this stage can compromise the entire optimization process. As a consequence, in our approach, we decided to dynamically choose the rule output site on a per-rule basis, when the evaluation of a rule can be actually fired. In more detail, the evaluation of a Datalog program in our approach is done by combining *(i)* rule unfolding with three methods for parallel/distributed evaluation of rules based on structural properties, i.e., *(ii)* inter-component and *(iii)* intra-component parallelism, and *(iv)* optimized distributed evaluation of (single) rules [1] which is based on Weighted Hypertree-Decomposition [7].

3 Experiments

In order to verify the applicability of the proposed approach, we generated a proof-of-concept instance of the framework presented in Section 2. As far as the data layer is concerned, we considered a variant of the well assessed ontology benchmark LUBM (see `http://swat.cse.lehigh.edu/projects/lubm/`) obtained as follows: A first copy obtained with the Univ-Bench data generator tool (UBA) has been replicated twice by randomly changing a fixed percentage of symbols (30% in our tests). The resulting set of three ontologies shares the same TBox, and has partially overlapping ABoxes. The three ontologies (for a total amount of about 20Gb), have been distributed over three different servers on our network, and have been linked with a set of virtual ABoxes, one for each concept and each role in the TBox.

For instance, given a role $r(X,Y)$ and its instances $r@s1(X,Y)$, $r@s2(X,Y)$, $r@s3(X,Y)$ on the three servers, the adopted virtual mapping had the form $r(X,Y) :$ $-r@s1(X,Y), r@s2(K,Y), r@s3(Z,Y)$ (analogous rules have been expressed for entities).

As for the Ontology and Query Rewriting layer, we considered the rewriting of four LUBM queries produced by Presto and Requiem on the standard ontology; the resulting Datalog programs have been enriched with the rules implementing the virtual ABox.

The Query Evaluation Layer has been realized with our distributed evaluation approach, which received as input the rewritten query and the specification of data distribution, and produced as output query results on the desired site. In order to verify the effectiveness of the approach, we compared running times of the setting presented above with a “naïve” one, where data involved by the query is first moved onto the destination site; in this way, the query evaluation process becomes completely local and is carried out by the standard version of DLV^{DB}. The computers used for the experiments are three rack mount HP ProLiant DL120 G6 equipped with Intel Xeon X3430, 2.4 GHz, with 4 Gb Ram, two of them running Windows 2003 Server Operating System and MS SQL Server, and one running Linux Debian Lenny and PostgreSQL. We set a time limit of 1200s (20 min) after which the execution of a system has been stopped.

The results of our experiments are shown in Table 1. This reports the execution times obtained by running the queries rewritten with both Presto and Requiem; for each query we report side by side (to be easily compared) the performance of the “naïve” evaluation and the one obtained by the “distributed” approach. The table shows the very good performance obtained by the distributed evaluator which is faster than the “naïve” strategy, even by an order of magnitude, independently of the query rewriting. This confirms that our techniques are both effective and orthogonal to the adopted rewriting technique. Our approach is, thus, a good candidate for the development of an efficient layer for the evaluation of queries posed on ontologies, and in particular when data is distributed among several sites.

References

1. Barilaro, R., Ricca, F., Terracina, G.: Optimizing the Distributed Evaluation of Stratified Programs via Structural Analysis. In: Delgrande, J.P., Faber, W. (eds.) LPNMR 2011. LNCS, vol. 6645, pp. 217–222. Springer, Heidelberg (2011)
2. Calì, A., Gottlob, G., Lukasiewicz, T.: A general datalog-based framework for tractable query answering over ontologies. In: Proc. of PODS 2009, pp. 77–86. ACM (2009)
3. Calvanese, D., Giacomo, G.D., Lembo, D., Lenzerini, M., Rosati, R.: Tractable reasoning and efficient query answering in description logics: The dl-lite family. JAR 39(3), 385–429 (2007)
4. Heymans, S., Eiter, T., Xiao, G.: Tractable reasoning with dl-programs over datalog-rewritable description logics. In: Proc. of ECAI 2010, pp. 35–40. IOS Press (2010)
5. Mugnier, M.-L.: Ontological Query Answering with Existential Rules. In: Rudolph, S., Gutierrez, C. (eds.) RR 2011. LNCS, vol. 6902, pp. 2–23. Springer, Heidelberg (2011)
6. Poggi, A., Lembo, D., Calvanese, D., Giacomo, G.D., Lenzerini, M., Rosati, R.: Linking Data to Ontologies. JoDS (10), 133–173 (2008)
7. Scarcello, F., Greco, G., Leone, N.: Weighted hypertree decompositions and optimal query plans. JCSS 73(3), 475–506 (2007)
8. Terracina, G., Leone, N., Lio, V., Panetta, C.: Experimenting with recursive queries in database and logic programming systems. TPLP 8(2), 129–165 (2008)

Argumentation and Voting for an Intelligent User Empowering Business Directory on the Web

Valentinos Evripidou and Francesca Toni

Department of Computing, Imperial College London, UK
valentinos.evripidou10@imperial.ac.uk, ft@imperial.ac.uk

Abstract. We describe a new argumentation method for analysing opinion exchanges between on-line users aiding them to draw informative, structured and meaningful information. Our method combines different factors, such as social support drawn from votes and attacking/supporting relations between opinions interpreted as abstract arguments. We show a prototype web application which puts into use this method to offer an intelligent business directory allowing users to engage in debate and aid them to extract the dominant, emerging public opinion.

1 Introduction

Social web platforms have grown exponentially in recent years creating new opportunities that were unthinkable in the pre-social web era. However, the amount of available information and its loose structure and semantics creates technical obstacles for exploiting such opportunities. Therefore, more intelligent systems are required that can filter and recommend information as well as provide resiliency against reputation attacks within such systems. Examples of such websites include business directories and e-commerce sites where it is very time-consuming to wade through opinions and combine rankings, which, in addition only give a partial view and do not identify important aspects.

We describe a prototype of an intelligent, user-empowering business directory that employs a method, based upon argumentation and extends the method of [4] for sharing and evaluating information in the Social Web. In this prototype, information may be qualitative, in terms of dialectical exchanges (where information can support or criticise other information), or quantitative, in terms of votes. These (voted) dialectical exchanges provide an abstraction for justified opinions and controversy amongst them. The evaluation of information takes into account the dialectical strength of opinions in terms of (i) their justifications, (ii) how they stand against criticism, and (iii) the aggregation of votes they and their supporters/critics receive, mediated by the dialectical strength of their justifications in turn.

2 Extended Social Abstract Argumentation

The *Social Abstract Argumentation (SAA) Framework* [4] is an extension of Abstract Argumentation [2] that incorporates a voting mechanism to take into account positive and negative votes on arguments. Instead of just having (i) a set of arguments A and (ii) a binary relation R on the arguments indicating the attacks between them, the framework also includes a function that maps each argument to a number of positive and negative votes. SAA is equipped with a method to evaluate the strength of each individual argument based on its votes and its attacks. This is calculated through the use of the *Social Model*.

We use a new *Extended SAA* which enhances the Social Abstract Argumentation Framework by including an additional binary relation R^+ indicating support between the arguments. Extended SAA incorporates a method to evaluate the strength of arguments (that we call the *Admissible Social Model*). Our new model produces intuitive results, that comply with admissibility [2], in the cases where an argument has not yet received any social support or when it only received negative votes as social support. To illustrate the use of the Social Model and the differences and enhancements by our proposed Admissible Social Model, lets consider $A = \{a, b, c\}$ and $R = \{(b, a), (c, a)\}$. Note that $\{b\}, \{c\}$ are both admissible sets of arguments in the abstract argumentation sense [2]. Let $V(a) = V(b) = V(c) = (5, 5)$, namely all arguments have accumulated the same amount of positive and negative votes (i.e. 5 positive and 5 negative votes). We can now calculate their social support τ_ϵ (given by the simple Vote Aggregation Function [4]) which will be (with $\epsilon = 0.1$): $\tau_\epsilon(V(a)) = \tau_\epsilon(V(b)) = \tau_\epsilon(V(c)) \simeq 0.50$. According to the Social Model [4] their strength will be: $M(a) = \tau_\epsilon(a)(1 - (M(c) + M(b) - M(c)M(b)) \simeq 0.13$, $M(b) = \tau_\epsilon(V(b)) \simeq 0.50$, $M(c) = \tau_\epsilon(V(c)) \simeq 0.50$. Fig. 1 shows the graphical representation of the framework, where nodes correspond to the arguments in the framework and the edges indicate the attack relations. Node sizes are scaled to their strength (M-valuation).

Let instead $V(b) = (0, 5)$, then $M(b) = 0$. This defies the concept of admissibility, since an unattacked argument is not accepted. Our proposed Admissible Social Model provides a definition for a new a vote aggregation function, the *Admissible Simple Vote Aggregation* function which solves this problem by introducing new conditions and a constant strength evaluation i in such cases. According to the new model the strength will be: $M'(a) = \tau_\epsilon(a)(1 - (M'(b) + M'(c) - M'(b)M'(c)) \simeq 0.245$, $M'(b) = \tau_\epsilon(b) \simeq \frac{i}{5}$where $i \in (0, 0.5]$, a predefined initial value (we assume here $i = 0.1$) and $M'(c) = \tau_\epsilon(c) \simeq 0.50$. The Admissible Social Model satisifies a number of intuitive properties [5].

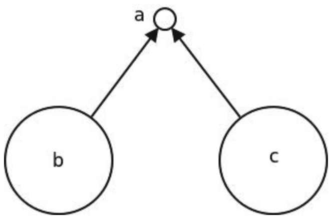
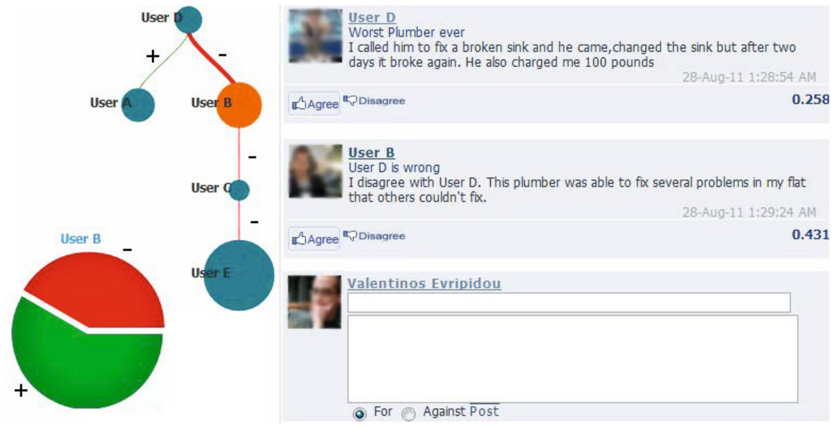


Fig. 1. The arguments of the example scaled to their strength

3 Prototype

The prototype web application, which is based on the Admissible Social Model, takes the form of an online business directory that lets users discuss about enlisted businesses in an expressive and interactive way. It gives users the capabilities of posting comments as structured arguments, giving feedback through votes, and engage into a meaningful debate over any business in the directory. Additionally, this information is visualised providing an intuitive and easy to understand structure of the conversation.



User D
Worst Plumber ever
I called him to fix a broken sink and he came,changed the sink but after two days it broke again. He also charged me 100 pounds
28-Aug-11 1:28:54 AM
0.258

User B
User D is wrong
I disagree with User D. This plumber was able to fix several problems in my flat that others couldn't fix.
28-Aug-11 1:29:24 AM
0.431

Valentinos Evripidou

Fig. 2. A screen shot of the prototype application showing the user interface

A screenshot of the application is shown in Fig. 2 were five users (A, B, C, D, E) share opinions, and those of users D and B are displayed on the right. Opinions are organised in a tree structure (depicted on the left). Supporters are indicated by links labeled with "+" and critics (attacks) are indicated by links labeled with "-". The pie chart (bottom-left) indicates positive ("+") and negative ("-") votes for the opinion of user B (the node selected). The strength of opinions, computed as per the Admissible Social Model, determines the size of nodes in the tree. For example, user E has the strongest opinion as no user disagrees with him. User B has the next strongest opinion due to indirect support (critic of critic) by the strongest opinion of E, combined with more positive than negative votes. The strength of opinions can help users with deciding whether to choose the item or service being discussed (use of a specific plumber in this case). Each user, when posting an argument, has to explicitly state beforehand the nature of the argument (i.e. support or attack to another argument). Additionally, the strength evaluation of the arguments is not static; a filtering mechanism provides a dynamic way of excluding or including arguments. For example a user can filter out all arguments from users that do not share the same interests or the same education/work background. This is done through the use of a social login mechanism which grants access to a user's public profile and arguments are filtered using this information.

4 Conclusions and Related Work

Several online platforms exist (e.g. debategraph.org, debate.org, livingvote.org) that enforce or visualise online debates on any subject. Such tools have not found their way into business directories or e-commerce sites in which users post unstructured reviews and no intelligent mechanism exists to aid them understand and evaluate the dominant public opinion. Through our prototype business directory, users are able to interact, exchange opinions and capture the overall idea about an enlisted business. An interesting approach is shown in [6] where the authors analyse an exchange of comments on Facebook in order to formalise and validate the opinions being expressed. We believe that our proposed extended Social Abstract Argumentation Framework, its semantics and the Admissible Social Model is a step towards the formalisation and analysis of such online conversations since it can successfully endorse relations of support between arguments and also take into account user voting. Some frameworks exist [1] that incorporate supporting relations but do not combine any notion of social support for the arguments. Additional planned future research includes sentiment analysis for automatically identifiying relations between arguments, as described in [3], and the introduction of different degrees for agreement and disagreement rather than having only attacks and supports; this would provide greater expressiveness and flexibility.

Acknowledgments. The authors would like to thank Michael Huth for his feedback. This research was partially funded by a Pathways to Impact Project by Imperial College London.

References

1. Amgoud, L., Cayrol, C., Lagasquie-Schiex, M.C.: On the bipolarity in argumentation frameworks (June 2004), ftp://ftp.irit.fr/pub/IRIT/RPDMP/article-nmr04.ps.gz
2. Dung, P.M.: On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and N-persons games (October 16, 1995), http://citeseer.ist.psu.edu/7707.html, http://www.di.unipi.it/~paolo/KIT011/dungpapers/arg-fin.ps.Z
3. Gabbriellini, S., Torroni, P.: Microdebates for policy-making. In: 3rd International Conference on Computational Sustainability, CompSust 2012, Copenhagen, Denmark, July 5-6 (2012)
4. Leite, J., Martins, J.: Social abstract argumentation. In: Walsh, T. (ed.) IJCAI, pp. 2287–2292. IJCAI/AAAI (2011), http://ijcai.org/papers11/Papers/IJCAI11-381.pdf
5. Matt, P.A., Toni, F.: A Game-Theoretic Measure of Argument Strength for Abstract Argumentation. In: Hölldobler, S., Lutz, C., Wansing, H. (eds.) JELIA 2008. LNCS (LNAI), vol. 5293, pp. 285–297. Springer, Heidelberg (2008)
6. Toni, F., Torroni, P.: Bottom-Up Argumentation. In: Modgil, S., Oren, N., Toni, F. (eds.) TAFA 2011. LNCS, vol. 7132, pp. 249–262. Springer, Heidelberg (2012), http://dx.doi.org/10.1007/978-3-642-29184-5

A Polynomial Reduction from ASPDA to ASP*

Wolfgang Faber

Department of Mathematics
University of Calabria
87030 Rende (CS), Italy
wf@wfaber.com

Abstract. ASPDA is a framework for expressing defeasibility in Answer Set Programs via so-called argumentation theories, proposed by Wan, Kifer, and Grosof in [2]. The authors describe a reduction from ASPDA to plain Answer Set Programming, which however exponentially inflate programs. In this note, we present an alternative reduction, which does not suffer from this problem. As a side-effect, complexity results for ASPDA are established.

1 Introduction

ASPDA is a framework for expressing defeasibility in Answer Set Programs via so-called argumentation theories, proposed by Wan, Kifer, and Grosof in [2]. ASPDA programs provide a very general means for defeating literals and rules and capture several earlier proposals for defeasibility in logic programming. In [2], the authors also provide a reduction from ASPDA to ASP (in the sense of [1]). However, this reduction can easily lead to ASP programs that are exponentially larger than the original ASPDA programs. In this paper, we show that this exponential behavior is not necessary, by providing an alternative reduction. Different to the reduction in [2], ours introduces new symbols and also needs a concept of rule identifier, which make proving correctness of the reduction slightly more cumbersome. However, this reduction immediately provides complexity results for ASPDA, in particular showing that (virtually all) computational tasks over ASPDA programs have the same complexity as those over ASP programs.

2 ASPDA: Syntax and Semantics

We briefly review syntax and semantics of ASPDA, for details we refer to [2]. The language assumes a set of atoms; in [2] this set is not fixed, here we assume it to consist of first-order or propositional atomic formulas. There are two kinds of negation, and a literal is either an atom A, $\texttt{neg}\ A$, $\texttt{naf}\ A$, or $\texttt{naf}\ \texttt{neg}\ A$. A rule is of the form

$$@r\ L_1 \vee \cdots \vee L_k :- \mathit{Body} \tag{1}$$

where $k \geq 0$, r is a term and the tag of the rule (different rules can share the same rule tag), each L_i ($0 < i \leq k$) is a literal, and Body is a conjunction of literals. Given a rule

* This work was supported by M.I.U.R. within the PRIN project LoDeN.

of the form (1), the term $h(r, L_i)$ ($handle(r, L_i)$ in [2]) is the handle for each of the head literals L_i ($0 \leq i \leq k$). Each rule can be either defeasible or strict.

An argumentation theory AT is a set of strict rules of the form (1), which makes use of a distinguished predicate $\$defeated_{AT}$ that may occur only in rule heads. The subscript AT is usually omitted when the context is clear. An answer-set program with defaults and argumentation theories (ASPDA) is a set of rules of the form (1), which may comprise an argumentation theory. In [2], the argumentation theory is usually considered separated from the program, but since it is syntactically and semantically the same as a special kind of program, we consider it as part of the program for simplicity.

Herbrand universe and base are defined in the standard way, where the Herbrand base consists not just of ground atoms, but of ground `naf`-free literals. An (Herbrand) interpretation is a subset of the Herbrand base, and we will assume consistent interpretations, i.e. no interpretation contains both A and `neg` A.

A `naf`-free literal L is true in an interpretation I if $L \in I$, `naf` L is true in I if $L \notin I$; otherwise these literals are false in I. A strict rule is satisfied in I if at least one head literal is true in I whenever all body literals are true in I. A defeasible rule of the form (1) is satisfied if it either meets the condition for a strict rule or if $\$defeated(h(r, L_i))$ is true in I for all $0 < i \leq k$. As usual, an interpretation I is a model of an ASPDA program P if I satisfies all rules in P. A model of a program P is minimal if none of its subsets is a model of P.

For defining answer sets, [2] define the quotient $\frac{P}{I}$ for an ASPDA program P and an interpretation I in four steps: (i) Delete every rule in P in which a `naf` body literal is false in I; (ii) in each defeasible rule of the form (1), delete all L_i that are true in I; if all L_i are deleted, delete the complete rule; (iii) remove all `naf`-literals of the remaining rules; (iv) remove tags from the remaining rules. An interpretation I is an answer set of an ASPDA P if I is a minimal model of $\frac{P}{I}$.

Traditional ASP can be viewed as a special case of ASPDA. ASPDA programs that have no defeasible rules and empty argumentation theory can be viewed as ASP programs. It is easy to show that the quotient $\frac{P}{I}$ coincides with the reduct P^I [1] for such programs (the only difference are the rule tags, which are irrelevant for these programs).

3 A Polynomial Reduction from ASPDA to ASP

In [2] a reduction from ASPDA to ASP is provided that preserves answer sets, which however, produces an exponential number of rules in general. We provide an alternative reduction, which does not suffer from this exponential increase in size.

Definition 1. *Given an ASPDA P, for each defeasible rule of the form (1), create*

$$\$der(r, L_1) \vee \cdots \vee \$der(r, L_k) :- \mathit{Body}, \texttt{naf}\ \$rdef(rid) \tag{2}$$

$$\$rdef(rid) :- \$defeated(h(r, L_1)), \ldots, \$defeated(h(r, L_k)) \tag{3}$$

where $\$rdef$ and $\$der$ are fresh predicates, rid is a rule identifier (obtained for example by the index of a fixed enumeration of rules; note that the rule tag cannot serve as the rule identifier), and for each $0 < i \leq k$ create

$$L_i :- \ \$der(r, L_i) \tag{4}$$

$$\$der(r, L_i) \;:-\; L_i, \mathtt{naf}\ \$defeated(h(r, L_i)) \qquad (5)$$

$$:-\; \$der(r, L_i), \$defeated(h(r, L_i)) \qquad (6)$$

For each strict rule, delete its rule tag. We refer to the obtained program as $tr(P)$.

Example 1. For $@r\ a \vee b :-$ the reduction of [2] generates

$$a \vee b :- \mathtt{naf}\ \$defeated(h(r, a)), \mathtt{naf}\ \$defeated(h(r, b))$$
$$a :- \mathtt{naf}\ \$defeated(h(r, a)), \$defeated(h(r, b))$$
$$b :- \$defeated(h(r, a)), \mathtt{naf}\ \$defeated(h(r, b))$$

(and also $:-\$defeated(h(r, a)), \$defeated(h(r, b))$, but this seems to be due to a typo). The reduction of Definition 1 generates

$$\$der(r, a) \vee \$der(r, b) :- \mathtt{naf}\ \$rdef(rid) \qquad a :- \$der(r, a) \qquad b :- \$der(r, b)$$
$$\$rdef(rid) :- \$defeated(h(r, a)), \$defeated(h(r, b))$$
$$\$der(r, a) :- a, \mathtt{naf}\ \$defeated(h(r, a)) \qquad :- \$der(r, a), \$defeated(h(r, a))$$
$$\$der(r, b) :- b, \mathtt{naf}\ \$defeated(h(r, b)) \qquad :- \$der(r, b), \$defeated(h(r, b))$$

In general, for each rule with k head literals, the reduction of Definition 1 creates $3k + 2$ rules, while the one of [2] creates $2^k - 1$ rules.

Theorem 1. *Given an ASPDA P, there is a one-to-one relationship between the answer sets of P and those of $tr(P)$. In particular, for each answer set A of P, $tr(A) = A \cup \{\$der(r, L) \mid$ a defeasible rule with tag r in P exists with true body and L in its head, s.t. $\$defeated(h(r, L)) \notin A$ and $L \in A\,\} \cup \{\$rdef(rid) \mid$ a defeasible rule with identifier rid and tag r in P exists s.t. for each head literal L, $\$defeated(h(r, L)) \in A$ holds $\}$ is an answer set of $tr(P)$, and these are the only answer sets of $tr(P)$.*

Proof. Assume that A is an answer set of P (hence a minimal model of $\frac{P}{A}$). We show that $tr(A)$ is a minimal model of $\frac{tr(P)}{tr(A)}$. First observe that for each rule in P which is deleted in step (i) of the definition of $\frac{P}{A}$, the rule itself or its corresponding rule (4) is not in $\frac{tr(P)}{tr(A)}$ either. Moreover, a defeasible rule in P for which $\$defeated(h(r, L)) \in A$ holds for all head literals L is not in $\frac{P}{A}$ due to step (ii) of the definition of $\frac{P}{A}$, and no reduct of the corresponding rule (4) is in $\frac{tr(P)}{tr(A)}$ either, since by construction $\$rdef(rid) \in tr(A)$. For all other defeasible rules of form (1) in P (i.e. those not deleted in steps (i) and (ii) of the definition of $\frac{P}{A}$), $\frac{P}{A}$ contains $\bigvee_{L \in K} L :- Body'$, s.t. K is the set of head literals s.t. $\$defeated(h(r, L) \notin A$ and $Body'$ is $Body$ without naf-literals. $\frac{tr(P)}{tr(A)}$ instead has $\$der(r, L_1) \vee \cdots \vee \$der(r, L_k) :- Body'$, $\$der(r, L_i) :- L_i$ for L_i s.t. $\$defeated(h(r, L_i)) \notin A$ and also all rules of type (3), (4), and (6). By construction, $tr(A)$ is a model of $\frac{tr(P)}{tr(A)}$. To see minimality, observe that $\$der(r, L_i)$ take the place of L_i in rule heads of reducts in $\frac{tr(P)}{tr(A)}$. So if there is a model $N \subsetneq tr(A)$ for $\frac{tr(P)}{tr(A)}$ not containing some $\$der(r, L_i)$, then also L_i must not be in that model because of a rule of type (4). Removing other literals from $tr(A)$ cannot yield a model of $\frac{tr(P)}{tr(A)}$.

Assume now that M is an answer set for $tr(P)$. We show that $M = tr(A)$ for some answer set A of P. Let M' denote the set M after removing all literals $\$der(r, L)$ and $\$rdef(rid)$. First we note that M' satisfies all strict rules in the reduct $\frac{P}{M'}$. Concerning defeasible rules, if $\$rdef(rid) \in M$, then the rule with identifier rid is not in $\frac{P}{M'}$. Otherwise, rule (2) is in $\frac{tr(P)}{M}$ and a corresponding rule r' obtained from r is also in $\frac{P}{M'}$. Note that r' in general has fewer head literals than (2), however, we observe that for each L_i that was removed from r when creating r' in $\frac{P}{M'}$ there is a rule (6) and for each of these L_i, $\$defeated(h(r, L_i)) \in M'$ and $\$defeated(h(r, L_i)) \in M$ and so $\$der(r, L_i) \notin M$. Hence if rule (2) has a true body in M, $\$der(r, L_j) \in M$ holds only if $\$defeated(h(r, L_j)) \notin M$ and $\$defeated(h(r, L_j)) \notin M'$. Moreover, rules (4) enforce that $L_j \in M$ and $L_j \in M'$. It follows that all rules in $\frac{P}{M'}$ that stem from defeasible rules are satisfied by M' and hence M' is a model of $\frac{P}{M'}$. Minimality of M' then follows from the minimality of M and the fact that the satisfaction patterns of rule heads of the reducts of defeasible rules and the corresponding reducts of rules (2) coincide. Therefore M' is an answer set of P and $M = tr(M')$.

The computational complexity of reasoning tasks over ASPDA programs was left open in [2]. It is obvious that the reduction in Definition 1 runs in polynomial time, hence the reduction provides a tight upper bound for the complexity of all computational tasks of ASPDA, where the corresponding task for ASP is at least polynomial.

Corollary 1. *Given a computational task over ASP programs wwhich is complete for located a complexity class that contains P, the corresponding task over ASPDA programs is located in the same complexity class.*

Since traditional ASP programs are a special case of ASPDA, lower bounds extend trivially from ASP to ASPDA.

4 Conclusion

We have provided an alternative reduction from ASPDA to ASP, which avoids an exponential increase in space and thus is an immediate improvement over an analogous reduction in [2]. Contrary to the earlier reduction, it makes use of additional symbols and also needs the concept of a rule identifier. As an immediate consequence of the reduction, we obtain results on the computational complexity of computational tasks over ASPDA, which coincide with those of ASP for practically all relevant tasks.

References

1. Gelfond, M., Lifschitz, V.: Classical Negation in Logic Programs and Disjunctive Databases. New Generation Computing 9, 365–385 (1991)
2. Wan, H., Kifer, M., Grosof, B.N.: Defeasibility in Answer Set Programs via Argumentation Theories. In: Hitzler, P., Lukasiewicz, T. (eds.) RR 2010. LNCS, vol. 6333, pp. 149–163. Springer, Heidelberg (2010)

The Definability Abduction Problem for Data Exchange
(Abstract)

Enrico Franconi[1], Nhung Ngo[1], and Evgeny Sherkhonov[2]

[1] KRDB Research Center for Knowledge and Data, Free University of Bozen-Bolzano, Italy
[2] ISLA, University of Amsterdam, The Netherlands

1 Introduction

Data exchange is the problem of transforming data structured according to a source schema into data structured according to a target schema, via a *mapping* specified by means of rules in the form of *source-to-target tuple generating dependencies* – rules whose body is a conjunction of atoms over the source schema and the head is a conjunction of atoms over the target schema, with possibly existential variables in the head. With this formalization, given a fixed source database, there might be more than one target databases satisfying a given mapping. That is, the target database is actually an *incomplete database* represented by a set of possible databases. Therefore, the problem of query answering the target data is inherently complex for general (non-positive) relational or aggregate queries.

In order to recover the good computational properties of standard relational database technologies, the data exchange framework restricts the target query language to just (unions of) conjunctive queries ((U)CQ): as a matter of fact, the answers to such a query over the incomplete target database are the same as the answers of the same query over a single specific unique database (called a *universal solution*) [7]. So, the goal in the data exchange framework is to have one single database in the target once the source database is fixed. Then this database can be considered as a normal complete database and which can be queried with SQL – restricted to positive select-from-where queries.

The goal of this work is to enrich the data exchange framework to allow for general relational and aggregate queries, by suggesting "reasonable" amendments to the initial mapping so that the new extended mapping will then produce a *unique* materialized target instance depending only on the given source instance and the mapping. To this end, we formalize the property for the target being unique, and introduce and solve the novel problem of *definability abduction*. We consider a specific semantic minimality criterion and propose minimal abductive solutions of specific form.

2 Data Exchange and Typical Problems

For a more detailed discussion of data exchange problem, see [7,4,6]. We fix two disjoint *source* (S) and *target* (T) schemas. Then a sentence over $S \cup T$ of the form $\forall \bar{x}(\varphi(\bar{x}) \rightarrow \psi(\bar{x}))$, where (1) φ is a $\mathcal{L}_1$-formula over S and (2) ψ a $\mathcal{L}_2$-formula over T, is called S-T (or source-to-target (s-t)) $\mathcal{L}_1$-to-$\mathcal{L}_2$ *dependency*. A tuple $\mathcal{M} = (S, T, \Sigma)$, where Σ is a set of s-t $\mathcal{L}_1$-to-$\mathcal{L}_2$ dependencies, is called a *data exchange problem*. The set Σ is called a *schema mapping*. Intuitively, Σ specifies how and what source data must be translated to the target. A *solution* to the data exchange problem $\mathcal{M}$ for a source instance I is a target instance J such that $(I, J) \models \Sigma$. Commonly, schema mappings are specified with the help of s-t CQ-to-CQ dependencies (also called as s-t tuple generating dependencies (tgd)) [7].

The data exchange framework gives rise to the following problems [3,9,8]. (1) Non-rewritability over a distinguished "uniformly" computed target instance [3], (2) non-intuitive answers to queries with negation [9,8] and (3) semantics for aggregative queries is trivial [1]. For each of the problems different solutions were proposed by considering different query answering semantics and notions of solutions. Among them, the notions of CWA-solutions [9], $GCWA^*$-solutions [8], and endomorphic images of the canonical solution [1]. Overall, there is no uniform approach to solve the above problems. Ultimately, one would like to have a data exchange setting where one can materialize a target database and be able to use the full SQL over it. In fact, the problems can be avoided by restricting a set of possible solutions to a single target database. Next we formalize the property of uniqueness of the target instance for a given source instance under the schema mapping. Also we propose a framework for fixing schema mappings to guarantee this property.

3 Definability and Definability Abduction in Data Exchange

How can we characterize the case when the target is uniquely defined given a source instance? This semantic property is given by the notion of *definability* from FOL [5,10].

Definition 1. *Let Σ be a set of sentences in FOL. A predicate p is* definable *from the set of predicates P under Σ if for every two interpretations $\mathcal{I} = \langle D^{\mathcal{I}}, \cdot^{\mathcal{I}} \rangle$ and $\mathcal{J} = \langle D^{\mathcal{J}}, \cdot^{\mathcal{J}} \rangle$ such that they are models of Σ, it holds that $P^{\mathcal{I}} = P^{\mathcal{J}}$ implies $p^{\mathcal{I}} = p^{\mathcal{J}}$.*
Intuitively, in all models of Σ the extension of p is fully determined by the extension of the predicates in P. Note, if Σ is specified by s-t tgds then the extensions of the target predicates are not determined by the extensions of the source predicates.

Proposition 1. *Let Σ be a set of s-t tgds. Then for every $p \in T$ it holds that p is not definable from S under Σ.*

Note that not every schema mapping gives non-definability, e.g. the FO-to-FO mapping $\Sigma = \{p(x) \to q(x), \neg p(x) \to \neg q(x)\}$ ensures that q is definable from p. Proposition 1 says that the classical schema mapping language is not powerful enough to enforce unique data exchange solutions. Therefore, given a schema mapping specified by s-t tgds we want to extend it with new dependencies which would thus lead to definability of the target from the source. Then some natural questions arise such as how these dependencies should look and which ones we prefer. In fact, similar problems arise in abductive reasoning [11,2]. This motivates us to introduce the following general problem. Here $\sigma(\Sigma)$ means the signature of Σ.

Definition 2. *Let Σ be a set of FOL sentences. A triple $D = (\mathcal{P}_1, \mathcal{P}, \Sigma)$ is a* definability abductive problem *(DAP) if it holds that $\Sigma \cup \widetilde{\Sigma} \not\models \forall \bar{x}.p(\bar{x}) \leftrightarrow \tilde{p}(\bar{x})$ for every $p \in \mathcal{P}$, where $\tilde{p}$ is a fresh predicate of the same arity as p, and $\widetilde{\Sigma}$ is obtained from Σ by replacing all predicates from $\sigma(\Sigma) \setminus \mathcal{P}_1$ with fresh predicates with the same arity.*

The non-entailment in Definition 2 means that no predicate in $\mathcal{P}$ is definable from $\mathcal{P}_1$ under Σ. A set of sentences Δ is called a *solution* to a DAP $D = (\mathcal{P}_1, \mathcal{P}, \Sigma)$ if every

$p \in \mathcal{P}$ is definable from $\mathcal{P}_1$ under $\Sigma \cup \Delta$. A solution Δ is *consistent* if $\Sigma \cup \Delta$ is consistent, *relevant* if $\mathcal{P}$ is not definable from $\mathcal{P}_1$ under Δ, and *minimal* if for every solution Δ_1 such that $\Sigma \cup \Delta \models \Delta_1$ it holds that $\Sigma \cup \Delta_1 \models \Delta$. These restrictions are natural as we are interested in finding meaningful solutions which minimally change the intended meaning of the schema mapping.

In view of Proposition 1, every data exchange setting (S, T, Σ), where Σ is a set of s-t tgds, can be viewed as a DAP. We are now interested in definability abduction for data exchange. First, however, how should the DAP solutions look like?

Definition 3 (sts mapping). *A set of sentences Δ over $S \cup T$ is an* sts mapping *if it consists of s-t CQ-to-CQ and t-s CQ-to-$UCQ^=$ dependencies.*

Thus given a DAP solution Δ in the form of sts, the interplay of both s-t and t-s dependencies together with Σ gives us definability of T from S. Moreover, we believe the proposed solutions in this form could be easily verified by the user.

Now our goal is to provide a user with solutions to a given DAP problem. There are two alternative scenarios related to this. In the first one, one is interested in an algorithm for generating all solutions (which might be infinitely many). The known tableaux and resolution techniques [2] can only generate abductive solutions in the form of a conjunction of literals, and it is not clear if these techniques can be modified to generate dependencies. In the second scenario, the user might only be interested in *some* solutions, e.g. minimal ones. Then redundant computation of the entire set of solutions is avoided. This scenario is more practically feasible, as only a restricted class of solutions needs to be found which are more likely to be acceptable by the user. Let's now consider a DAP $\mathcal{M} = (S, T, \Sigma)$. We distinguish the cases how Σ is specified.

Full tgds. These are s-t tgds that do not contain existential variables in the head. In this case we can always assume that a full s-t mapping has the following form: $\Sigma = \cup_{p_i \in T} \cup_{j=1}^{n_i} \{\varphi_i^j(\bar{x}, \bar{z}_i) \rightarrow p_i(\bar{x})\}$, where φ_i^j is a conjunction of source atoms, p_i contains no constant and the variables in the attributes of p_i are distinct.

Example 1. Let $\Sigma = \{p(x, y) \rightarrow q(x, x) \wedge t(x, y) \wedge h(x, c)\}$ be a full schema mapping, c a constant. Then Σ is logically equivalent to $\Sigma' = \{p(x, y) \rightarrow t(x, y), p(x, y) \wedge z = x \rightarrow q(x, z), p(x, y) \wedge u = c \rightarrow h(x, u)\}$.

We single out a good solution to a DAP in the case of full dependencies. This kind of solution is both intuitive and enjoys being relevant and minimal.

Theorem 1. *Let $\mathcal{M} = (S, T, \Sigma)$ be a DAP specified by full s-t tgds. Then $\Delta = \bigcup_{p_i \in T}\{p_i(\bar{x}) \rightarrow \vee_j \exists \bar{z}_j \varphi_j^i(\bar{x}, \bar{z}_j)\}$ is a minimal relevant consistent sts solution to $\mathcal{M}$.*

Example 2. Consider a DAP $\mathcal{M} = (S, T, \Sigma)$ where $S = \{GradStudent(\cdot), UGradStudent(\cdot)\}, T = \{Student(\cdot)\}$, and $\Sigma = \{GradStudent(x) \rightarrow Student(x), UGradStudent(x) \rightarrow Student(x)\}$. Then $\Delta = \{Student(x) \rightarrow UGradStudent(x) \vee GradStudent(x)\}$ is a minimal solution to $\mathcal{M}$.

Emebdded tgds. These are s-t tgds containing existential variables in the head.

Example 3. Let $\mathcal{M} = (S, T, \Sigma)$ be a DAP, with $\Sigma = \{Person(x) \rightarrow \exists y.PhoneN(x, y)\}$. This DAP doesn't have a relevant sts solution and other solutions are not intuitive at all, e.g. $\Delta = \{Person(x) \wedge Person(y) \leftrightarrow PhoneN(x, y)\}$ is hardly acceptable. One way to solve the problem with non-intuitive solutions is to relax the notion of a DAP solution by allowing additional source predicates.

Definition 4. *Let $\mathcal{M} = (\mathcal{P}_1, \mathcal{P}, \Sigma)$ be a DAP and Δ a set of sentences. Δ is a* weak *solution to $\mathcal{M}$ if it is a solution to $\mathcal{M}' = (\mathcal{P}_2, \mathcal{P}, \Sigma)$ for some $\mathcal{P}_2$ such that $\mathcal{P}_1 \subseteq \mathcal{P}_2$.*

One can see the motivation for weak DAP solutions in data exchange: the additional source predicates can tell us more information about existential values described by the embedded mappings. Thus, we might be able to find though weak, but minimal solutions which could be intuitive from practical point of view. It appears that a solution where we ask the user for the existential values in the target by using *fresh* predicates in the source, is minimal. For every $p \in T$ we pick a fresh predicate p_s of the same arity.

Theorem 2. *Let $\mathcal{M} = (S, T, \Sigma)$ be a DAP where Σ is a set of pure embedded s-t tgds. Then $\Delta = \bigcup_{p \in T}\{p_s \leftrightarrow p\}$ is a minimal weak sts solution to $\mathcal{M}$.*

Example 4. Consider a DAP $\mathcal{M} = (S, T, \Sigma)$ where $S = \{Person(\cdot), Phone(\cdot,\cdot)\}$, $T = \{Contact(\cdot,\cdot)\}$, and $\Sigma = \{Person(x) \rightarrow \exists y.Contact(x,y)\}$. Then we have $\Delta = \{Contact(x,y) \leftrightarrow Phone(x,y)\}$ is a minimal solution of $\mathcal{M}$. It also implies that every source instance must satisfy the source constraint $Person(x) \rightarrow Phone(x,y)$.

4 Conclusions

We considered the problem of gaining definability of the target from the source schema. We showed that this problem is abductive in nature, and that in the case of full mappings there are interesting minimal sts solutions. Embedded schema mappings, on the other hand, seem to be bad mappings for finding *conservative* solutions. Therefore, the notion of a DAP solution has to be relaxed and fresh source predicates are sometimes needed.

The third author is supported by NWO (project 612.001.012 (DEX)).

References

1. Afrati, F.N., Kolaitis, P.G.: Answering aggregate queries in data exchange. In: PODS, pp. 129–138 (2008)
2. Aliseda-Llera, A.: Seeking explanations: abduction in logic, philosophy of science and artificial intelligence. PhD thesis, Stanford, CA, USA (1998), UMI Order No. GAX98-10072
3. Arenas, M., Barceló, P., Fagin, R., Libkin, L.: Locally consistent transformations and query answering in data exchange. In: PODS, pp. 229–240 (2004)
4. Arenas, M., Barcelo, P., Libkin, L., Murlak, F.: Relational and XML Data Exchange, 1st edn. Morgan and Claypool Publishers (2010)
5. Beth, E.: On Padoa's method in the theory of definition. Indagationes Mathematicae 15, 330–339 (1953)
6. Fagin, R., Haas, L.M., Hernández, M., Miller, R.J., Popa, L., Velegrakis, Y.: Clio: Schema Mapping Creation and Data Exchange. In: Borgida, A.T., Chaudhri, V.K., Giorgini, P., Yu, E.S. (eds.) Conceptual Modeling: Foundations and Applications. LNCS, vol. 5600, pp. 198–236. Springer, Heidelberg (2009)
7. Fagin, R., Kolaitis, P.G., Miller, R.J., Popa, L.: Data exchange: semantics and query answering. Theor. Comput. Sci. 336(1), 89–124 (2005)
8. Hernich, A.: Answering non-monotonic queries in relational data exchange. In: ICDT, pp. 143–154 (2010)
9. Libkin, L.: Data exchange and incomplete information. In: PODS, pp. 60–69 (2006)
10. Nash, A., Segoufin, L., Vianu, V.: Views and queries: Determinacy and rewriting. ACM Trans. Database Syst., 35, 21:1–21:41 (2010)
11. Paul, G.: Approaches to abductive reasoning: An overview. AI Review 7, 109–152 (1993)

Rule-Based Context Assessment in Smart Cities

Beibei Hu, Theodore Patkos, Abdelghani Chibani, and Yacine Amirat

Lissi Laboratory, University of Paris-Est Creteil (UPEC), Creteil, France
{beibei.hu,theodore.patkos,chibani,amirat}@u-pec.fr

Abstract. This paper presents a rule-based architecture that enables causal and temporal reasoning of events and supports their relevance assessment given the user's situation, in order to provide contextualized services for citizens inhabiting a smart city. Our approach for context reasoning and assessment is illustrated by emergency scenarios.

Keywords: rule-based reasoning, context-awareness, context assessment.

1 Introduction

Urban environments are highly dynamic, with unexpected incidents, such as car collisions, arising at any time. People affected by such incidents act in different roles (e.g., citizens and first responders) and need to receive information relevant to the current context at real-time, in order to perform effective decision-making. Urban ecosystems are thus required to exploit the semantics of events and create urban-specific Linked Data. To facilitate the context representation and reasoning, the integration of Semantic Web technologies for realizing an adaptable context-aware infrastructure has attracted much research attention. For example, the MobiSem framework is designed specifically to operate on mobile systems [4], the software architecture of GEPSIR aims to manage the situation awareness [1], and the DBpedia Mobile provides a location-enabled linked data browser[1]. Rule-based architectures are explored, building on their flexibility and their simplicity in specifying the behavior of a system. However, few approaches exploit the causal properties of the events that comprise an incident or assess the relevance of information that reaches interested entities given urban situations.

In the context of the European PEOPLE project[2] that aims to realize smart cities, our research focuses on reasoning and assessing the relevancy of information for users involved in critical urban situations. Our contributions in this paper are two-folds: (1) from an application perspective, we design an architecture for the provision of information relevant to the current situation of users acting in different roles; and (2) from a technical perspective, we propose a rule-based approach for information relevance assessment given domain descriptions axiomatized as Event Calculus declarative rules.

[1] wiki.dbpedia.org/DBpediaMobile
[2] http://www.people-project.eu

2 Motivating Scenarios

Assume a smart city, where bus stations are equipped with sensors to track the physical context in real-time. To handle the incidents, the command center operators use a system that connects to sensors and standard databases.

Scenario I – Citizen Rescue. At the Paul Armangot bus stop around 5pm, an old man has suddenly fallen down and remains unconscious. One emergency ambulance is sent to rescue this citizen. According to the fact that the incident occurs near the UPEC university around closing time, the system infers that there will be a large crowd of students on the street, and further decides to send a traffic alert to the approaching ambulance. After ambulance arrival, the system searches the medical records of the patient with his identification provided by a witness and sends a heart attack record to the ambulance crews.

Scenario II – Collision Incident. Two buses collide at the intersection of two avenues, resulting in a traffic jam. This incident is reported by several sensors installed at different bus stations, providing data from specific observation scopes. The system can recognize that those different sensor data refers to the same event and avoids sending redundant information.

3 A Rule-Based Architecture

The same piece of information may have different levels of significance for people acting in different roles in the same urban space. Urban ecosystems, flooded by a bulk of events arriving from different sources, need to recognize important information given user's context. Our research hypothesis consists in going beyond the location-related information and exploiting the causal relationships among events, reasoning on how context affects their consequences and understating the common sense behind the pattern of event occurrences. We design a rule-based architecture as shown in Figure 1, consisting of a contextual information store, the Casual and Temporal Reasoner (CTR) and the Relevance Assessment Rule Engine (RARE). Exploiting the knowledge produced by CTR, the relevance of information can be assessed by RARE in a given context. The main function of each component is explained below:

Contextual Information Store. It stores low-level sensor context, high-level context from CTR, background information (e.g., medical records), and linked open data (e.g., school time schedules). The relationships between context dimensions, including user, activity, event, information items, time and location, are represented in our ontology-based context model, partially shown in Figure 1. Specifically, the User class has subclasses for defining the role of first responders and citizens; the Event class, subclass of which is the initiating incident, refers to both atomic and compound events; the Information Item class is related to the Incident class and provides high-level context inferred from CTR.

Causal and Temporal Reasoner. To formally characterize the causal and temporal properties of events and the way they affect the environment we apply Event Calculus axiomatizations. The Event Calculus [3] is an influential formalism that can be used to reason about action, change and time for a wide range of

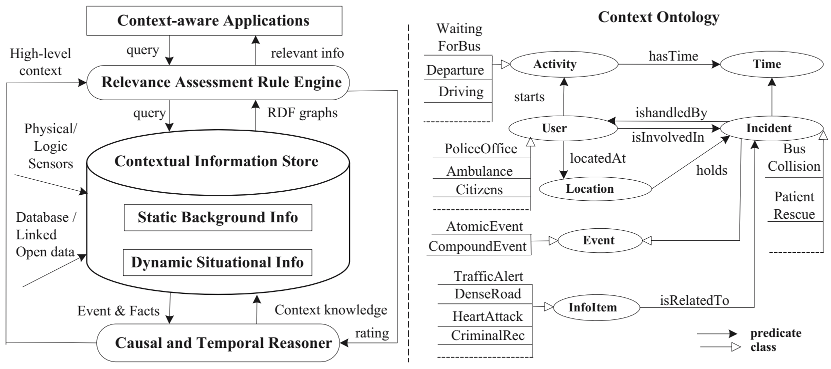


Fig. 1. A Rule-based Architecture

common sense phenomena. A set of axioms describes the properties of a domain and enables the modeling and derivation of the direct context-dependent effects of events, as well as their ramifications and qualifications. The contribution of CTR is twofold. First, the determination of which emergency entities to contact is decided, creating a connection between an occurring incident and the "Role" concept of the ontology. Second, the derivation of high-level context knowledge is attempted, given dynamic information from sensors and available urban-related data, such as hour schedules of nearby activities, medical profiles of involved persons if available etc. These are used to categorize the type of incident and to better characterize its features. For example, axioms (1), (2) below trigger an alert to inform about road activity conditions, whenever an incident occurs near a university around closing time (scenario I):

(1) $\neg$ HoldsAt(PublicHoliday($?day$),$?t$) $\Rightarrow$
Initiates(UniversityEnd(IUT,$?day$),DenseRoadActivity(RoadSegmentX),$?t$)

(2) Happens(Incident($?e$),$?t1$) $\wedge$ HoldsAt(LocatedAt($?e$,Loc1),$?t1$) $\wedge$
HoldsAt(DenseRoadActivity(Loc2),$?t2$) $\wedge$ ($?t1$-$?t2 \leq$ 15min)$\wedge$
HoldsAt(Near(Loc1,Loc2),$?t1$) $\Rightarrow$ Happens(InformativeEvent(Traffic,Loc2),$?t1$)

Notice that although the university closure occurs at a future time, the alert is triggered at the current time, in order to enable the RARE component evaluate its relevancy to the entities assigned to handle the incident. To accommodate reasoning tasks given such a scheme, we have designed a forward-chaining production system that can perform causal and temporal reasoning both offline and online. This component is implemented on top of Jess, an efficient rule engine that uses an enhanced version of the Rete algorithm to process rules. CTR combines the declarative semantics of the Event Calculus with features that enhance efficient online reasoning, such as rule-based operational semantics, semi-destructive assignment, negation-as-failure (NaF) and others.

Relevance Assessment Rule Engine. A rule engine (RARE) [2] is implemented for relevancy evaluation of information items inferred from CTR by taking into account conditions, such as the spatio-temporal relation between an event and a user's activity. Unlike SWRL that can infer facts, the Context

Rating Rule Language (CRRL) is defined for generating a cumulative rating for a given user/info pair. The body of CRRL rule contains a list of conditions that are combined together with logical operators such as the logical conjunction "&". Two special variables are appointed in the rule body: $?info$ and $?user$. The head of rule, with a numerical rating level in line with users' judgements from our user studies [2], specifies how the rating of a given item ($?info$) is updated for a given user ($?user$). Let us consider two rules for scenario I, for the same type of item – Traffic Alert, R1 specifies the relevance needed to be increased "+5.0" for the ambulance; while R2 specifies that the relevance should be decreased "-1.0" for the citizens. To deal with scenario II, certain rules are defined to decrease the rating, in case that a certain item has been presented.

R1: **type**($?info$, TrafficAlert) & **happens**($?info$, $?aleloc$) & **nearBy** ($?aleloc$, $?eveloc$) & **holds**($?eveloc$, $?event$) & **isHandledBy**($?event$, $?user$) & **starts**($?user$, DepartureActivity) & **hasRole**($?user$, Ambulance) ⇒ ***updateRating*** (+5.0)

R2: **type**($?info$, TrafficAlert) & **hasRole**($?user$, Citizen) ⇒ ***updateRating*** (−1.0)

RARE is triggered to execute CRRL rules once context is updated. For each rule, RARE translates the rule body into a SPARQL query which is evaluated by the Sesame query engine, and computes the relevance rating indicated in the rule head. All the translated rules interact together to increase or decrease the relevance, adding up the effect to produce a final ranking in a given context. To update the relevance, all the triples specifying the current values are firstly removed and then SPARQL construct queries are executed to create triple patterns with new values.

4 Conclusion and Future Work

We have presented a rule-based architecture for contextualized information provision in a smart city. Next, we will define a mechanism, which allows the relevance ratings generated by RARE to influence the weight of context knowledge inferred by CTR. We will implement a prototype and evaluate its behavior based on our scenarios. By measuring the results on different datasets, we will further investigate how our approach is feasible for providing relevant information to support user's urban activities.

References

1. Cimino, M.G., Lazzerini, B., Marcelloni, F., Ciaramella, A.: An adaptive rule-based approach for managing situation-awareness. Expert Systems with Applications 39(12), 10796–10811 (2012)
2. Hu, B.B.: Towards Contextualized Information Delivery: A Rule-based Architecture for the Domain of Mobile Police Work. PhD thesis, Delft University of Technology, Delft, The Netherlands (2011)
3. Kowalski, R., Sergot, M.: A logic-based calculus of events. New Generation Computing 4(1), 67–95 (1986)
4. Zander, S., Schandl, B.: Semantic web-enhanced context-aware computing in mobile systems: Principles and application. In: Mobile Computing Techniques in Emerging Markets: Systems, Applications and Services, pp. 47–96 (2012)

Recent Advances in Integrating OWL and Rules (Technical Communication)

Matthias Knorr[2], David Carral Martínez[1], Pascal Hitzler[1], Adila A. Krisnadhi[1], Frederick Maier[3], and Cong Wang[1]

[1] Kno.e.sis Center, Wright State University, U.S.A.
[2] CENTRIA, Universidade Nova de Lisboa, Portugal
[3] Aston Business School, Aston University, UK

Abstract. As part of the quest for a *unifying logic* for the Semantic Web Technology Stack,[1] a central issue is finding suitable ways of integrating description logics based on the Web Ontology Language (OWL) with rule-based approaches based on logic programming. Such integration is difficult since naive approaches typically result in the violation of one or more desirable design principles. For example, while both OWL 2 DL and RIF Core (a dialect of the Rule Interchange Format RIF) are decidable, their naive union is not, unless carefully chosen syntactic restrictions are applied.

We report on recent advances and ongoing work by the authors in integrating OWL and rules. We take an OWL-centric perspective, which means that we take OWL 2 DL as a starting point and pursue the question of how features of rule-based formalisms can be added without jeopardizing decidability. We also report on incorporating the closed world assumption and on reasoning algorithms. This paper essentially serves as an entry point to the original papers, to which we will refer throughout, where detailed expositions of the results can be found.

1 Rule-Extensions of OWL

In [4], Grosof et al. describe a fragment of the description logic $\mathcal{SHOIN}$ (a.k.a. OWL 1 DL) which, if syntactically transferred to first-order predicate logic (FOL) in a straightforward way, results in a set of function-free Horn clauses, i.e. a Datalog program under FOL semantics. This naive approach has been subsequently lifted to OWL 2 DL and given rise to the OWL 2 RL fragment [10]. This work does not, however, address the problem of identifying the rules of Datalog (under FOL semantics) expressible in OWL and its variants, and indeed recent results, including the work on description logic rules by Krötzsch et al. [7], show that OWL 2 RL can be improved significantly in this respect.

To formulate the recent findings, we first note that a directed graph G_r can be constructed from any given binary Datalog rule r, i.e. a rule containing only unary and binary predicates. The nodes of G_r are the variables occurring in the rule body of r, and there is exactly one directed edge between two variables x and y if there is at least one binary atom of the form $P(x, y)$ appearing in the

[1] http://www.w3.org/2007/03/layerCake.png

body of r. The following results then hold,[2] where z is the variable in the first argument of the head atom.

- If G_r is a tree with root z, then r can be expressed in $\mathcal{SROEL}$ (OWL 2 EL) [7].
- If G_r, with any edges inverted, is a tree with root z, then r can be expressed in $\mathcal{SROIEL}$ [7].
- If G_r, with edges considered undirected, does not contain four nodes which are path-connected by mutually disjoint paths in such a way that they constitute a 4-clique, then r can be expressed in $\mathcal{SROIEL}(\sqcap)$, i.e., in $\mathcal{SROIEL}$ extended by role conjunction [1].

The results above are based on the idea of retaining decidability by syntactically restricting the rules which are allowed to be used together with a DL knowledge base. A complementary line of work is based on the idea of weakening the semantics of rules in a suitable way. This was first voiced in the notion of DL-safe rules [13], which are rules in which the variables can bind only to known individuals, i.e. to constants present in the knowledge base, resulting in so-called *DL-safe SWRL*. This approach was then generalized in [9] in such a way that only *some* variables in rules—called *DL-safe variables*—were restricted this way. In [8], Krötzsch et al. ported this concept to description logics, resulting in a new syntactic construct called *nominal schemas*.

Nominal schemas can be understood as variable nominals. Syntactically, this new construct $\{x\}$ resembles a nominal, save that x is a variable rather than an individual, and it can only bind to individuals appearing in the knowledge base such that each occurrence of the nominal schema within one axiom is bound to the same individual. Semantically, this is realized by extending the interpretation with a first-order variable assignment binding variables to domain elements named by individuals in the knowledge base [8].

A DL extended with nominal schemas not only completely covers DL-safe SWRL, it also makes it possible to completely express any Datalog program under the Herbrand semantics—without any restriction on arities of predicates or on forms of rules [5,8]. Furthermore, $\mathcal{SROIQ}$ extended with nominal schemas, called $\mathcal{SROIQV}$, is of the same computational complexity as $\mathcal{SROIQ}$ [8].

So far, only monotonic rules are considered, despite the fact that the closed world assumption is often requested in order to be able to model defaults, exceptions, and integrity constraints. Following the spirit of description logics of minimal knowledge and negation as failure (MKNF) [3], two modal operators **K** and **A** are added to $\mathcal{SROIQV}$, yielding a more expressive yet still uniform formalism [5]. The two operators allow the inspection of the knowledge base, i.e. **K** represents minimal knowledge, while **A** is interpreted as autoepistemic assumption and corresponds to ¬**not**, where **not** is identical with default negation in non-monotonic rules. As is common in MKNF semantics, a set of interpretations is used instead of one interpretation, and the non-monotonic semantics is defined based on a preference relation among such sets, minimizing derivable knowledge.

This language trivially covers $\mathcal{SROIQV}$ (hence $\mathcal{SROIQ}$, the tractable OWL 2 profiles, and arbitrary Datalog rules as pointed out above), and $\mathcal{ALCK}_{\mathcal{NF}}$

[2] Some of these statements can be improved, as detailed in the indicated papers.

[3]. Thus, default reasoning, epistemic queries, closure of roles and concepts, and integrity constraints are available in the language. It also covers Hybrid MKNF [12], a tight integration of DLs and non-monotonic rules based on MKNF logics. Indeed, it is the first approach that covers the two distinct MKNF-based formalisms, [3] and [12]. Moreover, a decidable fragment of the full language is identified in [5], which contains most of the covered languages.

2 Algorithms for Reasoning with Nominal Schemas

There is a naive way of algorithmizing reasoning with nominal schemas, which we call *full grounding*: Replace each axiom with all *grounded* axioms, where nominal schemas are replaced by nominals, in all possible combinations which respect variable bindings. This yields a semantically equivalent knowledge base without nominal schemas, and a traditional reasoning algorithm can then be used. While this approach permits reasoning with nominal schemas, it is problematic in the sense that it is combinatorially explosive in cases involving axioms having many nominal schemas [2]. We have therefore started to investigate alternative approaches which ground nominal schemas in a dynamic fashion, thus reducing the overhead of full grounding. We briefly report on the preliminary findings of this ongoing work.

An alternative approach for dealing with nominal schemas is to extend standard tableau algorithms with grounding rules in such a way that grounding can be delayed until *required* [6]. In the following situations, grounding a nominal schema $\{x\}$ occurring within a concept C in the label of a tableau node is required before any standard tableau rules can be applied: (i) when $C = D \sqcup E$ and $\{x\}$ occurs in both D and E; (ii) when $C = \exists R.D$ and $\{x\}$ occurs in the top level of D, e.g., $\exists R.(\{x\} \sqcap E)$ needs grounding whereas $\exists R.\exists S.(A \sqcap \{x\})$ does not. On the other hand, when C is a conjunction or a universal restriction, grounding can be safely delayed. Other details such as when grounding should be performed, which variables should be grounded first, or to which individual names they are grounded are left for an actual implementation to specify. Work is still in progress to extend this idea to other expressive constructors, such as qualified number restrictions.

In addition to intelligent grounding as in the tableau approach just described, can we avoid grounding from the beginning? In pursuit of this idea, we have also started to investigate the resolution calculus for algorithmization, where grounding is handled on the fly via unification. Previous work on general resolution for DLs [11] was unable to deal with role chains, as it introduces further complications with termination. We solved this problem by using ordered chaining rules such that the inferred clauses will not be longer than the premises. Nominal schemas add yet another complication to the termination issue as normal forms of globally limited size are no longer readily available. We successfully addressed this by using a lifting lemma to show that resolution on nominal schema axioms takes fewer resolution steps than performing resolution on fully grounded knowledge bases. Details can be found in [14].

Acknowledgement. This work was supported by the National Science Foundation under award 1017225 ”III: Small: TROn - Tractable Reasoning with Ontologies” and by Fundação para a Ciência e a Tecnologia under project “ERRO – Efficient Reasoning with Rules and Ontologies” (PTDC/EIA-CCO/121823/2010). The fourth author acknowledges the support of Fulbright Indonesia Presidential Scholarship PhD Grant 2010 – 2013.

References

1. Carral Martínez, D., Hitzler, P.: Extending Description Logic Rules. In: Simperl, E., Cimiano, P., Polleres, A., Corcho, O., Presutti, V. (eds.) ESWC 2012. LNCS, vol. 7295, pp. 345–359. Springer, Heidelberg (2012)
2. Carral Martínez, D., Krisnadhi, A., Maier, F., Sengupta, K., Hitzler, P.: Reconciling OWL and rules. Tech. rep., Kno.e.sis Center, Wright State University, Dayton, Ohio (2011), http://www.pascal-hitzler.de/
3. Donini, F.M., Nardi, D., Rosati, R.: Description logics of minimal knowledge and negation as failure. ACM Transactions on Computational Logic 3(2), 177–225 (2002)
4. Grosof, B., 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 (WWW 2003), pp. 48–57. ACM (2003)
5. Knorr, M., Hitzler, P., Maier, F.: Reconciling OWL and non-monotonic rules for the Semantic Web. In: Proceedings of the 20th European Conference on Artificial Intelligence, Montpellier, France, August 27-31 (to appear, 2012)
6. Krisnadhi, A., Hitzler, P.: A Tableau Algorithm for Description Logics with Nominal Schema. In: Krötzsch, M., Straccia, U. (eds.) RR 2012. LNCS, vol. 7497, pp. 234–237. Springer, Heidelberg (2012)
7. Krötzsch, M., Rudolph, S., Hitzler, P.: Description logic rules. In: Ghallab, M., Spyropoulos, C.D., Fakotakis, N., Avouris, N.M. (eds.) Proceedings of the 18th European Conference on Artificial Intelligence, Patras, Greece, July 21-25, pp. 80–84. IOS Press, Amsterdam (2008)
8. Krötzsch, M., Maier, F., Krisnadhi, A.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 (WWW 2011), pp. 645–654. ACM (2011)
9. Krötzsch, M., Rudolph, S., Hitzler, P.: ELP: Tractable Rules for OWL 2. In: Sheth, A.P., Staab, S., Dean, M., Paolucci, M., Maynard, D., Finin, T., Thirunarayan, K. (eds.) ISWC 2008. LNCS, vol. 5318, pp. 649–664. Springer, Heidelberg (2008)
10. Motik, B., Cuenca Grau, B., Horrocks, I., Wu, Z., Fokoue, A., Lutz, C. (eds.): OWL 2 Web Ontology Language: Profiles. W3C Recommendation (October 27, 2009), http://www.w3.org/TR/owl2-profiles/
11. Motik, B.: Reasoning in Description Logics using Resolution and Deductive Databases. Ph.D. thesis, Univesität Karlsruhe (TH), Karlsruhe, Germany (2006)
12. Motik, B., Rosati, R.: Reconciling Description Logics and Rules. Journal of the ACM 57(5), 93–154 (2010)
13. Motik, B., Sattler, U., Studer, R.: Query answering for OWL-DL with rules. Journal of Web Semantics 3(1), 41–60 (2005)
14. Wang, C., Hitzler, P.: A tractable resolution procedure for $\mathcal{SROELV}_n(\sqcap_s, \times)$. Tech. rep., Kno.e.sis Center, Wright State University, Dayton, Ohio (2012), http://www.pascal-hitzler.de/

Building Virtual Earth Observatories Using Ontologies and Linked Geospatial Data*

Manolis Koubarakis[1], Manos Karpathiotakis[1], Kostis Kyzirakos[1], Charalampos Nikolaou[1], Stavros Vassos[1], George Garbis[1], Michael Sioutis[1], Konstantina Bereta[1], Stefan Manegold[2], Martin Kersten[2], Milena Ivanova[2], Holger Pirk[2], Ying Zhang[2], Charalampos Kontoes[3], Ioannis Papoutsis[3], Themistoklis Herekakis[3], Dimitris Mihail[4], Mihai Datcu[5], Gottfried Schwarz[5], Octavian Dumitru[5], Daniela Molina[5], Katrin Molch[5], Ugo Giammatteo[6], Manuela Sagona[6], Sergio Perelli[6], Eva Klien[7], Thorsten Reitz[7], and Robert Gregor[7]

[1] National and Kapodistrian University of Athens
[2] Centrum Wiskunde & Informatica
[3] National Observatory of Athens
[4] Harokopio University of Athens
[5] German Aerospace Center (DLR)
[6] Advanced Computer Systems
[7] Fraunhofer Institute for Computer Graphics Research

1 Introduction

Advances in remote sensing technologies have enabled public and commercial organizations to send an ever-increasing number of satellites in orbit around Earth. As a result, Earth Observation (EO) data has been constantly increasing in volume in the last few years, and is currently reaching petabytes in many satellite archives. For example, the multi-mission data archive of the TELEIOS partner German Aerospace Center (DLR) is expected to reach 2 PB next year, while ESA estimates that it will be archiving 20 PB of data before the year 2020. As the volume of data in satellite archives has been increasing, so have the scientific and commercial applications of EO data. Nevertheless, it is estimated that up to 95% of the data present in existing archives has never been accessed, so the potential for increasing exploitation is very big.

TELEIOS[1] is a recent European project that addresses the need for scalable access to PBs of Earth Observation data and the effective discovery of knowledge hidden in them. TELEIOS started on September 2010 and it will last for 3 years. Up to now, the project has made significant progress in the development of state-of-the-art techniques in Scientific Databases, Semantic Web and Image Mining and have applied them to the management of EO data.

In the rest of this technical communication we outline the contributions of TELEIOS, and explain why it goes significantly beyond operational systems

* This work has been funded by the FP7 project TELEIOS (257662).
[1] http://www.earthobservatory.eu/

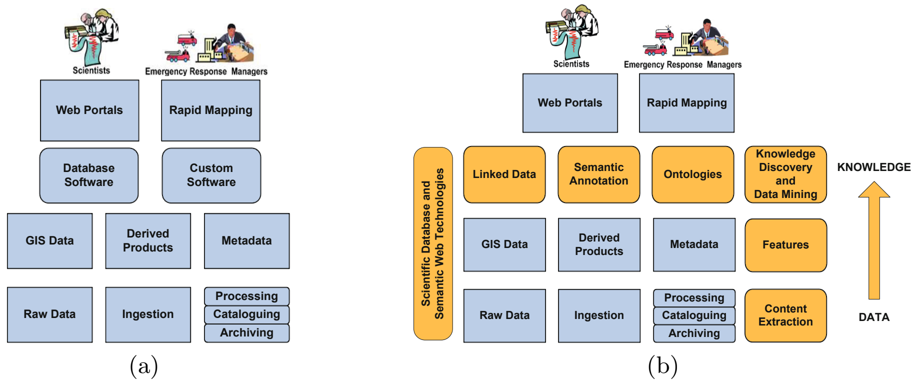


Fig. 1. Pre-TELEIOS EO data centers and the Virtual Earth Observatory

currently deployed in various EO data centers and Earth Observation portals such as EOWEB-NG. We also present its main technical contributions related to ontologies and linked geospatial data.

2 Basic Concepts of the Virtual Earth Observatory

Satellite missions continuously send to Earth huge amounts of EO data providing snapshots of the surface of the Earth or its atmosphere. The management of the so-called *payload data* is an important activity of the ground segments of satellite missions. Figure 1(a) gives a high-level view of some of the basic data processing and user services available at EO data centers today, e.g., at the German Remote Sensing Data Center (DFD) of TELEIOS partner DLR through its Data Information and Management System (DIMS).

Raw data, often from multiple satellite missions, is ingested, processed, catalogued and archived. Processing results in the creation of various *standard products* (Level 1, 2, etc., in EO jargon; raw data is Level 0) together with extensive metadata describing them. For example, in the NOA application of TELEIOS [2], images from the SEVIRI sensor are processed (cropped, georeferenced and run through a pixel classification algorithm) to detect pixels that are hotspots. Then these pixels are stored as standard products in the form of shapefiles. Raw data and derived products are complemented by *auxiliary data*, e.g., various kinds of geospatial data such as maps, land use/land, and cover data.

Raw data, derived products, metadata and auxiliary data are stored in various storage systems and are made available using a variety of policies depending on their volume and expected future use. For example, in the TerraSAR-X archive managed by DFD, long term archiving is done using a hierarchy of storage systems (including a robotic tape library) which offers batch to near-line access, while product metadata are available on-line by utilizing a relational DBMS and an object-based query language.

EO data centers such as DFD also offer a variety of user services. For example, for scientists that want to utilize EO data in their research, DFD offers the Web interface EOWEB-NG for searching, inspection, and ordering of products. Space agencies such as DLR and NOA might also make various other services available aimed at specific classes of users. For example, the Center for Satellite Based Crisis Information (ZKI) of DLR provides a 24/7 service for the rapid provision, processing and analysis of satellite imagery during natural and environmental disasters, for humanitarian relief activities and civil security issues worldwide. Similar emergency support services for fire mapping and damage assessment are offered by NOA through its participation in the GMES SAFER program.

We now summarize the TELEIOS advancements to today's state of the art in EO data processing that are shown graphically with yellow color in Figure 1(b). Firstly, traditional raw data processing is augmented by *content extraction* methods that deal with the specificities of satellite images and derive image descriptors (e.g., texture features, spectral characteristics of the image). Knowledge discovery techniques combine image descriptors, image metadata and auxiliary data (e.g., GIS data) to determine concepts from a domain ontology (e.g., forest, lake, fire, burned area) that characterize the content of an image [1]. Hierarchies of domain concepts are formalized using OWL ontologies and are used to annotate standard products. Annotations are expressed in RDF and are made available as linked data so that they can be easily combined with other publicly available linked data sources (e.g., GeoNames, LinkedGeoData, DBpedia) to allow for the expression of rich user queries.

Web interfaces to EO data centers and specialized applications (e.g., rapid mapping) can now be improved significantly by exploiting the semantically-enriched standard products and linked data sources made available by TELEIOS. For example, the advanced query builder for EO data archives that is now being developed in TELEIOS based on the data model stRDF and the query language stSPARQL can enable end-users to pose easily very expressive queries.

stRDF is an extension of the W3C standard RDF that allows the representation of geospatial data that changes over time [3,5]. stRDF is accompanied by stSPARQL, an extension of the query language SPARQL 1.1 for querying and updating stRDF data. stRDF and stSPARQL use OGC standards (WKT and GML) for the representation of temporal and geospatial data [5,4]. stRDF and stSPARQL have been implemented in the system Strabon which is freely available as open source software[2]. Strabon extends the well-known open source RDF store Sesame 2.6.3 and uses PostGIS as the backend spatially-enabled DBMS. Recent work on geospatial extensions to SPARQL has also resulted in the creation of an OGC standard for querying geospatial data encoded in RDF, called GeoSPARQL [7], which is a superset of stSPARQL. To the best of our knowledge, Strabon and the implementation of GeoSPARQL Parliament[3] are currently the RDF stores offering the richest functionality regarding geospatial data.

In TELEIOS, stRDF is used to represent satellite image metadata (e.g., time of acquisition, geographical coverage), knowledge extracted from satellite images (e.g., a certain image pixel is a fire hotspot), and auxiliary geospatial data sets

[2] http://www.strabon.di.uoa.gr/
[3] http://parliament.semwebcentral.org/

encoded as linked data. One can then use stSPARQL to express in a single query an information request such as the following: "Find an image taken by a Meteosat second generation satellite on August 25, 2007 which covers the area of Peloponnese and contains hotspots corresponding to forest fires located within 2 km from a major archaeological site". Encoding this information request today in a typical interface to an EO data archive such as EOWEB-NG is impossible, because domain-specific concepts such as "forest fires" are not included in the archive metadata, thus they cannot be used as search criteria. In EOWEB-NG and other similar Web interfaces, search criteria include a hierarchical organization of available products (e.g., high resolution optical data, Synthetic Aperture Radar data) together with a temporal and geographic selection menu.

We have been developing image information mining techniques that allow us to characterize satellite image regions with concepts from appropriate ontologies (e.g., landcover ontologies with concepts such as water-body, lake, and forest, or environmental monitoring ontologies with concepts such as forest fires, and flood) [1,6]. These concepts are encoded in OWL ontologies and are used to annotate EO products. Thus, we attempt to close the semantic gap that exists between user requests and searchable information available explicitly in the archive.

But even if semantic information was included in the archived annotations, one would need to join it with information obtained from auxiliary data sources to answer the above query. Although such open sources of data are available to EO data centers, they are not used currently to support sophisticated ways of end-user querying in Web interfaces such as EOWEB-NG. In TELEIOS, we assume that auxiliary data sources, especially geospatial ones, are encoded in stRDF and are available as linked geospatial data, thus stSPARQL can easily be used to express information requests such as the above. The linked data web is being populated with geospatial data quickly, thus we expect that languages such as stSPARQL and GeoSPARQL [7] will soon be mainstream extensions of SPARQL that can be used to access such data effectively.

References

1. Dumitru, C.O., Molina, D.E., Cui, S., Singh, J., Quartulli, M., Datcu, M.: KDD concepts and methods proposal: report & design recommendations. Del. 3.1, FP7 project TELEIOS (2011)
2. Kontoes, H., Papoutsis, I., Michail, D.: Requirements specification for the real-time fire monitoring application. Del. D7.1, FP7 project TELEIOS (2012)
3. Koubarakis, M., Kyzirakos, K.: Modeling and Querying Metadata in the Semantic Sensor Web: The Model stRDF and the Query Language stSPARQL. In: Aroyo, L., Antoniou, G., Hyvönen, E., ten Teije, A., Stuckenschmidt, H., Cabral, L., Tudorache, T. (eds.) ESWC 2010. LNCS, vol. 6088, pp. 425–439. Springer, Heidelberg (2010)
4. Koubarakis, M., Kyzirakos, K., Karpathiotakis, M.: Data models, Query Languages, Implemented Systems and Applications of Linked Geospatial Data. In: Tutorials at ESWC (2012), http://www.strabon.di.uoa.gr/tutorial-eswc-2012

5. Koubarakis, M., Kyzirakos, K., Nikolaou, B., Sioutis, M., Vassos, S.: A data model and query language for an extension of RDF with time and space. Del. 2.1, FP7 project TELEIOS (2011)
6. Koubarakis, M., Sioutis, M., Kyzirakos, K., Karpathiotakis, M., Nikolaou, C., Vassos, S., Garbis, G., Bereta, K., Dumitru, O.C., Molina, D.E., Molch, K., Schwarz, G., Datcu, M.: Building Virtual Earth Observatories using Ontologies, Linked Geospatial Data and Knowledge Discovery Algorithms. In: ODBASE (2012)
7. OGC: GeoSPARQL - A geographic query language for RDF data (November 2010)

A Tableau Algorithm for Description Logics with Nominal Schema[⋆]

Adila Krisnadhi[⋆⋆] and Pascal Hitzler

Kno.e.sis Center, Wright State University, Dayton OH
{adila,pascal}@knoesis.org

Nominal schema is an expressive description logic (DL) construct that was proposed in recent efforts to integrate DLs and (logic programming) rule-based paradigms for the Semantic Web [1] represented by two "diverging" W3C[1] standards: the DL-based Web Ontology Language (OWL) [2] whose major variant, OWL 2 DL, is based on the description logic (DL) $\mathcal{SROIQ}$ [3]; and the rule-based Rule Interchange Format (RIF) whose core variant, called RIF Core [4], is essentially Datalog, i.e., function-free Horn logic.

Nominal schema is introduced to description logics (DLs) as a way to express DL-safe rules [5] with predicates of arbitrary arity within the DL syntax [6,7]. Syntactically, a nominal schema is a nominal-like expression $\{x\}$ with x a variable, instead of an individual name. Semantically, each occurrence of the same nominal schema within one axiom binds to the same individual and the axiom is then understood as a macro that represents all axioms obtained by completely instantiating the variables. This makes DLs with nominal schemas decidable, in fact, adding nominal schemas to the DL $\mathcal{SROIQ}$ does not increase the worst-case complexity [6]. This decidability result, however, hinged upon a straightforward but inefficient algorithmization based on *full grounding upfront* which induces an exponential blow up already at the beginning of reasoning.

In this paper, we explore one alternative for full grounding upfront by adapting a standard tableau algorithm for the DL $\mathcal{ALCOV}$, i.e., $\mathcal{ALCO}$ plus nominal schemas, with for a *delayed grounding* strategy. The rationale is that by delaying the grounding until it is absolutely necessary to do so we can minimize the blow up compared to performing it fully at the beginning of reasoning. Due to space restrictions, we refer the reader to [8] for the corresponding formal definitions, proofs, and other technical details.

A Tableau Algorithm for DL with Nominal Schemas

We now briefly describe tableau-based algorithm for the DL $\mathcal{ALCOV}$ which, given an $\mathcal{ALCOV}$ KB $\mathcal{K}$, determines whether $\mathcal{K}$ is satisfiable. Since nominals are allowed, we assume that the ABox part of $\mathcal{K}$ is internalized: its axioms

[⋆] This work is supported by the National Science Foundation (NSF) project "TROn – Tractable Reasoning with Ontologies" under the award 1017225 III: Small.

[⋆⋆] The first author acknowledges the support of Fulbright Indonesia Presidential Scholarship PhD Grant. 2010–2013.

[1] World Wide Web Consortium, http://www.w3.org/

are rewritten into equivalent TBox axioms. The general framework remains the same as in standard tableau algorithms for other DLs. The algorithm works on the so-called *completion graph* whose nodes (resp. edges) are labeled with concepts (resp. roles) occurring in the KB. The nodes themselves are divided into nominal nodes, whose label contains some nominal, and blockable nodes whose label contains no nominal.

Suppose that there are n nominals $\{a_1\}, \ldots, \{a_n\}$ occurring in $\mathcal{K}$. Then, the completion graph is initialized with n nominal nodes $r_1, \ldots, r_n$ where the label $\mathsf{L}(r_i) \coloneqq \{\{a_i\}, \top\}$, $i = 1, \ldots, n$. After the initialization, tableau rules are then applied exhaustively to the graph until no rule is applicable (in which the graph is said to be *complete*) or a *clash* occurs, i.e., when there is some node r such that $\bot \in \mathsf{L}(v)$ or $\{C, \neg C\} \subseteq \mathsf{L}(r)$ where C is either a concept name or a nominal. If the algorithm terminates and no clash occurs, then $\mathcal{K}$ is satisfiable. Otherwise, $\mathcal{K}$ is unsatisfiable. As in standard tableau algorithms, blocking is employed to ensure termination. For $\mathcal{ALCOV}$, it suffices to employ *equality blocking* as defined here: a node r is blocked if r is blockable and one of the following holds: (i) either r has a blocked ancestor; or (ii) r has a blockable ancestor r' such that $\mathsf{L}(r) = \mathsf{L}(r')$ and the path between r and r' consists only of blockable nodes. Termination is a consequence of blocking and the fact that there are only finitely many possible labels to the nodes and edges in the graph. The latter holds partly because there can only be finitely many possible groundings of nominal schemas in a node's label. Soundness and completeness of this tableau algorithm are shown using standard technique of unraveling the completion graph.

$\mathcal{ALCOV}$ tableau rules are obtained from standard $\mathcal{ALCO}$ tableau rules [9], but sufficiently modified and extended with grounding rules to handle nominal schemas. Fig. 1 the adaptation of standard tableau rules for $\mathcal{ALCOV}$. Note that tableau rules for dealing with conjunction, universal restriction, nominals and adding TBox axioms to a node are omitted because those rules are the same as in standard tableau algorithms. On the other hand, rules for handling disjunction and existential restriction are modified, and grounding rules are added.

In Fig. 1, $\mathsf{Var}(C)$ is the set of all variables appearing (as nominal schemas) in C. The set $\mathsf{Tvar}(C)$ is defined as: $\mathsf{Tvar}(\{v\}) = \{\{v\}\}$ for any $v \in \mathsf{N}_V$; $\mathsf{Tvar}(\neg C) = \mathsf{Tvar}(C)$; $\mathsf{Tvar}(C \sqcap D) = \mathsf{Tvar}(C \sqcup D) = \mathsf{Tvar}(C) \cup \mathsf{Tvar}(D)$; $\mathsf{Tvar}(\forall R.C) = \mathsf{Tvar}(\exists R.C) = \emptyset$; and $\mathsf{Tvar}(C) = \emptyset$ if $\mathsf{Var}(C) = \emptyset$. In grounding rules, given a concept C, $C[v/a]$ is the concept obtained from C by grounding the nominal schema $\{v\}$ into the nominal $\{a\}$. The mapping M is used to store for each node r, all grounding pairs v/a that have been used earlier. All grounding rules non-deterministically choose a variable assignment that can be used to partially ground the corresponding concept. This provides some flexibility for incorporating some heuristics that allow an implementation to ground variables in such a way that a clash can be found as early as possible.

The intuition behind the grounding rules comes from the way concept expressions with nominal schemas are read as FOL formulas with equality. For example, consider a concept $C \coloneqq \exists R.(\{x\} \sqcap \exists S.\{y\})$ occurring in the label of some node s. Assume that $a, b \in \mathsf{N}_I$ appear in KB, so initialization implies the

$\sqcup$-rule: if: $C \sqcup D \in \mathsf{L}(r)$, $\mathsf{Var}(C) \cap \mathsf{Var}(D) = \emptyset$, and $\{C, D\} \cap \mathsf{L}(r) = \emptyset$
then: $\mathsf{L}(r) := \mathsf{L}(r) \cup \{C'\}$ for some $C' \in \{C, D\}$

$\exists^{\sqcup}$-rule: if: $\exists R.(C \sqcup D) \in \mathsf{L}(r)$, $\mathsf{Var}(C) \cap \mathsf{Var}(D) = \emptyset$,
then: $\mathsf{L}(r) := \mathsf{L}(r) \cup \{\exists R.C'\}$ for some $C' \in \{C, D\}$

$\exists$-rule: if: $\exists R.C \in \mathsf{L}(r)$, C is not a disjunction $C' \sqcup D'$, $\mathsf{Tvar}(C) = \emptyset$, r has no R-successor s with $C \in \mathsf{L}(s)$
then: create a node s with $\mathsf{L}(r, s) := \{R\}$ and $\mathsf{L}(s) := \{C\}$

grv-rule: if: (i) for some $v \in \mathsf{N}_V$ and some $a \in \mathsf{N}_I$, either $\{v\} \in \mathsf{L}(r)$ and $\{a\} \notin \mathsf{L}(r)$, or $\neg\{v\} \in \mathsf{L}(r)$ and $\neg\{a\} \notin \mathsf{L}(r)$; (ii) $v/a \notin \mathsf{M}(r)$;
then: $\mathsf{L}(r) := \mathsf{L}(r) \cup \{D[v/a] \mid D \in \mathsf{L}(r)\}$ and $\mathsf{M}(r) := \mathsf{M}(r) \cup \{v/a\}$

gr$\sqcup$-rule: if: (i) $C \sqcup D \in \mathsf{L}(r)$; (ii) $(C \sqcup D)[v/a] \notin \mathsf{L}(r)$ for some $v \in \mathsf{Var}(C) \cap \mathsf{Var}(D)$ and $a \in \mathsf{N}_I$; (iii) $v/a \notin \mathsf{M}(r)$;
then: $\mathsf{L}(r) := \mathsf{L}(r) \cup \{D[v/a] \mid D \in \mathsf{L}(r)\}$ and $\mathsf{M}(r) := \mathsf{M}(r) \cup \{v/a\}$

gr$\exists^{\sqcup}$-rule: if: (i) $\exists R.(C \sqcup D) \in \mathsf{L}(r)$; (ii) $\exists R.(C \sqcup D)[v/a] \notin \mathsf{L}(r)$ for some $v \in \mathsf{Var}(C) \cap \mathsf{Var}(D)$ and $a \in \mathsf{N}_I$; (iii) $v/a \notin \mathsf{M}(r)$;
then: $\mathsf{L}(r) := \mathsf{L}(r) \cup \{D[v/a] \mid D \in \mathsf{L}(r)\}$ and $\mathsf{M}(r) := \mathsf{M}(r) \cup \{v/a\}$

gr$\exists$-rule: if: (i) $\exists R.C \in \mathsf{L}(r)$ and C is not a disjunction; (ii) $\exists R.C[v/a] \notin \mathsf{L}(r)$ for some $v \in \mathsf{Tvar}(C)$ and $a \in \mathsf{N}_I$; (iii) $v/a \notin \mathsf{M}(r)$;
then: $\mathsf{L}(r) := \mathsf{L}(r) \cup \{D[v/a] \mid D \in \mathsf{L}(r)\}$ and $\mathsf{M}(r) := \mathsf{M}(r) \cup \{v/a\}$

gr-rule: if: (i) $C \in \mathsf{L}(r)$; (ii) $C[v/a] \notin \mathsf{L}(r)$, but $v/a \in \mathsf{M}(r)$ for some $v \in \mathsf{Var}(C)$ and some $a \in \mathsf{N}_I$;
then: $\mathsf{L}(r) := \mathsf{L}(r) \cup \{D[v/a] \mid D \in \mathsf{L}(r)\}$

Fig. 1. Tableau expansion rules for $\mathcal{ALCOV}$ with rules for conjunction, universal restriction, nominal and TBox axiom omitted. All rules require r to be not blocked

existence of two nodes s_a and s_b whose label resp. contains $\{a\}$ and $\{b\}$. The above concept can be read as the formula

$$\forall x.\forall y.\exists z'.(R(s, z') \wedge x = z' \wedge \exists z''.(S(z', z'') \wedge y = z'')) \tag{1}$$

where s is the meta-variable that represents a domain element for which the formula holds. The variables x, y and z are universally quantified in a special way: they range over the set of named individuals, i.e., not the whole domain. Thus, grounding the nominal schemas to every possible individual names is equivalent to dropping those universal quantifiers.

The *gr*$\exists$-rule grounds C to yield: $\exists R.(\{a\} \sqcap \exists S.\{y\})$ and $\exists R.(\{b\} \sqcap \exists S.\{y\})$ which can be read as: $\forall y \exists z'.(R(s, z') \wedge a = z' \wedge \exists z''.(S(z', z'') \wedge y = z''))$ and $\forall y \exists z'.(R(s, z') \wedge b = z' \wedge \exists z''.(S(z', z'') \wedge y = z''))$. The algorithm will process these two concepts in by generating two new nodes, each of which are then merged by the nominal rule to s_a and s_b. This makes both s_a's and s_b's labels contains $\exists S.\{y\}$. This situation is the same as having the following two formulas:

$$R(s, a) \wedge \forall y \exists z''.(S(a, z'') \wedge y = z'') \tag{2}$$

$$R(s, b) \wedge \forall y \exists z''.(S(b, z'') \wedge y = z'') \tag{3}$$

Clearly, the conjunction of (2) and (3) is equivalent to (1) if we interpret x to range only over named individuals. More importantly, notice that the universal quantifier that binds y now correctly moves "inside" because the existential quantifier that binds z' has been instantiated beforehand and all possible instantiations of z' must be equal to some named individuals which generate, in this case, two different formulas. Note that this cannot be done for the universal quantifier that binds x on (1) because even if z' is instantiated beforehand, there is only one formula which does not capture all possible instantiations of z'.

The above example also illustrates that grounding y can be done after ∃-rule is applied. This epitomises the delayed grounding strategy. A full grounding upfront would have fully grounded y (in addition to x) in the labeling of s. This results in four new labels, instead of two as in the above example. If this is followed by applications of ∃-rule, there will be four new individuals generated, although they will eventually be merged into two due to nominal rule. This unnecessary creation of new individuals is avoided if grounding y can be delayed.

References

1. Knorr, M., Martinez, D.C., Hitzler, P., Krisnadhi, A.A., Maier, F., Wang, C.: Recent Advances in Integrating OWL and Rules (Technical Communication). In: Krötzsch, M., Straccia, U. (eds.) RR 2012. LNCS, vol. 7497, pp. 225–228. Springer, Heidelberg (2012)
2. Hitzler, P., Krötzsch, M., Parsia, B., Patel-Schneider, P.F., Rudolph, S. (eds.): OWL 2 Web Ontology Language: Primer. W3C Recommendation, October, 27 (2009), `http://www.w3.org/TR/owl2-primer/`
3. Horrocks, I., Kutz, O., Sattler, U.: The even more irresistible SROIQ. In: Doherty, P., Mylopoulos, J., Welty, C. (eds.) Proceedings of the 10th International Conference on Principles of Knowledge Representation and Reasoning (KR 2006), pp. 57–67. AAAI Press (2006)
4. Boley, H., Hallmark, G., Kifer, M., Paschke, A., Polleres, A., Reynolds, D. (eds.): RIF Core Dialect. W3C Recommendation, June 22 (2010), `http://www.w3.org/TR/rif-core/`
5. Motik, B., Sattler, U., Studer, R.: Query answering for OWL DL with rules. Journal of Web Semantics 3(1), 41–60 (2005)
6. Krötzsch, M., Maier, F., Krisnadhi, A.A., Hitzler, P.: A better uncle for OWL: Nominal schemas for integrating rules and ontologies. In: Sadagopan, S., Ramamritham, K., Kumar, A., Ravindra, M., Bertino, E., Kumar, R. (eds.) Proceedings of the 20th International World Wide Web Conference, WWW 2011, Hyderabad, India, pp. 645–654. ACM, New York (March/April 2011)
7. Carral Martinez, D., Krisnadhi, A., Maier, F., Sengupta, K., Hitzler, P.: Reconciling OWL and rules. Technical report, Kno.e.sis Center, Wright State University, Dayton, Ohio, U.S.A. (2011), `http://www.pascal-hitzler.de/`
8. Krisnadhi, A., Hitzler, P.: A tableau algorithm for description logics with nominal schemas. Technical report, Kno.e.sis Center, Wright State University, Dayton, OH, U.S.A. (2012), `http://knoesis.wright.edu/researchers/adila/`
9. Horrocks, I., Sattler, U.: Ontology reasoning in the SHOQ(D) description logic. In: Proceedings of the 17th International Joint Conference on Artificial Intelligence (IJCAI), vol. 1, pp. 199–204 (2001)

Toward Scalable Reasoning over Annotated RDF Data Using MapReduce

Chang Liu[1] and Guilin Qi[2]

[1] Shanghai Jiaotong University, China
liuchang@apex.sjtu.edu.cn
[2] Southeast University, China
gqi@seu.edu.cn

1 Introduction

The Resource Description Framework (RDF) is one of the major representation standards for the Semantic Web. RDF Schema (RDFS) is used to describe vocabularies used in RDF descriptions.

Recently, there is an increasing interest to express additional information on top of RDF data. Several extensions of RDF were proposed in order to deal with time, uncertainty, trust and provenance. All these specific domains can be modeled by a general framework called *annotated RDF* data [3][5]. A recent work reported millions of triples with temporal information [1] and the number is still increasing. It is reasonable to expect more annotated RDF triples to be handled by semantic web applications. Therefore scalability will become an important issue for these applications.

Existing work [4] has shown that MapReduce is a scalable framework to perform large scale RDFS reasoning. This inspires us to solve the large scale annotated RDFS reasoning problem using MapReduce. We find that most of the optimizations for RDFS reasoning is also applicable for annotated RDFS reasoning. However, to reason with an arbitrary annotation domain, there are still some unique challenges that need to be handled. In this paper, we will discuss these challenges and solutions to tackle them. Our preliminary results show that our method is scalable for a specific domain: fuzzy domain.

2 Annotated RDFS Reasoning

2.1 Syntax

An annotated RDF triple is in form of $(s, p, o) : \lambda \in \mathbf{UBL} \times \mathbf{UB} \times \mathbf{UBL} \times D$[1]. Here $\mathbf{U}$, $\mathbf{B}$ and $\mathbf{L}$ are the sets of *URI references*, *blank nodes* and *literals* respectively. D is an *annotated domain* which is an idempotent commutative semi-ring $D = \langle L, \oplus, \otimes, \bot, \top \rangle$, where $\oplus$ is $\top$-annihilating, $\otimes$ is $\bot$-annihilating, and $\otimes$ is distributive over $\oplus$.

[1] Here we allow the predicate to be blank node to avoid the implicit typing rules.

2.2 Deductive System

Our discussion is based on the annotated extension of ρdf proposed in [5]. It is easy to extend to the full annotated RDFS semantics. For the annotated RDFS framework, there is a complete and sound rule set for an arbitrary annotation domain $D = \langle L, \oplus, \otimes, \perp, \top \rangle$. The rule set contains the following rules. Please note that the generalisation rule is destructive, *i.e.*, this rule removes the premises as the conclusion is inferred.

Table 1. Annotated RDFS Entailment Rules

Subproperty (a)	$(A, \mathtt{sp}, B) : \lambda_1, (B, \mathtt{sp}, C) : \lambda_2$	$(A, \mathtt{sp}, C) : \lambda_1 \otimes \lambda_2$
Subproperty (b)	$(D, \mathtt{sp}, E) : \lambda_1, (X, D, Y) : \lambda_2$	$(X, E, Y) : \lambda_1 \otimes \lambda_2$
Subclass (a)	$(A, \mathtt{sc}, B) : \lambda_1, (B, \mathtt{sc}, C) : \lambda_2$	$(A, \mathtt{sc}, C) : \lambda_1 \otimes \lambda_2$
Subclass (b)	$(A, \mathtt{sc}, B) : \lambda_1, (X, \mathtt{type}, A) : \lambda_2$	$(X, \mathtt{type}, B) : \lambda_1 \otimes \lambda_2$
Typing (a)	$(D, \mathtt{dom}, B) : \lambda_1, (X, D, Y) : \lambda_2$	$(X, \mathtt{type}, B) : \lambda_1 \otimes \lambda_2$
Typing (b)	$(D, \mathtt{range}, B) : \lambda_1, (X, D, Y) : \lambda_2$	$(Y, \mathtt{type}, B) : \lambda_1 \otimes \lambda_2$
Generalisation	$(X, A, Y) : \lambda_1, (X, A, Y) : \lambda_2$	$(X, A, Y) : \lambda_1 \oplus \lambda_2$

3 MapReduce Algorithm for Annotated RDFS Reasoning

3.1 Naive Implementation

We will use typing rule (a) to illustrate how to use MapReduce program to encode a rule. To apply this rule, we essentially need to perform a join between $(D, \mathtt{dom}, B) : \lambda_1$ and $(X, D, Y) : \lambda_2$. Algorithm 1 and Algorithm 2 provide the map and reduce function for this rule respectively. In this MapReduce program, the mappers scan all the annotated triples, and check for each triple if it has the form $(D, \mathtt{dom}, B) : \lambda_1$ or $(X, D, Y) : \lambda_2$. If so, the mapper will emit a key/value pair where the key is D and the value is the triple. Then the reducers collect all triple pairs, and output the derived results.

Algorithm 1. map function for typing rule (a)

```
Input: key, triple
1: if triple.predicate == 'dom' then
2:     emit({p=triple.subject}, {flag=0, u=triple.object, a=triple.annotation};
3: end if
4: emit({p=triple.predicate}, {flag=1, u=triple.subject, a=triple.annotation};
```

3.2 Challenges and Solutions

Such a naive implementation, however, suffers severe efficiency problem. For example, the naive join implementation will introduce an expensive shuffling stage; arbitrary attempts of rule applications will result in fixpoint iterations. For this reason, previous work [4] for RDFS reasoning has proposed several optimizations. These optimizations include loading schema triples into memory, grouping data to avoid duplicates, and ordering the rule applications to avoid fixpoint iterations. For the annotated RDFS reasoning problem, most of these optimizations are also applicable. However, there are still additional challenges to deal with the annotation. In the following, we shall discuss these challenges and their solutions.

Algorithm 2. reduce function for typing rule (a)

```
Input: key, iterator values
1: set[0].clear(); set[1].clear();
2: for value $\in$ values do
3:     set[value.flag].update(value.u, value.a);
4: end for
5: for $(i, j) \in$ set[0]$\times$set[1] do
6:     emit(null, new AnnotatedTriple(i.u, 'type', j.u, i.a$\otimes$j.a));
7: end for
```

Generalization Rule. One major difference between the general annotated RDFS reasoning and the RDFS reasoning is the generalization rule. In RDFS reasoning, we can use data grouping technique to avoid generating duplicates, and therefore the generalization rule is not necessary. In the general setting, however, this optimization is not applicable. We have to reproduce the whole dataset for this rule, which is expensive. On the other hand, such generalization rule is important to improve the efficiency in both time and space. Therefore the challenge is what is the best tradeoff. For annotated RDFS reasoning, the best decision is to apply the generalization rule twice: once at the beginning and once at the end. Furthermore, even though an explicit application of the generalization rule is unavoidable, the grouping technique is still useful to avoid duplicates as many as possible.

Unnecessary Derivation. If applying a rule derives an annotated triple $\tau : \lambda$ where $\lambda = \perp$, such derivation is an *unnecessary derivation*. Unnecessary derivations will cause the reasoner to waste a lot of time to perform useless rule applications. Therefore we want to have as few unnecessary derivations as possible. To tackle this problem, we can design the map key according to the annotation such that two annotated triples, which definitely produce empty result, will not be grouped together. Therefore the reducers can generate fewer empty results. Notice that these optimizations for specific domains might not be applicable for the general setting. However, our general methods can handle the general annotated RDFS reasoning, even though they might be inefficient. Therefore we can treat these specific optimizations as 'plugins' to the general reasoner so that they do not affect the generality.

Fixpoint Calculation. Calculating the complete results by applying the subclass rules and subproperty rules requires fixpoint iteration. For RDFS reasoning, as discussed in [4], we can load all schema triples into memory to solve this problem. The key problem is how to calculate the subclass (or subproperty) closure. In RDFS semantics, calculating the subclass (or subproperty) closure is essentially calculating the transitive closure over the subclass (or subproperty) graph. In the general setting, however, calculating the transitive closure is not enough: we have to deal with the annotations. To handle this challenge, we find that calculating the closure is essentially a variation of all-pairs shortest path calculation problem, *i.e.*, calculating the shortest pathes between each pair of nodes in a weighted graph. We can modify the shortest path algorithm to deal with this problem.

Table 2. Scalability for fuzzy RDFS reasoning

Number of units	128	64	32	16	8	4	2
Time (seconds)	122.653	136.861	146.393	170.859	282.802	446.917	822.269
Speedup	6.70	6.01	5.62	4.81	2.91	1.84	1.00

4 Preliminary Results

We have implemented a reasoner for fuzzy domain. In this section, we will report some preliminary results of our fuzzy RDFS reasoner. These results were originally reported in our journal paper [2].

We use Hadoop as our MapReduce platform. We randomly generate fuzzy degrees on top of the DBPedia core ontology as our dataset. After performing fuzzy RDFS reasoning algorithm, 133656 new fuzzy triples were derived from the original dataset which contains 26996983 fuzzy triples. Table 2 lists the running time results. The results show that the running time speedup increases along with the number of computing units used. However, this speedup increase is not as linear as expected. The reason is that there is a warmup overhead of the Hadoop system which is unavoidable no matter how many computing units we used. Therefore we basically conclude that our method is scalable with respect to that the running time decreases as the number of units increases.

5 Conclusion

In this paper, we showed how to use MapReduce techniques to achieve scalable annotated RDFS reasoning. We discuss some major challenges that will cause efficiency issues, and propose several solutions to tackle them. We report our preliminary results over fuzzy RDFS dataset, and the results show that our method is scalable.

References

1. Hoffart, J., Suchanek, F.M., Berberich, K., Lewis-Kelham, E., de Melo, G., Weikum, G.: YAGO2: exploring and querying world knowledge in time, space, context, and many languages. In: Proc. of WWW 2011, pp. 229–232 (2011)
2. Liu, C., Qi, G., Wang, H., Yu, Y.: Reasoning with Large Scale Ontologies in Fuzzy pD^* Using MapReduce. IEEE Computational Intelligence Magazine 7(2), 54–66 (2012)
3. Udrea, O., Recupero, D., Subrahmanian, V.: Annotated RDF. ACM Transactions on Computational Logic 11(2), 1–41 (2010)
4. Urbani, J., Kotoulas, S., Oren, E., van Harmelen, F.: Scalable Distributed Reasoning Using MapReduce. In: Bernstein, A., Karger, D.R., Heath, T., Feigenbaum, L., Maynard, D., Motta, E., Thirunarayan, K. (eds.) ISWC 2009. LNCS, vol. 5823, pp. 634–649. Springer, Heidelberg (2009)
5. Zimmermann, A., Lopes, N., Polleres, A., Straccia, U.: A general framework for representing, reasoning and querying with annotated Semantic Web data. Journal of Web Semantics 11, 72–95 (2012)

On Integrating Description Logics and Rules under Minimal Hypotheses

Anees Mehdi[1,⋆], Alexandre Miguel Pinto[2], and Sebastian Rudolph[1,⋆]

[1] Institute AIFB—Karlsruhe Institute of Technology
{mehdi,rudolph}@kit.edu
[2] CISUC-DEI-FCT—Universidade de Coimbra
ampinto@uc.pt

Abstract. A central and much debated topic in the Knowledge Representation and Reasoning community is how to combine open-world with closed-world formalisms, such as Description Logics (DLs) with Logic Programming. We propose an approach to defining the semantics of hybrid theories, composed of a DL and a Normal Logic Program (NLP) parts, which employs standard open-world semantics for the former and Pinto and Pereira's Minimal Hypotheses semantics (MHs) for the latter. As opposed to the currently employed semantics for hybrid DL-NLP KBs based on Stable Model (SM) semantics, our hybrid semantics guarantees the existence of models for any hybrid DL-NLP theory with consistent DL fragment and consistent DL-NLP ensemble. Because MHs features beneficial theoretical properties, like relevance and cumulativity, existential query answering tasks may not need to consider the whole hybrid KB, as it is necessarily the case with current state-of-the-art approaches based on the SM semantics.

1 Introduction

Description Logics (DLs) are a family of knowledge representation formalisms that are decidable fragments of first-order logic [2], where decidability is ensured via several syntactic restrictions. These restrictions lead to problems when expressing some non-tree like relationships. Such relations can easily be expressed using logic programming rules. Nevertheless, rule-based formalisms have their own shortcomings because typically they do not allow reasoning with unbounded infinite domains and hence cannot be used in many scenarios where modeling incomplete information is required.

A hybrid knowledge base (KB) has two components: a DL-KB[1] and a Logic Program (LP). In this work we focus on the same direction as, say, [8] and present a new approach of integrating DLs with normal Logic Programs (NLPs). Unlike the SM [3] based approaches like [8,6], in our approach, odd loops over negation[2] in the rule part of

⋆ Supported by the DFG project ExpresST.

[1] DL-KBs are usually called ontologies in the Semantic Web community. In this paper, we use these two terms interchangeably.

[2] When two rules depend on each other we say they form a loop. When such a loop is formed through default negated literals (DNLs) in the bodies of rules, we dub it loop over negation (LON). When there is an even (odd) number of DNLs through which the LON is formed we dub it even (odd) loop over negation (ELON/OLON).

a hybrid KB are not treated as modeling errors and hence not every hybrid KB containing OLONs needs to be inconsistent. Approaches based on Well-Founded Semantics (WFS) like [5] are three-valued and handle OLONs via the third *undefined* truth value.

Example 1. The affordable car problem.
Consider an online recommendation system for selling vehicles. The knowledge of the car sales company is described by the following ontology and NLP rule:

$$\begin{aligned} \mathit{Vehicle} &\equiv \mathit{Car} \sqcup \mathit{Van} \sqcup \mathit{Truck} && (1)\\ \mathit{Car} &\equiv \mathit{ABS} \sqcup \mathit{Airbagged} \sqcup \mathit{Automatic} && (2)\\ \mathit{AffordableCar} &\equiv \mathit{Car} \sqcap \neg(\mathit{ABS} \sqcap \mathit{Airbagged} \sqcap \mathit{Automatic}) \sqcap \mathit{StandardSeats} && (3)\\ \mathit{LuxuryCar} &\equiv \mathit{Car} \sqcap \mathit{ABS} \sqcap \mathit{Airbagged} \sqcap \mathit{Automatic} \sqcap \mathit{LeatherSeats} && (4)\end{aligned}$$

$$\mathit{StandardSeats(C)} \leftarrow not\ \mathit{LeatherSeats(C)} \qquad (5)$$

Vehicles for sale are cars, vans, or trucks (Axiom (1)); all cars always come with at least one additional feature (Axiom (2)); affordable car misses at least one of these features (Axiom (3)) and has standard seats; luxury cars have all three features and special leather seats (Axiom (4)). By default, a car is sold with standard seats, unless it is explicitly demanded by the customer that the car must have leather seats – Rule (5).

Suppose now there is a customer who will be happy if she gets an affordable car c, and her preferences regarding car systems are given as in the following rules:

$$\mathit{Automatic(c)} \leftarrow not\ \mathit{ABS(c)} \quad (6) \qquad \mathit{ABS(c)} \leftarrow not\ \mathit{Airbagged(c)} \quad (7)$$
$$\mathit{Airbagged(c)} \leftarrow not\ \mathit{Automatic(c)} \quad (8) \qquad \mathit{Happy} \leftarrow \mathit{AffordableCar(c)} \quad (9)$$

We need to find an affordable car while satisfying her preferences. Using the stable models as the semantic basis for the NLP part leads to no solution because the SMs are unable to assign models to the OLON formed by the rules (6), (7) and (8). However, such a system is easily realizable in our approach. ◊

LONs in NLPs can be used to represent alternative choices, not unlike SAT problems, and in these cases the existence of a solution is guaranteed as long as no Integrity Constraints[3] (ICs) are added to the program. The Closed World Assumption (CWA) principle associated with the not operator is intended to enforce a skeptical stance, i.e., holding minimal beliefs. Although with LPs with no LONs we can always apply the CWA, with LPs with LONs there are several alternative minimal sets of beliefs one can assume — in this case we no longer use the CWA, but instead a Alternative World Assumption (AWA). The approach taken by the Minimal Hypotheses (MH) semantics [9], upon which our current work is based, considers its models to be the consequences of (set-inclusion) minimally assumed hypotheses, where the assumable hypotheses come from the atoms of DNLs in LONs. In the example the NLP part is used to represent

[3] An IC is a special kind of logic rule where the head is $\perp$. ICs are not part of NLPs, but (non-Normal) LPs are unions of "normal" rules with ICs. This way, a generate-and-test problem can be modeled by a LP using the normal rules as generators of candidate solutions (the models), and using the ICs as filters to discard unsatisfying candidate solutions.

a customer's preferences which we want to satisfy in a 2-valued fashion: a SM-based approach provides no solution, whereas a MH-based one does.

2 Minimal Hypotheses-Based Semantics for Hybrid DL-NLP KBs

Our semantics for hybrid DL-NLP KBs is based upon a guess-and-check declarative fixed-point definition, an approach not unlike that of SMs (which are fixed-points of the Gelfond-Lifschitz[3] operator and are also defined via a guess-and-check).

A hybrid DL-NLP KB is a pair $\mathcal{K} = (\mathcal{O}, \mathcal{P})$ where $\mathcal{O}$ is a DL-KB and $\mathcal{P}$ is an NLP. $\Sigma_{\mathcal{O}}$ denotes the signature (the set of predicate symbols and constants occurring in) of $\mathcal{O}$, $\Sigma_{\mathcal{P}}$ denotes the signature of $\mathcal{P}$, and $\Sigma_{\mathcal{K}}$ denotes the common signature of $\mathcal{K}$ — $\Sigma_{\mathcal{K}} = \Sigma_{\mathcal{O}} \cap \Sigma_{\mathcal{P}}$. $\mathcal{AB}_{\Sigma}$ denotes the set of all possible atoms over signature Σ. Our semantics for $\mathcal{K}$ takes into account the semantics of both of its components $\mathcal{O}$ and $\mathcal{P}$, where we consider the MH semantics for $\mathcal{P}$. MHs allows for several alternative models for $\mathcal{P}$, and the $\mathcal{O}$ has several models, thus the hybrid $\mathcal{K}$ must have several hybrid models. The literals of a model of each of $\mathcal{O}$ and $\mathcal{P}$ must be used by the other to allow for the possible entailment of more consequences. Coherence is enforced: explicitly negated literals entailed from $\mathcal{O}$ imply their default negated shared $\Sigma_{\mathcal{K}}$ counterparts in $\mathcal{P}$.

Definition 2. MH-based semantics of hybrid KB.
Let $\mathcal{O}$ be a consistent DL theory and $\mathcal{K} = (\mathcal{O}, \mathcal{P})$ be a hybrid DL-NLP KB. A pair (I, M) is an MH-based hybrid model of $\mathcal{K}$ iff

- M is an MH model of $\mathcal{P} \cup (I^+ \cap \mathcal{AB}_{\Sigma_{\mathcal{K}}})$ with
- $\{not\ B : \neg B \in I^- \wedge B \in \mathcal{AB}_{\Sigma_{\mathcal{K}}}\} \subseteq M^-$ (coherence) and
- $\big(\mathcal{O} \cup (M^+ \cap \mathcal{AB}_{\Sigma_{\mathcal{K}}}) \cup (\{\neg B : not\ B \in M^- \wedge B \in \mathcal{AB}_{\Sigma_{\mathcal{K}}}\})\big) \cup I$ is consistent,

where $M = M^+ \cup M^-$, $M^+ \subseteq \mathcal{AB}_{\Sigma_{\mathcal{P}}}$, $M^- = \{not\ B : B \in \mathcal{AB}_{\Sigma_{\mathcal{P}}} \setminus M^+\}$; and $I = I^+ \cup I^-$, $I^+ \subseteq \mathcal{AB}_{\Sigma_{\mathcal{O}}}$, $I^- = \{\neg B : B \in \mathcal{AB}_{\Sigma_{\mathcal{O}}} \setminus I^+\}$. We use the term hybrid model instead of MH-based hybrid model whenever it is obvious from the context. $\diamondsuit$

In words, we define the semantics as a coupling of two different semantics via a synchronizing "interface" of ground atoms. A work closely related thereto is that of the so-called multi-context systems (MCSs): a framework that allows for combining arbitrary monotonic and non-monotonic logics [1]. A Hybrid KB in our approach can be taken as a multi-context system with two contexts, an ontology context and a program context. See [7] for a detail comparison with existing approaches.

In Example 1, a hybrid model would be (I, M) with (abbreviating predicate names)

$$I = \{SS(c), \neg LS(c), Air(c), ABS(c), AC(c), \neg Aut(c), Car(c), \neg LC(c), Veh(c)\} \text{ and}$$
$$M = \{SS(c), not\ LS(c), Air(c), ABS(c), not\ Aut(c), H, AC(c)\}.$$

Non-monotonicity in the NLP part is naturally supported by our formalism.

3 Reasoning

Given a model (I, M) for $\mathcal{K} = (\mathcal{O}, \mathcal{P})$ and an atom A, $(I, M) \models A$ iff

- $\mathcal{O} \cup (M^+ \cap \mathcal{AB}_{\Sigma_\mathcal{K}}) \cup (\{\neg B : \mathit{not}\ B \in M^- \wedge B \in \mathcal{AB}_{\Sigma_\mathcal{K}}\}) \cup I \models A$ whenever $A \in \mathcal{AB}_{\Sigma_\mathcal{O}}$, and
- $A \in M$ whenever $A \in \mathcal{AB}_{\Sigma_\mathcal{P}}$

Two reasoning tasks are essential: *consistency* and *entailment*. $\mathcal{K}$ is *consistent* iff there is at least one MH-based model for $\mathcal{K}$. For a given first-order atom A we say A is *credulously/skeptically entailed from* $\mathcal{K}$ (written as $\mathcal{K} \models_C A / \mathcal{K} \models A$) iff for some/every MH-based hybrid model (I, M) of $\mathcal{K}$ we have $(I, M) \models A$. The rules never violate the DL-safety restriction as the only way for $\mathcal{O}$ and $\mathcal{P}$ to communicate is via a finite set of shared ground atoms. Hence, the DL-safety restriction is trivially satisfied for all the rules. It follows from Def. 2 that both the consistency and entailment problems require guessing sets I and M such that the conditions imposed by the definitions are satisfied. In [7] we have shown that the complexity of these problems highly depends on the DL in the which the ontology part of the hybrid DL-NLP KB is formulated. E.g., for $\mathcal{SROIQ}$ [4] we get:

Theorem 3. *Complexity of the entailment and consistency problems.*
The consistency and the entailment problems in a hybrid DL-NLP KB are both N2EXPTIME*-complete.*

For the proof, we refer to [7] where we also provide a straight forward method for checking the entailment of an atom from a hybrid DL-NLP KB.

References

1. Brewka, G., Eiter, T.: Equilibria in heterogeneous nonmonotonic multi-context systems. In: AAAI, pp. 385–390 (2007)
2. Baader, F., et al. (eds.): The Description Logic Handbook: Theory, Implementation, and Applications. Cambridge University Press (2003)
3. Gelfond, M., Lifschitz, V.: The stable model semantics for logic programming. In: Kowalski, R. (ed.) Procs. ICLP 1988, pp. 1070–1080. MIT Press (1988)
4. Horrocks, I., Kutz, O., Sattler, U.: The even more irresistible $\mathcal{SROIQ}$. In: Doherty, P., Mylopoulos, J., Welty, C.A. (eds.) Proceedings of the 10th International Conference on Principles of Knowledge Representation and Reasoning (KR 2006), pp. 57–67. AAAI Press (2006)
5. Knorr, M., Alferes, J.J., Hitzler, P.: A coherent well-founded model for hybrid mknf knowledge bases. In: Procs. ECAI 2008, pp. 99–103. IOS Press (2008)
6. Lukasiewicz, T.: A novel combination of answer set programming with description logics for the semantic web. IEEE Trans. Knowl. Data Eng. 22, 1577–1592 (2010)
7. Mehdi, A., Pinto, A.M., Rudolph, S.: On integrating description logic and rules under mh semantics. Tech. Rep. 3028, Institute AIFT, Karlsruhe Institute of Technology (2012)
8. Motik, B., Rosati, R.: Reconciling Description Logics and Rules. J. ACM 57(5), 1–62 (2010)
9. Pinto, A.M., Pereira, L.M.: Each normal logic program has a 2-valued minimal hypotheses semantics. CoRR abs/1108.5766 (2011)

On the (Non-)Succinctness of Uniform Interpolation in General $\mathcal{EL}$ Terminologies

Nadeschda Nikitina and Sebastian Rudolph

Karlsruhe Institute of Technology
Karlsruhe, Germany
{nikitina,rudolph}@kit.edu

1 Introduction

In view of the practical deployment of OWL [9] based on description logics [2], the importance of non-standard reasoning services for supporting ontology engineers was pointed out, for instance, in [8]. An example of such reasoning services is that of *uniform interpolation*: given a theory using a certain vocabulary, and a subset Σ of "relevant terms" of that vocabulary, find a theory that uses only Σ terms and gives rise to the same consequences (expressible via Σ) as the original theory. In particular for the understanding and the development of complex knowledge bases, e.g., those consisting of *general concept inclusions* (GCIs), the appropriate tool support would be beneficial. We consider the task of uniform interpolation in the very lightweight description logic $\mathcal{EL}$, the basic member of the $\mathcal{EL}$ family [1] which provides the logical backbone of the OWL EL profile. The existing related approaches ([3,6,4]) do not provide a solution for the task of uniform interpolation in general $\mathcal{EL}$ terminologies. Up to now, also the bounds on the size of uniform $\mathcal{EL}$ interpolants have been unknown. We propose a worst-case-optimal approach to computing a finite uniform $\mathcal{EL}$ interpolant for a general terminology. After a normalization, we construct two regular tree grammars generating subsumees and subsumers of atomic concepts interpreted as tree languages. Using a Gentzen-style proof calculus for general subsumptions in $\mathcal{EL}$, we show that, in case a uniform interpolant exists, the corresponding sublanguages with an exponential bound on the role depth are sufficient to obtain a uniform $\mathcal{EL}$ interpolant of at most triple exponential size. Further, we show that, in the worst-case, no smaller interpolants exist, thereby establishing the triple exponential tight bounds on the size of uniform interpolants in $\mathcal{EL}$. This is a report on our recent work accepted at ECAI 2012 [7].

2 Preliminaries

Let N_C and N_R be countably infinite and mutually disjoint sets of concept symbols and role symbols. An $\mathcal{EL}$ concept C is defined as $C ::= A|\top|C \sqcap C|\exists r.C$, where A and r range over N_C and N_R, respectively. In the following, we use symbols A, B to denote atomic concepts and C, D to denote arbitrary concepts. A *terminology* or *TBox* consists of *concept inclusion* axioms $C \sqsubseteq D$ and *concept equivalence* axioms $C \equiv D$ used as a shorthand for $C \sqsubseteq D$ and $D \sqsubseteq C$. While knowledge bases in general can also include a specification of individuals with the corresponding concept and role assertions

(ABox), in this paper we do not consider them. The signature of an $\mathcal{EL}$ concept C or an axiom α, denoted by $\mathrm{sig}(C)$ or $\mathrm{sig}(\alpha)$, respectively, is the set of concept and role symbols occurring in it. To distinguish between the set of concept symbols and the set of role symbols, we use $\mathrm{sig}_C(C)$ and $\mathrm{sig}_R(C)$, respectively. The signature of a TBox $\mathcal{T}$, in symbols $\mathrm{sig}(\mathcal{T})$ (correspondingly, $\mathrm{sig}_C(\mathcal{T})$ and $\mathrm{sig}_R(\mathcal{T})$), is defined analogously. The semantics of the above introduced DL constructs is standard and can be found, for instance, in [2].

In this paper, we investigate uniform interpolation based on concept- inseparability, i.e., the aim is to preserve all Σ-concept inclusions. Thus, the task of *uniform interpolation* is defined as follows: Given a signature Σ and a TBox $\mathcal{T}$, determine a TBox $\mathcal{T}'$ with $\mathrm{sig}(\mathcal{T}') \subseteq \Sigma$ such that for all $\mathcal{EL}$ concepts C, D with $\mathrm{sig}(C) \cup \mathrm{sig}(D) \subseteq \Sigma$ holds: $\mathcal{T} \models C \sqsubseteq D$ iff $\mathcal{T}' \models C \sqsubseteq D$. $\mathcal{T}'$ is also called a *uniform $\mathcal{EL}$ Σ-interpolant* of $\mathcal{T}$. In practice, uniform interpolants are required to be finite, i.e., expressible by a finite set of finite axioms using only the language constructs of $\mathcal{EL}$.

3 Lower Bound

While deciding the existence of uniform interpolants in $\mathcal{EL}$ is exponential [4], i.e., one exponential less complex than the same decision problem for the more complex logic $\mathcal{ALC}$ [6], the size of uniform interpolants remains triple-exponential. We demonstrate that this is in fact the lower bound by the means of the following example (obtained by a slight modification of an example given in [5] originally demonstrating a double exponential lower bound in the context of conservative extensions).

Example 1. The $\mathcal{EL}$ TBox $\mathcal{T}_n$ for a natural number n is given by

$$A_1 \sqsubseteq \overline{X_0} \sqcap \ldots \sqcap \overline{X_{n-1}} \tag{1}$$

$$A_2 \sqsubseteq \overline{X_0} \sqcap \ldots \sqcap \overline{X_{n-1}} \tag{2}$$

$$\sqcap_{\sigma\in\{r,s\}} \exists\sigma.(\overline{X_i} \sqcap X_0 \sqcap \ldots \sqcap X_{i-1}) \sqsubseteq X_i \qquad i < n \tag{3}$$

$$\sqcap_{\sigma\in\{r,s\}} \exists\sigma.(X_i \sqcap X_0 \sqcap \ldots \sqcap X_{i-1}) \sqsubseteq \overline{X_i} \qquad i < n \tag{4}$$

$$\sqcap_{\sigma\in\{r,s\}} \exists\sigma.(\overline{X_i} \sqcap \overline{X_j}) \sqsubseteq \overline{X_i} \quad j < i < n \tag{5}$$

$$\sqcap_{\sigma\in\{r,s\}} \exists\sigma.(X_i \sqcap \overline{X_j}) \sqsubseteq X_i \quad j < i < n \tag{6}$$

$$X_0 \sqcap \ldots \sqcap X_{n-1} \sqsubseteq B \tag{7}$$

In $\mathcal{T}_n$, the atomic concepts X_i and $\overline{X_i}$ represent the bit number i of a binary counter being set and unset, respectively. Axiom 3 ensures that an unset bit will be set in the successor number, if all smaller bits are already set. The subsequent Axiom 4 ensures that a set bit will be unset in the successor number, if all smaller bits are also set. Axioms 5 and 6 ensure that in all other cases, bits do not flip. For instance, Axiom 5 states that, if any bit before bit i is still unset, then bit i will remain unset also in the successor number. If we now consider sets C_i of concept descriptions inductively defined by $C_0 = \{A_1, A_2\}$, $C_{i+1} = \{\exists r.C_1 \sqcap \exists s.C_2 \mid C_1, C_2 \in C_i\}$, then we find that $|C_{i+1}| = |C_i|^2$ and consequently $|C_i| = 2^{(2^i)}$. Thus, the set C_{2^n-1} contains triply exponentially many different concepts, each of which is doubly exponential in the size of $\mathcal{T}_n$ (intuitively, we obtain concepts having the shape of binary trees of exponential depth, thus having doubly exponentially many leaves, each of which can be endowed with A_1 or A_2, giving rise to triply exponentially many different such trees). It can be shown that for each concept $C \in C_{2^n-1}$ it holds $\mathcal{T}_n \models C \sqsubseteq B$ and that there cannot be a smaller uniform interpolant for $\mathcal{T}_n$ w.r.t. the signature $\Sigma = \{A_1, A_2, B, r, s\}$ than $\{C \sqsubseteq B \mid C \in C_{2^n-1}\}$.

Hence we have found a class $\mathcal{T}_n$ of TBoxes giving rise to uniform interpolants of triple-exponential size in terms of the original TBox. In the following, we show that this is also an upper bound by providing a procedure for computing uniform interpolants with a triple-exponentially bounded output.

4 Upper Bound

The upper bound can be shown by providing an algorithm, which computes a uniform interpolant, in case it exists, of at most triple-exponential size in the size of the original TBox. The algorithm relies on a normalization, which assigns to each sub-expression occurring in the original TBox and not being equivalent to any atomic concept a fresh concept name. This can be done recursively by replacing sub-expressions $C_1 \sqcap ... \sqcap C_n$ and $\exists r.C$ by fresh concept symbols until each axiom in the TBox $\mathcal{T}$ is one of $\{A \sqsubseteq B, A \equiv B_1 \sqcap ... \sqcap B_n, A \equiv \exists r.B\}$, where $A, B, B_i \in \text{sig}_C(\mathcal{T}) \cup \{\top\}$ and $r \in \text{sig}_R(\mathcal{T})$. Given a normalized TBox additionally extended with classification results, we can show using a deduction calculus for $\mathcal{EL}$ terminologies that the uniform interpolant UI can be obtained from the sets of subsumers and subsumees of all atomic concepts in $\mathcal{T}$ as follows.

Definition 1. *Let $\mathcal{T}$ be a normalized $\mathcal{EL}$ TBox and , for each $A \in sig_C(\mathcal{T})$, let $R_1(A)$ and $R_2(A)$ be the set of subsumees and the set of subsumers of A in $\mathcal{T}$. Then, the $\mathcal{EL}$ TBox* $\mathtt{UI}(\mathcal{T}, \Sigma, R_1, R_2)$ *is given by*

$$\{C \sqsubseteq A \mid A \in \Sigma, C \in R_1(A)\} \cup \{A \sqsubseteq D \mid A \in \Sigma, D \in R_2(A)\} \cup$$
$$\{C \sqsubseteq D \mid \textit{there is } A \notin \Sigma, C \in R_1(A), D \in R_2(A)\}.$$

In our approach, we represent the (possibly infinite) sets of subsumees and subsumers as tree languages $L(G)$ generated by regular tree grammars G, where concept expressions C are interpreted as a trees according to their term structure.

Theorem 1. *Let $\mathcal{T}$ be a normalized $\mathcal{EL}$ TBox, Σ a signature. For each $A \in sig_C(\mathcal{T})$, we can compute from $\mathcal{T}$ in exponential time a grammar $G^{\sqsupseteq}(\mathcal{T}, \Sigma, A)$ generating subsumees of A and a grammar $G^{\sqsubseteq}(\mathcal{T}, \Sigma, A)$ generating subsumers of A with the following properties:*

- *$G^{\sqsupseteq}(\mathcal{T}, \Sigma, A)$ and $G^{\sqsubseteq}(\mathcal{T}, \Sigma, A)$ are exponentially bounded in the size of $\mathcal{T}$, while the number of non-terminals corresponds to the number of atomic concepts in $\mathcal{T}$.*
- *For each C with $sig(C) \subseteq \Sigma$ such that $\mathcal{T} \models C \sqsubseteq A$ there is a concept C' generated by $G^{\sqsupseteq}(\mathcal{T}, \Sigma, A)$ such that C can be obtained from C' by adding arbitrary conjuncts to arbitrary sub-expressions.*
- *Each D satisfying $sig(D) \subseteq \Sigma$ and $\mathcal{T} \models A \sqsubseteq D$ is generated by $G^{\sqsubseteq}(\mathcal{T}, \Sigma, A)$.*

While the languages generated by the grammars are usually infinite, we require finite subsets of $L(G^{\sqsupseteq}(\mathcal{T}, \Sigma, A))$ and $L(G^{\sqsubseteq}(\mathcal{T}, \Sigma, A))$ to obtain the corresponding upper bound. Based on the following lemma presented in [4], we obtain a bound on the role depth of minimal uniform $\mathcal{EL}$ interpolants, allowing us to restrict the role depth of relevant elements in $L(G^{\sqsupseteq}(\mathcal{T}, \Sigma, A))$ and $L(G^{\sqsubseteq}(\mathcal{T}, \Sigma, A))$:

Lemma 1. *Let $\mathcal{T}$ be a normalized $\mathcal{EL}$ TBox, Σ a signature. There exists a uniform $\mathcal{EL}$ Σ-interpolant of $\mathcal{T}$ if and only if there exists a uniform $\mathcal{EL}$ Σ-interpolant $\mathcal{T}'$ of $\mathcal{T}$ whose maximal role depth is exponentially bounded by $|\mathcal{T}|$.*

Based on this bound and the size of $G^{\sqsupseteq}(\mathcal{T},\Sigma,A)$,$G^{\sqsubseteq}(\mathcal{T},\Sigma,A)$, we can describe a way to materialize a role-depth-bounded part of $G^{\sqsupseteq}(\mathcal{T},\Sigma,A)$,$G^{\sqsubseteq}(\mathcal{T},\Sigma,A)$ into subsumer and subsumee sets $R_1(A)$ and $R_2(A)$, respectively, obtaining the following result:

Theorem 2. *Given an $\mathcal{EL}$ TBox $\mathcal{T}$ and a signature Σ, there exists a uniform $\mathcal{EL}$ Σ-interpolant of $\mathcal{T}$ iff there exists a uniform $\mathcal{EL}$ Σ-interpolant $\mathcal{T}'$ with $|\mathcal{T}'| \in O(2^{2^{2^{|\mathcal{T}|}}})$.*

5 Summary

In this paper, we summarize an approach to computing uniform interpolants of general $\mathcal{EL}$ terminologies based on proof theory and regular tree languages. Moreover, we noted that, if a finite uniform $\mathcal{EL}$ interpolant exists, then there exists one of at most triple exponential size in terms of the original TBox, and that, in the worst-case, no shorter interpolant exists, thereby establishing the triple exponential tight bounds.

Acknowledgements. This work was supported by the DFG project ExpresST.

References

1. Baader, F., Brandt, S., Lutz, C.: Pushing the $\mathcal{EL}$ envelope. In: Proc. of the 19th Int. Joint Conf. on Artificial Intelligence (IJCAI 2005), pp. 364–369 (2005)
2. Baader, F., Calvanese, D., McGuinness, D., Nardi, D., Patel-Schneider, P. (eds.): The Description Logic Handbook: Theory, Implementation, and Applications, 2nd edn. Cambridge University Press (2007)
3. Konev, B., Walther, D., Wolter, F.: Forgetting and uniform interpolation in large-scale description logic terminologies. In: Proc. of the 21st Int. Joint Conf. on Artificial Intelligence (IJCAI 2009), pp. 830–835 (2009)
4. Lutz, C., Seylan, I., Wolter, F.: An automata-theoretic approach to uniform interpolation and approximation in the description logic EL. In: Proc. of the 13th Int. Conf. on Principles of Knowledge Representation and Reasoning (KR 2012) (2012)
5. Lutz, C., Wolter, F.: Deciding inseparability and conservative extensions in the description logic $\mathcal{EL}$. Journal of Symbolic Computation 45(2), 194–228 (2010)
6. Lutz, C., Wolter, F.: Foundations for uniform interpolation and forgetting in expressive description logics. In: Proc. of the 22nd Int. Joint Conf. on Artificial Intelligence (IJCAI 2011), pp. 989–995 (2011)
7. Nikitina, N., Rudolph, S.: ExpExpExplosion: Uniform interpolation in general EL terminologies. In: Proc. of the 20th European Conf. on Artificial Intelligence, ECAI 2012 (to appear, 2012), Extended version available via `http://dl.dropbox.com/u/10637748/11.pdf`
8. Nikitina, N., Rudolph, S., Glimm, B.: Reasoning-supported interactive revision of knowledge bases. In: Proc. of the 22nd Int. Joint Conf. on Artificial Intelligence (IJCAI 2011), pp. 1027–1032 (2011)
9. OWL Working Group, W: OWL 2 Web Ontology Language: Document Overview. W3C Recommendation, October 27 (2009), `http://www.w3.org/TR/owl2-overview/`

On Definability and Redundancy in $\mathcal{EL}$–TBoxes[⋆]

Denis Ponomaryov[1] and Dmitry Vlasov[2]

[1] Institute of Informatics Systems, Novosibirsk, Russia
[2] Institute of Mathematics, Novosibirsk, Russia
ponom@iis.nsk.su, vlasov@academ.org

Abstract. We describe a technical solution for solving the following entailment problem in the Description Logic $\mathcal{EL}$: $(\mathcal{T} \setminus U) \cup f^*(\mathcal{T} \setminus U) \models C_1 \sqsubseteq C_2$, where $\mathcal{T}$ is a TBox, C_1,C_2 are concepts, U is a subset of axioms of $\mathcal{T}$, and f^* is a renaming function which gives a copy of TBox with some signature symbols injectively renamed. We introduce an enriched model structure for $\mathcal{EL}$ TBoxes: once built for a given TBox $\mathcal{T}$, it allows for checking the entailment efficiently for different choices of C_1,C_2, U, and f^*. This problem is concerned with computing explicit definitions of concepts wrt a signature and is an important part of the recently proposed algorithms for ontology decomposition.

1 Introduction

In this paper, we address the problem of eliminating redundant axioms from $\mathcal{EL}$–Tboxes and deciding existence of explicit definition of a $\mathcal{EL}$–concept wrt a given signature in a $\mathcal{EL}$–TBox. These procedures are the important part of the decomposition algorithm for $\mathcal{EL}$–TBoxes proposed in [2]. Given a TBox $\mathcal{T}$, an axiom $\varphi \in \mathcal{T}$ is called *redundant* if $\mathcal{T} \setminus \{\varphi\} \models \varphi$. The notion has been discussed in [4], where general algorithms for eliminating redundant axioms have been proposed. In the decomposition algorithms of [2], eliminating redundant axioms from the input TBox turns out to be very important and allows to avoid excessive calls of computationally hard subroutines. Given a TBox $\mathcal{T}$, a concept C is said to be explicitly definable in $\mathcal{T}$ wrt a signature Σ if $\mathcal{T} \models C \equiv E$, where E is a concept over Σ. Explicit definability was studied in Description Logics in context of query rewriting (Sect. 3 in [5]) and is intimately related to modularity properties of ontologies via the notion of uniform interpolation [3]. Unsurprisingly, it happens to be the crucial point in the decomposition algorithms from [2]. As the logic $\mathcal{EL}$ enjoys parallel interpolation property and concept interpolation (Theorems 13 and 45 in [2]), a concept C is explicitly definable in a $\mathcal{EL}$–TBox $\mathcal{T}$ wrt a signature $\Sigma \subseteq \mathtt{sig}(\mathcal{T})$ (a subset of signature of $\mathcal{T}$) iff $\mathcal{T} \cup \mathcal{T}^* \models C^* \sqsubseteq C$, where $\mathtt{sig}(\mathcal{T}) \cap \mathtt{sig}(\mathcal{T}^*) = \Sigma$, $\mathtt{sig}(C^*) \cap \mathtt{sig}(C) \subseteq \Sigma$, and $\mathcal{T}^*$ and C^* are obtained from $\mathcal{T}$ and C, respectively, by replacing all non-Σ-symbols simultaneously with fresh ones, not present in $\mathtt{sig}(C) \cup \mathtt{sig}(\mathcal{T})$. For instance, if A, B, C, D are concept names,

[⋆] This work was funded by the grant of the President of Russian Federation (Grant No. MK-2037.2011.9), the Russian Academy of Sciences (Grant No. 15/10), and the Siberian Division of the Russian Academy of Sciences (Integration Project No. 3).

$\mathcal{T} = \{A \sqcap B \sqsubseteq C, A \equiv D, A \sqsubseteq B\}$ and $\Sigma = \{D\}$ then $\mathcal{T} \models A \sqcap B \equiv D$ and we have the entailment $\mathcal{T} \cup \{A^* \sqcap B^* \sqsubseteq C^*, A^* \equiv D, A^* \sqsubseteq B^*\} \models A^* \sqcap B^* \sqsubseteq A \sqcap B$ proving definability of concept $A \sqcap B$ wrt Σ in $\mathcal{T}$[1]. Thus, definability is polynomially reduced to entailment in $\mathcal{EL}$ which is tractable, although the explicit definition itself can be of exponential length wrt the size of $\mathcal{T}$ and C (cf. Example 28 in [2]). Deciding definability allows to avoid excessive search for explicit definitions which is provably harder than checking entailment (in many Description Logics). At the same time, it suffices for computing signature decompositions of $\mathcal{EL}$–TBoxes ([2]). There are two complications in this context, however. First, the signature Σ may not be known in advance and only has to be a subset of $\mathtt{sig}(C)$ (probably, in union with a subset $\Delta \subseteq \mathtt{sig}(\mathcal{T})$). Second, it may be necessary to verify entailment of $C^* \sqsubseteq C$ independently of some subset $U \subseteq \mathcal{T}$ of axioms of TBox (cf. Figure 5 in [2]). For instance, in the example above, the axiom $A \sqcap B \sqsubseteq C$ was not needed to prove the equivalence $A \sqcap B \equiv D$ in $\mathcal{T}$. We introduce *enriched models* of $\mathcal{EL}$–TBoxes, which, given a TBox $\mathcal{T}$, allow for deciding efficiently the entailment

$$(\mathcal{T} \setminus U) \cup f^*(\mathcal{T} \setminus U) \models C_1 \sqsubseteq C_2$$

for different subsets $U \subseteq \mathcal{T}$, concepts C_1, C_2, and function f^* renaming signature symbols injectively. By definition, this allows for efficiently removing redundant axioms and checking for existence of explicit definitions of concepts wrt different subsignatures of $\mathcal{T}$. Since the main application of this procedure is the decomposition algorithm working with a given TBox (Figure 5 in [2]), we restrict ourselves to the case when C_1, C_2 are concepts occurring in axioms of $(\mathcal{T} \setminus U) \cup f^*(\mathcal{T} \setminus U)$. Though, the proposed construction is straightforwardly extended to the case of arbitrary concepts. The main idea behind the enriched models is the observation applicable to any system with non-trivial search problems: do a preprocessing of information in order to handle numerous standard calls to it faster.

2 Enriched Models of $\mathcal{EL}$–TBoxes

An enriched model is a structure related to syntactic form of axioms in a TBox, recall the syntactic model considered in the standard decision procedure for $\mathcal{EL}$ [1]. It is built over a TBox in an *enriched normal form* obtained as follows. Given a TBox $\mathcal{T}$, the conservative extension $\mathcal{T}_{prim}$ of $\mathcal{T}$ is constructed which satisfies the following property: for each axiom $\varphi = (C \sqsubseteq D)$ in $\mathcal{T}$, there is a formula $A \sqsubseteq B \in \mathcal{T}_{prim}$ augmented with a link to the original axiom φ such that $A \sqsubseteq B$ is equivalent to φ in $\mathcal{T}_{prim}$ and A, B are concept names. This conservative extension is called *primitivization* of TBox $\mathcal{T}$ (for concept names are primitive objects in TBox). Then $\mathcal{T}_{prim}$ is transformed into normal form in a standard way [1] with the only exception that arbitrary number of conjuncts is allowed on the left–hand side of concept inclusions and each formula in the normal form is augmented with a link to the corresponding original axiom φ from $\mathcal{T}$. After these transformations, a TBox is said to be in *enriched normal*

[1] The explicit definition is exactly the concept interpolant between $A^* \sqcap B^*$ and $A \sqcap B$.

form (notation $\mathcal{T}_{norm}$). The main idea behind enriched model is that for every concept inclusion $A \sqsubseteq B$, with $\mathcal{T}_{norm} \models A \sqsubseteq B$ and A, B concept names from $\mathtt{sig}(\mathcal{T}_{norm})$, the enriched model stores paths corresponding to derivations of $A \sqsubseteq B$, with all the intermediate formulas from $\mathcal{T}_{norm}$ used in these derivations and links to the original axioms of $\mathcal{T}$. If $U \subseteq \mathcal{T}$, then $\mathcal{T} \setminus U \models A \sqsubseteq B$ holds iff there is a path corresponding to a derivation of $A \sqsubseteq B$ in $\mathcal{T}$, where no intermediate formula refers to an axiom of U. Specifically, an enriched model is a bipartite graph $\langle \Gamma^+, \Gamma^-, E \rangle$ augmented with four mappings, where (informally speaking) vertices of Γ^+ (+–vertices) correspond to signature symbols of $\mathcal{T}_{norm}$ and vertices of Γ^- correspond to axioms of $\mathcal{T}_{norm}$. The four mappings are:

– $\delta : \Gamma^- \to \mathcal{T}$ which maps formula vertices to axioms of the original TBox;
– $\rho : \Gamma^+ \to \mathsf{N}_\mathsf{C}^\mathcal{T} \cup (\mathsf{N}_\mathsf{C}^\mathcal{T} \times \mathsf{N}_\mathsf{C}^\mathcal{T})$ mapping +–vertices to concept names from $\mathcal{T}_{norm}$ (the set $\mathsf{N}_\mathsf{C}^\mathcal{T}$) and pairs of concept names from $\mathsf{N}_\mathsf{C}^\mathcal{T} \times \mathsf{N}_\mathsf{C}^\mathcal{T}$;
– $f^c : \mathsf{N}_\mathsf{C}^\mathcal{T} \to \mathcal{P}(\Gamma^+)$ giving a set of +–vertices corresponding to a concept name;
– $f^r : \mathsf{N}_\mathsf{R}^\mathcal{T} \to \mathcal{P}(\Gamma^+)$ giving a set of +–vertices corresponding to a role name from $\mathcal{T}_{norm}$ (the set $\mathsf{N}_\mathsf{R}^\mathcal{T}$).

It holds that $\mathcal{T}_{norm} \models A \sqsubseteq B$ for concept names A and B iff there is a vertex $v \in \Gamma^+$ in the enriched model of $\mathcal{T}_{norm}$ such that $v \in f^c(B)$ and $\rho(v) = A$.

Theorem 1. *The standard model for TBox $\mathcal{T}_{norm}$ is obtained by taking the set $\mathsf{N}_\mathsf{C}^\mathcal{T}$ as the universe and the composition mappings $\rho \circ f^c : \mathsf{N}_\mathsf{C}^\mathcal{T} \to \mathcal{P}(\mathsf{N}_\mathsf{C}^\mathcal{T})$ and $\rho \circ f^r : \mathsf{N}_\mathsf{R}^\mathcal{T} \to \mathcal{P}(\mathsf{N}_\mathsf{C}^\mathcal{T} \times \mathsf{N}_\mathsf{C}^\mathcal{T})$ as the interpretation for concept names and roles.*

Given a TBox $\mathcal{T}$, an enriched model $\mathcal{M}$ of $\mathcal{T}_{norm}$, a subset $U \subseteq \mathcal{T}$, and concepts C_1, C_2 occurring in axioms of $\mathcal{T}$, the entailment $\mathcal{T} \setminus U \models C_1 \sqsubseteq C_2$ is decided as follows. By primitivization and normalization steps, we may assume that we have concept names A_1 and A_2 such that $\mathcal{T}_{norm} \models A_i \equiv C_i$, for $i = 1, 2$. We consider the set of vertices $f^c(A_2) \subseteq \Gamma^+$. If there is no $v \in f^c(A_2)$ with $\rho(v) = A_1$, then $\mathcal{T}_{norm} \not\models A_1 \sqsubseteq A_2$ and hence, $\mathcal{T} \setminus U \not\models C_1 \sqsubseteq C_2$. Otherwise, consider a substructure $\mathcal{N} \subseteq \mathcal{M}$ such that $v \in \Gamma^+ |_\mathcal{N}$ and for each vertex $u \in \Gamma^- |_\mathcal{N}$, it holds $\delta(u) \notin U$. We call the substructure $\mathcal{N}$ *witness* for v in $\mathcal{T} \setminus U$.

Theorem 2. *$\mathcal{T} \setminus U \models A_1 \sqsubseteq A_2$ holds iff there is a vertex $v \in f^c(A_2)$ with $\rho(v) = A_1$ and a witness $\mathcal{N}$ for v in $\mathcal{T} \setminus U$.*

The algorithm for searching substructure $\mathcal{N}$ reduces to recursive traversal of the graph representation of model $\mathcal{M}$, starting from vertex v. In particular, this procedure allows to check axioms of a TBox for redundancy using a single prebuilt enriched model of TBox. To decide the entailment $(\mathcal{T} \setminus U) \cup f^*(\mathcal{T} \setminus U) \models C_1 \sqsubseteq C_2$ for various selections of subset U and renaming function f^*, we start from a prebuilt enriched model $\mathcal{M}_0$ of TBox $(\mathcal{T} \cup \mathcal{T}')_{norm}$, where $\mathcal{T}'$ is a signature–disjoint copy of $\mathcal{T}$ (with all symbols from $\mathtt{sig}(\mathcal{T})$ injectively renamed). Having function f^*, w.l.o.g. we assume that f^* agrees on fresh symbols with the initial renaming used for $\mathcal{T}'$, $\mathtt{sig}(f^*(\mathcal{T})) \subseteq \mathtt{sig}(\mathcal{T}') \cup \mathtt{sig}(\mathcal{T})$ and construct the enriched model $\mathcal{M}_f$ for $\mathcal{T} \cup f^*(\mathcal{T})$ in two steps. First, we factorize model $\mathcal{M}_0$ by identifying symbols from $\mathtt{sig}(\mathcal{T})$ which are not changed by f^* with their counterparts from $\mathtt{sig}(\mathcal{T}')$. Then we extend the factorized model $\mathcal{M}_0$ to satisfy all concept inclusions from $(\mathcal{T} \cup f^*(\mathcal{T}))_{norm}$ which are not already contained in $(\mathcal{T} \cup \mathcal{T}')_{norm}$. Finally, we use the obtained model $\mathcal{M}_f$ to decide $(\mathcal{T} \setminus U) \cup f^*(\mathcal{T} \setminus U) \models A_1 \sqsubseteq A_2$, where A_1 and A_2 are concept names corresponding to C_1 and C_2 as described above for redundancy testing. If the set of signature symbols not renamed by f^* is "small enough" in comparison with cardinality of $\mathtt{sig}(\mathcal{T})$ (this is the case for the decomposition algorithm in Figure 5 in [2]), then computing $\mathcal{M}_f$ based on the prebuilt model $\mathcal{M}_0$ is cheaper than building the enriched model for $(\mathcal{T} \cup f^*(\mathcal{T}))_{norm}$ from scratch.

3 Conclusions

An enriched model is an extended representation of a TBox employed for eliminating redundant axioms from TBox and deciding definability of a concept wrt TBox and concept subsignature. Essentially, computing an enriched model is a preprocessing which gives a moderate increase in the size of representation of a TBox in computer memory and allows to avoid multiple calls to an external reasoner. To give an example, we provide some numerical data for one of the versions of Plant Ontology (*PO*, Revision 1.64) consisting of 1274 $\mathcal{EL}$ subclass axioms. In our implementation, the size of the internal representation of *PO* after loading was 941 Kb, while the size of the enriched model for PO_{norm} required 4.43 Mb. The enriched model was computed in 0.97 seconds and, based on this prebuilt model, a maximal irredundant subset of axioms of PO was found in 0.69 seconds. The order in which axioms were selected from PO for redundancy testing was arbitrary; seven axioms have been found redundant. With the same order on axioms, the simple algorithm for redundancy testing based on the standard reasoning procedure for $\mathcal{EL}$ computes the same maximal irredundant set in more than two minutes. While redundancy elimination becomes efficient unconditionally, a gain in speed of deciding explicit definability with the proposed technique depends much on the context of application. As the main purpose of our methods is related to decomposition algorithms which are out of the scope of this paper, we have to leave discussion of relevant details for the future.

References

1. Brandt, S.: Polynomial Time Reasoning in a Description Logic with Existential Restrictions, GCI Axioms, and—What Else? In: Proc. Sixteenth European Conference on Artificial Intelligence, ECAI 2004 (2004)
2. Konev, B., Lutz, C., Ponomaryov, D., Wolter, F.: Decomposing description logic ontologies. In: Proc. Twelfth International Conference on the Principles of Knowledge Representation and Reasoning, KR 2010 (2010)
3. Konev, B., Lutz, C., Walther, D., Wolter, F.: Formal Properties of Modularisation. In: Parent, C., Spaccapietra, S., Stuckenschmidt, H. (eds.) Ontology Modularisation. Springer (2009)
4. Grimm, S., Wissmann, J.: Elimination of Redundancy in Ontologies. In: Antoniou, G., Grobelnik, M., Simperl, E., Parsia, B., Plexousakis, D., De Leenheer, P., Pan, J. (eds.) ESWC 2011, Part I. LNCS, vol. 6643, pp. 260–274. Springer, Heidelberg (2011)
5. Seylan, I., Franconi, E., de Bruijn, J.: Effective query rewriting with ontologies over dboxes. In: Proc. Twenty-first International Joint Conference on Artificial Intelligence, IJCAI 2009 (2009)

A Mobile Reasoner for Semantic-Based Matchmaking

Michele Ruta, Floriano Scioscia, Giuseppe Loseto, Filippo Gramegna, and Eugenio Di Sciascio

Politecnico di Bari, via E. Orabona 4, I-70125 Bari, Italy
{m.ruta,f.scioscia,disciascio}@poliba.it,
{loseto,gramegna}@deemail.poliba.it

Abstract. Reasoning in pervasive computing has to face computational issues inherited by mobile platforms. This paper presents a prototypical reasoner for mobile devices, which leverages Semantic Web technologies to implement both standard (subsumption, satisfiability, classification) and non-standard (abduction, contraction) inferences for moderately expressive knowledge bases. System features are surveyed, followed by early performance analysis.

1 Introduction

Semantic Web technologies have been proposed as a candidate to promote interoperability and intelligent decision support in ubiquitous computing contexts –*e.g.*, supply chain management and u-commerce [6], peer-to-peer resource discovery [7] and so on– keeping rich and structured the exchanged information. Reasoning and query answering for resource discovery in ubiquitous contexts is a critical issue. Mobile computing platforms –albeit increasingly effective and powerful– are still featured by hardware/software limitations. Most mobile inference engines currently provide only rule processing for materialization of entailments in a Knowledge Base (KB) [4,5], not supporting advanced inferences and extensive reasoning over ontologies [5]. Standard satisfiability and subsumption provide only boolean "yes/no" answers to queries. Non-standard inferences, like Concept Abduction and Concept Contraction, are needed to enable a more fine-grained semantic ranking as well as explanations of outcomes [2]. Implementation of tableaux algorithms on mobile devices exhibited serious memory impact [8]. Moreover, current Semantic Web reasoners cannot be ported without a significant re-write effort. In [7] a different design approach was followed to adapt non-standard inferences to ubiquitous computing. Expressiveness of logic language and axioms was limited in a way that structural algorithms could be adopted, but maintaining it enough for broad application areas. Similar principles motivated independent research work on $\mathcal{EL}^{++}$ structural reasoners [1] and the definition of OWL 2 *profiles*. This paper introduces a prototypical mobile reasoner[1] compliant with Semantic Web technologies through the OWL API

[1] Mini-ME, the Mini Matchmaking Engine – http://sisinflab.poliba.it/swottools/minime/

Table 1. Syntax and semantics of $\mathcal{ALN}$ constructs and simple-TBoxes

Name	**Syntax**	**Semantics**
Top	$\top$	$\Delta^{\mathcal{I}}$
Bottom	$\bot$	$\emptyset$
Intersection	$C \sqcap D$	$C^{\mathcal{I}} \cap D^{\mathcal{I}}$
Atomic negation	$\neg A$	$\Delta^{\mathcal{I}} \setminus A^{\mathcal{I}}$
Universal quantification	$\forall R.C$	$\{d_1 \mid \forall d_2 : (d_1, d_2) \in R^{\mathcal{I}} \rightarrow d_2 \in C^{\mathcal{I}}\}$
Number restriction	$\geq nR$	$\{d_1 \mid \sharp\{d_2 \mid (d_1, d_2) \in R^{\mathcal{I}}\} \geq n\}$
	$\leq nR$	$\{d_1 \mid \sharp\{d_2 \mid (d_1, d_2) \in R^{\mathcal{I}}\} \leq n\}$
Inclusion	$A \sqsubseteq D$	$A^{\mathcal{I}} \subseteq D^{\mathcal{I}}$
Equivalence	$A \equiv D$	$A^{\mathcal{I}} = D^{\mathcal{I}}$

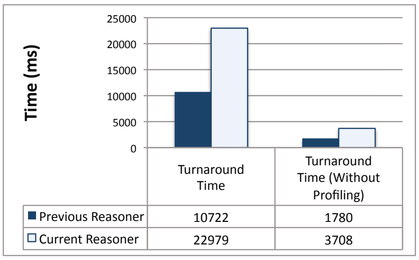
[3] and implementing both standard reasoning tasks (subsumption, classification, satisfiability) and non-standard inferences for semantic-based matchmaking (abduction and contraction [2]). It is developed in Java, with Android as target platform. The paper is so structured: Section 2 describes the system, while Section 3 reports on early experiments before concluding in Section 4.

2 System Outline

The system prototype is compatible with the Android Platform version 2.1 (API level 7) or later. It runs either as a *service* (*i.e.*, a background daemon) invoked by Android applications, or as a library. In the latter form, it runs unmodified on Java Standard Edition, version 6 or later. The system supports OWL 2 ontology language, in all syntaxes allowed by the adopted OWL API library. The adopted logic language is $\mathcal{ALN}$ (Attributive Language with unqualified Number restrictions) Description Logics in *simple-TBox* hypothesis [7], which keeps polynomial the computational complexity of standard and non-standard inferences. Main constructs are summarized in Table 1. The reasoning framework is based on structural algorithms. When a knowledge base is loaded, it is preprocessed with *unfolding* and *Conjunctive Normal Form* (CNF) *normalization* [7]. Beyond standard ***Concept Satisfiability*** (a.k.a. consistency) and ***Subsumption test***, the proposed system supports ***Concept Contraction*** and ***Concept Abduction*** non-standard inferences, enabling a semantic matchmaking and relevance ranking of available resources w.r.t. a request [2,7]. ***Ontology Satisfiability*** and ***Classification*** reasoning services over ontologies are also supported.

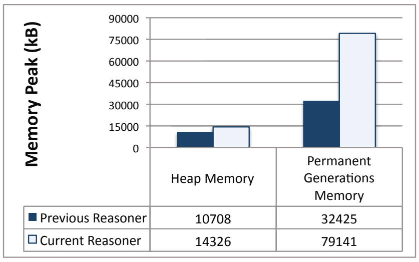
3 Performance Evaluation

Performance evaluation has been carried out as a comparison with a previous version [7] of the matchmaker, which was developed for the Java ME (Micro Edition) platform. This choice is motivated due to the lack of a mobile device supporting both Java ME and Android, hence it was unfeasible to test the old and new versions on a unique target handheld. The comparison has been done by running the same test suite with both matchmakers on a PC equipped with an Intel Core2 Duo T7300 CPU, 2 GB of RAM, Fedora 16 operating system



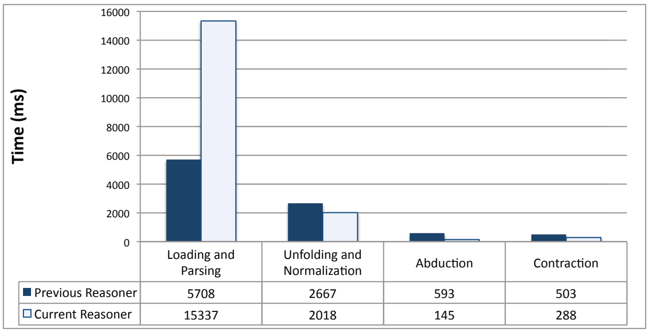
	Turnaround Time	Turnaround Time (Without Profiling)
■ Previous Reasoner	10722	1780
□ Current Reasoner	22979	3708

Fig. 1. Turnaround Time



	Heap Memory	Permanent Generations Memory
■ Previous Reasoner	10708	32425
□ Current Reasoner	14326	79141

Fig. 2. Memory Usage



	Loading and Parsing	Unfolding and Normalization	Abduction	Contraction
■ Previous Reasoner	5708	2667	593	503
□ Current Reasoner	15337	2018	145	288

Fig. 3. Processing Time

and Oracle Java SE 7u3. The test performs unfolding and normalization on an ontology in [7] –having a size of 528 kB– and 100 request/supply pairs –with an average size of 4.2 kB– and finally executes Concept Contraction and Concept Abduction inferences between each pair. The VisualVM Java profiler was used to measure execution times and memory consumption. The computational impact of the profiler is evident (see Figure 1): the actual execution time may be up to 6 times smaller. We further analyzed the time required by the main processing steps: loading and parsing input documents; unfolding and normalization; abduction; contraction. As shown in Figure 3, most of the time (66.7%) was spent by the OWL API to load and parse input files. It took about 15 seconds to process the ontology and 100 request/supply pairs whereas the previous code took 5.7 seconds (53.2% of total time). As far as normalization and unfolding are concerned, the old tool completed in 2667 ms, while the new one did the same in 2018 ms (1.32 times faster, approximately). Figure 3 also shows execution times for the abduction and contraction tasks. The old reasoner ran 100 abduction tests in 593 ms, while the current one did the same in 145 ms, *i.e.*, 4 times faster. For the contraction service, the previous software called the method 100 times without checking for concept compatibility first, resulting in a total time of 503 ms. The new reasoner took only 26.4 ms to perform 100 compatibility checks; 64 of them returned false and triggered contraction for a total of 262 ms, which is approximately 1.74 times faster than the old one.

Figure 2 shows the *heap* –used for dynamic allocation of new class instances and arrays– and *permanent generation* –where long-lived objects are moved– memory peak that was reached during the test. Memory consumption appears to have increased for both areas w.r.t. the previous system. Experiments evidence design and adopted optimization resulted in faster execution of non-standard inference services for semantic matchmaking w.r.t. previous tools. The integration of the OWL API improved flexibility in using KBs, and compatibility with Semantic Web languages. Nevertheless, it is a performance bottleneck for processing time and presumably for memory consumption.

4 Conclusion

A prototypical reasoner for mobile computing was presented. Early experiments evidenced correctness and efficiency of non-standard reasoning services, whereas the integration of the OWL API introduced a performance penalty. Future work will be devoted to OWL interface optimization and to the enhancement of both allowed reasoning features and supported logic languages.

Acknowledgments. The authors acknowledge partial support of national project ERMES (Enhance Risk Management through Extended Sensors) - PON (2011-2014).

References

1. Baader, F., Brandt, S., Lutz, C.: Pushing the EL envelope. In: International Joint Conference on Artificial Intelligence, vol. 19, p. 364. Lawrence Erlbaum Associates (2005)
2. Colucci, S., Di Noia, T., Pinto, A., Ragone, A., Ruta, M., Tinelli, E.: A Non-Monotonic Approach to Semantic Matchmaking and Request Refinement in E-Marketplaces. International Journal of Electronic Commerce 12(2), 127–154 (2007)
3. Horridge, M., Bechhofer, S.: The OWL API: a Java API for working with OWL 2 ontologies. In: Proc. of OWL Experiences and Directions 2009 (2009)
4. Kim, T., Park, I., Hyun, S., Lee, D.: MiRE4OWL: Mobile Rule Engine for OWL. In: 2010 IEEE 34th Annual Computer Software and Applications Conference Workshops (COMPSACW), pp. 317–322. IEEE (2010)
5. Motik, B., Horrocks, I., Kim, S.: Delta-Reasoner: a Semantic Web Reasoner for an Intelligent Mobile Platform. In: Twentyfirst International World Wide Web Conference (WWW 2012). ACM (to appear, 2012)
6. Ruta, M., Di Noia, T., Di Sciascio, E., Piscitelli, G., Scioscia, F.: RFID meets Bluetooth in a semantic based u-commerce environment. In: Proc. of the Ninth International Conference on Electronic Commerce, pp. 107–116. ACM (2007)
7. Ruta, M., Di Sciascio, E., Scioscia, F.: Concept abduction and contraction in semantic-based P2P environments. Web Intelligence and Agent Systems 9(3), 179–207 (2011)
8. Sinner, A., Kleemann, T.: KRHyper - In Your Pocket. In: Nieuwenhuis, R. (ed.) CADE 2005. LNCS (LNAI), vol. 3632, pp. 452–457. Springer, Heidelberg (2005)

(Re)Configuration Using Web Data: A Case Study on the Reviewer Assignment Problem*

Anna Ryabokon[1], Axel Polleres[2], Gerhard Friedrich[1], Andreas A. Falkner[2], Alois Haselböck[2], and Herwig Schreiner[2]

[1] Alpen-Adria Universität, Klagenfurt, Austria
{anna.ryabokon,gerhard.friedrich}@ifit.uni-klu.ac.at
[2] Siemens AG Österreich, Vienna, Austria
{axel.polleres,andreas.a.falkner,alois.haselboeck, herwig.schreiner}@siemens.com

Abstract. Constraint-based configuration is – on the one hand – one of the classical problem domains in AI and also in industrial practice. Additional problems arise, when configuration objects come from an open environment such as the Web, or in case of a reconfiguration. On the other hand, (re)configuration is a reasoning task very much ignored in the current (Semantic) Web reasoning literature, despite (i) the increased availability of structured data on the Web, particularly due to movements such as the Semantic Web and Linked Data, (ii) numerous practically relevant tasks in terms of using Web data involve (re)configuration. To bridge these gaps, we discuss the challenges and possible approaches for reconfiguration in an open Web environment, based on a practical use case leveraging Linked Data as a "component catalog" for configuration. In this paper, we present techniques to enhance existing review management systems with (re)configuration facilities and provide a practical evaluation.

1 Introduction

Constraint-based configuration, i.e. picking and linking a suitable set of components from a component catalog s.t. some predefined constraints are satisfied is a classical problem in AI and also in industrial practice. As users of the Web, we often solve such configuration tasks where in theory the "component catalog" is the Web, e.g. as private persons configuring an itinerary (flight, accommodation, hotel, etc.), or as academics, in the task of assigning expert reviewers to papers. Emerging availability of Linked Data on the Web [6] allows us to apply known configuration techniques to such problems that have been solved manually by Web search. However, Linked Data and adjacent Semantic Web communities focus mainly on taxonomic reasoning and ontologies (RDF Schema, OWL), to better structure Web data or infer implicit Web data, thus largely ignoring, to the best of our knowledge, reasoning tasks required for the configuration.

In this paper we show how configuration can be implemented in the framework of Linked Data. Moreover, we discuss an extension of the original problem for the cases

* This work was funded by FFG FIT-IT within the scope of the project RECONCILE (grant number 825071). An extended version of the paper can be found at http://proserver3-iwas.uni-klu.ac.at/reconcile/index.php/benchmarks

when configuration does not start from scratch. In this case a previously consistent configuration has to be adapted, i.e. a reconfiguration is required. Reconfiguration is an important task in the after-sale life-cycle of configurable products and services, because requirements are changing and there is a need to keep a product or a service up-to-date [2]. As it has been shown in the previous work [3], (re)configuration tasks can be efficiently handled by Answer Set Programming (ASP) [1] which extends logic programming and includes an expressive modeling language and solving tools [4].

The feasibility of (re)configuration based on Open Web Data in a practical scenario is demonstrated on the reviewer assignment problem (RAP): The decision if a paper is accepted on a conference depends on reviews made by the program committee. Therefore, it is required to assign every paper to a number of reviewers such that on the one hand these reviewers are interested in reading the paper and on the other hand have enough expertise. Our experiments show that the reviewer (re-)assignment task, leveraging Open Data and deploying methods of reconfiguration using SPARQL[9] and ASP, can efficiently be applied in practice.

2 (Re)Configuration Using Web Data

The reviewer assignment problem can be viewed as a configuration task where papers must be linked to reviewers such that a set of problem specific constraints are fulfilled. A preferred solution can be determined based on an optimization function which ranks the set of valid reviewer/paper assignments (i.e. configurations). In addition, reviewers typically specify their preferences in a process of bidding on the one hand, and on the other hand papers should be reviewed by the most competent reviewers among the program committee (PC). Whereas bidding preference are usually collected by a conference management system, the "expertise match" between reviewers and papers is normally not given explicitly and has to be estimated by program or area chairs while assigning the papers in existing systems, if it is taken into account at all.

In our approach we consider four categories of expertise: *conflict* if a reviewer is an author of the paper or biased by some other circumstances; *low*, *moderate* and *high* expertise. Moreover, the preferences of reviewers provided by the bidding process are encoded as: *conflict* of interest declared by a reviewer; *indifference*, i.e. no bid is provided; *weak* and *strong* willingness to review the paper. The latter two categories correspond to "I can review" and "I want to review" in EasyChair. The goal is to find a match between reviewers and papers s.t. different preferences are reconciled.

Goldsmith and Sloan [5] propose to view RAP as a variant of the stable marriage problem. A paper/reviewer assignment (marriage) is stable if there does not exist an alternative assignment in which paper *and* reviewer are individually better off than in their current assignment. Consequently, a reviewer cannot spot a paper which she prefers more and for which she has more competence compared to the current assignments. There are several variants of the stable matching which differ from the classic stable marriage problem: *polygamy* – reviewers can get more than one paper and vice versa; *incomplete lists* – some reviewers or papers cannot be assigned to each other; and *indifference* – the preferences express a preset number of preference classes. Each variation of the Stable Marriage Problem mentioned above can be solved in polynomial time [5]. The problem becomes more complicated and is known to be NP-hard if both incomplete lists and indifference occur [8] as in the paper assignment variant of the stable marriage problem. Therefore, a problem solving method which is able to deal with NP-hard problems is required and justifies the usage of ASP as a problem representation and solving framework.

In order to reduce the load on the solver we consider stability as a soft constraint and minimize the number of assignments which do not fulfill the stability property. In addition, we minimize the number of assignments of papers to reviewers with low and moderate expertise as well as of reviewers to papers with indifference and weak willingness. The encoding includes also the following hard constraints: (1) each paper must be assigned to a fixed number of reviewers and (2) fairness of the workload should be achieved. In order to distribute the papers among the reviewers as uniformly as possible, we add a *balancing criterion* as a hard constraint, which limits the minimum and maximum number of papers assigned to each reviewer.

We use Linked Data [6] to extract valuable information about connections between authors, such as recent co-authorship, joint affiliation or create expertise profiles. The first two types of connections allow automatic recognition of conflicts of interests. The profiles can be used to compare abstracts or keywords of published papers to submissions, thus determining the level of expertise. For a proof-of-concept implementation we have selected a fictitious set of reviewers composed of persons mentioned at `data.semanticweb.org`, as well as a subset of papers mentioned there as fictitious set of submissions. We also retrieve information about recent co-authorship from `http://dblp.l3s.de/d2r/` where we only link authors with unambiguous unique names present in both DBLP and `data.semanticweb.org`. In this paper, we make the reasonable assumption that the more similar the paper abstract and the abstracts of a reviewer are, the more competent the reviewer is to evaluate the paper. In order to compute these similarities we extracted abstracts of submitted papers and papers written by reviewers using SPARQL queries to `data.semanticweb.org`. The set of abstracts was analyzed by established methods from information retrieval and recommender systems [7] as follows: First, we derived a list of keywords relevant to the abstracts of papers and reviewers by considering only those terms which are provided by the PC chair of a conference in form of keywords. Next, we clean the keywords by employing a lemmatizer such as `http://morphadorner.northwestern.edu/`. The result of this process is a term vector for each reviewer and each paper, which we use in a standard *term frequency – inverse document frequency* (TF/IDF) weighting of the paper's abstract as well as of the union of abstracts for each reviewer. The similarities of vectors describing the papers and vectors describing the reviewers are computed by the *cosine similarity measure* [7].

The similarities were used to generate RAP instances of different size including a set of reviewers and papers as well as their bids and expertises. For each instance we applied ASP solver to find solutions of both configuration and reconfiguration problems. The instances to the latter problem are obtained by modifying corresponding solutions of the configuration problem. In the case of RAP modifications include situations when reviewers may drop out, papers could be withdrawn, or additional conflicts of interests may be discovered. The transformation of the legacy configuration possibly requires that some of its parts are deleted. Therefore, each reconfiguration problem instance includes requirements and transformation knowledge regarding reuse or deletion of parts of a legacy configuration.

We employ the modeling patterns described in [3] to formulate a reconfiguration problem instance. The principle idea is that for every element of the legacy configuration a decision has to be made whether or not to delete or reuse this element. The reused elements are complemented on demand by addition of new elements in order to fulfill all requirements. Note that in the reconfiguration case the optimization criteria of a configuration problem are extended with a criteria minimizing the costs associated with the transformation actions such as delete, reuse or create.

The evaluation results were performed using Potassco ASP collection [4] show that the reasoner was able to find a solution for all test instances and we obtained the best configuration solutions that can be computed within a timeout period of 900 seconds; proving optimality for such (re)configuration instances seems to be infeasible in practice. Note that, the performed experiments have realistic number for PC members and submissions comparable with e.g. the last ISWC conferences from which we took the data. For the reconfiguration problem instances the solver was able to find solutions with optimal reconfiguration costs in all but the two biggest cases from 16 which were tested. In these two cases the solver found solutions which reconfiguration costs are 20% and 8% higher than the optimum. A solution with the optimal reconfiguration costs was usually identified by the solver in the first 10 seconds of the solving process excluding the grounding time. The obtained results show that the proposed method is feasible for realistic reviewer assignment problems.

References

1. Brewka, G., Eiter, T., Truszczynski, M.: Answer set programming at a glance. Communications of the ACM 54(12), 92–103 (2011)
2. Falkner, A., Haselböck, A.: Challenges of Knowledge Evolution in Practice. In: Workshop on Intelligent Engineering Techniques for Knowledge Bases (IKBET 2010), pp. 1–5 (2010)
3. Friedrich, G., Ryabokon, A., Falkner, A., Haselböck, A., Schenner, G., Schreiner, H. (Re)configuration using Answer Set Programming. In: Proceedings of the IJCAI 2011 Workshop on Configuration, pp. 17–25 (2011)
4. Gebser, M., Kaufmann, B., Schaub, T.: Conflict-driven answer set solving: From theory to practice. Artif. Intell. 187, 52–89 (2012)
5. Goldsmith, J., Sloan, R.: The AI conference paper assignment problem. In: Proceedings of AAAI Workshop on Preference Handling for Artificial Intelligence, Vancouver, BC, Canada, pp. 53–57 (2007)
6. Heath, T., Bizer, C.: Linked Data: Evolving the Web into a Global Data Space, Synthesis Lectures on the Semantic Web: Theory and Technology, vol. 1. Morgan & Claypool (2011)
7. Jannach, D., Zanker, M., Felfernig, A., Friedrich, G.: Recommender Systems: An Introduction. Cambridge University Press (2010)
8. Manlove, D., Irving, R., Iwama, K., Miyazaki, S., Morita, Y.: Hard variants of stable marriage. Theoretical Computer Science 276(1-2), 261–279 (2002)
9. Prud'hommeaux, E., Seaborne, A.: SPARQL Query Language for RDF. W3C Recommendation, World Wide Web Consortium (2008), http://www.w3.org/TR/rdf-sparql-query/

Ontology-Based Data Access Using Views

Juan F. Sequeda[1], Marcelo Arenas[2], and Daniel P. Miranker[1]

[1] Department of Computer Science, The University of Texas at Austin
{jsequeda,miranker}@cs.utexas.edu
[2] Department of Computer Science, PUC Chile
marenas@ing.puc.cl

1 Our Position

The OWL 2 QL profile, which is based on DL-Lite$_R$, has been designed so that query answering is possible using relational database technology via query rewriting. Unfortunately, given a query Q posed in terms of an OWL 2 QL ontology[1] O, the size of the rewritten query, Q_o, which can be evaluated directly on the relational database, is worst case exponential w.r.t the size of Q and O [1]. This means that the computation and evaluation of Q_o can be costly. Recent research focuses on creating rewriting algorithms that generates Q_o with a smaller size [3].

In this paper, we propose a new approach to answering SPARQL queries on OWL 2 QL ontologies over existing relationally stored data. Our proposal is to replace answering queries via query rewriting with *answering queries using views*. Our position is that SQL infrastructure can be leveraged in order to support effective SPARQL query answering on OWL 2 QL ontologies over existing relationally stored data. We present preliminary results that support our position.

Our position is inspired by our previous work on Ultrawrap [5], a system that can execute SPARQL queries at almost equivalent execution speed as its semantically equivalent SQL queries. Previous experimental studies demonstrated that existing approaches were several magnitudes slower. Our main insight was to represent the relational data as RDF triples using views. SPARQL queries are syntactically translated to SQL queries which operate on the views. We observed that two relational optimizations are needed in order for relational database to effectively execute SPARQL queries: detection of unsatisfiable conditions and self-join elimination. Given this past history, we ask ourselves if we can apply this same approach to answering SPARQL queries on OWL 2 QL ontologies.

We first present the status quo with a running example adapted from [3]. We then present our proposed approach and show initial results that support our position. Finally, we present the open issues and our next steps.

[1] In this paper, we use the term OWL 2 QL ontology to refer only to a TBox. Therefore, we assume that it only contains axioms.

2 The Status Quo

The status quo of systems that answer queries over an OWL 2 QL ontology mapped to a relational database, also known as Ontology-based Data Access (OBDA), consists of the following input: an OWL 2 QL ontology O, a relational database D and an initial mapping M from the database D to the ontology O. Queries posed over the ontology O are answered in three steps:

1. *Query Rewriting:* given a conjunctive query Q, and the ontology O, compute a union of conjunctive queries Q_o, which is a rewriting of Q w.r.t O.
2. *Query Unfolding:* given the mapping M, and the rewritten query Q_o, compute a SQL query Q_{SQL}.
3. *Query Evaluation:* the SQL query Q_{SQL} is evaluated on the relational database.

OBDA systems such as Quest [4] or Mastro [2] implement these three steps. Additionally, Calvanese et al. [1] and Perez-Urbina et al. [3] present query rewriting algorithms and part of systems such as QuOnto[2], Owlgres[3] and REQUIEM[4].

Throughout this paper, we will use the following running example. Consider the following OWL 2 QL ontology O with concepts Student, Professor and Person, and the following axioms:

$$\text{Student} \sqsubseteq \text{Person}$$
$$\text{Professor} \sqsubseteq \text{Person}$$

Consider the following relational database D consisting of tables `PROF(PID,NAME)` and `STUD(SID,NAME)`, and consider a mapping M from the database D to the ontology O defined as follows using Datalog notation:

$$\text{Student}(x) \leftarrow \texttt{STUD}(x, y)$$
$$\text{Professor}(x) \leftarrow \texttt{PROF}(x, y)$$

Now assume that $Q(x)$ is the query Person(x) posed over O. The first step of the methodology just described is query rewriting: given the query Q and ontology O, Q is rewritten to Q_o:

$$Q_o(x) = \text{Student}(x) \vee \text{Professor}(x)$$

The next step is query unfolding: given the mapping M and the rewritten query Q_o, generate a SQL query Q_{SQL}:

```
SELECT SID FROM STUD   UNION ALL   SELECT PID FROM PROF
```

Finally, query Q_{SQL} is sent to the RDBMS where it is evaluated. Note that `UNION ALL` is used because it does not eliminate duplicate rows.

[2] http://www.dis.uniroma1.it/quonto/
[3] http://pellet.owldl.com/owlgres/
[4] http://www.cs.ox.ac.uk/projects/requiem/

3 Our Proposal: Answering Queries Using Views

Our proposal is to replace answering queries via query rewriting with answering queries using views. We do this in two steps. First, we represent the mappings using SQL views, which is a union of SQL queries. We call this view the *Tripleview*. Second, we compile the axioms of the OWL 2 QL ontology into SQL queries and add them to the Tripleview. Note that this is only done once and not for each time a query is executed. With our approach, we avoid rewriting queries at run time and let the relational database do the query unfolding. In order to further explain our approach, consider the same running example: the OWL 2 QL ontology O, the relational database D and the initial mapping M shown in Section 2. First, we represent the mapping M in the Tripleview:

```
CREATE VIEW Tripleview(s, p, o) AS
SELECT SID AS s, "rdf:type" AS p, "Student" AS o FROM STUD
UNION ALL
SELECT PID AS s, "rdf:type" AS p, "Professor" AS o FROM PROF
```

The next step is to compile the OWL 2 QL ontology axioms into SQL queries. Following our example, we need to add two additional queries to the Tripleview, generating the following:

```
CREATE VIEW Tripleview(s, p, o) AS
SELECT SID AS s, "rdf:type" AS p, "Student" AS o FROM STUD
UNION ALL
SELECT PID AS s, "rdf:type" AS p, "Professor" AS o FROM PROF
UNION ALL
SELECT SID AS s, "rdf:type" AS p, "Person" AS o FROM STUD
UNION ALL
SELECT PID AS s, "rdf:type" AS p, "Person" AS o FROM PROF
```

Everything up to now is done before any query is executed. After the Tripleview is created, we can execute queries. For example, consider the query Q written in SPARQL:

```
SELECT ?x WHERE { ?x rdf:type Person }
```

This query is then syntactically translated to a SQL query on the Tripleview and evaluated directly on the RDBMS without any further processing:

```
SELECT s FROM Tripleview WHERE p = "rdf:type" AND o = "Person"
```

4 Does This Work in Practice?

To test if our approach works in practice, we created the example database and implemented the Tripleview on Microsoft SQL Server and executed the query. The resulting query plan is shown in Fig. 1.

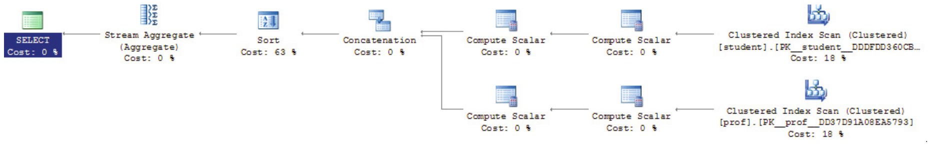


Fig. 1. The physical query plan for our running example on Microsoft SQL Server

The logical query plan consists of the Tripleview with the union of four queries. We observe that the physical query plan generated by SQL Server consists of a union of two queries. Therefore, the SQL optimizer determined which queries in the Tripleview were not going to satisfy the original query and transformed the original logical query plan into the optimal physical plan shown in Fig. 1. This transform is the *detection of unsatisfiable conditions* optimization. Additionally, we observe that query Q_{SQL} generates the same query plan.

This preliminary result supports our position that by answering queries using views, the SQL infrastructure can be leveraged to support effective SPARQL query answering on OWL 2 QL ontologies over existing relationally stored data. Notice that we are not claiming that query rewriting is not needed. On the contrary, we are trying to get the best of both worlds. For example, query rewriting algorithms, such as the ones presented in [2,3], would need to be modified to generate the views. Additionally, we need to investigate how much of OWL 2 QL can be represented in views. Our first findings show that any axioms that is not of the form $A \sqsubseteq \exists R$ (i.e. sub-class, sub-property, equivalent class, equivalent property, etc), can be represented in views. Nevertheless, the possibility of representing existential axioms of the form $A \sqsubseteq \exists R$ depends on the ability of a relational database to generate the equivalent of a blank node.

We are currently implementing this approach in Ultrawap and planning to evaluate it against other OBDA systems.

References

1. Calvanese, D., Giacomo, G., Lembo, D., Lenzerini, M., Rosati, R.: Tractable reasoning and efficient query answering in description logics: The dl-lite family. J. Autom. Reason. 39(3), 385–429 (2007)
2. Calvanese, D., Giacomo, G.D., Lembo, D., Lenzerini, M., Poggi, A., Rodriguez-Muro, M., Rosati, R., Ruzzi, M., Savo, D.F.: The mastro system for ontology-based data access. Semantic Web 2(1), 43–53 (2011)
3. Pérez-Urbina, H., Horrocks, I., Motik, B.: Efficient Query Answering for OWL 2. In: Bernstein, A., Karger, D.R., Heath, T., Feigenbaum, L., Maynard, D., Motta, E., Thirunarayan, K. (eds.) ISWC 2009. LNCS, vol. 5823, pp. 489–504. Springer, Heidelberg (2009)
4. Rodríguez-Muro, M., Calvanese, D.: Quest, a system for ontology based data access. In: OWLED (2012)
5. Sequeda, J.F., Miranker, D.P.: Ultrawrap: Sparql execution on relational data. Technical Report TR-12-10, The University of Texas at Austin, Department of Computer Sciences (2012)

Description Logic Knowledge Base Exchange

Elena Botoeva
Supervised by Diego Calvanese

KRDB Research Centre, Free Univ. of Bozen-Bolzano
piazza Dominicani 3, 39100 Bolzano, Italy
Botoeva@inf.unibz.it

1 Problem Definition

1.1 Introduction

Data exchange is a field of database theory that deals with transferring data between differently structured databases, with motivation coming from industry [21,17]. The starting point of intensive investigation of the problem of data exchange was given in [14] where it was defined as, given data structured under a source schema and a mapping specifying how it should be translated to a target schema, to transform the source data into data structured under the target schema such that it accurately reflects the source data w.r.t. the mapping. This problem has been studied for different combinations of languages used to specify the source and target schema, and the mappings [8]. Most of the results in the literature consider tuple generating dependencies (tgds) as the language to specify mappings. Tgds allow one to express containment of conjunctive queries, and have been widely employed in other areas of database theory. Furthermore, once a target instance is materialized, one might want to perform query answering over it.

A fundamental assumption in the traditional data exchange setting is that the source is a complete database: every fact is either true or false. Conversely, it is not the case for target instances: incomplete information can be introduced by the mapping layer (see also [19]). This implies that we cannot perform data exchange repeatedly on the obtained target instance. It also creates difficulties when combining composition and inversion operators, which are basic operators for metadata management [9].

Recently there has been an interest in data exchange with data in the source incompletely specified, which means that (possibly infinitely) many source instances are implicitly represented. The work by [4] is fundamental in this respect: they propose a general framework for exchanging incomplete information based on the notion of *representation system*, as a mechanism to describe in a finite way (infinitely) many complete instances of a data schema. And more importantly, they develop a general knowledge exchange framework for the case when the source is a knowledge base, not just a database instance.

1.2 My Research Problem

In this thesis, inspired by [4], we go beyond the traditional data exchange setting, and study in depth knowledge base exchange, that is, exchange between knowledge bases.

We start by refining the representation systems framework to Description Logic (DLs) knowledge bases (KBs) [7]. DLs are decidable fragments of first order logic (FOL), and widely used as (underlying logical basis for) formalisms in Knowledge Representation, Semantic Web, Ontology-Based Data Access, etc. They provide nice modeling and reasoning capabilities exhibiting a trade-off between expressiveness of a logic and computational complexity of reasoning in that logic[1] [7,18].

DLs appeal as an appropriate specification language, both for representing mappings and implicit knowledge in the source, as their limited expressive power does not allow for complex cases that emerge when using tgds where one has the full expressive power of conjunctive queries. Therefore many undecidable problems may turn our to be decidable when using DLs. First candidates to be considered are logics of two families of light-weight DLs: the *DL-Lite* family [11,5], and the $\mathcal{EL}$ family [6].

With the development of Semantic Web and Linked Data the problems of restructuring, combining and translating DL KBs become relevant in many contexts. This could be translating a KB in one natural language to another; combining several KBs in order to create one in a new, unique vocabulary; simply modifying the structure of existing KBs according to the needs.

The *goal of my research work* is to study exchange of DL KBs, namely:

- to define relevant notions and reasoning problems in the context of DL KB exchange,
- to develop results and techniques for KB exchange for various choices of DLs as the specification language, establishing tractability and decidability bounds,
- to explore the behavior of the DL KB exchange framework w.r.t. metadata management operators.

2 Related Work

2.1 Data Exchange

Data exchange deals with transferring data between differently structured databases. As it was mentioned in the introduction, the basic problem of data exchange is to materialize an instance of the target schema given an instance of the source schema and a schema mapping, where traditionally a schema mapping is constituted by a set of source-to-target tuple generating dependencies (st-tgds) and a set of target constraints, which can be target tgds or target equality generating dependencies (egds). Both tgds and egds are classes of constraints widely explored in the database community, for formal definitions refer to [14]. Thus, the materialized target instance (called solution) should be a correct translation of the source instance with respect to the st-tgds such that it also satisfies the target constraints. Various syntactic restrictions on (target) tgds have been considered in order to guarantee termination of the chase procedure employed to compute solutions: full tgds and weakly acyclic tgds. Full tgds do not contain existentially quantified variables, so the chase is always finite. The latter notion of weak acyclicity allows for more expressivity than full tgds, while still providing termination of the chase. Due to

[1] See e.g., `http://www.cs.man.ac.uk/~ezolin/dl/` for a summary of complexity results.

existentially quantified variables in arbitrary tgds, there can exist infinitely many non-isomorphic solutions. Fagin et al. [14] argued that *universal* solutions are the preferred solutions to be materialized in data exchange as they are the 'most general' solutions: they can be homomorphically embedded into other solutions. Next, the 'best' universal solution was defined to be the smallest universal solution, called the *core* [15].

Known results on data exchange can be summarized as follows.

- There exists a data exchange setting for which the problem of existence of a solution is undecidable.
- For a source instance and a mapping consisting of a set of st-tgds, a set of target egds, and a weakly acyclic set of target tgds the problems of deciding existence of a solution and computing a universal solution [14]; computing the core of the universal solutions [16]; and computing certain answers to a query over the target solutions [14] can be solved in polynomial time.

2.2 Ontology Alignment

There exists a whole body of work on ontology alignment, also called ontology mapping and ontology matching [12,13]. The task of ontology alignment is given two ontologies to find correspondences between semantically related entities of ontologies resulting in a mapping between the ontologies. Such a problem arises in the contest of ontology merging, integration and alignment, which can be considered as an ontology reuse process. Despite seeming similarity of the ontology alignment and knowledge base exchange problems, their problem can be seen as the opposite of ours, since we are given a source KB and a mapping, and the task is to materialize a target KB, while they are given two ontologies and their task is to find a mapping between them (which might not have a direction).

2.3 Knowledge Exchange

The field of knowledge and/or incomplete data exchange is relatively new. Here we shortly present the framework defined in [4].

Arenas, Perez and Reutter [4] proposed a general framework for exchange of incomplete data. It extends the standard data exchange framework by allowing the source data to be incompletely specified and, thus, (possibly infinitely) many source instances to be represented. A notion of *representation system* is introduced as a mechanism to represent multiple instances of a data schema. Then they consider the problem of knowledge exchange, where KBs are constituted by database instances (explicit information) and sets of tgds (implicit information), and mappings are sets of st-tgds, and show an undecidability result for the case of arbitrary tgds (used to specify KBs) and full st-tgds, and a tractability result for the case of full tgds and arbitrary st-tgds.

3 The Research Plan

3.1 Research Questions

In this PhD research work we deal with exchanging Description Logic (DLs) knowledge bases (KBs). The main research tasks can be summarized as follows:

- define the framework of DL KB exchange:
  - give necessary definitions, and
  - determine relevant reasoning problems.
- study the problem of KB exchange for various choices of DLs
  - start with lightweight DLs, such as *DL-Lite* and $\mathcal{EL}$,
  - investigate which DLs have 'nice' properties in the context of KB exchange, such as tractability, decidability and closure under metadata operators.

More precisely, the research questions we are trying to answer within this thesis are:

- what is a 'good' notion of solution in the context of KB exchange;
- for a given DL
  - what is the complexity of computing a (good) solution;
  - is a solution always expressible in the same DL;
  - for each defined reasoning problem, what is the complexity of deciding it;
  - how to compute composition and inverse of mappings, and can they be expressed in the same DL;
- which DL exploits nice computational and modeling properties for KB exchange.

3.2 Contribution

Firstly, we have specialized the framework for KB exchange proposed in [4] to the case where as a representation system we use DL KBs constituted by a TBox, representing implicit information, and an ABox, representing explicit information, and where mappings are sets of DL inclusions. In such a setting, one is given a source DL KB and a DL mapping, and the problem is to materialize a target DL KB such that it is a solution for the source KB under the mapping.

Then, we defined a novel notion of representability whose aim was to understand the capacity of (universal) solutions to transfer implicit knowledge. Presence of the latter in the target is crucial: maximizing implicit knowledge (TBox axioms) allows for minimizing explicit information (ABox assertions), hence, for having smaller solutions (specially in the case of data intensive applications). A source TBox is said to be representable under a mapping if there exists a target TBox such that combined with an ABox that depends only on the source ABox and the mapping, for an arbitrary source ABox, gives a universal solution for the source KB under the mapping. Representability of a source TBox under a mapping implies that it only remains to transfer the source ABox via the mapping.

Next, we argued that the preferred solutions in our framework should not be the standard universal solutions based on the correspondence between models of source and target KBs [2]. Indeed, we showed that such kinds of solutions present serious limitations: if a universal solution exists then its TBox is empty, so it is impossible to represent implicit source information, which in turn may lead to exponentially large target ABoxes. To overcome these drawbacks, we introduced the weaker notion of *(universal) $\mathcal{Q}$-solution*, for a query language $\mathcal{Q}$. Intuitively, these solutions cannot be distinguished from the universal solutions by means of $\mathcal{Q}$-queries, therefore they are as good as the universal solutions in the scenarios where query answering is the reasoning task performed over the target.

Finally, we developed results and techniques for KB exchange and for the $\mathcal{Q}$-represent-ability problem in the case where $\mathcal{Q}$ is unions of conjunctive queries (UCQs), and KBs are expressed in two significant DLs of the *DL-Lite* family [11], namely *DL-Lite*$_{RDFS}$ and *DL-Lite*$_{\mathcal{R}}$. The case of *DL-Lite*$_{RDFS}$ is particularly interesting, since this DL corresponds to the FOL fragment of RDFS [10], the widely adopted standard Semantic Web language. We obtained that for *DL-Lite*$_{RDFS}$ source TBoxes and mappings, the problem of computing (universal) solutions is decidable in polynomial time [1]. In the case of *DL-Lite*$_{\mathcal{R}}$, the problem of computing/checking (universal solutions) turned out to be PSPACE-hard and in EXPTIME (not published yet). Moreover, we showed that the problem of computing UCQ-representations can be solved in polynomial time for *DL-Lite*$_{\mathcal{R}}$ TBoxes and mappings without disjointness assertions [3]. Later we addressed also general *DL-Lite*$_{\mathcal{R}}$ TBoxes and mappings, and showed that the problem of checking whether the given target TBox is a UCQ-representation of the given source TBox under the given mapping can be solved in polynomial time. Notice that in *DL-Lite*$_{\mathcal{R}}$, representability of a TBox implies an algorithm to construct universal UCQ-solutions of polynomial size. The results we have obtained till now can be found in more detail in [1], [2], and [3].

3.3 Open Problems

The area of research on KB exchange, in particular DL KBs, is new and there are many interesting problems that remain open. From the theoretical point of view, the investigation can be done in two 'ortogonal' dimensions, one dimension along the reasoning problems to be considered, another one along the logics used as the specification language (for mappings, and source and target TBoxes). Below we list some open problems and the logics for which they should be tackled.

1) reasoning problems:
   - the problem of deciding (and computing) universal solutions (done for *DL-Lite*$_{\mathcal{R}}$);
   - the problem of deciding (and computing) representability (done for *DL-Lite*$_{\mathcal{R}}$);
   - the problem of deciding (and computing) composition and inversion of mappings;
2) DLs as the underlying formalism
   - *DL-Lite*$_{\mathcal{R}}$, *DL-Lite*$_{\mathcal{F}}$, *DL-Lite*$_{\mathcal{A}}$, *DL-Lite*$^{\mathcal{H}}_{horn}$, *DL-Lite*$^{\mathcal{H}}_{bool}$,
   - $\mathcal{EL}$, $\mathcal{ELH}$, $\mathcal{EL}^+$,
   - other expressive DLs ? (decidability borders to be understood).

We also plan to implement some of the devised algorithms (first of all, representability) and possibly a prototype system for KB exchange. It would be interesting to see how we can exploit the existing systems such as Clio [17] for exchanging the ABox information and, for instance, Quest [20] for answering queries over the target KB.

References

1. Arenas, M., Botoeva, E., Calvanese, D.: Knowledge base exchange. In: Proc. of DL 2011, vol. 745. CEUR (2011), ceur-ws.org
2. Arenas, M., Botoeva, E., Calvanese, D., Ryzhikov, V., Sherkhonov, E.: Exchanging description logic knowledge bases. In: Proc. of KR 2012 (2012)

3. Arenas, M., Botoeva, E., Calvanese, D., Ryzhikov, V., Sherkhonov, E.: Representability in dl-liter knowledge base exchange. In: Proc. of the 25th Int. Workshop on Description Logics, DL 2012 (2012)
4. Arenas, M., Pérez, J., Reutter, J.L.: Data exchange beyond complete data. In: Proc. of PODS 2011, pp. 83–94 (2011)
5. Artale, A., Calvanese, D., Kontchakov, R., Zakharyaschev, M.: The *DL-Lite* family and relations. J. of Artificial Intelligence Research 36, 1–69 (2009)
6. Baader, F., Brandt, S., Lutz, C.: Pushing the $\mathcal{EL}$ envelope. In: Proc. of the 19th Int. Joint Conf. on Artificial Intelligence (IJCAI 2005), pp. 364–369 (2005)
7. Baader, F., Calvanese, D., McGuinness, D., Nardi, D., Patel-Schneider, P.F. (eds.): The Description Logic Handbook: Theory, Implementation and Applications. Cambridge University Press (2003)
8. Barcelò, B.: Logical foundations of relational data exchange. SIGMOD Record 38(1), 49–58 (2009)
9. Bernstein, P.A.: Applying model management to classical meta data problems. In: CIDR (2003)
10. Brickley, D., Guha, R.V.: RDF vocabulary description language 1.0: RDF Schema. W3C Recommendation, World Wide Web Consortium (February 2004), http://www.w3.org/TR/rdf-schema/
11. 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. of Automated Reasoning 39(3), 385–429 (2007)
12. Choi, N., Song, I.-Y., Han, H.: A survey on ontology mapping. SIGMOD Rec. 35(3), 34–41 (2006)
13. Euzenat, J., Meilicke, C., Stuckenschmidt, H., Shvaiko, P., Trojahn, C.: Ontology Alignment Evaluation Initiative: Six Years of Experience. In: Spaccapietra, S. (ed.) Journal on Data Semantics XV. LNCS, vol. 6720, pp. 158–192. Springer, Heidelberg (2011)
14. Fagin, R., Kolaitis, P.G., Miller, R.J., Popa, L.: Data exchange: Semantics and query answering. Theoretical Computer Science 336(1), 89–124 (2005)
15. Fagin, R., Kolaitis, P.G., Popa, L.: Data exchange: Getting to the core. ACM Trans. on Database Systems 30(1), 174–210 (2005)
16. Gottlob, G., Nash, A.: Data exchange: Computing cores in polynomial time. In: Proc. of the 25th ACM SIGACT SIGMOD SIGART Symp. on Principles of Database Systems (PODS 2006), pp. 40–49 (2006)
17. Haas, L.M., Hernández, M.A., Ho, H., Popa, L., Roth, M.: Clio grows up: from research prototype to industrial tool. In: Proceedings of the 2005 ACM SIGMOD International Conference on Management of Data, SIGMOD 2005, pp. 805–810. ACM, New York (2005)
18. Horrocks, I., Sattler, U.: A tableaux decision procedure for $\mathcal{SHOIQ}$. In: Proc. of the 19th Int. Joint Conf. on Artificial Intelligence (IJCAI 2005), pp. 448–453 (2005)
19. Libkin, L., Sirangelo, C.: Data exchange and schema mappings in open and closed worlds. J. of Computer and System Sciences 77(3), 542–571 (2011)
20. Rodriguez-Muro, M., Calvanese, D.: Quest, a system for ontology based data access. In: Proc. of the 9th Int. Workshop on OWL: Experiences and Directions (OWLED 2012) (to appear, 2012)
21. Shu, N.C., Housel, B.C., Taylor, R.W., Ghosh, S.P., Lum, V.Y.: Express: a data extraction, processing, and restructuring system. ACM Trans. Database Syst. 2, 134–174 (1977)

Research Summary: Datalog-Based Data Access

Cristina Civili

Dipartimento di Ingegneria Informatica, Automatica e Gestionale Antonio Ruberti
Sapienza Università di Roma, Italy
civili@dis.uniroma1.it

Abstract. This paper presents the research summary of a Ph.D. plan concerning the study of Ontology-Based Data Access (OBDA) for ontologies expressed in Datalog-based formalisms, i.e., Datalog rules that allow the use of existential variables in the head.

1 Introduction

The research project I am undertaking during my Ph.D. program carries on the work I started in my Master Thesis and concerns the use of Datalog-based formalisms as an alternative to Description Logics for modeling ontologies in Ontology-Based Data Access systems.

The interest in ontological languages has both theoretical reasons and practical implications in several fields such as Knowledge Representation, Semantic Web and Information Integration. Despite the existence in the literature of several works concerning this topic, the identification of an ontological language showing an acceptable balance between its expressive power and the computational complexity of reasoning tasks is still an open challenge that is slowing down the commercial spread of semantic tools.

I believe that Datalog-based formalisms could be the answer to this issue, therefore I am carrying forward a formal investigation on a broad family of languages, called Datalog$^{\pm}$.

2 Current State of the Art of the Research Field

The last few years have seen a growing interest in ontologies, a flexible tool for the formalization of knowledge bases whose use in Semantic Web [4] and Information Integration is well-established.

Lately, the focus has been on their application to data access: Ontology-Based Data Access (OBDA) systems are a new promising frontier for knowledge representation and database research.

In OBDA systems, ontologies are used as an additional layer of information placed upon traditional databases with the purpose of semantically enriching them. Typically, the ontology contains only the terminological part of the knowledge base, while the DBMS is used to manage the actual data. Query answering in such systems is often accomplished through expansion techniques, such as query rewriting [6,14,8].

Up to now, Description Logics (DLs) have been used as the logical languages for modeling ontologies. In the most expressive of these languages, the main ontological reasoning and query answering tasks are undecidable, whereas even in the decidable fragments they are still computationally very hard.

One recent research direction is to find more expressive formalisms for which query answering is decidable and also tractable in data complexity, in order to effectively use them on top of large relational databases. To this research field belong both the DL-Lite family of languages [10], i.e., lightweight DLs designed with the purpose of reducing the data complexity of query answering (polynomial-time data complexity), and the Datalog$^{\pm}$ family of languages [5,6,7,9], i.e., formalisms whose syntax is based on variants of the Datalog language [1,11].

More generally, in order to propose more expressive ontological languages, there is a high interest in classifying rules with existential variables in the conclusion (∀∃-rules) [3,15]. Since this feature, called value invention, typically makes reasoning tasks such as conjunctive query answering undecidable, it is crucial to identify classes that are decidable, tractable and FOL-Rewritable (see below).

3 Beyond the State of the Art

The most recent trend concerning ontologies suggests to use them not only as a modeling tool, but also as a way for enhancing traditional databases. Currently there is a huge interest in developing database management systems enhanced with advanced reasoning and query processing mechanisms. This interest has not only theoretical reasons, but also a practical one: developing a new technology concerning data management is seen as a great commercial opportunity, since enterprise data would be the ideal target for ontological reasoning.

The necessity of combining ontological reasoning with database techniques has emerged both in the database and in the knowledge representation research communities, but we have yet to see to the actual breakthrough of semantic technologies.

In OBDA systems, an extensional relational database (the ABox) is combined with an ontological theory (the TBox) describing rules and constraints that derive new knowledge from the extensional data. A query is not just answered against the database, but against the whole logical theory. This requires a new approach to query answering, based on expansions and query rewriting techniques.

To be actually competitive on the market, these systems must guarantee a complexity of query answering not greater than that of the ones currently in use. In other words, the current research challenge is to find formalisms for representing ontologies that, on one hand, are powerful enough to satisfy the most common modeling needs and, on the other hand, keep the tractability of conjunctive query answering.

This is already true for some DLs, such as DL-Lite, for which the complexity of conjunctive query answering is LOGSPACE, and for other formalisms belonging to the Datalog$^{\pm}$ family, namely Linear, Sticky and Sticky Join Datalog$^{\pm}$.

These languages share one important and well-studied property: the FOL-Rewritability of query-answering, i.e., a pair (Σ, q), where Σ is an ontology and q is a conjunctive query (CQ), can be rewritten as a first order query q' such that, given a database D, it holds that $D \cup \Sigma \models q$ iff $D \models q'$.

Since each FOL query can be equivalently written in SQL, this means that a CQ q based on an ontology Σ can be rewritten as an SQL query over the original database. This is an important property that can be conveniently exploited in ontological reasoning applied to existing relational databases, therefore there is a particular interest in languages that are FOL-Rewritable.

It has been shown that the languages of the DL-Lite family are the maximal DLs supporting efficient query answering over large amounts of instances [10,2] and that DLs are probably a more suitable tool for representing ontologies than the Datalog-based formalisms [16], mostly because of the lack of value invention in plain Datalog. However, in the last few years, the interest in Datalog-based formalisms as an alternative to DLs for modeling ontologies has grown. Extending plain Datalog with the possibility of existential variables in the head rules was the first step forward in this direction. In the following I will refer to this extension as Datalog$^{\pm}$, but in the literature such rules are also generally referred to as $\forall\exists$-rules [3], Datalog$^{\exists}$ [15] or tuple-generating dependencies (TGDs) and equality-generating dependencies (EGDs).

Different syntactic restrictions on the form of rules determine the existence of different variants of Datalog$^{\pm}$ that belong to the family and that are suited for efficient ontological reasoning, in particular, for tractable ontology-based query answering.

I want to take this study further by using the Datalog$^{\pm}$ class as a point of reference. I believe that it is possible to identify classes in Datalog$^{\pm}$ that generalize DL-Lite and other ontology languages, while, at the same time, keep the property of FOL-Rewritability (or at least tractability) of query answering with respect to data complexity.

Considering the advantages as well as the limits of the formalisms described in the literature, I propose thus a broader approach to the problem, with the aim of identifying more expressive formalisms that can be used in OBDA systems.

Hence, the research I am undertaking concerns pursuing the following objectives.

1. The identification of new *decidable* and *tractable* classes of Datalog$^{\pm}$ programs. In particular, I am interested in FOL-Rewritable classes, for which the technique of query answering through query rewriting has proved to be successful.
2. The identification of effective algorithms for some of such classes, i.e., sound, complete and terminating algorithms, which includes both techniques for checking the membership of programs to such classes and techniques for query answering under such classes.
3. The development of an actual OBDA system based on Datalog$^{\pm}$, in order to check the efficiency of the above-mentioned classes and to provide measurements about their performance in real reasoning systems.

In order to accomplish these results, I started from current studies on the subject and I am trying to apply a mix of existing and new techniques. In particular:

1. I aim to accomplish the first objective with techniques used in current literature together with demonstration techniques used in DLs, in particular in DL-Lite;
2. I aim to accomplish the second objective generalizing query answering and query rewriting techniques used in DLs, together with answer set programming (ASP) techniques;
3. I aim to accomplish the third objective following the path of QuOnto/Mastro system, developed at Sapienza University of Rome, which is currently used in real OBDA projects. The basic idea is to develop an analogous system based on Datalog$^{\pm}$.

4 Current Status of the Research Plan

During the first months of my Ph.D. program, I studied reasoning, and in particular query answering, over databases with tuple-generating dependencies (TGDs), focusing on classes of TGDs for which conjunctive query answering is FOL-Rewritable.

In my first paper [12], we defined the class of weakly recursive TGDs and, under some restrictions (i.e. we focused our attention on simple TGDs), we proved that this class comprises and generalizes every previously known FOL-rewritable class of TGDs. Moreover, we defined a new algorithm that is able to compute the first-order rewriting of conjunctive queries over weakly recursive TGDs.

The intuition behind weakly recursive TGDs is related to the notion of position graph, a structure that encodes some important characteristics of a program P that can be used to determine whether P is FOL-rewritable or not. In particular, through the concept of m-edges and s-edges, respectively related to the "loss" of distinguished variables moving from the head to an atom of the body, and to the "split" of a head variable in two or more atoms of the body, we are able to identify "dangerous" cycles in the position graph. For more details on the definition of weakly recursive TGDs refer to [12].

In another work [13], we studied query answering over ontologies expressed in Datalog$^{\pm}$ from a different point of view, i.e. focusing on subclasses UCQs, rather than on the whole class, through the introduction of the notion of conjunctive query pattern (CQP). Given a class of queries $\mathcal{Q}$ expressed by a CQP, we studied decidability and complexity of answering queries in $\mathcal{Q}$ over a Datalog$^{\pm}$ program, and we defined an algorithm that, given a Datalog$^{\pm}$ program P and a class of queries $\mathcal{Q}$, is able to compute a simplified Datalog$^{\pm}$ program P' that is equivalent to P with respect to answering queries in $\mathcal{Q}$.

Currently, we are studying more expressive classes of FOL-rewritable (or at least decidable) TGDs that comprise the class of weakly recursive TGDs, as well as more efficient algorithms for computing the first-order rewriting of conjunctive queries over such TGDs.

5 Expected Achievements

The main purpose of my Ph.D. research program is to generalize the OBDA approach to classes of ontological languages that are more complex and more expressive than DLs.

DLs, despite their good capabilities in modeling ontologies, cannot capture everything that can be expressed through TGDs, in particular, they are unable to express complex forms of joins. Considering the nature of OBDA systems, in which the management of the factual part of the ontology is delegated to a DBMS, such a lack in the ontological languages represents a huge limitation in a significant amount of real life cases.

Actually, there is no need to give up on such an important feature: we already know some Datalog-based formalism, like Sticky and Sticky-Join Datalog$^{\pm}$, that can express powerful joins and keep the FOL-Rewritability of conjunctive query answering, while even Linear Datalog$^{\pm}$ is able to express inclusion dependencies that generalize DL-Lite.

However, it is important to notice that there is a huge margin of development in this field: up to now, the two most expressive classes known in the literature do not maximize the set of Datalog$^{\pm}$ rules that can be translated in FOL. In particular, Sticky Datalog$^{\pm}$ is able to express only body joins that involve distinguished variables, while Sticky-Join Datalog$^{\pm}$, which can express more complex forms of join, cannot be checked in polynomial time, i.e., the complexity of deciding whether a Datalog program belongs to the class is PSPACE.

Therefore I believe that a broader approach to Datalog$^{\pm}$ rules, aimed at studying classes that show relevant property such as FOL-Rewritability, could lead to extremely important and valuable results.

The introduction of several new Datalog$^{\pm}$-based formalisms for which query answering is decidable, tractable and FOL-Rewritable would make it possible to use already existing query answering techniques based on query rewriting. These new formalisms could constitute an alternative to DLs for modeling ontologies in OBDA systems and be useful in all the cases in which the modeling capabilities of DLs have proved unsatisfactory.

Moreover, the introduction of sound, complete and terminating algorithms for such new formalisms would allow to effectively check the membership of Datalog$^{\pm}$ programs to such classes and to answer queries, in order to use them in real working systems.

Finally, the development of an actual OBDA system based on Datalog$^{\pm}$, on one hand would be able to show the capabilities of such new formalisms, and, on the other hand, could provide measurements about their performance in real reasoning systems that could be compared to the ones of current analogous systems.

Since OBDA is currently a hot topic of study in both knowledge representation and database research community, I believe that these results would be an actual breakthrough in these fields, with both theoretical and practical implications.

In particular, such results would pave the way towards the development of semantic-based data access systems much more powerful than the current technologies, and this is especially important in Semantic Web research.

References

1. Abiteboul, S., Hull, R., Vianu, V.: Foundations of Databases. Addison Wesley Publ. Co. (1995)
2. Artale, A., Calvanese, D., Kontchakov, R., Zakharyaschev, M.: The *DL-Lite* family and relations. J. of Artificial Intelligence Research 36, 1–69 (2009)
3. Baget, J.-F., Leclère, M., Mugnier, M.-L., Salvat, E.: On rules with existential variables: Walking the decidability line. Artificial Intelligence 175(9-10), 1620–1654 (2011)
4. Berners-Lee, T., Hendler, J., Lassila, O.: The Semantic Web. Scientific American 284(5), 34–43 (2001)
5. Calì, A., Gottlob, G., Lukasiewicz, T.: A general Datalog-based framework for tractable query answering over ontologies. In: Proc. of the 28th ACM SIGACT SIGMOD SIGART Symp. on Principles of Database Systems (PODS 2009), pp. 77–86 (2009)
6. Calì, A., Gottlob, G., Pieris, A.: Advanced processing for ontological queries. Proc. of the VLDB Endowment 3(1), 554–565 (2010)
7. Calì, A., Gottlob, G., Pieris, A.: Query Answering under Non-guarded Rules in Datalog+/-. In: Hitzler, P., Lukasiewicz, T. (eds.) RR 2010. LNCS, vol. 6333, pp. 1–17. Springer, Heidelberg (2010)
8. Calì, A., Gottlob, G., Pieris, A.: Query rewriting under non-guarded rules. In: Proc. of the 4th Alberto Mendelzon Int. Workshop on Foundations of Data Management (AMW 2010) (2010)
9. Calì, A., Gottlob, G., Pieris, A.: New expressive languages for ontological query answering. In: Proc. of the 25th AAAI Conf. on Artificial Intelligence (AAAI 2011) (2011)
10. 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. of Automated Reasoning 39(3), 385–429 (2007)
11. Ceri, S., Gottlob, G., Tanca, L.: What You Always Wanted to Know About Datalog (And Never Dared to Ask). IEEE Trans. on Knowl. and Data Eng. 1, 146–166 (1989)
12. Civili, C., Rosati, R.: A broad class of first-order rewritable tuple-generating dependencies. In: Proc. of the 2nd Datalog 2.0 Workshop (to appear, 2012)
13. Civili, C., Rosati, R.: Query Patterns for Existential Rules. In: Krötzsch, M., Straccia, U. (eds.) RR 2012. LNCS, vol. 7497, pp. 42–57. Springer, Heidelberg (2012)
14. Gottlob, G., Orsi, G., Pieris, A.: Ontological queries: Rewriting and optimization. In: Proc. of the 27th IEEE Int. Conf. on Data Engineering (ICDE 2011), pp. 2–13 (2011)
15. Leone, N., Manna, M., Terracina, G., Veltri, P.: Efficiently computable Datalog$^{\exists}$ programs. In: Proc. of the 13th Int. Conf. on the Principles of Knowledge Representation and Reasoning, KR 2012 (to appear, 2012)
16. Patel-Schneider, P.F., Horrocks, I.: Position paper: a comparison of two modelling paradigms in the Semantic Web. In: Proc. of the 15th Int. World Wide Web Conf. (WWW 2006), pp. 3–12 (2006)

A Quadratic Approach for Trend Detection in Folksonomies

Mohamed Nader Jelassi[1,2,3]

[1] Faculty of Sciences of Tunis, Tunis, Tunisia
[2] Clermont Université, Université Blaise Pascal, LIMOS, BP 10448, F-63000 CLERMONT-FERRAND
[3] CNRS, UMR 6158, LIMOS, F-63173 AUBIERE
nader.jelassi@isima.fr

Abstract. This proposal is concerned with the addition of a time stamp (a date) to the triples normally used in the representation of *folksonomies*. We motived our approach by helping the detection of trends in social networks.

1 Problematic

In the current PhD research, we tackle the problem of mining and analyzing data from social networks *aka folksonomies* [8]. Considered as a tripartite hyper-graph [11] of tags, users and resources, a *folksonomy* also mimics the structure of a triadic context [10]. The new data of *folksonomy* systems provides a rich resource for data analysis, information retrieval, and knowledge discovery applications. However, the main problem with *folksonomies* is their huge size since such structures grasped increasingly attention with the rise of the so called Web 2.0. Hence, many researches focuses on the extraction of lossless concise representations of interesting patterns, *i.e.,* triadic concepts from such structure. However, a scrutiny of the related work unveils that the time stamp dimension has not been considered [15]. The increasing use of these systems shows that *folksonomy*-based works are then able to offer a better solution in the domain of Web Information Retrieval (*WIR*) [9] by considering time when dealing with a query or during the user's taggings, *i.e.,* by suggesting the appropriate trendy tags. Moreover, the *WIR* will be without loss of information by keeping track all *folksonomy*'s actors: users, tags and resources. Hence, we will consider a *folksonomy* commonly composed of triples <users, tags, resources> and we shall consider **time** as a new dimension. Time is considered one of the most important factors in detecting emerging subjects. Thus, we are getting a ***d-folksonomy*** composed of quadruples <users, tags, resources, **dates**> which mimics the structure of a quadratic context [14]. Considering time when tackling this problem aims to discover which users, tags or resources are the most relevant at identified periods *w.r.t* a specific social network/*d-folksonomy*.

For this task, we first introduce a new algorithm, called QUADRICONS, as an extension of TRICONS [13] dedicated to the triadic contexts. QUADRICONS aims at getting out quadratic concepts, *i.e.,* quadri-concepts from quadratic contexts/*d-folksonomies*. A quadri-concept have the property that none of these sets can be extended without shrinking one of the other three dimensions. We also introduce a new closure operator that splits the induced search space into equivalence classes whose smallest elements are the quadri-minimal generators. Then, we use extracted quadri-concepts for the trend detection and popularity analyze in *d-folksonomies*. Such quadri-concepts are also useful to derive *quadratic rules*; reasoning about them in order to output relations between tags, users or resources.

The problematic that we tackle is relevant due to the increasing popularity of *folksonomies*, *i.e.,* social networks which grasped attention since the last decade. Analyzing data of such structure would be interesting and useful for a battery of potential application that we discuss in the last section.

2 Related Work

Trend Detection in *Folksonomies*

In 2006, Hotho *et al.* presented an approach for discovering topic-specific trends within *folksonomies* [7]. The authors analyze the emergence of common semantics by exploring trends in social networks with the FOLKRANK algorithm. This algorithm is able to compute topic-specific ranking on users, tags and resources. However, by transforming the *folksonomy* into an hypergraph, the edges between any kind two-dimensions leads to an information loss about the third dimension. For example, an edge between a user u and a tag t tells us that there is a resource r such as (u,t,r) is a triple of the *folksonomy*, however, we have no precise information concerning the resource r.

Dubinko *et al.* present in [4] a new approach based on a characterization of the most interesting tags associated with a sliding interval of time. They introduce a new measure called *interestingness* which is able to assess which tags are qualified by *"interesting"* within an interval of time. However, it turns out that the *interestingness* is better to detect the most popular tags than the interesting ones. Indeed, it computes the total number of occurence of tags over time regardless to users or resources related to these tags. For example, a tag t used several times by a single user would be considered as interesting even it is likely seen as a spam.

Considering the importance of time in *folksonomies*, Amitay *et al.* [1] discuss several aspects and uses of the time dimension in the context of Web Information Retrieval. The authors explain that they are able to detect and expose significant events and trends. A new measure called *TLP*[1] is introduced and used to gauge the relative number of timestamped (*i.e.,* dated) links that are associated with every time interval. Several applications are proposed by the authors: predictions of events, studying popularity change, etc. However, resources under study are not related to users or tags resulting in some loss of information.

Quadratic Concept Mining

In [14], Voutsadakis generalized the constructs and results of Wille [10] to the n-adic contexts. The author gives a definition of an n-adic concept. Voutsadakis gives an example of quadratic concepts and their associated complete Boolean 4-lattice. Despite

[1] Timestamped Link Profile.

robust theoretical study, no algorithm has been proposed by Voutsadakis for an efficient extraction of quadratic concepts. In addition, despite that of a n-adic concept, no basic notion of data mining (minimal generator, equivalence class, etc.) was adapted to the n-adic context. Indeed, it will be useful to localize small representative elements, *i.e.,* minimal generators, when mining huge data in *folksonomies*. Recently, Cerf *et al.* proposed the DATA-PEELER algorithm [2] which is able to extract all closed concepts from n-ary relations. When $n = 4$, the DATA-PEELER algorithm is able to extract quadratic concepts. However, DATA-PEELER is hampered by the large number of elements that may contain any of the dimensions. Thus, its strategy becomes ineffective and leads to a complex computation of n-adic concepts when dealing with large data as *folksonomies*.

3 Contributions and Research Plan

Contrarily to these aforementioned approaches, the approach that we introduce straightforwardly handles: *(i)* the detection of minimal generators for a scalable mining of quadri-concepts and *(ii)* keeping track of differents elements (users, tags, resources, dates) for an efficient trend detection in *d-folksonomies*. Visualizing the evolution of users, tags and resources over time is therefore a challenging task. Our contribution is summarized therefore to:

The Extraction of Quadratic Concepts from Four-Dimensional Contexts. We will first introduce a new closure operator for a four-dimensional context that splits the induced search space into equivalence classes whose smallest elements are the quadri-minimal generators. Minimal Generators (*MGs*) have been shown to play an important role in many theoretical and practical problem settings involving closure systems. Such minimal generators can offer a complementary and simpler way to understand the concept, because they may contain far fewer attributes than closed concepts. Indeed, *MGs* represent the smallest elements within an equivalence class. Complementary to closures, minimal generators provide a way to characterize formal concepts [3].

Trend Detection in Social Networks. [7] As application to this approach, we analyze trends in social networks through extracted quadratic concepts. According to each dimension, we are looking to which users, tags or resources are gaining (or losing) in popularity through time. Such approach is able to consider any kind of data in the same analysis. Taking into account quadratic concepts for such analyze allows lossless information since such structure keeps track of all actors of the *d-folksonomy*: users, tags, resources and dates. Moreover, to gauge the popularity of these actors through time, we use a measure[2] that takes into account the different elements of a quadri-concept based on the notion enunciated in [7] *"A resource which is tagged with important tags by important users becomes important itself. The same holds, symmetrically, for tags and users."*

[2] For instance, this measure consists in computing the number of quadri-concepts related to the structure under study. We plan to improve this measure by integrating the social aspects related to users and the relations between them.

4 Current Status of the PhD Research

At this level, we made the following advancements in the current PhD research:

1. The proposition of the QUADRICONS algorithm for **an efficient extraction of quadri-concepts** thanks to the first localisation of quadri-generators and the appropriate use of a closure operator for a quadratic context. Besides regrouping under one concept tags and resources shared in common by users of a *d-folksonomy* (discovery of hidden conceptualizations [8]), the time dimension, that we consider, plays a key role. Indeed, it turns out that the user's mode of tagging depends on time: a user who tagged a resource r with a tag t does perhaps not assign it the same tag at a different period. Several experiments (*c.f.,* Section 5) show that QUADRICONS outperforms its competitor, *i.e.,* DATA PEELER.
2. Since it is certainly not possible to analyze all the information provided by *d-folksonomies*, this huge data must be "cooked" into a succinct representation. Therefore, the quadri-concepts extracted by QUADRICONS are used for the experimentations in order to **discover trends in *d-folksonomies***. Thanks to the new introduced dimension, *i.e.,* time, we define a framework for the trend detection in social networks in order to discover popular users, tags and resources within time. The obtained diagrams depict which users are the most popular within a social network, which tags are the most trendy through time or even which resources (movies, artists, websites, ...) made the *buzz* at particular periods of time.

Among perspectives under study for the current PhD research, the following can be listed:

1. Discovering communities of interest (stable or volatile) in *d-folksonomies* based on discovered popular users.
2. Defining the quadratic form of association rules according to quadri-concepts [12]. Such rules do not exist in the literature yet.
3. Finding more applications using quadri-concepts: tag recommendation, suggesting friends, *folksonomy* evaluation [5], etc.

5 Experimental Results

For the experiments, we use two real-world datasets. First, MOVIELENS (*http://movielens.org*) which is a movie recommendation website where users are asked to note movies they like and dislike. Second, LAST.FM (*http://last.fm*) which is a music website where users are asked to tag their preferred artists. The MOVIELENS and LAST.FM datasets used for the experiments are freely downloadable from *http://movielens.org*.

We compare the performances (in sec) of the QUADRICONS algorithm *vs.* those of the DATA-PEELER algorithm for different values of the number of quadruples over the aforementioned datasets. The experiments show that QUADRICONS outperforms its competitor for both datasets and for all values of the number of quadruples. QUADRICONS does not exceed $1,51$ seconds versus $667,11$ for its competitor. Note that the number of extracted quadri-concepts reached, respectively, 991 and 897 for the MOVIELENS and LAST.FM datasets.

Trend Detection

Extracted quadri-concepts are useful for trend detection in *d-folksonomies* as the Figure 1 show some achievements carried on both datasets. The diagram on the left depicts the evolution of tags that were affected to the movie *'Harry Potter'* between December 2005 and August 2008. It tell us how the users of MovieLens see the movie. Indeed, it was seen first as a *'kid movie'* as the tags *magic*, *kids* and *witch* are massively used, then, with the release of new films, the popularity of these tags faded due to the emergence of new tags like *fantasy* with a rise of 200%. Interestingly enough, it is important to consider these trends when recommending tags to describe that movie. While, The diagram on the right of Figure 1 shows the evolution of users which tagged *pop/rock* artists. In May 2011, we see the rise of four outstanding users; we decide to analyse the first user's data in the system, *i.e.,* the user 470. We observe that he (she) has fifty-one friends [3]; such user must have some influence in this social network and could be targeted by the social network to promote such artists. However, among his friends, no one appears on the diagram in Figure 1. Hence, it could be interesting to recommend him(her) the users 58, 392 or 227 which share common interests.

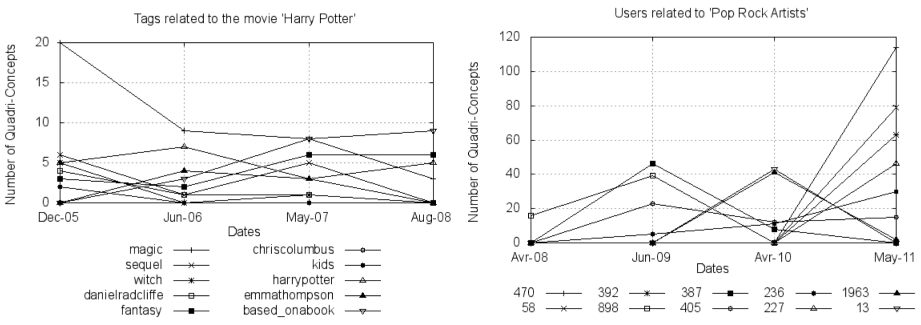


Fig. 1. (**Left**) Evolution of MovieLens' tags related to the movie 'Harry Potter' over time. (**Right**) Evolution of LastFM' users related to the 'Pop Rock artists' over time. User names are omitted for privacy reasons.

Quadratic concepts are also useful in order to derive rules with a quadratic form as explained in the following.

Quadratic Rules

The rule extraction is an active area of data mining and is applied in many domains as the study of the consumer basket, the analysis of the behavior of Internet users, computer security, bioinformatics and music to cite but a few. Starting from quadratic concepts, we are able to define two kinds of rules: intra-class and inter-class such as each quadri-concept represents an equivalence class. Such rules involve implication between tags while keeping track of users, resources and dates as introduced in [12].

[3] The Last.fm dataset show us friendship relations between the users.

The same kind of rules holds, symmetrically, for resources, users and dates. The first kind of rules (intra) are implications between a quadri-generator and the quadri-concept of the same equivalence class while the second kind of rules (inter) involves implications between a quadri-generator and a quadri-concept of different equivalence classes.

Example 1. Considering a tri-concept $\{\{u_1,u_2,u_3\}, \{t_2,t_3,t_4\}, \{r_1\}, \{d_1,d_2\}\}$ and a quadri-generator $\{\{u_1,u_2,u_3\}, \{t_3\}, \{r_1\}, \{d_1\}\}$ of a same equivalence class, an example of a quadratic rule (intra-class) is given as follows:
$t_3 \rightarrow_{\{u_1,u_2,u_3\},\{r_1\},\{d_1,d_2\}} t_2t_4$. The rule tell us that the tag t_3 implies the tags t_2 and t_4 with respect to the users u_1 ,u_2 and u_3, the resource r_1 and the dates d_1 and d_2.

Such rule can be useful for tag recommendation, analyzing tag popularity in time or spam detection. To the best of our knowledge, there are no approaches combining both implications between tags and temporality in rules yet.

References

1. Amitay, E., Carmel, D., Herscovici, M., Lempel, R., Soffer, A.: Trend detection through temporal link analysis. J. Am. Soc. Inf. Sci. Technol. 55(14), 1270–1281 (2004)
2. Cerf, L., Besson, J., Robardet, C., Boulicaut, J.F.: Closed patterns meet n-ary relations. ACM TKDD 3, 3:1–3:36 (2009)
3. Dong, G., Jiang, C., Pei, J., Li, J., Wong, L.: Mining Succinct Systems of Minimal Generators of Formal Concepts. In: Zhou, L.-z., Ooi, B.-C., Meng, X. (eds.) DASFAA 2005. LNCS, vol. 3453, pp. 175–187. Springer, Heidelberg (2005)
4. Dubinko, M., Kumar, R., Magnani, J., Novak, J., Raghavan, P., Tomkins, A.: Visualizing tags over time. ACM Trans. Web 1(2), 193–202 (2007)
5. Helic, D., Strohmaier, M., Trattner, C., Muhr, M., Lerman, K.: Pragmatic evaluation of folksonomies. In: Proc. of the 20th International Conference on WWW 2011, New York, NY, USA, pp. 417–426 (2011)
6. Hotho, A., Jäschke, R., Schmitz, C., Stumme, G.: Folkrank: A ranking algorithm for folksonomies. In: Proc. of FGIR, Hildesheim, Germany, pp. 111–114 (2006)
7. Hotho, A., Jäschke, R., Schmitz, C., Stumme, G.: Trend detection in folksonomies. In: Avrithis, Y., Kompatsiaris, Y., Staab, S., O'Connor, N.E. (eds.) SAMT 2006. LNCS, vol. 4306, pp. 56–70. Springer, Heidelberg (2006)
8. Jäschke, R., Hotho, A., Schmitz, C., Ganter, B., Stumme, G.: Discovering shared conceptualizations in folksonomies. Web Semantics 6, 38–53 (2008)
9. Krause, B., Hotho, A., Stumme, G.: A Comparison of Social Bookmarking with Traditional Search. In: Macdonald, C., Ounis, I., Plachouras, V., Ruthven, I., White, R.W. (eds.) ECIR 2008. LNCS, vol. 4956, pp. 101–113. Springer, Heidelberg (2008)
10. Lehmann, F., Wille, R.: A Triadic Approach to Formal Concept Analysis. In: Ellis, G., Rich, W., Levinson, R., Sowa, J.F. (eds.) ICCS 1995. LNCS, vol. 954, pp. 32–43. Springer, Heidelberg (1995)
11. Mika, P.: Ontologies are us: A unified model of social networks and semantics. Web Semantics 5(1), 5–15 (2007)
12. Trabelsi, C., Jelassi, N., Ben Yahia, S.: Auto-complétion de requêtes par une base générique de règles triadiques. In: Proc. of CORIA, Avignon, France, pp. 9–24 (2011)
13. Trabelsi, C., Jelassi, N., Ben Yahia, S.: Scalable Mining of Frequent Tri-concepts from *Folksonomies*. In: Tan, P.-N., Chawla, S., Ho, C.K., Bailey, J. (eds.) PAKDD 2012, Part II. LNCS, vol. 7302, pp. 231–242. Springer, Heidelberg (2012)
14. Voutsadakis, G.: Polyadic concept analysis. Order 19(3), 295–304 (2002)
15. Wolff, K.E.: Temporal concept analysis. In: Proc. of the 9th ICCS 2001, pp. 91–107 (2001)

Non-termination Analysis and Cost-Based Query Optimization of Logic Programs

Senlin Liang

Department of Computer Science
Stony Brook University
Stony Brook, NY 11794, USA
`sliang@cs.stonybrook.edu`

Abstract. There have been many studies in termination analysis of logic programming but little has been done on analyzing *non*-termination of logic programs, which is even more important in our opinion. Non-termination analysis examines program execution history when non-termination is suspected and informs the programmer of non-termination causes and possible ways to fix them. In the first part of this thesis, we study the problem of non-termination in tabled logic engines with subgoal abstraction, such as XSB,[1] and propose a suite of algorithms, called non-*Termin*ation anal*yzer*, `Terminyzer`, for automatic detection and explanation of non-termination.

The second part of this thesis focuses on cost-based query optimization. Database query optimizers rely on data statistics in selecting query execution plans and rule-based systems can greatly benefit from such optimizations as well. To this end, one first needs to collect data statistics for base and propagate them to derived predicates. However, there are two difficulties: dependencies among arguments and recursion. To address these problems, we implement a *Cost*-based query opt*imizer*, `Costimizer`, which consists of a cost estimator and an optimizing unit. The optimizing unit performs a greedy search optimization based on predicate statistics computed by the cost estimator. We validate the effectiveness of `Costimizer` on both size estimation and query optimization through experimental studies.

Keywords: non-termination analysis, termination analysis, forest logging, tabling, subgoal abstraction, derived predicate statistics, cost estimation, query optimization, logic programming, rule-based systems.

1 Non-Termination Analysis

1.1 Motivation and Problem Statement

The development of high-level logic languages such as $\mathcal{F}$LORA-$\mathcal{2}$[2] and SILK[3] aims at making logic-based knowledge representation accessible to knowledge

[1] `http://xsb.sourceforge.net`
[2] `http://flora.sourceforge.net`
[3] `http://silk.semwebcentral.org`

engineers who are not programmers. This type of users cannot be expected to debug the rule bases that they create and thus they require special support. In the course of the SILK project we discovered that non-termination due to the use of HiLog [2] and function symbols is one of the most vexing problems such users are facing, which motivates the current work.

There are three main scenarios where programs may not terminate. First, recursion can cause non-termination under the usual Prolog evaluation strategy. For instance, the query `?- p(a)` against the program with a single rule "`p(X) :- p(X).`" will not terminate. This kind of problems have been successfully addressed by adding SLG resolution (also known as tabling) [24] to Prolog, and a number of systems support it to various degrees (XSB, Yap, B-Prolog, Ciao).

The second scenario where programs might not terminate, even under the SLG resolution, occurs when increasingly deep nested calls are generated during the evaluation. Consider the query `?- p(a)` against the program with a single rule "`p(X) :- p(f(X)).`", the following calls will be successively generated: `p(a), p(f(a)), p(f(f(a)))`, and so on. Since neither call subsumes the other, tabling will not be able to evaluate the query and terminate. However, the technique known as *subgoal abstraction* [17] can take care of this problem. The essence of the technique is to modify the calls by replacing ("abstracting") subterms with new variables once certain term depth limit has been reached. For instance, in our example we could abstract calls once the depth limit of 4 has been reached. As a result, `p(f(f(f(f(a)))))` and all the subsequent calls would be abstracted to `p(f(f(f(X))))`.

For instance, in XSB (which to our knowledge is the only system that supports both tabling and subgoal abstraction), the above program will terminate. Generally, tabling with subgoal abstraction will evaluate queries that have a finite number of answers. Thus, the only remaining scenario is when both tabling and subgoal abstraction are used, but query evaluation does not stop because the number of answers to the query or its subqueries is infinite. Consider querying `?- p(X)` against the rule set "`p(a). p(f(X)) :- p(X).`", these answers will be successively derived: `p(a), p(f(a)), p(f(f(a)))` and so on.

In general, such queries cannot be evaluated completely, but if the program is what the user intended, the user could ask the system to stop after getting the first few answers. The problem arises when this was not the intended result. For small programs with only a few rules, expert programmers might be able to find the causes of the problem. However, for large knowledge bases with hundreds or thousands of rules this becomes a difficult task even for a seasoned logic programmer. For a knowledge engineer who is not a programmer, debugging non-termination is out of the question.

Note that the neither program termination (the halting problem) nor the problem of whether the number of answers is finite is decidable [19,21]. Sufficient conditions for termination of logic programs have been proposed in the literature [19,10,1,13,14,18,12,16,26], but most dealt with Prolog or Prolog-like evaluation strategies. Neither tabling nor subgoal abstraction were taken into account. This thesis therefore takes a different track on the problem: developing techniques that can help users analyze the causes of non-termination when both tabling and subgoal abstraction are used. We introduce a suite of algorithms, called `Terminyzer`, which are based on the analysis of logs produced by table operations for calls to tabled predicates [23]. The algorithms report the potential causes of non-termination with increasing level of fidelity and precision. As expected, the higher-fidelity algorithms have higher complexity.

1.2 Terminyzer

`Terminyzer` includes four non-termination analysis approaches of different computational complexity: *call sequence*, *answer flow*, *functor pattern* and *rule sequence* analyses. Call sequence analysis provides the sequences of tabled unfinished calls (calls that have not been completely evaluated) that are likely involved in non-termination, answer flow approach detects how information flows among these unfinished calls, and functor pattern analysis finds the sequences of functors that are applied repeatedly to generate infinitely many answers. The most advanced algorithm in the suite, the rule sequence method, finds the actual sequence of repeated rule applications that are the cause of non-termination. Based on these analysis, `Terminyzer` recommends to the programmer possible ways to remove non-termination causes by delaying the evaluation of certain subgoals.

Example 1. Consider the evaluation of the query `?- r(X)` against the following program in XSB with all predicates tabled:

```
p(a).  q(b).  s(f(b)).
p(f(X)) :- q(X).                        %% rule1
q(g(X)) :- p(X).                        %% rule2
r(X) :- p(X), s(X).                     %% rule3
```

The following sequence of subgoal calls will be made: `r(X)` calls `p(X)`, `p(X)` calls `q(X)`, and `q(X)` calls `p(X)`. Since there are infinitely many answers for `p(X)` and `q(X)` due to the recursion and the use of function symbols, the evaluation does not terminate.

By analyzing the logs of table operations for this evaluation, call sequence analysis provides the sequence of unfinished subgoal calls as `[r(X), p(X), q(X)]`, answer flow analysis identifies that answer information flows from subgoal `p` to `q` and `q` to `p`, functor pattern analysis detects the functor sequences that are applied repeatedly to produce new answers as `[f], [f,g], [g], [g,f]`. Rule sequence analysis will identify the sequence of rules that are fired and caused non-termination as `[rule3, rule1, rule2]`.

Based on the analysis, we know that XSB are producing infinitely answers for subgoals `p(X)` and `q(X)`. Thus, `Terminyzer` can recommend users that the evaluation of these two subgoals should be delayed until their argument `X` becomes bound. For instance, the rule `rule3` can be rewritten as "`r(X) :- s(X), p(X).`" to delay the evaluation of `p(X)`. Then, the query will terminate successfully and output the correct answer `r(f(b))`. □

2 Cost-Based Query Optimization

2.1 Motivation and Problem Statement

Database query optimizers depend on accurate and fast algorithms for estimating predicate (relation) sizes, which in turn depend on *joint data distributions* of values in different arguments. Traditionally, query optimizers estimate these sizes using statistical summaries for base predicates and propagate them to relational expressions assuming *argument independence* [3]. To make size estimates practical, data distributions must be summarized accurately and efficiently. Histogram is one such summarization technique that is in wide use in all major database systems (DB2, Oracle, Microsoft, etc.). Different histograms have been proposed in the literature, which differ in their complexity, cost, and accuracy [7,5]. Query optimizers search to improve join orders within predetermined budgets of time and space using statistics-based cost models. Research on optimization algorithms is quite extensive [15,11,25,4], and database systems implement greedy algorithms such as restricting the search space to left-deep trees [20].

However, there are several challenges in applying cost-based query optimization for rule-based systems. First, the argument independence assumption is rarely true for real world datasets, so size estimates based on histograms can be off by orders of magnitude [22]. It has been shown that estimation errors grow exponentially with the number of joins involved [6,8]. For rules, this problem is exacerbated by the presence of recursive predicates. Second, optimization algorithms have to take into account indexing and how the basic operations are performed by logic engines.

2.2 Costimizer

To address the above problems, this thesis proposes a cost based query optimizer, `Costimizer`, which is implemented as a pre-processor in XSB. The architecture of `Costimizer` is given in Figure 1 and it consists of two major components: *a cost estimator* and *an optimizing unit*. The cost estimator computes predicate statistics and it implements two different estimation algorithms: *log extraction* and *statistics for derived predicates* (SDP). These cost estimates from the estimator are then fed into the optimizing unit whose task is to find the optimal join ordering within given budget and to add necessary indexing commands. Finally, the optimized query and knowledge base are loaded into XSB for evaluation.

Log extraction performs information retrieval on query execution traces and provides the *real* (but *partial*) cost information. It can be used to optimize programs with many recursive rules which may not terminate. The SDP approach computes predicate statistics, in the form of *dependency matrices*, by an abstract evaluation of program rules [8,9] and it consists of a query spanning graph (QSG) Builder and an interpreter. The SDP approach performs best when a knowledge base has a relatively small number of rules but many more facts.

The optimizing unit implements a greedy search algorithm which finds a join ordering for a set of predicates and adds appropriate indexing commands for

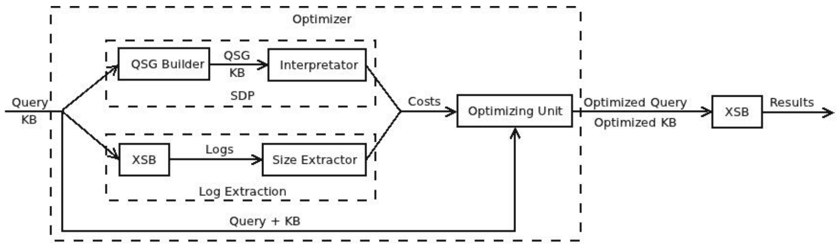


Fig. 1. System Architecture of `Costimizer`

each predicate. It is worth mentioning that the cost estimator is independent of the optimizing unit. Since virtually all cost-based optimizers use size estimates, our cost estimation will benefit most of such optimizers.

3 Conclusion and Future Works

This thesis focuses on non-termination analysis and cost-based optimization of logic programs. `Terminyzer` utilizes traces to help the programmer *debug* his programs. Currently, the call sequence, answer flow and functor pattern analyses are already implemented in XSB. We have performed several tests of `Terminyzer` and manually verified their results. Our future plan is to implement the rule sequence analysis. For `Costimizer`, we have implemented SDP in XSB and verified that it provides better estimates than histograms [8,9]. We plan to fully implement `Costimizer` and perform more extensive experiments.

References

1. Bruynooghe, M., Codish, M., Gallagher, J.P., Genaim, S., Vanhoof, W.: Ter- mination analysis of logic programs through combination of type-based norms. ACM Trans. Program. Lang. Syst. 29 (April 2007), `http://doi.acm.org/10.1145/1216374.1216378`
2. Chen, W., Kifer, M., Warren, D.S.: HiLog: A foundation for higher-order logic programming. J. Log. Program. 15(3), 187–230 (1993)
3. Christodoulakis, S.: Implications of certain assumptions in database performance evauation. ACM Trans. Database Syst. 9(2), 163–186 (1984)
4. DeHaan, D., Tompa, F.W.: Optimal top-down join enumeration. In: Proceedings of the 2007 ACM SIGMOD International Conference on Management of Data, SIGMOD 2007, pp. 785–796. ACM, New York (2007)
5. Ioannidis, Y.: The history of histograms (abridged). In: Proc. of VLDB Conference (2003)
6. Ioannidis, Y.E., Christodoulakis, S.: On the propagation of errors in the size of join results. SIGMOD Rec. 20(2), 268–277 (1991)
7. Ioannidis, Y.E., Poosala, V.: Balancing histogram optimality and practicality for query result size estimation. In: SIGMOD 1995: Proceedings of the 1995 ACM SIGMOD International Conference on Management of Data, pp. 233–244. ACM, New York (1995)

8. Liang, S., Kifer, M.: Deriving predicate statistics in datalog. In: Kutsia, T., Schreiner, W., Fernández, M. (eds.) PPDP, pp. 45–56. ACM (2010)
9. Liang, S., Kifer, M.: Deriving Predicate Statistics for Logic Rules. In: Krötzsch, M., Staraccia, U. (eds.) RR 2012. LNCS, vol. 7497, pp. 139–155. Springer, Heidelberg (2012)
10. Lindenstrauss, N., Sagiv, Y., Serebrenik, A.: Proving termination for logic programs by the query-mapping pairs approach. In: Program Developments in Computational Logic. LNCS, pp. 453–498. Springer (2004)
11. Moerkotte, G., Neumann, T.: Dynamic programming strikes back. In: SIGMOD 2008: Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data, pp. 539–552. ACM, New York (2008)
12. Neumerkel, U., Mesnard, F.: Localizing and Explaining Reasons for Non-Terminating Logic Programs with Failure-Slices. In: Nadathur, G. (ed.) PPDP 1999. LNCS, vol. 1702, pp. 328–342. Springer, Heidelberg (1999), http://dl.acm.org/citation.cfm?id=645815.668882
13. Nguyen, M.T., De Schreye, D.: Polytool: Proving Termination Automatically Based on Polynomial Interpretations. In: Puebla, G. (ed.) LOPSTR 2006. LNCS, vol. 4407, pp. 210–218. Springer, Heidelberg (2007), http://dl.acm.org/citation.cfm?id=1759187.1759208
14. Nguyen, M.T., Giesl, J., Schneider-Kamp, P., De Schreye, D.: Termination Analysis of Logic Programs Based on Dependency Graphs. In: King, A. (ed.) LOPSTR 2007. LNCS, vol. 4915, pp. 8–22. Springer, Heidelberg (2008), http://dx.doi.org/10.1007/978-3-540-78769-3_2
15. Ono, K., Lohman, G.M.: Measuring the complexity of join enumeration in query optimization. In: Proceedings of the Sixteenth International Conference on Very Large Databases, pp. 314–325. Morgan Kaufmann Publishers Inc., San Francisco (1990), http://portal.acm.org/citation.cfm?id=94362.94436
16. Payet, E., Mesnard, F.: Nontermination inference of logic programs. ACM Trans. Program. Lang. Syst. 28, 256–289 (2006), http://doi.acm.org/10.1145/1119479.1119481
17. Riguzzi, F., Swift, T.: Well-definedness and efficient inference for probabilistic logic programming under the distribution semantics. In: TPLP (to appear, 2012)
18. Schneider-Kamp, P., Giesl, J., Ströder, T., Serebrenik, A., Thiemann, R.: Automated termination analysis for logic programs with cut*. Theory Pract. Log. Program. 10, 365–381, http://dx.doi.org/10.1017/S1471068410000165
19. Schreye, D.D., Decorte, S.: Termination of logic programs: The never-ending story. J. Log. Program. 19/20, 199–260 (1994)
20. Selinger, P.G., Astrahan, M.M., Chamberlin, D.D., Lorie, R.A., Price, T.G.: Access path selection in a relational database management system. In: SIGMOD 1979: Proceedings of the 1979 ACM SIGMOD International Conference on Management of Data, pp. 23–34. ACM, New York (1979)
21. Sipser, M.: Introduction to the Theory of Computation, 1st edn. International Thomson Publishing (1996)
22. Stillger, M., Lohman, G.M., Markl, V., Kandil, M.: Leo - db2's learning optimizer. In: VLDB 2001: Proceedings of the 27th International Conference on Very Large Data Bases, pp. 19–28. Morgan Kaufmann Publishers Inc., San Francisco (2001)
23. Swift, T., Warren, D.S., Sagonas, K., Freire, J., Rao, P., Cui, B., Johnson, E., de Castro, L., Marques, R.F., Saha, D., Dawson, S., Kifer, M.: The XSB system, version 3.3.x., vol. 1. Programmer's manual, http://xsb.sourceforge.net

24. Swift, T., Warren, D.S.: Xsb: Extending prolog with tabled logic programming. CoRR abs/1012.5123 (2010)
25. Vance, B., Maier, D.: Rapid bushy join-order optimization with cartesian products. In: Proceedings of the 1996 ACM SIGMOD International Conference on Management of Data, SIGMOD 1996, pp. 35–46. ACM, New York (1996)
26. Voets, D., De Schreye, D.: Non-termination Analysis of Logic Programs Using Types. In: Alpuente, M. (ed.) LOPSTR 2010. LNCS, vol. 6564, pp. 133–148. Springer, Heidelberg (2011), http://dl.acm.org/citation.cfm?id=2008282.2008291

When Data, Knowledge and Processes Meet Together

Ario Santoso
Supervised by Diego Calvanese

KRDB Research Centre,
Faculty of Computer Science, Free University of Bozen/Bolzano,
Piazza Domenicani, 3. I-39100 Bolzano, Italy
`Santoso@inf.unibz.it`

Abstract. The aim of this work is to investigate the integration of data, knowledge, and processes. In particular, we study the combination of research in Ontology-Based Data Access (OBDA) with Data-Centric Dynamic Systems (DCDSs). The idea of OBDA is to combine data and knowledge by providing a conceptual view over data repositories in terms of an ontology, while DCDSs provide a holistic framework for modeling business processes in which both data and processes are treated as first-class citizens. Thanks to this combination, we obtain Semantically-Governed Data-Aware Processes (SGDAPs), which represent a significant step towards the envisioned unifying framework. We position SGDAPs in the state of the art and briefly discuss the current status of our research. We then identify several research directions along which we intend to continue our work.

1 Introduction

The marriage between data and knowledge has been studied extensively in the Ontology-Based Data Access (OBDA) setting [5,19]. The idea is to provide a conceptual view over data repositories through ontologies. On the other hand, recent work in business processes, services, and databases brought about the necessity of considering both data and processes as first-class citizens during system design. This holistic view of data and processes together has given rise to a line of research under the name of *artifact-centric business processes* [17,10,21,1] that aims at avoiding the notorious discrepancy of traditional approaches where these aspects are considered separately [4].

Verification is in this setting an important task to guarantee that the designed systems obey to desired properties, but is highly challenging because the presence of data causes them to be infinite-state in general. Recently, interesting decidability results for verification of temporal properties over such systems have been obtained (see e.g. [3,4,9]). In particular, [12,11] recently studied these issues in the context of Data-Centric Dynamic Systems (DCDSs), in which the data component is constituted by a relational database, and the process layer is described declaratively through condition-action rules.

Another line of research that spanned over the last years is related to the combination of processes and knowledge. The purpose here is to represent the process component of the system using semantically-rich representation languages that can be processed by machines to realize different reasoning tasks such as automated discovery, execution support, composition and interoperation. Notable examples in this area are Semantic Web Services (SWS) [16,20] and Semantic Business Process Management (c.f. [22,13]).

Differently from all these approaches, our aim is to combine data, processes and knowledge, in a setting where the process manipulates the data and the knowledge component (ontology) is used to understand, govern, and verify the overall system as well as its dynamics over time. Towards this goal, we are interested in investigating how the line of research on OBDA can be fruitfully combined with the one of DCDSs. In this setting the presence of the ontology provides a unified, high-level conceptual view of the system, reflecting the relevant concepts and relations of the domain of interest and abstracting away how processes and data are concretely realized and stored at the concrete implementation level. This, in turn, is the basis for different important reasoning task such as verification of conceptual dynamic properties, track and query the evolving system through the conceptual level, regulate how new processes can be injected into the system. synthesize new processes starting from high level conceptual requirements, reasoning under implicit and incomplete information.

The interplay between the three components of data, knowledge, and process requires to establish connections among different research directions, and paves the way for several fascinating issues to be investigated. We identify some of them and discuss them in the following.

2 Preliminaries

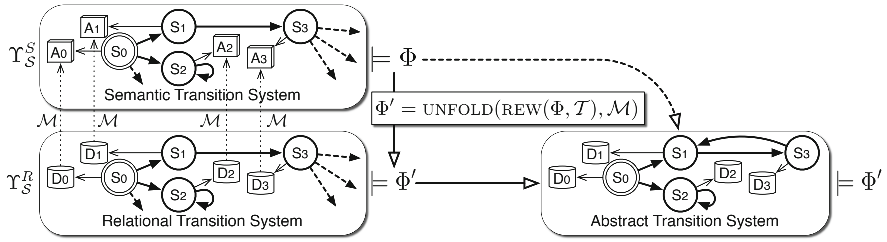
Ontology-Based Data Access System (c.f. [5]). An OBDA system provides a conceptual view over data stored in a relational database. It includes *(i)* (the intensional level of) an ontology, *(ii)* a relational database schema, and *(iii)* a mapping between the ontology and the database. In [5], the ontology which represents the domain of interest, is represented as a TBox in $DL\text{-}Lite_{\mathcal{A},id}$, which is a Description Logic (DL) that has been specifically designed for efficient ontology-based access to large amounts of data. In an OBDA system, a relational database is connected to an ontology through mappings, which relate queries over the database to queries over the ontology. Each mapping specifies how to populate the concepts and roles of the ontology from the tuples contained in the relational database. Function symbols are used to construct (abstract) objects from the concrete values retrieved from the database. However in practice these objects are not concretely materialized. Instead certain answers of queries formulated over the ontology can be computed by directly accessing the underlying data. In fact, thanks to the first-order rewritability of lightweight DLs such as $DL\text{-}Lite_{\mathcal{A},id}$, a query over the ontology can be rewritten by compiling away the TBox and the unfolded into SQL query over the database by exploiting the mapping assertions [5].

Data-Centric Dynamic System (c.f. [11]). A DCDS captures the manipulation of the data that is done by the available processes in the system. A DCDS consist of *(i)* The *data component* which represents the data of interest in the application. *(ii)* The *process component* which represents the progression mechanism for the DCDS. The semantic of DCDS is defined in terms of a possibly infinite transition system whose states are labeled by databases and where transitions represent the execution of process actions. Such transition system represents all possible computations that the process component can do on the data component starting from the initial database instance. In DCDSs, we are interested in verifying whether the transition system of a given DCDS satisfies a certain temporal dynamic property of interest, specified in some first-order temporal logic. Even though this kind of verification is in general undecidable, suitable property specification languages (based on first-order variants of μ-calculus) as well as restrictions on the allowed DCDSs that ensure decidability have been proposed in [11].

3 Contributions: Semantically-Governed Data-Aware Processes

In this section we briefly review the current status of our research. More details can be found in [7]. Our idea is to augment DCDS with an ontology, connected to the data layer of the DCDS through mappings, in the style of OBDA. The resulting system is called Semantically-Governed Data-Aware Process (SGDAP) and is constituted by three components: *(i)* An *OBDA system*, which keeps all the data of interest and provides a conceptual view over it in terms of a $DL\text{-}Lite_{\mathcal{A},id}$ TBox. *(ii)* A *process component* as in the DCDS, which characterize the evolution (dynamic aspect) of the system. *(iii)* An *initial database instance*. Intuitively, the OBDA system keeps all the data of interest, while the process component modifies and evolves such data, starting from the initial database. Conceptually, an SGDAP separates the system into two layers, the *relational layer* and the *semantic layer*. The relational layer captures the database evolution (manipulation) done by the process execution, while the semantic layer exploits the ontology for providing a conceptual view of the system evolution. This enables to: *(i) understand and query* the evolving system through the semantic layer, and *(ii) govern* the evolution of the system at the semantic layer by rejecting those process actions that, currently executed at the relational layer, would lead to new system states that violates some constraint of the ontology.

The semantics of SGDAP is defined in terms of two transition systems: a *Relational Layer Transition System* (RTS) and a *Semantic Layer Transition System* (STS). The RTS is the same as the transition system of a classical DCDSs, which captures the evolution of the systems at the relational layer, tracking how the database is evolved by the process component. On the other hand, the STS is a "virtualization" of the RTS in the semantic layer and provides a conceptual view of the system evolution. In particular, the STS maintains the structure of the RTS unaltered, reflecting that the process component is executed



Υ_S^S
A0
A1
A2
A3
S0
S1
S2
S3
Semantic Transition System
$\models \Phi$
$\mathcal{M}$
$\Phi' = \textsc{unfold}(\textsc{rew}(\Phi, \mathcal{T}), \mathcal{M})$
Υ_S^R
D0
D1
D2
D3
Relational Transition System
$\models \Phi'$
Abstract Transition System
$\models \Phi'$

Fig. 1. Verification of dynamic $\mu\mathcal{L}_{\mathrm{C}}^{\mathrm{EQL}}$ properties over SGDAP

over the relational layer, but it associates to each state the set of concept and role assertions obtained from the application of the mappings starting from the corresponding database instance.

An interesting task that we have addressed for SGDAPs is the verification of *conceptual temporal properties*, i.e., temporal properties that constrain the dynamics of the system understood at the semantic layer. In particular we have introduced the property specification language $\mu\mathcal{L}_{\mathrm{C}}^{\mathrm{EQL}}$ which is a variant of μ-calculus where epistemic conjunctive queries [6] are used to arbitrarily query the states of the system. We have then shown that verification of $\mu\mathcal{L}_{\mathrm{C}}^{\mathrm{EQL}}$ properties over the STS of an SGDAP can be reduced to verification of corresponding properties over the underlying RTS by exploiting the first-order rewritability of *DL-Lite*$_{\mathcal{A},id}$. The rough idea to tackle the verification problem is basically to bring down the conceptually specified temporal properties into the relational layer, by adopting the concept of "rewriting" and "unfolding" in OBDA, and then exploit the decidability results of temporal specification property in DCDS (The idea of the approach is depicted in Figure 1). This, in turn, allows to directly import all the decidability results studied for DCDSs into the framework of SGDAPs.

4 Future Plan and Open Challenges

As far as now we have mainly studied the framework of SGDAPs and investigated decidability issues related to verification of sophisticated temporal properties. Ongoing work is being dedicated now to the investigation of complexity issues and to the adoption of the framework in concrete case studies (see the ACSI EU project[1]).

As further work, we plan to investigate different ways of combining data, knowledge, and processes starting from SGDAPs. One interesting setting is to consider the situation where the process component is specified over the semantic layer. In this setting, we are interested in studying how the high level process specification can be realized as a concrete process over the relational layer. This leads to a problem of synthesizing a process in the relational layer from the given high level specification in the semantic layer, which might also be expected to satisfy some conceptual temporal properties. Another interesting setting is

[1] http://www.acsi-project.eu/

to consider the situation where we have a process specification on both the semantic and relational layer. Here it is interesting to see how these two process specifications relate to each other and to check whether they are aligned.

Another direction is to investigate the situation where there is a change explicitly over the ontology. The question here is how such a change will/can affect the relational layer. This research direction has a closed relation to the research on view updates [2]. A possible way to address this issue is to investigate bidirectional transformations (c.f. [18]) to formalize the relation between the two layer.

Another interesting line of research is to connect our approach to research on verification of different classes of Petri Nets [14]. In particular we are interested in considering classes of Petri Nets equipped with data, such as Colored Petri Nets or Petri Data Nets. The success on this direction will open a connection between two well-studied fields and enrich the results in each field.

Another direction is to consider quantitative properties of the system. In this case, we are interested in verification and synthesis in presence of quantitative requirements, by leveraging on [15,8].

References

1. Abiteboul, S., Bourhis, P., Galland, A., Marinoiu, B.: The AXML Artifact Model. In: Proc. of TIME 2009, pp. 11–17 (2009)
2. Bancilhon, F., Spyratos, N.: Update Semantics of Relational Views. ACM Trans. Database Syst. 6(4), 557–575 (1981)
3. Belardinelli, F., Lomuscio, A., Patrizi, F.: An Abstraction Technique for the Verification of Artifact-Centric Systems. In: Proc. of KR 2012 (2012)
4. Bhattacharya, K., Gerede, C., Hull, R., Liu, R., Su, J.: Towards Formal Analysis of Artifact-Centric Business Process Models. In: Alonso, G., Dadam, P., Rosemann, M. (eds.) BPM 2007. LNCS, vol. 4714, pp. 288–304. Springer, Heidelberg (2007)
5. Calvanese, D., De Giacomo, G., Lembo, D., Lenzerini, M., Poggi, A., Rodriguez-Muro, M., Rosati, R.: Ontologies and Databases: The *DL-Lite* Approach. In: Tessaris, S., Franconi, E., Eiter, T., Gutierrez, C., Handschuh, S., Rousset, M.-C., Schmidt, R.A. (eds.) Reasoning Web. LNCS, vol. 5689, pp. 255–356. Springer, Heidelberg (2009)
6. Calvanese, D., Giacomo, G.D., Lembo, D., Lenzerini, M., Rosati, R.: EQL-Lite: Effective First-Order Query Processing in Description Logics. In: Proc. of IJCAI 2007 (2007)
7. Calvanese, D., Giacomo, G.D., Lembo, D., Montali, M., Santoso, A.: Ontology-Based Governance of Data-Aware Processes. In: Krötzsch, M., Straccia, U. (eds.) RR 2012. LNCS, vol. 7497, pp. 25–41. Springer, Heidelberg (2012)
8. Calvanese, D., Santoso, A.: Best Service Synthesis in the Weighted Roman Model. In: Proc. of the 4th Central-European Workshop on Services and their Composition (ZEUS 2012), CEUR Electronic Workshop Proceedings, vol. 847 (2012), http://ceur-ws.org/
9. Cangialosi, P., Giacomo, G.D., Masellis, R.D., Rosati, R.: Conjunctive Artifact-Centric Services. In: Maglio, P.P., Weske, M., Yang, J., Fantinato, M. (eds.) ICSOC 2010. LNCS, vol. 6470, pp. 318–333. Springer, Heidelberg (2010)

10. Cohn, D., Hull, R.: Business Artifacts: A Data-centric Approach to Modeling Business Operations and Processes. IEEE Bull. on Data Engineering 32(3), 3–9 (2009)
11. Hariri, B.B., Calvanese, D., Giacomo, G.D., Deutsch, A., Montali, M.: Verification of Relational Data-Centric Dynamic Systems with External Services. CoRR Technical Report arXiv:1203.0024, arXiv.org e-Print archive (2012), http://arxiv.org/abs/1203.0024
12. Bagheri Hariri, B., Calvanese, D., Giacomo, G.D., De Masellis, R., Felli, P.: Foundations of Relational Artifacts Verification. In: Rinderle-Ma, S., Toumani, F., Wolf, K. (eds.) BPM 2011. LNCS, vol. 6896, pp. 379–395. Springer, Heidelberg (2011)
13. Hepp, M., Leymann, F., Domingue, J., Wahler, A., Fensel, D.: Semantic Business Process Management: A Vision Towards Using Semantic Web Services for Business Process Management. In: IEEE International Conference on e-Business Engineering, ICEBE 2005, pp. 535–540 (October 2005)
14. Jensen, K., Kristensen, L.: Coloured Petri Nets, Modelling and Validation of Concurrent Systems. Springer (2009)
15. Kwiatkowska, M.: Quantitative Verification: Models Techniques and Tools. In: Proc. of the the 6th Joint Meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on The Foundations of Software Engineering, ESEC-FSE 2007, pp. 449–458. ACM, New York (2007)
16. McIlraith, S., Son, T., Zeng, H.: Semantic Web Services. IEEE Intelligent Systems 16(2), 46–53 (2001)
17. Nigam, A., Caswell, N.S.: Business Artifacts: An Approach to Operational Specification. IBM Systems Journal 42(3), 428–445 (2003)
18. Pierce, B.C.: Linguistic Foundations for Bidirectional Transformations: Invited Tutorial. In: Proc. of the 31st Symposium on Principles of Database Systems, PODS 2012, pp. 61–64. ACM, New York (2012)
19. Rodríguez-Muro, M., Calvanese, D.: Dependencies: Making Ontology Based Data Access Work in Practice. In: Proc. of the 5th Alberto Mendelzon Int. Workshop on Foundations of Data Management (AMW 2011), CEUR Electronic Workshop Proceedings, vol. 749 (2011), http://ceur-ws.org/
20. Swartz, A.: MusicBrainz: A Semantic Web Service. IEEE Intelligent Systems 17(1), 76–77 (2002)
21. van der Aalst, W.M.P., Barthelmess, P., Ellis, C.A., Wainer, J.: Proclets: A Framework for Lightweight Interacting Workflow Processes. Int. J. of Cooperative Information Systems 10(4), 443–481 (2001)
22. Wetzstein, B., Ma, Z., Filipowska, A., Kaczmarek, M., Bhiri, S., Losada, S., Lopez-Cobo, M.J., Cicurel, L.: Semantic Business Process Management: A Lifecycle Based Requirements Analysis. In: Proc. of the Workshop on Semantic Business Process and Product Lifecycle Management, CEUR Electronic Workshop Proceedings, vol. 251 (2007), http://ceur-ws.org/

Author Index