Proceedings

of the

2006 International Workshop on Description Logics DL'06

May 30 – June 1, 2006

Lake District, United Kingdom

Edited by Bijan Parsia, Ulrike Sattler, and David Toman

Preface

Welcome to the 2006 International Workshop on Description Logics, DL'06, in the Lake District, UK, May 30–June 1, 2006. The workshop continues the long-standing tradition of international workshops devoted to discussing developments and applications of knowledge representation formalisms and systems based on Description Logics. The list of the International Workshops on Description Logics can be found at `http://dl.kr.org`.

There were 38 papers submitted to DL'06 each of which was reviewed by at least three members of the program committee or additional reviewers recruited by the PC members.

In addition to the presentation of the accepted papers, posters, and demos the following speakers agreed to give invited talks at the workshop:

- Franz Baader: *Description Logics: a personal history*
- Anthony Hunter: *Living with Inconsistency: Theory + Applications*

The organizers and participants of the Description Logics 2006 workshop gratefully acknowledge the support of *Cerebra* (`cerebra.com`) that helped to provide reduced registration fees for students, and the University of Manchester for providing logistical support, in particular with registration.

Our thanks go to all the authors for submitting to DL, and to the invited speakers, PC members, and all additional reviewers who made the technical programme possible. The organization of the workshop also greatly benefited from the help of many people at the University of Manchester, in particular Sam Creighton for helping with all the invitation letters for visas, and Bernardo Cuenca Grau, Hector Perez-Urbina and Birte Glimm for helping with the local organization. Finally, we would like to acknowledge that the work of the PC was greatly simplified by using the *EasyChair* conference management system (`www.easychair.org`) developed by Andrei Voronkov.

Bijan Parsia, Ulrike Sattler, and David Toman
Description Logics 2006 PC chairs and organizers

Programme Committee

Carlos Areces, INRIA Lorraine, France
Giuseppe De Giacomo, Università di Roma "La Sapienza", Italy
Enrico Franconi, Free University of Bozen-Bolzano, Italy
Volker Haarslev, Concordia University, Quebec, Canada
Ian Horrocks, University of Manchester, UK
Ralf Küsters, Christian-Albrechts-Universitt zu Kiel, Germany
Carsten Lutz, Technical University Dresden, Germany
Ralf Möller, Hamburg University of Technology, Germany
Bijan Parsia, University Of Maryland, College Park, USA
Peter Patel-Schneider, Bell Laboratories, USA
Ulrike Sattler, University of Manchester, UK
David Toman, University of Waterloo, Canada
Chris Welty, IBM Watson Research Center, USA
Frank Wolter, University of Liverpool, UK

Additional Reviewers

Alex Borgida
Sebastian Brandt
Diego Calvanese
Xi Deng
Sofia Espinosa
Birte Glimm
Daniel Gorin
Christian Halaschek-Wiener
Jan Hladik
Aditya Kalyanpur
Alissa Kaplunova
Atila Kaya
Sergio Mera
Maja Miličić
Boris Motik
Hsueh-Ieng Pai
Boontawee Suntisrivaraporn
Stephan Tobies
Dmitry Tsarkov
Anni-Yasmin Turhan
Dirk Walther
Grant Weddell
Michael Wessel
Michael Zakharyaschev
Hui Zhang
Evgeny Zolin

Contents

I Long Papers 1

Conjunctive Query Answering for Description Logics with Transitive Roles.
Birte Glimm, Ian Horrocks and Ulrike Sattler 3

Efficient Reasoning in $\mathcal{EL}+$.
Franz Baader, Carsten Lutz and Boontawee Suntisrivaraporn . . 15

DL Actions with GCIs: a Pragmatic Approach.
Hongkai Liu, Carsten Lutz, Maja Miličić and Frank Wolter . . 27

Discrete Tableau Algorithms for $\mathcal{FSHI}$.
Yanhui Li, Baowen Xu, Jianjiang Lu and Dazhou Kang 39

Epistemic First-order Queries over Description Logic Knowledge Bases.
Diego Calvanese, Giuseppe De Giacomo, Domenico Lembo, Maurizio Lenzerini and Riccardo Rosati 51

Data Complexity of Answering Unions of Conjunctive Queries in $\mathcal{SHIQ}$.
Magdalena Ortiz, Diego Calvanese and Thomas Eiter 62

PSPACE Automata for Description Logics.
Jan Hladik and Rafael Peñaloza 74

Binary Absorption in Tableaux-Based Reasoning for Description Logics.
Alexander K. Hudek and Grant Weddell 86

A Description Logic of Change.
Alessandro Artale, Carsten Lutz and David Toman 97

II Regular Papers 109

Reasoning for Fuzzy Description Logic with Comparison Expressions.
Dazhou Kang, Baowen Xu, Jianjiang Lu and Yanhui Li 111

Handling Imprecise Knowledge with Fuzzy Description Logic.
Giorgos Stoilos, Giorgos Stamou and Jeff Pan 119

Finding Subsumers for Natural Language Presentation.
Chris Mellish and Jeff Pan 127

SHIN ABox Reduction.
Achille Fokoue, Aaron Kershenbaum and Li Ma 135

Tableau Caching for Description Logics with Inverse and Transitive Roles.
Yu Ding and Volker Haarslev 143

On the Scalability of Description Logic Instance Retrieval.
Ralf Möller Volker Haarslev and Michael Wessel 151

High Performance Absorption Algorithms for Terminological Reasoning.
Ming Zuo and Volker Haarslev 159

Automated Benchmarking of Description Logic Reasoners.
Tom Gardiner, Ian Horrocks and Dmitry Tsarkov 167

Will my Ontologies Fit Together?.
Bernardo Cuenca Grau, Ian Horrocks, Oliver Kutz and Ulrike Sattler 175

Description logic reasoning using the PTTP approach.
Zsolt Nagy, Gergely Lukácsy and Peter Szeredi 183

Extending the $\mathcal{SHOIQ}(\mathcal{D})$ Tableaux with DL-safe Rules: First Results.
Vladimir Kolovski, Bijan Parsia and Evren Sirin 192

Description Logic Reasoning for Dynamic ABoxes.
Christian Halaschek-Wiener, Bijan Parsia, Evren Sirin and Aditya Kalyanpur 200

Beyond Asserted Axioms: Fine-Grain Justifications for OWL-DL Entailments.
Aditya Kalyanpur, Bijan Parsia and Bernardo Cuenca Grau . . 208

Optimizations for Answering Conjunctive ABox Queries.
Evren Sirin and Bijan Parsia 215

Model checking the basic modalities of CTL with Description Logic.
Shoham Ben-David, Richard Trefler and Grant Weddell 223

Towards Mobile Reasoning.
Thomas Kleemann 231

III Posters 239

Euclidian Roles in Description Logics.
Giorgos Stoilos and Giorgos Stamou 241

Experiences with Load Balancing and Caching for Semantic Web Applications.
Alissa Kaplunova, Atila Kaya and Ralf Möller 243

How sensor data interpretation could benefit from description logics: A practical case study.
Ralf Möller and Sylvia Melzer 245

Cost-Efficient Web Service compositions for querying processes over reasoning services.
Irma Sofia Espinosa Peraldi and Ralf Möller 247

Static Knowledge Representation of Multi-Agent System's Specification by Description Logic.
Haiyan Che and Jigui Sun 249

A Tableaux-based Mobile DL Reasoner - An Experience Report.
Felix Müller, Michael Hanselmann, Thorsten Liebig and Olaf Noppens 251

Computing Maximally Satisfiable Terminologies for the Description Logic $\mathcal{ALC}$ with Cyclic Definitions.
Kevin Lee, Thomas Meyer, Jeff Pan and Richard Booth 253

Pellet System Description.
Evren Sirin and Bijan Parsia 255

Topological Reasoning in Basic Description.
Matteo Cristani, Nicoletta Gabrielli and Paolo Torelli 257

The new ICOM Ontology Editor.
Pablo Fillottrani, Enrico Franconi and Sergio Tessaris 259

Part I
Long Papers

Conjunctive Query Answering for Description Logics with Transitive Roles

Birte Glimm* Ian Horrocks Ulrike Sattler
The University of Manchester, UK
[glimm,horrocks,sattler]@cs.man.ac.uk

Abstract

An important reasoning task, in addition to the standard DL reasoning services, is conjunctive query answering. In this paper, we present algorithms for conjunctive query answering in the expressive Description Logics $\mathcal{SHQ}$ and $\mathcal{SHOQ}$. In particular, we allow for transitive (or non-simple) roles in the query body, which is a feature that is not supported by other existing conjunctive query answering algorithms. For $\mathcal{SHQ}$, we achieve this by extending the logic with a restricted form of $\downarrow$ binders and state variables as known from Hybrid Logics. We also highlight, why the addition of inverse roles makes the task of finding a decision procedure for conjunctive query answering more complicated.

1 Introduction

Existing Description Logic (DL) reasoners[1] provide automated reasoning support for checking concepts for satisfiability and subsumption, and also for answering queries that retrieve instances of concepts and roles. The development of a decision procedure for conjunctive query answering in expressive DLs is, however, still a partially open question. *Grounded* conjunctive queries for $\mathcal{SHIQ}$ are supported by KAON2, Pellet, and Racer's query language nRQL. However, the semantics of grounded queries is different from the usually assumed open-world semantics in DLs since existentially quantified variables are always bound to named individuals.

*This work was supported by an EPSRC studentship.

[1] For example, FaCT++ `http://owl.man.ac.uk/factplusplus/`, KAON2 `http://kaon2.semanticweb.org/`, Pellet `http://www.mindswap.org/2003/pellet/`, or Racer Pro `http://www.racer-systems.com/`

None of the existing conjunctive query answering techniques [14, 10, 4, 9, 13] is able to handle transitive roles or nominals in the query body.[2] In this paper, we present an extension of $\mathcal{SHQ}$ with a restricted form of the $\downarrow$ binder operator and state variables as known from Hybrid Logics [3]. This extended logic enables an extension of the rolling-up technique [14, 4] to transitive roles; a feature which was left open by Tessaris [14]. Unfortunately, the $\downarrow$ binder already makes the DL $\mathcal{ALC}$ undecidable. For query answering, however, the $\downarrow$ binder occurs in a very restricted form and, as we show in Section 4, this extension of $\mathcal{SHQ}$ is decidable. Further on, we adapt the tableaux algorithm for $\mathcal{SHOQ}$ [7] in order to decide conjunctive query entailment in $\mathcal{SHQ}$.

Query answering for a logic that includes nominals, e.g., a logic such as $\mathcal{SHOQ}$, has the additional difficulty that cycles are no longer limited to ABox individuals or shortcuts via transitive roles. Therefore, the non-deterministic assignment of individual names to variables in a cycle, as suggested for the DL $\mathcal{DLR}$ by Calvanese et al. [4], can no longer be applied. In Section 5, however, we show that, by also introducing new variables, the non-deterministic assignment of individual names to variables can be used in order to provide a query answering algorithm for $\mathcal{SHOQ}$.

Finally, we highlight why the extension of $\mathcal{SHQ}$ or $\mathcal{SHOQ}$ with inverse roles makes the design of a decision procedure for conjunctive query answering more complicated.

2 Preliminaries

We assume readers to be familiar with the syntax and semantics of the DL $\mathcal{SHOIQ}$ (for details see [1]) and of $\mathcal{SHOIQ}$ knowledge bases. Therefore, we introduce only the syntax and semantics of the $\downarrow$ binder and of conjunctive queries here.

Let $\mathcal{L}$ be a Description Logic, C an $\mathcal{L}$-concept, and N_V a finite set of variable names with $y \in N_V$. With $\mathcal{L}_\downarrow$, we denote the language obtained by allowing, additionally, y and $\downarrow y.C$ as $\mathcal{L}$-concepts. For an interpretation $\mathcal{I} = (\Delta^\mathcal{I}, \cdot^\mathcal{I})$, an element $d \in \Delta^\mathcal{I}$, and a variable $y \in N_V$, we denote with $\mathcal{I}_{[y/d]}$ the interpretation that extends $\mathcal{I}$ such that $y^\mathcal{I} = \{d\}$. The $\mathcal{L}_\downarrow$-concept $\downarrow y.C$ is then interpreted as $(\downarrow y.C)^\mathcal{I} = \{d \in \Delta^\mathcal{I} \mid d \in C^{\mathcal{I}_{[y/d]}}\}$.

Let $\vec{y}$ be a vector of *variables* and $\vec{c}$ a vector of individual names. A *Boolean conjunctive query* q has the form $\langle\rangle \leftarrow conj_1(\vec{y};\vec{c}) \wedge \ldots \wedge conj_n(\vec{y};\vec{c})$. We call $\mathbf{T}(q) = \vec{y} \cup \vec{c}$ the set of *terms* in q,[3] and we call each $conj_i(\vec{y};\vec{c})$ for $1 \leq i \leq n$

[2] Although the algorithm presented by Calvanese et al. [4] allows the use of regular expressions in the query (in particular the transitive reflexive closure), it has been shown that the algorithm is incomplete [8, 5].

[3] For readability, we sometimes abuse our notation and refer to $\vec{y}$ as a set. When referring

an atom. *Atoms* are either concept or role atoms: a *concept atom* has the form $t\colon C$, and a *role atom* the form $\langle t, t'\rangle\colon r$, for $\{t, t'\} \subseteq \mathbf{T}(q)$, C a possibly complex $\mathcal{L}$-concept, and r an $\mathcal{L}$-role.

Let $\mathcal{K}$ be an $\mathcal{L}$-knowledge base ($\mathcal{L}$-KB), $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ a model for $\mathcal{K}$, and q a Boolean conjunctive query for $\mathcal{K}$. An *assignment in* $\mathcal{I}$ is a mapping $\cdot^{A} : \mathbf{T}(q) \to \Delta^{\mathcal{I}}$. We say that q *is true in* $\mathcal{I}$ and write $\mathcal{I} \models q$ if there exists an assignment $\cdot^{A}$ in $\mathcal{I}$ s.t. $t^{A} \in \{t^{\mathcal{I}}\}$ for every individual $t \in \vec{c}$, $t^{A} \in C^{\mathcal{I}}$ for every concept atom $t\colon C$ in q, and $\langle t^{A}, t'^{A}\rangle \in r^{\mathcal{I}}$ for every role atom $\langle t, t'\rangle\colon r$ in q. We call such an assignment *a q-mapping w.r.t.* $\mathcal{I}$. If $\mathcal{I} \models \mathcal{K}$ implies $\mathcal{I} \models q$ for all models $\mathcal{I}$ of $\mathcal{K}$, then we say that q *is true in* $\mathcal{K}$, and write $\mathcal{K} \models q$; otherwise we say that q *is false in* $\mathcal{K}$, and write $\mathcal{K} \not\models q$.

Since answering non-Boolean conjunctive queries can be reduced to answering (possibly several) Boolean queries, we consider only Boolean queries here.

3 Reducing Query Answering to Concept Unsatisfiability

The main ideas used in this paper were first introduced by Calvanese et al. [4] for deciding conjunctive query entailment for the expressive DL $\mathcal{DLR}_{reg}$. Since query answering can be reduced to query entailment, the algorithm in [4] is capable of deciding conjunctive query answering as well. The authors show how a query q can be expressed as a concept C_q, such that q is true w.r.t. a given knowledge base if adding $\top \sqsubseteq \neg C_q$ makes the KB unsatisfiable. In order to obtain the concept C_q, the query q is represented as a directed, labelled graph. This graph, called a tuple-graph or a query graph, is traversed in a depth-first manner and, during the traversal, nodes and edges are replaced with appropriate concept expressions, leading to the concept C_q after completing the traversal.

The nodes in a query graph correspond to the terms in the query and are labelled with the concepts that occur in the corresponding concept atoms. The edges correspond to the role atoms in q and are labelled accordingly. For example, let q_1 be the query $\langle\rangle \leftarrow x\colon C \wedge \langle x, y\rangle\colon s \wedge y\colon D$ and q_2 the query $\langle\rangle \leftarrow x\colon C \wedge \langle x, y\rangle\colon s \wedge \langle y, x\rangle\colon r \wedge y\colon D$. The query graphs for q_1 and q_2 are depicted in Fig. 1 and Fig. 2 respectively.

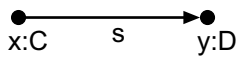


Figure 1: The (acyclic) query graph for q_1.

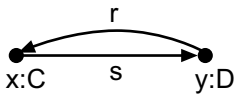


Figure 2: The (cyclic) query graph for q_2.

Since q_1 is acyclic, we can build the concept that represents q_1 as follows:

to a vector $\vec{y}$ as a set, we mean the set $\{y_i \mid y_i \text{ occurs in } \vec{y}\}$.

start at x and traverse the graph to y. Since y is a leaf node, remove y and its incoming edge and conjoin $\exists s.D$ to the label C of x, resulting in $C \sqcap \exists s.D$ for C_{q_1}. A given KB $\mathcal{K}$ entails q_1 iff $\mathcal{K} \cup \{\top \sqsubseteq \neg C_{q_1}\}$ is unsatisfiable. Since the graph is collapsed with each step, the technique is often called rolling-up. Full details can be found in the original paper [4], although the description there is slightly different since reification and name formulae are used in order to reduce $\mathcal{DLR}_{reg}$ (which allows also for roles of arity $n > 2$) to *converse*-PDL. Tessaris [14] proposes a similar approach for the DL $\mathcal{SHQ}$ (without transitive roles in the query).

This reduction is not directly extendable to cyclic queries since, due to the tree-model property of most DLs, a concept cannot capture cyclic relationships. However, for simpler DLs, this also means that cycles can only be enforced by ABox assertions and hence, Calvanese et al. suggest to replace variables in a cycle, i.e., x and y in q_2, non-deterministically with individual names from the KB (or with terms from the more specific query in the case of query entailment).

Although the technique presented by Calvanese et al. provides the basic foundation of several query answering algorithms, including the extensions presented here, it was found later that the algorithm in its original form is not a decision procedure. Horrocks et al. [8] point out that, by identifying variables with each other, some cyclic queries become acyclic, and hence a replacement of variables with named individuals might not find all solutions. For example, identifying y and y' in the query graph depicted in Fig. 3, leads to the acyclic query graph in Fig. 4, which can be expressed as $C \sqcap \exists r.\exists s.D$. Clearly, the KB containing the ABox assertion $a : \exists r'.(C \sqcap \exists r.\exists s.D)$ would make the query true, although in the relevant assignments none of the variables can be bound to the individual name a.

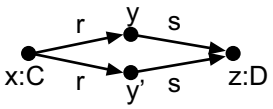


Figure 3: The original query graph.

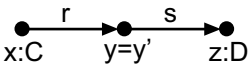


Figure 4: The query graph after identifying y with y'.

Another problem arises through the regular expressions in $\mathcal{DLR}_{reg}$, which are, for example, capable of expressing inverse roles in the query. Let q_3 be the query $\langle\rangle \leftarrow \langle x, y\rangle : r \wedge \langle y, x\rangle : s^-$ (see Fig. 5). Identifying x and y still leaves us with a cyclic query graph. Let $\mathcal{K}$ be a knowledge base containing the role axiom $r \sqsubseteq s$ and the ABox assertion $a : \exists r'.\exists r.\top$. Clearly, we have that $\mathcal{K} \models q_3$ (see Fig. 6), although again in the relevant assignments none of the variables can be bound to the individual name a. Similar examples can be given for regular expressions that use the Kleene star for expressing the transitive closure. These discoveries motivated the work presented in the following sections, which in particular targets the problems arising with cyclic queries.

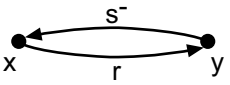


Figure 5: The query graph for q_3.

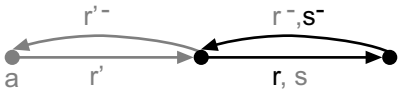


Figure 6: A representation of the canonical model of $\mathcal{K}$. The parts that make q_3 true are highlighted.

4 The Rolling-Up Technique with ↓ Binders

An extension of the rolling-up technique to cyclic queries is not directly possible, as we have tried to explain in the previous section. Due to the tree-model property of most DLs, a concept cannot capture cyclic relationships, but it is also not always correct to replace variable names in a cycle with individual names from the knowledge base. In this section, we show how cyclic queries for an $\mathcal{L}$-KB can be rolled-up into $\mathcal{L}_{\downarrow}$-concepts. The $\downarrow$ binder can label elements in a model with a variable, and this variable can be used to enforce a co-reference. Consider again the query $q_2\langle\rangle \leftarrow x\!:\!C \wedge \langle x,y\rangle\!:\!s \wedge \langle y,x\rangle\!:\!r \wedge y\!:\!D$ (see Fig. 2). Using the $\downarrow$ binder, we can construct the following concept C_{q_2} for q_2: $\downarrow x.(C \sqcap \exists s.(D \sqcap \exists r.x))$. In order to obtain this concept, we still traverse the query graph (see Fig. 7): we start at x, traverse to y, then we reach x again and replace the r-edge by conjoining $\exists r.x$ to the label of y, now y is a leaf node and we replace y and its incoming s-edge by conjoining $\exists r.\downarrow y.(D \sqcap \exists r.x)$ to the label of x, since x is the last node, we obtain C_{q_2} as $\downarrow x.(\exists r.\downarrow y.(D \sqcap \exists r.x)$. Since y never occurs as a variable, we can omit the $\downarrow$ binder for y. Now we have again that $\mathcal{K} \models q_2$ iff $\mathcal{K} \cup \{\top \sqsubseteq \neg C_{q_2}\}$ is unsatisfiable. Further details and a proof that C_{q_2} indeed captures q_2 can be found in a technical report [6].

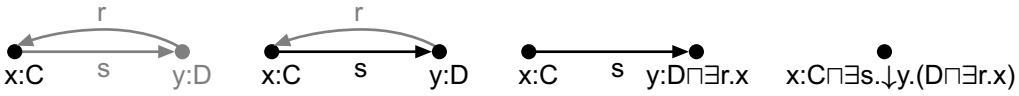


Figure 7: The graph traversal for the query graph of q_2 step by step. Parts already seen during the traversal are highlighted.

Even for cyclic queries, the rolling-up is now straightforward. If, however, the logic $\mathcal{L}$ under consideration does not provide for inverse roles, then this technique is only applicable in case each component in the query graph is also strongly connected. In the presence of weakly connected components, we would need inverse roles in order to roll-up a query. For example, let q_4 be the query $\langle\rangle \leftarrow x\!:\!C \wedge \langle x,y\rangle\!:\!s \wedge \langle x,y\rangle\!:\!r \wedge y\!:\!D$, i.e., both edges go now from x to y and x is not reachable from y. Therefore, the query graph is composed of two weakly connected components and a traversal of the query graph would get stuck at y. With inverse roles, q_4 could easily be expressed as $\downarrow x.(C \sqcap \exists s.(D \sqcap \exists \mathsf{Inv}(r).x))$. Tessaris [14] shows how arbitrary conjunctive queries for $\mathcal{SHQ}$ can be rolled-up

even without inverse roles, and an extension of this technique might work for $\downarrow$ binders and variables as well.

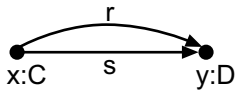


Figure 8: The query graph for q_4.

Unfortunately, there are no decision procedures for expressive DLs with the $\downarrow$ binder operator. It is even known that $\mathcal{ALC}$ extended with binders and state variables is undecidable [2]. However, we can observe some interesting properties for our query concepts. (1) After the rolling-up, all variables occur only positively. (2) Only existential restrictions are introduced in the rolling-up. Hence, negating the query concept for q_2 and transforming it into negation normal form yields $\downarrow x.(\neg C \sqcup \forall s.(\neg D \sqcup \forall r.\neg x))$ and we observe that all variables occur negated and are under the scope of universal quantifiers. For $\mathcal{ALC}$, these restrictions are enough to regain decidability [11].

4.1 A Tableaux Algorithm for $\mathcal{SHQ}$ Query Concepts

Our algorithm is based on the tableaux algorithm for $\mathcal{SHOQ}$ [7], which obviously decides KB satisfiability for $\mathcal{SHQ}$. Using this algorithm has the advantage that we can use nominals for the constants in the query. Alternatively, representative concepts and additional ABox assertions for each constant in a query could be used [14]. A tableaux algorithm builds a finite representation of a model, which is called a completion graph. The nodes in the completion graph represent elements in the domain and are labelled with a set of concepts. Labelled edges represent the relational structures among the elements. An initial completion graph is expanded according to a set of expansion rules. The expansion stops when an obvious contradiction, called a clash, occurs, or when no more rules are applicable. In the latter case, the completion graph is called complete. Termination is guaranteed by a technique called blocking. A complete and clash-free completion graph can be "unravelled" into a model for the given KB.

In order to handle query concepts that may contain binders and state variables, some adaptations of the existing algorithm are necessary. In order to store the bindings of variables, we modify the node labels s.t. they contain tuples of the form $\langle C, B\rangle$, where C is a concept and B is a (possibly empty) set of bindings of the form $\{y_1/v_1, \ldots, y_n/v_n\}$, where $y_1, \ldots, y_n$ are the free variables in C and $v_1, \ldots, v_n$ are nodes. The next obvious addition is a rule to handle concepts of the form $\downarrow y.C$: if $\langle \downarrow y.C, B\rangle$ is in the label of a node v, we add $\langle C, \{y/v\} \cup B\rangle$ and $\langle y, \{y/v\}\rangle$ to the label of v. The latter states that y is bound to v at v, and the former that the free variable y in C is bound to v. All other existing rules

have to propagate the bindings as well, e.g., the $\forall$-rule applied to $\langle \forall r.C, B\rangle$ in the label of a node v adds $\langle C, B\rangle$ to the labels of v's r-successors. The set B contains all and only the bindings for the free variables in C. Another obvious consequence is the addition of a new clash condition: if both $\langle y, \{y/v\}\rangle$ and $\langle \neg y, \{y/v\}\rangle$ are in the label of the node v, then this is a clash.

A more challenging task is the extension of the blocking condition. For $\mathcal{SHQ}$, however, we argue that we can simply ignore the bindings, i.e., if $\langle C, B\rangle$ is in the label, we consider only C in the blocking condition. This clearly ensures termination. But why does this guarantee that we can unravel a complete and clash-free completion forest into a model? Obviously, in $\mathcal{SHQ}$, there is no way for a node to propagate information "back" to its ancestors, and clashes according to the new clash condition can only occur through a cyclic structure. This is so because a node v is only labelled with $\langle y, \{y/v\}\rangle$ by an application of the new $\downarrow$-rule to some concept $\langle \downarrow y.C, B\rangle$ in the label of v. Furthermore, the only way $\neg y$ can occur with v as a binding is when $\langle C, \{y/v\} \cup B\rangle$ is expanded to $\langle \neg y, \{y/v\}\rangle$ via a cyclic path back to v. This is obviously only possible among individual nodes in $\mathcal{SHQ}$ and, therefore, no clash in the tableau can by caused by unravelling. Hence, transitive roles alone are not causing major problems.

An interesting consequence is, however, that we loose the finite model property. For example, let $\mathcal{K}$ contain the axioms $\top \sqsubseteq \exists r.\top$ and $\top \sqsubseteq \neg C_q$ with r a transitive role and $\neg C_q := \downarrow x.(\forall r.\neg x)$. The first axiom enforces an infinite r-chain for every individual. Normally, a finite model could contain an r-cycle instead of an infinite chain, but this would clearly violate $\neg C_q$. Hence, every model of $\mathcal{K}$ must be acyclic and therefore contain an infinite r-chain.

5 A Rolling-up Technique for $\mathcal{SHOQ}$

In this section, we show how the rolling-up technique can be extended to the DL $\mathcal{SHOQ}$, i.e., $\mathcal{SHQ}$ plus nominals. We do not use $\downarrow$ binders here, but rather propose an extension of the guessing technique as sketched in Section 3. In the presence of nominals, a simple non-deterministic assignment of individual names to variables that occur in a cycle is not sufficient. For example, Fig. 9 represents a model for the KB containing the axioms $\{a\} \sqsubseteq \neg C \sqcap \neg D \sqcap \exists s.(C \sqcap \exists r.(D \sqcap \exists s.\{a\}))$ and $\mathsf{trans}(s)$. The query $\langle\rangle \leftarrow x : C \wedge y : D \wedge \langle x, y\rangle : r \wedge \langle y, x\rangle : s$ (see Fig. 10) would clearly be true, although in the relevant assignments a cannot be bound to either x or y. Since the query graph for this query contains just one strongly connected component, no inverse roles are required in the rolling-up and we should be able to deal with this query.

For $\mathcal{SHOQ}$, our main argument for the correctness of an extension of the guessing of variable assignments is that a cycle among new nodes can only occur due to a transitive role that provides a shortcut for "skipping" the nominal.

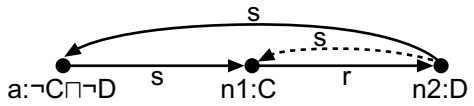


Figure 9: The dashed line indicates the relationship added due to s being transitive. Therefore, there is a cycle not directly containing the nominal a.

Hence, a nominal is always at least indirectly involved in a cycle. In this case, we have only one nominal a, and we may guess that it is either in the position of x, in the position of y, or it is "splitting" the (transitive) role s (see Fig. 11). In each case, we can roll-up the query graph into the nominal, obtaining the three query concepts $C_q^1 := \{a\} \sqcap C \sqcap \exists r.(D \sqcap \exists s.\{a\}), C_q^2 := \{a\} \sqcap D \sqcap \exists s.(C \sqcap \exists r.\{a\}), C_q^3 := \{a\} \sqcap \exists s.(C \sqcap \exists r.(D \sqcap \exists s.\{a\}))$. Identifying x with y gives the additional forth query concept $C_q^4 := \{a\} \sqcap C \sqcap D \sqcap \exists r.\{a\} \sqcap \exists s.\{a\}$. The query is true just in case one of these query concepts is entailed by the KB. Clearly, in our example, only the concept C_q^3 that corresponds to the "new" guess is entailed. If we had n nominals, then we would need to try $4n$ guesses.

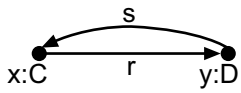


Figure 10: The original query graph.

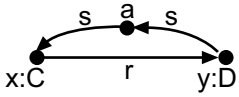


Figure 11: An alternative query graph in which the nominal a is assumed to be involved in the cycle.

Since it was, to the best of our knowledge, not even known if conjunctive query answering for logics with nominals and transitive roles is decidable, this technique is clearly valuable, although it is, due to its highly non-deterministic nature, not very practical.

6 The Challenges of Inverses and Nominals

Neither the arguments used for the extension of $\mathcal{SHQ}$ with binders (i.e., blocking can ignore different bindings), nor the extended guessing strategy for $\mathcal{SHOQ}$, can be used with inverse roles. Fig. 12 shows a representation of a model for the concept $\{a\} \sqcap \exists r.(\exists s.\top)$ for s a transitive and symmetric role. The query $\langle\rangle \leftarrow \langle x, x\rangle : s$ is obviously true in this model. The nominal a is, however, not even indirectly involved in the cycle.

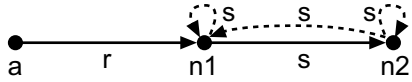


Figure 12: The dashed lines represent the additional relationships added for s being transitive and symmetric.

Since completion graphs are finite representations of models, the question is how far do we have to expand a completion graph before blocking is "safe", i.e., unravelling into a tableau does not lead to a clash.

In order to see the same problem from another perspective, we briefly sketch a different query answering algorithm in the style of CARIN [10]. CARIN provides a conjunctive query answering algorithm for the DL $\mathcal{ALCNR}$. Recall from Section 2 that a query q is true in a knowledge base $\mathcal{K}$, if there is a q-mapping for every model $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ of $\mathcal{K}$. Similarly, we define a mapping ϕ from terms in a query to nodes in a completion graph $\mathbf{G}$ and we call ϕ *a q-mapping w.r.t.* $\mathbf{G}$ if, for each constant t in q, $\{t\}$ is in the label of $\phi(t)$ (i.e., constants are mapped to the corresponding nominal nodes), for each concept atom $t\!:\!C$ in q, C is in the label of $\phi(t)$ and, for each role atom $\langle t, t'\rangle\!:\!r$ in q, $\phi(t')$ is an r-descendant of $\phi(t)$, where r-descendant implicitly closes the transitive roles. Since ϕ is purely syntactic, we extend the KB with an axiom $\top \sqsubseteq C \sqcup \neg C$ for each concept C s.t. $t\!:\!C$ is a concept atom in q. Hence, we obtain a decision procedure for conjunctive query answering if we can show the following:

Claim 6.1
For each model $\mathcal{I}$ of $\mathcal{K}$, there is a q-mapping w.r.t. $\mathcal{I}$ iff for each completed and clash-free completion graph $\mathbf{G}$ of $\mathcal{K}$, there is a q-mapping w.r.t. $\mathbf{G}$.

In order to take the length of the paths in the query into account, blocking must be delayed appropriately. In CARIN (i.e., for a logic without transitive roles) this is achieved by using two isomorphic trees (instead of two isomorphic pairs as it is the case for normal pairwise blocking as in $\mathcal{SHIQ}$) s.t. the depth of the trees corresponds to the longest path in the query.

The "if" direction of Claim 6.1 is relatively straight forward but, for the "only if" direction, we have to show that (in contrapositive form), if there is a completion graph $\mathbf{G}$ for $\mathcal{K}$ s.t. there is no q-mapping ϕ w.r.t. $\mathbf{G}$, then there is a model $\mathcal{I}$ of $\mathcal{K}$ s.t. there is no q-mapping σ w.r.t. $\mathcal{I}$. We now give an example that shows that, even if we find such a completion graph $\mathbf{G}$, we cannot guarantee that there is no mapping σ w.r.t. the canonical model $\mathcal{I}$ of $\mathbf{G}$, if q contains a cycle with a transitive role occurring in it. Of course there may be another completion graph, for which this problem does not occur, but it is hard to prove that there is always a canonical counter-model for the query.

Let $\mathcal{K}$ be a KB containing the axiom $\top \sqsubseteq \exists r.\top \sqcup \exists s.\top$ for r a transitive and symmetric role and let q be the query $\langle\rangle \leftarrow \langle x, x\rangle\!:\!r \wedge \langle x, y\rangle\!:\!s \wedge \langle y, z\rangle\!:\!r \wedge \langle z, z\rangle\!:\!r \wedge x\!:\!C \wedge z\!:\!C$, see Fig. 13 right. The upper part of Fig. 13 shows a possible (simplified) completion graph $\mathbf{G}$ for $\mathcal{K}$, where C or $\neg C$ is added to the node labels to allow for a purely syntactic mapping ϕ. The grey underlying triangle shapes illustrate the two isomorphic trees used for blocking, and clearly there is no q-mapping w.r.t. $\mathbf{G}$. The partial representation of a model depicted in the lower part of Fig. 13, however, shows that, by unravelling $\mathbf{G}$, we would get

a model $\mathcal{I}$ of $\mathcal{K}$ s.t. there is a q-mapping w.r.t. $\mathcal{I}$. This is the case because there is no longer only one element in the extension of C, and all role relationships for q are satisfied since r is transitive and symmetric (inferred relationships are only pictured where they are relevant for the mapping). Even choosing a larger depth for the two trees would allow this to happen, since the decision for C or $\neg C$ and for an r- or s-successor was purely non-deterministic.

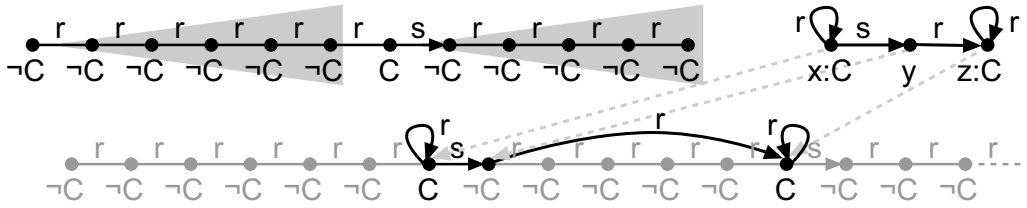


Figure 13: A completion graph **G** with trees for blocking (top), a partial unravelling of **G** (bottom), and a query graph (right).

Therefore, the absence of a q-mapping for the completion graph does not prove that there is a model for which there is no q-mapping. Requiring several isomorphic trees or pairs before blocking occurs obviously does not help either since only one part between the isomorphic trees/pairs could contain the twice required C in the node label. The problem is that, by repeating the unique part between the blocking and the blocked node, we can produce a structure that allows for a q-mapping.

A tempting solution for "safe" blocking conditions would be to require a path of n isomorphic sequences directly following each other, where n is the length of the longest path in the query. However, this will not guarantee termination, as the following argument shows: we could interpret each path from a root or nominal node as a word over the edge labels, e.g., the above example would produce a word starting with $rrrrrrsrrrr$. For the suggested blocking condition, such a path must contain a substring of the form W^n, i.e., n repetitions of the word W, where W is any sequence of letters over our alphabet of roles. The additional blocking conditions for the node labels are irrelevant here. In 1944 Morse and Hedlund [12] showed that the so called Thue-Morse sequence, which forms a word over an alphabet of only two letters, is cube-free, which would correspond to a path of unbounded length in which no blocking occurs. Hence, the algorithm would not terminate for $n \geqslant 3$. For a three letter alphabet, this holds already for $n \geqslant 2$.

We encounter similar difficulties when using $\downarrow$ binders and variables for a logic like $\mathcal{SHIQ}$ or $\mathcal{SHOIQ}$. It is difficult to determine how "deep" the completion graph has to be before blocking is safe. If the completion graph is not "deep enough", unravelling does not yield a model.

Although for the CARIN technique, some canonical models with a q-mapping have completion graphs without a q-mapping, it seems as if by testing all possi-

ble completed and clash-free completion graphs, one would always find at least one completion graph such that also its canonical model does not provide for a q-mapping. However, this is only an observation, which could support the view that the problem is, despite all these problems, decidable. A different proof technique might be necessary to show this, and our future work is aimed at investigating this.

7 Conclusions

In the previous sections, we have presented two ideas that allow an extension of the rolling-up technique also to cyclic conjunctive queries for $\mathcal{SHQ}$ and $\mathcal{SHOQ}$. For $\mathcal{SHQ}$, we achieved this by extending the logic with a restricted form of $\downarrow$ binders and state variables as known from Hybrid Logics. This allows the expression of cyclic queries as concepts since, with $\downarrow$ binders and variables we can express a co-reference. Query entailment can then be reduced to deciding concept satisfiability for the extended DL. However, adding the $\downarrow$ binder, even in the very restricted form needed for query answering, has a notable impact on the resulting logic; e.g., for $\mathcal{SHQ}$, this extension leads to the loss of the finite model property. In Section 4.1, we illustrate how a tableaux algorithm for $\mathcal{SHQ}$ can be extended in order to handle query concepts. Although each $\downarrow$ introduces a fresh nominal on the fly, we show that, for $\mathcal{SHQ}$, termination can be regained by ignoring them in the blocking condition. Thus we have shown how conjunctive queries with transitive roles in the query body can be answered.

In Section 5, we extend the work by Calvanese et al. [4] to $\mathcal{SHOQ}$, i.e., to a logic that allows for nominals. We do this by proposing a more sophisticated guessing technique, which then again enables the rolling-up of a query into a $\mathcal{SHOQ}$-concept.

In Section 6, we highlight why none of the proposed techniques extends easily to a logic with inverse roles. We also show this for a query entailment algorithm in the style of CARIN [10]. The main problem is to decide when a completion graph is expanded "far enough" in order to safely predict that the query is also not entailed in its possibly infinite canonical model.

The rolling-up technique with binders and variables integrates seamlessly into the existing tableaux algorithms, whereas, for the CARIN-style algorithms, the search for a q-mapping is an additional and completely separated step. Moreover, the added axioms $\top \sqsubseteq C \sqcup \neg C$ for every concept C in the query, can significantly increase the amount of non-determinism. This makes binders and variables the more attractive choice from an implementation point of view.

Our future work will include efforts to show that answering arbitrary conjunctive queries for $\mathcal{SHIQ}$ and $\mathcal{SHOIQ}$ is decidable. If that is the case — and we believe it is — we aim at extending the rolling-up technique with binders

and state variables to $\mathcal{SHIQ}$, $\mathcal{SHOQ}$ and $\mathcal{SHOIQ}$. This would provide a (hopefully practical) decision procedure for arbitrary conjunctive queries over OWL-DL knowledge bases.

References

[1] F. Baader, D. Calvanese, D. L. McGuinness, D. Nardi, and P.F̃. Patel-Schneider, editors. *The Description Logic Handbook*. Cambridge University Press, 2003.

[2] P. Blackburn and J. Seligman. Hybrid languages. *Journal of Logic, Language and Information*, 4(3):251–272, 1995. Special issue on decompositions of first-order logic.

[3] P. Blackburn and J. Seligman. *Advances in Modal Logic*, volume 1, chapter What are hybrid languages?, pages 41–62. CSLI, 1998.

[4] D. Calvanese, G. De Giacomo, and M. Lenzerini. On the decidability of query containment under constraints. In *Proc. of PODS'98*, pages 149–158. ACM, 1998.

[5] B. Glimm and I. Horrocks. Handling cyclic conjunctive queries. In *Proc. of DL'05*, Edinburgh, UK, July 26–28 2005. CEUR.

[6] B. Glimm, I. Horrocks, and U. Sattler. Conjunctive query answering for the description logic $\mathcal{SHOIQ}$. Technical report, The University of Manchester, 2006. Available online at `http://www.cs.man.ac.uk/~glimmbx/download/GlHS06b.pdf`.

[7] I. Horrocks and U. Sattler. Ontology reasoning in the $\mathcal{SHOQ}$ description logic. In *Proc. of IJCAI'01*, pages 199–204. Morgan Kaufmann, 2001.

[8] I. Horrocks, U. Sattler, S. Tessaris, and S. Tobies. How to decide query containment under constraints using a description logic. In *Proc. of LPAR'00*, LNAI. Springer, 2000.

[9] U. Hustadt, B. Motik, and U. Sattler. A decomposition rule for decision procedures by resolution-based calculi. In *Proc. of LPAR'04*, volume 3452 of *LNCS*. Springer, 2004.

[10] A. Y. Levy and M.-C. Rousset. CARIN: A representation language combining horn rules and description logics. In *European Conf. on Artificial Intelligence*, pages 323–327, 1996.

[11] M. Marx. Narcissists, stepmothers and spies. In *Proc. of DL'02*, volume 53. CEUR, 2002.

[12] M. Morse and G. A. Hedlund. Unending chess, symbolic dynamics, and a problem in semigroups. *Duke Mathematical Journal*, 11(1):1–7, 1944.

[13] M. M. Ortiz, D. Calvanese, and T. Eiter. Characterizing data complexity for conjunctive query answering in expressive description logics. In *Proc. of the 21st Nat. Conf. on Artificial Intelligence (AAAI'06)*, 2006. to appear.

[14] S. Tessaris. *Questions and answers: reasoning and querying in Description Logic*. PhD thesis, University of Manchester, 2001.

Efficient Reasoning in $\mathcal{EL}^+$

Franz Baader, Carsten Lutz, Boontawee Suntisrivaraporn
Institute for Theoretical Computer Science
TU Dresden, Germany
{baader,clu,meng}@tcs.inf.tu-dresden.de

1 Introduction

The early dream of a description logic (DL) system that offers both sound and complete polynomial-time algorithms and expressive means that allow its use in real-world applications has since the 1990ies largely been considered to be a pipe dream. This was, on the one hand, due to complexity results showing intractability even in very inexpressive DLs [5], in particular in the presence of TBoxes [13]. On the other hand, many of the applications considered then required more expressive power rather than less, which led to the development of more and more expressive DLs. The use of such intractable DLs in applications was made possible by the fact that highly-optimized tableau-based reasoners for them behaved quite well in practice.

However, more recent developments regarding the $\mathcal{EL}$ family of DLs have shed a new light on the realizability of this dream.[1] On the one hand, theoretical results [1, 6, 2] have shown that reasoning in $\mathcal{EL}$ and several of its extensions remains tractable in the presence of TBoxes and even of general concept inclusions (GCIs). On the other hand, it has turned out that, despite its relatively low expressivity, the $\mathcal{EL}$ family is highly relevant for a number of important applications, in particular in the bio-medical domain: for example, medical terminologies such as the Systematized Nomenclature of Medicine (Snomed) [9] and the Galen Medical Knowledge Base (Galen) [14] are formulated in $\mathcal{EL}$ or small extensions thereof, and the Gene Ontology (Go) [8] used in bioinformatics can also be viewed as an $\mathcal{EL}$ TBox.

In this paper, we address the question of whether the polynomial-time algorithms for reasoning in $\mathcal{EL}$ and its extensions really behave better in practice than intractable, but highly-optimized tableau-based algorithms. To this end, we have implemented a refined version of the algorithm described in [2] in our

[1] An alternative contender for a usable tractable DL is the one introduced in [7].

$$
\begin{array}{rcl}
\mathsf{Endocardium} & \sqsubseteq & \mathsf{Tissue} \sqcap \exists \mathsf{cont\text{-}in}.\mathsf{HeartWall} \sqcap \\
 & & \quad \exists \mathsf{cont\text{-}in}.\mathsf{HeartValve} \\
\mathsf{HeartWall} & \sqsubseteq & \mathsf{BodyWall} \sqcap \exists \mathsf{part\text{-}of}.\mathsf{Heart} \\
\mathsf{HeartValve} & \sqsubseteq & \mathsf{BodyValve} \sqcap \exists \mathsf{part\text{-}of}.\mathsf{Heart} \\
\mathsf{Endocarditis} & \sqsubseteq & \mathsf{Inflammation} \sqcap \exists \mathsf{has\text{-}loc}.\mathsf{Endocardium} \\
\mathsf{Inflammation} & \sqsubseteq & \mathsf{Disease} \sqcap \exists \mathsf{acts\text{-}on}.\mathsf{Tissue} \\
\mathsf{Heartdisease} \sqcap \exists \mathsf{has\text{-}loc}.\mathsf{HeartValve} & \sqsubseteq & \mathsf{CriticalDisease} \\
\mathsf{Heartdisease} & \doteq & \mathsf{Disease} \sqcap \exists \mathsf{has\text{-}loc}.\mathsf{Heart} \\
\mathsf{part\text{-}of} \circ \mathsf{part\text{-}of} & \sqsubseteq & \mathsf{part\text{-}of} \\
\mathsf{part\text{-}of} & \sqsubseteq & \mathsf{cont\text{-}in} \\
\mathsf{has\text{-}loc} \circ \mathsf{cont\text{-}in} & \sqsubseteq & \mathsf{has\text{-}loc}
\end{array}
$$

Figure 1: An example $\mathcal{EL}^+$ ontology.

CEL reasoner,[2] and compared its performance on the large bio-medical ontologies SNOMED, GO, and a trimmed-down version of GALEN with the performance of the most advanced tableau-based reasoners FaCT^{++},[3] RacerMaster,[4] and Pellet[5] on these ontologies.

2 $\mathcal{EL}^+$ and the algorithm implemented in CEL

The CEL system currently supports the DL $\mathcal{EL}^+$, whose concepts are formed according to the syntax rule

$$C ::= A \mid \top \mid C \sqcap D \mid \exists r.C,$$

where A ranges over concept names, r over role names, and C, D over concepts. Thus, the concept language of $\mathcal{EL}^+$ is identical to that of $\mathcal{EL}$. The $\cdot^+$ in its name refers to the more powerful ontology formalism. An $\mathcal{EL}^+$ *ontology* is a finite set of *general concept inclusions (GCIs)* $C \sqsubseteq D$ and *role inclusions (RIs)* $r_1 \circ \cdots \circ r_n \sqsubseteq r$. Note that $\mathcal{EL}^+$ ontologies can thus express transitive roles, role hierarchies, and so-called right-identities on roles ($r \circ s \sqsubseteq s$), which are very useful in medical ontologies [15, 11]. The semantics of concepts and ontologies is defined in the usual way (see, e.g., [2]). We write $C \sqsubseteq_{\mathcal{O}} D$ if the concept C is subsumed by the concept D w.r.t. the ontology $\mathcal{O}$. Figure 1 shows a small medical ontology formulated in $\mathcal{EL}^+$. Based on this ontology, it is not hard to see that the

[2] CEL can be downloaded from `http://lat.inf.tu-dresden.de/systems/cel/`

[3] See `http://owl.man.ac.uk/factplusplus/`

[4] See `http://www.racer-systems.com/`

[5] See `http://www.mindswap.org/2003/pellet/`

Normal Form	Completion Rules
$A_1 \sqcap \cdots \sqcap A_n \sqsubseteq B$	**R1** If $A_1, \ldots, A_n \in S(X)$, $A_1 \sqcap \cdots \sqcap A_n \sqsubseteq B \in \mathcal{O}$, and $B \notin S(X)$ then $S(X) := S(X) \cup \{B\}$
$A \sqsubseteq \exists r.B$	**R2** If $A \in S(X)$, $A \sqsubseteq \exists r.B \in \mathcal{O}$, and $(X, B) \notin R(r)$ then $R(r) := R(r) \cup \{(X, B)\}$
$\exists r.A \sqsubseteq B$	**R3** If $(X, Y) \in R(r)$, $A \in S(Y)$, $\exists r.A \sqsubseteq B \in \mathcal{O}$, and $B \notin S(X)$ then $S(X) := S(X) \cup \{B\}$
$r \sqsubseteq s$	**R4** If $(X, Y) \in R(r)$, $r \sqsubseteq s \in \mathcal{O}$, and $(X, Y) \notin R(s)$ then $R(s) := R(s) \cup \{(X, Y)\}$
$r \circ s \sqsubseteq t$	**R5** If $(X, Y) \in R(r)$, $(Y, Z) \in R(s)$, $r \circ s \sqsubseteq t \in \mathcal{O}$, and $(X, Z) \notin R(t)$ then $R(t) := R(t) \cup \{(X, Z)\}$

Figure 2: Normal form and completion rules

concept Endocarditis is subsumed by the concept Heartdisease and CriticalDisease (i.e. Endocarditis $\sqsubseteq_{\mathcal{O}}$ Heartdisease and Endocarditis $\sqsubseteq_{\mathcal{O}}$ CriticalDisease).

The main reasoning service that CEL offers is classification, i.e., the computation of the complete subsumption hierarchy between all concept names occurring in the input ontology $\mathcal{O}$. The CEL reasoner is based on the polynomial time algorithm developed in [2] for the extension $\mathcal{EL}^{++}$ of $\mathcal{EL}^{+}$, which can additionally handle the bottom concept (and thus disjoint concepts), restricted concrete domains, and nonimals (and thus ABoxes). In the following, we briefly introduce the restriction of this algorithm to $\mathcal{EL}^{+}$, and then describe the refined version of this algorithm implemented in our CEL reasoner.

To classify an ontology, the algorithm first transforms it into *normal form*, which requires that all concept and role inclusions are of one of the forms shown in the left part of Figure 2. By introducing new concept and role names and applying a number of simple transformation rules, any ontology $\mathcal{O}$ can be transformed in linear time into a normalized one such that subsumption between the concept names occurring in $\mathcal{O}$ is preserved [16, 2].

For the rest of this section, we assume without loss of generality that the input ontology $\mathcal{O}$ is in normal form. Let $\mathsf{RN}_{\mathcal{O}}$ be the set of all role names occurring in $\mathcal{O}$, $\mathsf{CN}_{\mathcal{O}}$ the set of all concept names occurring in $\mathcal{O}$, and $\mathsf{CN}_{\mathcal{O}}^{\top} := \mathsf{CN}_{\mathcal{O}} \cup \{\top\}$. The algorithm computes

- a mapping S assigning to each element of $\mathsf{CN}_{\mathcal{O}}^{\top}$ a subset of $\mathsf{CN}_{\mathcal{O}}^{\top}$, and
- a mapping R assigning to each element of $\mathsf{RN}_{\mathcal{O}}$ a binary relation on $\mathsf{CN}_{\mathcal{O}}^{\top}$.

The intuition is that these mappings make implicit subsumption relationships explicit in the sense that $B \in S(A)$ implies $A \sqsubseteq_{\mathcal{O}} B$, and $(A, B) \in R(r)$ implies

$A \sqsubseteq_{\mathcal{O}} \exists r.B$. The mappings are initialized by setting $S(A) := \{A, \top\}$ for each $A \in \mathsf{CN}_{\mathcal{O}}^{\top}$ and $R(r) := \emptyset$ for each $r \in \mathsf{RN}_{\mathcal{O}}$. Then the sets $S(A)$ and $R(r)$ are extended by applying the completion rules shown in the right part of Figure 2 until no more rule applies.

In [2], it is shown that this algorithm always terminates in time polynomial in the size of the input ontology, and that it is sound and complete in the following sense: after termination we have $B \in S(A)$ iff $A \sqsubseteq_{\mathcal{O}} B$, for all $A, B \in \mathsf{CN}_{\mathcal{O}}^{\top}$. Thus, the sets $S(A)$ yield a complete representation of the subsumption relation $\sqsubseteq_{\mathcal{O}}$ on the concept names from $\mathcal{O}$ (including $\top$).

It is obvious that, when implementing this algorithm, an efficient approach for finding an applicable rule must be developed. Though a naïve brute-force search for applicable rules would still yield a polynomial time algorithm, its complexity on very large knowledge bases would be prohibitive. As a solution to this problem, we use a refined version of the algorithm, which is inspired by the linear-time algorithm for satisfiability of propositional Horn formulas proposed in [10]. More precisely, our algorithm uses a set of queues, one for each element of $\mathsf{CN}_{\mathcal{O}}^{\top}$, to guide the application of completion rules. Intuitively, the queues list additions to $S(A)$ and $R(r)$ that still have to be carried out. The possible entries of the queues are of the form

$$B_1 \sqcap \cdots \sqcap B_n \rightarrow B' \quad \text{and} \quad \exists r.B$$

with $B_1, \ldots, B_n$, B, and B' concept names, r a role name, and $n \geq 0$. For the case $n = 0$, we simply write the queue entry $B_1 \sqcap \cdots \sqcap B_n \rightarrow B'$ as B'. Intuitively,

- an entry $B_1 \sqcap \cdots \sqcap B_n \rightarrow B'$ in $\mathsf{queue}(A)$ means that B' has to be added to $S(A)$ if $S(A)$ already contains $B_1, \ldots, B_n$, and
- $\exists r.B \in \mathsf{queue}(A)$ means that (A, B) has to be added to $R(r)$.

The fact that such an addition triggers other rules will be taken into account by appropriately extending the queues when the addition is performed.

To facilitate describing the manipulation of the queues, we view the (normalized) input ontology $\mathcal{O}$ as a mapping $\widehat{\mathcal{O}}$ from concepts to sets of queue entries as follows: for each concept name $A \in \mathsf{CN}_{\mathcal{O}}^{\top}$, $\widehat{\mathcal{O}}(A)$ is the minimal set of queue entries such that

- if $A_1 \sqcap \cdots \sqcap A_n \sqsubseteq B \in \mathcal{O}$ and $A = A_i$, then

$$A_1 \sqcap \cdots \sqcap A_{i-1} \sqcap A_{i+1} \sqcap \cdots \sqcap A_n \rightarrow B \in \widehat{\mathcal{O}}(A);$$

- if $A \sqsubseteq \exists r.B \in \mathcal{O}$, then $\exists r.B \in \widehat{\mathcal{O}}(A)$.

procedure process(A, X)
begin
  if $X = B_1, \ldots, B_n \to B$ **and** $B \notin S(A)$ **then**
    if $\{B_1, \ldots, B_n\} \subseteq S(A)$ **then**
      $S(A) := S(A) \cup \{B\}$;
      queue(A) := queue$(A) \cup \widehat{\mathcal{O}}(B)$;
      for all concept names A' and role names r
          with $(A', A) \in R(r)$ **do**
        queue(A') := queue$(A') \cup \widehat{\mathcal{O}}(\exists r.B)$;
  if $X = \exists r.B$ **and** $(A, B) \notin R(r)$ **then**
    process-new-edge(A, r, B)
end;

procedure process-new-edge(A, r, B)
begin
  for all role names s with $r \sqsubseteq^*_{\mathcal{O}} s$ **do**
    $R(s) := R(s) \cup \{(A, B)\}$;
    queue(A) := queue$(A) \cup \bigcup_{\{B' \mid B' \in S(B)\}} \widehat{\mathcal{O}}(\exists s.B')$;
    for all concept names A' and role names t, u with
        $t \circ s \sqsubseteq u \in \mathcal{O}$ **and** $(A', A) \in R(t)$ **and** $(A', B) \notin R(u)$ **do**
      process-new-edge(A', u, B);
    for all concept names B' and role names t, u with
        $s \circ t \sqsubseteq u \in \mathcal{O}$ **and** $(B, B') \in R(t)$ **and** $(A, B') \notin R(u)$ **do**
      process-new-edge(A, u, B');
end;

Figure 3: Processing the queue entries

Likewise, for each concept $\exists r.A$, $\widehat{\mathcal{O}}(\exists r.A)$ is the minimal set of queue entries such that, if $\exists r.A \sqsubseteq B \in \mathcal{O}$, then $B \in \widehat{\mathcal{O}}(\exists r.A)$.

In the modified algorithm, the queues are used as follows: since the sets $S(A)$ are initialized with $\{A, \top\}$, we initialize queue(A) with $\widehat{\mathcal{O}}(A) \cup \widehat{\mathcal{O}}(\top)$, i.e., we add to the queues the *immediate* consequences of being an instance of A and $\top$. Then, we repeatedly fetch (and thereby remove) entries from the queues and process them using the procedure process displayed in Figure 3. To be more precise, process(A, X) is called when the queue of A was non-empty and we fetched the queue entry X from queue(A) to be treated next. Observe that the first if-clause of the procedure process implements **R1** and (part of) **R3**, and the second if-clause implements **R2**, (the rest of) **R3**, as well as **R4** and **R5**. The procedure process-new-edge(A, r, B) is called by process to handle the effects of adding a new pair (A, B) to $R(r)$. The notation $\sqsubseteq^*_{\mathcal{O}}$ used in its top-most **for**-

loop stands for the reflexive-transitive closure of the statements $r \sqsubseteq s$ occurring in $\mathcal{O}$. Queue processing is continued until all queues are empty. Observe that the refined algorithm need not perform *any* search to check which completion rules are applicable.

With a relatively straightforward implementation (in Common LISP) of this algorithm, we were able to classify the large SNOMED ontology (see below) in less than 4 hours (see [4] for this and other experimental results). Since then, however, we have further improved the implementation by changing the strategy of rule applications, changing the encoding of concept and role names, and by using low-level optimizations of the underlying data structure. These optimizations have enhanced the performance of CEL on large real-world ontologies. In particular, CEL can now classify SNOMED in less than half an hour (see below).

3 The experimental results

To test whether CEL can compete with modern tableau based reasoners, we have conducted a number of experiments based on three important bio-medical ontologies: the Gene Ontology (GO) [8], the Galen Medical Knowledge Base (GALEN) [14], and the Systematized Nomenclature of Medicine (SNOMED) [9]. These ontologies provide us with the following benchmark $\mathcal{EL}^+$ ontologies:

- $\mathcal{O}^{\text{GO}}$, which is the latest OWL version of GO;
- $\mathcal{O}^{\text{GALEN}}$, which is a stripped-down version of GALEN obtained by removing inverse role axioms and treating functional roles as ordinary ones;
- $\mathcal{O}^{\text{SNOMED}}$, which is the complete SNOMED ontology;
- $\mathcal{O}^{\text{SNOMED}}_{\mathsf{core}}$, which is the definitorial fragment of $\mathcal{O}^{\text{SNOMED}}$, obtained by keeping only complete concept definitions ($A \equiv C$) , but not the primitive ones ($A \sqsubseteq C$). We consider this fragment in order to obtain another benchmark that is easier to tackle for standard DL reasoners.

Some information about the size and structure of these ontologies is shown in the upper part of Table 1, where CDefs stands for complete concept definitions ($A \equiv C$), PCDefs for primitive concept definitions ($A \sqsubseteq C$), and GCIs for concept inclusions that are neither CDefs nor PCDefs. It is interesting to note that all ontologies except for $\mathcal{O}^{\text{GALEN}}$ are actually acyclic TBoxes.

We have compared the performance of CEL with three of the most advanced tableau-based reasoning systems: FaCT^{++} (v1.1.0), RacerMaster (v1.9.0), and Pellet (v1.3b). All these systems implement expressive DLs in which subsumption is EXPTIME-complete. The experiments have been performed on a PC with 2.8GHz Intel Pentium 4 processor and 512MB memory running Linux v2.6.14.

	$\mathcal{O}^{\text{Go}}$	$\mathcal{O}^{\text{Galen}}$	$\mathcal{O}^{\text{Snomed}}_{\text{core}}$	$\mathcal{O}^{\text{Snomed}}$
No. of CDefs.	0	699	38,719	38,719
No. of PCDefs.	20,465	2041	0	340,972
No. of GCIs	0	1214	0	0
No. of role axioms	1	438	0	11 + 1
$\lvert\mathsf{CN}_{\mathcal{O}}\rvert$	20,465	2,740	53,234	379,691
$\lvert\mathsf{RN}_{\mathcal{O}}\rvert$	1	413	52	52
CEL	5.8	14	95	1,782
FaCT^{++}	6.9	50	740	3,859
RacerMaster	19	14	34,709	*unattainable*
Pellet	1,357	75	*unattainable*	*unattainable*

Table 1: Benchmarks and Evaluation Results

For Pellet, we used JVM v1.5 and set the Java heap space to 256MB (as recommended by the implementor). In the case of Galen, for the sake of fairness also the tableau reasoners have been used with the restricted version of Galen that includes neither functional nor inverse roles. In the case of Snomed, the only existing right-identity rule was passed to CEL, but not to the other reasoners as they do not support right identities. The results of our experiments are summarized in the lower part of Table 1, where all classification times are shown in seconds and *unattainable* means the reasoner failed due to memory exhaustion. Notably, CEL outperforms all the reasoners in all benchmarks except for $\mathcal{O}^{\text{Galen}}$, where CEL and RacerMaster show the same performance.

CEL and FaCT^{++} are the only reasoners that can classify $\mathcal{O}^{\text{Snomed}}$, whereas RacerMaster and Pellet fail. Pellet and the original version of FaCT (not shown in the table) also fail already to classify $\mathcal{O}^{\text{Snomed}}_{\text{core}}$. It seems worth noting that the performance of FaCT^{++} on $\mathcal{O}^{\text{Snomed}}$ degrades dramatically if $\mathcal{O}^{\text{Snomed}}$ is extended with real GCIs. For instance, FaCT^{++} needs about 3,000 more seconds to classify $\mathcal{O}^{\text{Snomed}}$ for each additional randomly generated GCI of the form $\exists r.C \sqsubseteq D$, whereas the performance of CEL does not change noticeably if we add such GCIs.

4 Computing the Subsumption DAG

The innate classification output of CEL is simply the computed sets $S(A)$ for all concept names A. We call these sets *subsumer sets* in what follows. In contrast, tableau-based reasoners usually employ the enhanced traversal method from [3] to generate a directed acyclic graph (DAG) describing the *direct* subsumption relationships, i.e., for every concept name A they compute the sets of its direct

subsumers and subsumees, which are the sets of concept names B such that $A \sqsubseteq_{\mathcal{O}} B$ ($B \sqsubseteq_{\mathcal{O}} A$) and there is no concept name $B' \notin \{A, B\}$ with $A \sqsubseteq_{\mathcal{O}} B' \sqsubseteq_{\mathcal{O}} B$ ($B \sqsubseteq_{\mathcal{O}} B' \sqsubseteq_{\mathcal{O}} A$). We will call this graph the *subsumption DAG*. Since the subsumption relation is a quasi-order rather than a partial order (i.e., in general not antisymmetric), one node of the DAG actually corresponds to an equivalence class of concept names rather than a single concept name. The advantage of using subsumption DAGs over subsumer sets is that this format is more compact, and it directly supports browsing the subsumption hierarchy by going from a concept name to its direct subsumers or subsumees. The disadvantage is that answering a subsumption question $A \sqsubseteq^{?}_{\mathcal{O}} B$ then requires to test reachability of B from A in the DAG, and not just a look-up in the subsumer set $S(A)$.

Since many applications require subsumption DAGs rather than (or in addition to) subsumer sets, CEL allows to construct the former from the latter in an efficient way. In principle, converting subsumer sets into a subsumption DAG is easy. We can simply compute, for each concept name A,

- the set $SS(A) := \{B \in S(A) \mid A \notin S(B)\}$ of strict subsumers of A, i.e., subsumers of A that are not equivalent to A;
- the set $DS(A) := SS(A) \setminus \left(\bigcup_{B \in SS(A)} SS(B)\right)$ of direct subsumers of A;
- the set $DS^{-}(A) := \{B \mid A \in DS(B)\}$ of direct subsumees of A.

Clearly, the sets $DS(A)$ and $DS^{-}(A)$ yield a representation of the subsumption DAG.

However, we do not use this direct construction since computing the sets $DS^{-}(A)$ is expensive (it needs quadratic time) and it is possible to avoid the direct computation of these sets according to the above definition by using an approach that is inspired by the enhanced traversal method in [3]. Another virtue of our alternative approach is that the potentially costly set operations in the computation of $DS(A)$ are replaced by an inexpensive marking algorithm.

In order to explain the main idea underlying our algorithm, assume that we have already computed a restriction of the subsumption DAG to some subset of the concept names, and that we now want to insert the concept name A into this DAG. We start by computing the set $SS(A)$ of strict subsumers according to the definition given above. The elements of $S(A) \setminus SS(A)$ are the concepts that are equivalent to A. To find all the *direct* subsumers of A among the elements of $SS(A)$, we proceed as follows. If all elements of $SS(A)$ belong to the already computed DAG, we can find the direct subsumers by using a simple graph traversal algorithm to mark all the strict subsumers of elements of $SS(A)$ in the DAG. The direct subsumers of A are then those elements of $SS(A)$ that are not marked. If there are elements of $SS(A)$ that do not belong to the already computed DAG, then we simply first insert these elements into the DAG (by

issuing recursive calls of the insertion procedure) before inserting A. By following this strategy, we ensure that, when inserting a concept name A into the DAG, all subsumers of A are already in the DAG, but no subsumee of A is. Hence, our algorithm need not compute the direct subsumees explicitly. Instead, it is enough to extend the set of direct subsumees of B by A in case B is found to be a direct subsumer of A.

Figure 4 shows a pseudo code representation of our algorithm. The sets $\mathsf{parents}(A)$ are used to store the direct subsumers of A, the sets $\mathsf{children}(A)$ are used to store the direct subsumees of A, and the sets $\mathsf{equivalents}(A)$ are used to store the concepts that are equivalent to A. Note that the description of the algorithm is a bit sloppy in that we do not distinguish between a concept name and the node in the DAG representing (the equivalence class of) this name.

An algorithm similar to ours is obtained if we describe the subsumer sets as a primitive TBox, i.e. a set of primitive concept definitions $A \sqsubseteq \sqcap_{B_i \in S(A)} B_i$ for each concept name A, and then employ a simplified version of the enhanced traversal method [3] using told subsumer information and some of the optimizations described in [12] to compute the subsumption DAG from the resulting TBox.

The time required by CEL for computing subsumption DAGs is very small. For example, even in the case of $\mathcal{O}^{\text{SNOMED}}$, which has almost 380,000 concepts and huge subsumer sets, it takes only 9 seconds. This is negligible compared to the time needed to compute the subsumer sets. In particular, if we add this time to CEL's run-time on $\mathcal{O}^{\text{SNOMED}}$ in Table 1, CEL is still more than twice as fast as FaCT^{++}.

There is an obvious alternative to first computing the full subsumer sets, and only then deriving the subsumption DAG from them: we could modify our classification algorithm, which computes the whole subsumption hierarchy, into a subsumption algorithm that answers only a single subsumption query, and then use this subsumption algorithm inside a standard enhanced traversal algorithm as described in [3]. We have experimented with this strategy, which is closer to the approach employed by tableau-based systems. To turn our algorithm into a subsumption algorithm, we have developed a goal-directed variant of it, which is based on activating a concept name if computing its subsumer set is required for answering the subsumption question at hand. If the aim is to answer the subsumption query $A \sqsubseteq^?_{\mathcal{O}} B$, then initially only A is activated. Intuitively, completion rules are only applied to activated names. We activate a concept name B' whenever B' is the second component of a tuple added to some $R(r)$. The set $S(A')$ and the queue of A' is initialized only when the concept name becomes activated, and thus the subsumer sets of concept names that do not become activated are not populated by the algorithm. During the construction of the whole subsumption DAG, the enhanced traversal procedure makes repeated calls to the subsumption algorithm. To avoid redoing work, we retain the already

```
procedure compute-dag
    for all concept names X ∈ CN_O^⊤ do
        classified(X) := false
        parents(X) := children(X) := equivalents(A) := ∅
    for each concept name A ∈ CN_O^⊤ do
        if not classified(A) then
            dag-classify(A);
end;

procedure dag-classify(A)
    candidates := ∅;
    for all subsumers B ∈ S(A) do
        if A ∈ S(B) then
            classified(B) := true;
            equivalents(A) := equivalents(A) ∪ {B};
        else
            if not classified(B) then
                dag-classify(B);
            candidates := candidates ∪ {B};
    dag-insert(A, candidates);
    classified(B) := true;
end;

procedure dag-insert(A, candidates)
    marked(X) := false for all X ∈ CN_O^⊤;
    for all B ∈ candidates do
        for all X ∈ parents(B) do
            marked(X) := true
    while there are X, Y ∈ CN_O^⊤ with marked(X), Y ∈ parents(X), and
          not marked(Y) do
        marked(Y) := true
    parents(A) := {B ∈ candidates | marked(B) = false};
    for all B ∈ parents(A) do
        children(B) := children(B) ∪ {A};
end;
```

Figure 4: Computing the DAG from the subsumer sets

computed parts of the mappings $S(\cdot)$ and $R(\cdot)$ for such repeated calls.

However, our current implementation of this idea cannot compete with the runtime of the original CEL implementation described before. For example, the classification of $\mathcal{O}^{\text{SNOMED}}$ takes 3,750 seconds. This is still slightly better than

the performance of FaCT^{++}, but more than twice of the 1,791 seconds needed when first computing the subsumer sets and then constructing the subsumption DAG. The reason is probably that, in sum, the single subsumption tests do the same work as the full classification algorithm, but then there is the additional overhead of the enhanced traversal method (which is more complicated than the simplified version employed to compute the subsumption DAG from the subsumer sets).

5 Conclusion

The performance evaluations show that our tractable reasoner CEL outperforms modern reasoners for intractable DLs based on tableau algorithms. It should be noted that the good performance of CEL is achieved with a relatively straightforward implementation of the tractable algorithm, whereas the tableau-based systems are the result of many years of research into optimization techniques. The robustness and scalability of tractable reasoning is visible in the case of SNOMED, which comprises almost 380,000 concept definitions. Only CEL and FaCT^{++} can classify this terminology, whereas RacerMaster, Pellet, and the original version of FaCT fail. Additionally, FaCT^{++} shows a significant degradation in performance if SNOMED, which is an acyclic TBox, is extended with GCIs. In contrast, the runtime of CEL is not noticeably affected by such an extension.

Developing CEL is ongoing work. We plan to extend its capabilities to the DL $\mathcal{EL}^{++}$ [2], which includes, among other things, *nominals* and *the bottom concept* (thus one can express *ABoxes* and *disjoint concepts*). We also plan to implement DIG and OWL interfaces,[6] so that CEL can be used as a backend reasoner for ontology editors like OilEd[7] and Protégé,[8] which would also make their sophisticated graphical user-interfaces available to users of CEL.

References

[1] F. Baader. Terminological cycles in a description logic with existential restrictions. In *Proc. IJCAI'03*, 2003.

[2] F. Baader, S. Brandt, and C. Lutz. Pushing the $\mathcal{EL}$ envelope. In *Proc. IJCAI'05*, 2005.

[3] F. Baader, E. Franconi, B. Hollunder, B. Nebel, and H.-J. Profitlich. An empirical analysis of optimization techniques for terminological representation systems or: Making KRIS get a move on. *Applied Artificial Intelligence*, 4:109–132, 1994.

[6] See `http://dl.kr.org/dig/` and `http://www.w3.org/2004/OWL/`

[7] See `http://oiled.man.ac.uk/`

[8] See `http://protege.stanford.edu/`

[4] F. Baader, C. Lutz, and B. Suntisrivaraporn. Is tractable reasoning in extensions of the description logic $\mathcal{EL}$ useful in practice? In *Proc. M4M'05*, 2005.

[5] R. J. Brachman and H. J. Levesque. The tractability of subsumption in frame-based description languages. In *Proc. AAAI'84*, 1984.

[6] S. Brandt. Polynomial time reasoning in a description logic with existential restrictions, GCI axioms, and—what else? In *Proc. ECAI'04*, 2004.

[7] D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, and R. Rosati. DL-Lite: Tractable description logics for ontologies. In *Proc. AAAI'05*, 2005.

[8] The Gene Ontology Consortium. Gene Ontology: Tool for the unification of biology. *Nature Genetics*, 25:25–29, 2000.

[9] R. Cote, D. Rothwell, J. Palotay, R. Beckett, and L. Brochu. The systematized nomenclature of human and veterinary medicine. Technical report, SNOMED International, Northfield, IL: College of American Pathologists, 1993.

[10] W. F. Dowling and J. Gallier. Linear-time algorithms for testing the satisfiability of propositional horn formulae. *J. Logic Programming*, 1(3):267–284, 1984.

[11] I. Horrocks and U. Sattler. Decidability of SHIQ with complex role inclusion axioms. *Artificial Intelligence*, 160(1–2):79–104, 2004.

[12] I. Horrocks and D. Tsarkov. Optimised classification for taxonomic knowledge bases. In *Proc. DL'05*, 2005.

[13] B. Nebel. Terminological reasoning is inherently intractable. *Artificial Intelligence*, 43:235–249, 1990.

[14] A. Rector and I. Horrocks. Experience building a large, re-usable medical ontology using a description logic with transitivity and concept inclusions. In *Proc. Ontological Engineering Workshop, AAAI Spring Symposium*, 1997.

[15] K.A. Spackman. Managing clinical terminology hierarchies using algorithmic calculation of subsumption: Experience with SNOMED-RT. *J. of the American Medical Informatics Association*, 2000. Fall Symposium Special Issue.

[16] B. Suntisrivaraporn. Optimization and implementation of subsumption algorithms for the description logic $\mathcal{EL}$ with cyclic TBoxes and general concept inclusion axioms. Master thesis, TU Dresden, Germany, 2005.

DL Actions with GCIs: a Pragmatic Approach

H. Liu[1], C. Lutz[1], M. Miličić[1], F. Wolter[2]

[1]Institut für Theoretische Informatik
TU Dresden, Germany
lastname@tcs.inf.tu-dresden.de

[2]Department of CS
Univ. of Liverpool, UK
frank@csc.liv.ac.uk

Abstract

We recently proposed action formalisms based on description logics (DLs) as decidable fragments of well-established action theories such as the Situation Calculus and the Fluent Calculus. One short-coming of our initial proposal is that the considered formalisms admit only acyclic TBoxes, but not GCIs. In this paper, we define DL action formalisms that admit GCIs, propose a pragmatic approach to addressing the ramification problem that is introduced in this way, show that our formalim is decidable and investigate its computational complexity.

1 Introduction

Action theories such as the Situation Calculus (SitCalc) and the Fluent Calculus aim at describing actions in a semantically adequate way [8, 10]. They are usually formulated in first- or higher-order logic and do not admit decidable reasoning. For reasoning about actions in practical applications, such theories are thus not directly suited. There are two obvious ways around this problem: the first one is to accept undecidability and replace reasoning by programming. This route is taken by the inventors of action-oriented programming languages such as Golog [3] and Flux [11], whose semantics is based on the SitCalc and Fluent Calculus, respectively. The second one is to try to identify fragments of action theories such as SitCalc that are sufficiently expressive to be useful in applications, but nevertheless admit decidable reasoning. For example, a simple such fragment is obtained by allowing only propositional logic for describing the state of the world and pre- and post-conditions of actions. A much more expressive formalism was identified in our recent paper [1], where we define action formalisms that are based on description logics (DLs). More precisely, we use DL ABoxes to describe the state of the world and pre- and post-conditions of actions

and prove that reasoning in the resulting formalism is decidable [1]. We also show in [1] that, in this way, we actually get a decidable fragment of SitCalc. However, the DL action formalism defined in [1] has two major limitations: first, we only admit acyclic TBoxes, but no general TBoxes involving GCIs. And second, we allow only concept names (but no complex concepts) in post-conditions and additionally stipulate that these concept names are *not* defined in the TBox. In the present paper, we present a pragmatic approach to overcoming these limitations while retaining decidability of reasoning. In particular, we show how to incorporate general TBoxes into DL action formalisms. Since there is no clear notion of a concept name "being defined" in a general TBox, we also drop the second restriction and admit arbitrary concepts in post-conditions.

The main reason for adopting the mentioned restrictions in [1] was that they disarm the frame and ramification problem, which pose major difficulties in reasoning about actions. When admitting general TBoxes, in particular the ramification problem becomes a serious issue. Attempts to *automatically* solve this problem, e.g. by adopting a Winslett-style PMA semantics [14], lead to semantic and computational problems: we show in [1] that counter-intuitive results and undecidability of reasoning are the consequence of adopting such a semantics. Since there appears to be no general automated solution to the ramification problem introduced by general TBoxes, we take a rather pragmatic approach and leave it to the designer of an action description to fine-tune the ramifications of the action. This is similar to what is done in the SitCalc and the Fluent Calculus to address the ramification problem. There, the designer of an action description can control the ramifications of the action by specifying causal relationships between predicates [4, 9]. While causality appears to be a satisfactory approach for addressing the ramification problem in the case of propositional state constraints (which correspond to a TBox formulated in propositional logic), it seems not powerful enough for attacking the ramifications introduced by general TBoxes, which may involve complex quantification patterns. We therefore adopt a different strategy for controlling ramifications: when describing an action, the user can specify the predicates that can change by executing the action, as well as those that cannot change. To allow an adequate fine-tuning of ramifications, we admit rather complex statements about the change of predicates such as "the concept name A can change from positive to negative only at the indidivual a, and from negative to positive only where the complex concept C was satisfied before the action was executed".

The family of action formalisms introduced in this paper can be parameterized with any description logic. We show that, for many standard DLs, the reasoning problems *executability* and *projection* in the corresponding action formalism are decidable. We also pinpoint the exact computational complexity of these reasoning problems for several members of the $\mathcal{ALCQIO}$ family of DLs. As a rule of thumb, our results show that reasoning in the action formalism

instantiated with a description logic $\mathcal{L}$ is of the same complexity as subsumption in $\mathcal{L}$ extended with nominals. For fine-tuning ramifications, deciding the consistency of actions is of prime importance. We introduce two notions of consistency (weak and strong) and show that one of them is of the same complexity as deciding projection while the other one is undecidable even when the action formalism is instantiated with $\mathcal{ALC}$.

2 Describing Actions

The action formalism proposed in this paper is an extension of the one from [1] and is not restricted to a particular DL. However, for our complexity results we consider the DL $\mathcal{ALCQIO}$ and its fragments. We refrain from introducing the syntax and semantics of $\mathcal{ALCQIO}$ in full detail, referring e.g. to [2], and only give a few central definitions. A *concept literal* is a concept name or the negation thereof, and a *role literal* is a role name or the negation thereof. An *ABox assertion* is of the form $C(a)$ or $r(a, b)$, where a, b are individual names, C is a concept, and r a role literal. An *ABox* $\mathcal{A}$ is a finite set of ABox assertions. A *general concept inclusion axiom (GCI)* is an expression of the form $C \sqsubseteq D$, where C and D are an concepts. A *(general) TBox* $\mathcal{T}$ is a finite set of GCIs.

The main ingredients of our approach to reasoning about actions are action descriptions (as defined below), ABoxes for describing the current knowledge about the state of affairs in the application domain, and TBoxes for describing general knowledge about the application domain similar to state constraints in the SitCalc and Fluent Calculus. On the semantic side, interpretations are used to describe the state of affairs in the application domain. Thus, the knowledge described by an ABox is incomplete: ABoxes may admit more than a single model, and all the corresponding states of affairs are considered possible. Before we go deeper into the semantics, we introduce the syntax of action descriptions. We use $\mathcal{LO}$ to denote the extension of a description logic $\mathcal{L}$ with nominals.

Definition 1 (Action). Let $\mathcal{L}$ be a description logic. An *atomic $\mathcal{L}$-action* $\alpha = (\mathsf{pre}, \mathsf{occ}, \mathsf{post})$ consists of

- a finite set pre of $\mathcal{L}$-ABox assertions, the *pre-conditions*;
- the *occlusion pattern* occ which is a set of mappings $\{\mathsf{occ}_{\varphi_1}, \ldots, \mathsf{occ}_{\varphi_n}\}$ indexed by $\mathcal{L}$-ABox assertions $\varphi_1, \ldots, \varphi_n$ such that each occ_{φ_i} assigns
  - to every concept literal A an $\mathcal{LO}$-concept $\mathsf{occ}_{\varphi_i}(A)$,
  - to every role literal r a finite set $\mathsf{occ}_{\varphi_i}(r)$ of pairs of $\mathcal{LO}$-concepts.
- a finite set post of *conditional post-conditions* of the form φ/ψ, where φ and ψ are $\mathcal{L}$-ABox assertions.

A *composite $\mathcal{L}$-action* is a finite sequence of atomic $\mathcal{L}$-actions $\alpha_1, \ldots, \alpha_n$.

Applying an action changes the state of affairs, and thus transforms an interpretation $\mathcal{I}$ into an interpretation $\mathcal{I}'$. Intuitively, the pre-conditions specify under which conditions the action is applicable. The post-condition φ/ψ says that, if φ is true in the original interpretation $\mathcal{I}$, then ψ is true in the interpretation $\mathcal{I}'$ obtained by applying the action. The purpose of the occlusion pattern is to control the changes that are allowed to occur during the execution of the action. This is necessary because, as will be discussed in more detail later, the presence of TBoxes often requires to allow more changes than those that are directly enforced via the post-conditions. To illustrate how the occlusion pattern works, suppose $\mathsf{occ} = \{\mathsf{occ}_{\varphi_1}, \ldots, \mathsf{occ}_{\varphi_n}\}$ and $\varphi_{i_1}, \ldots, \varphi_{i_m}$ are the assertions among the indexes in occ that are true in the original interpretation $\mathcal{I}$. If A is a concept name, then instances of the concept

$$\mathsf{occ}_{\varphi_{i_1}}(A) \sqcup \cdots \sqcup \mathsf{occ}_{\varphi_{i_m}}(A)$$

in $\mathcal{I}$ may change from A in $\mathcal{I}$ to $\neg A$ in $\mathcal{I}'$, but non-instances may not. Likewise, instances of

$$\mathsf{occ}_{\varphi_{i_1}}(\neg A) \sqcup \cdots \sqcup \mathsf{occ}_{\varphi_{i_m}}(\neg A)$$

may change from $\neg A$ to A. For role names, $(C, D) \in \mathsf{occ}_{\varphi_{i_k}}(r)$ means that pairs from $C^{\mathcal{I}} \times D^{\mathcal{I}}$ that have been connected by r in $\mathcal{I}$ may lose this connection in $\mathcal{I}'$, and similarly for the occlusion of negated role names. Note that the indexing of the mappings occ_φ with ABox assertions makes the occlusions conditional: the occlusion occ_φ is only active if φ is satisfied by the original interpretation $\mathcal{I}$. We will explain later why nominals are always admitted in the occlusion pattern, even if they are not provided by $\mathcal{L}$.

For defining the semantics in a succinct way, it is convenient to introduce the following abbreviation. For an action α with $\mathsf{occ} = \{\mathsf{occ}_{\varphi_1}, \ldots, \mathsf{occ}_{\varphi_n}\}$, an interpretation $\mathcal{I}$, a concept literal A, and a role literal r, we set

$$\begin{aligned}(\mathsf{occ}(A))^{\mathcal{I}} &:= \bigcup_{\mathcal{I} \models \varphi_i} (\mathsf{occ}_{\varphi_i}(A))^{\mathcal{I}} \\ (\mathsf{occ}(r))^{\mathcal{I}} &:= \bigcup_{(C,D) \in \mathsf{occ}_{\varphi_i}(r), \mathcal{I} \models \varphi_i} (C^{\mathcal{I}} \times D^{\mathcal{I}})\end{aligned}$$

Thus, $\mathsf{occ}(B)^{\mathcal{I}}$ describes those elements of $\Delta^{\mathcal{I}}$ that may change from B to $\neg B$ when going to $\mathcal{I}'$.

Definition 2 (Action semantics). Let $\alpha = (\mathsf{pre}, \mathsf{occ}, \mathsf{post})$ be an atomic action and $\mathcal{I}, \mathcal{I}'$ interpretations sharing the same domain and interpretation of all individual names. We say that α *may transform* $\mathcal{I}$ *to* $\mathcal{I}'$ w.r.t. a TBox $\mathcal{T}$ ($\mathcal{I} \Rightarrow^{\mathcal{T}}_{\alpha} \mathcal{I}'$) iff the following holds:

- $\mathcal{I}, \mathcal{I}'$ are models of $\mathcal{T}$;
- for all $\varphi/\psi \in \mathsf{post}$: $\mathcal{I} \models \varphi$ implies $\mathcal{I}' \models \psi$ (written $\mathcal{I}, \mathcal{I}' \models \mathsf{post}$);

- for each $A \in \mathsf{N_C}$ and $r \in \mathsf{N_R}$, we have

$$\begin{array}{rcll} A^{\mathcal{I}} \setminus A^{\mathcal{I}'} & \subseteq & (\mathsf{occ}(A))^{\mathcal{I}} & \neg A^{\mathcal{I}} \setminus \neg A^{\mathcal{I}'} \subseteq (\mathsf{occ}(\neg A))^{\mathcal{I}} \\ r^{\mathcal{I}} \setminus r^{\mathcal{I}'} & \subseteq & (\mathsf{occ}(r))^{\mathcal{I}} & \neg r^{\mathcal{I}} \setminus \neg r^{\mathcal{I}'} \subseteq (\mathsf{occ}(\neg r))^{\mathcal{I}} \end{array}$$

The composite action $\alpha_1, \ldots, \alpha_n$ *may transform* $\mathcal{I}$ to $\mathcal{I}'$ w.r.t. $\mathcal{T}$ ($\mathcal{I} \Rightarrow^{\mathcal{T}}_{\alpha_1,\ldots,\alpha_n} \mathcal{I}'$) iff there are models $\mathcal{I}_0, \ldots, \mathcal{I}_n$ of $\mathcal{T}$ with $\mathcal{I} = \mathcal{I}_0$, $\mathcal{I}' = \mathcal{I}_n$, and $\mathcal{I}_{i-1} \Rightarrow^{\mathcal{T}}_{\alpha_i} \mathcal{I}_i$ for $1 \leq i \leq n$.

We now give a simple example for DL actions that illustrates the ramification problem introduced by general TBoxes and how occlusion patterns can be used to control this problem. The TBox $\mathcal{T}$ contains the following GCI, which says that everybody registered for a course has access to the university library:

$$\exists \mathsf{registered\_for}.\mathsf{Course} \sqsubseteq \exists \mathsf{access\_to}.\mathsf{Library}$$

The ABox $\mathcal{A}$, which describes the current state of the world, is defined as:

$$\mathsf{Course}(\mathsf{cs}), \quad \neg\exists \mathsf{registered\_for}.\mathsf{Course}(\mathsf{peter}), \quad \neg\exists \mathsf{access\_to}.\mathsf{Library}(\mathsf{peter}).$$

Obviously, $\mathcal{A}$ states that computer science is a course and that Peter is neither registered for a course nor has access to a library. Now, the action

$$\alpha := (\emptyset, \mathsf{occ}, \{\mathsf{taut}/\mathsf{registered\_for}(\mathsf{peter}, \mathsf{cs})\})$$

describes the registration of Peter for the computer science course. For simplicity, the set of pre-conditions is empty and taut is some valid ABox assertion (say $\top(\mathsf{cs})$), i.e., the post-condition is unconditional.

Following the law of inertia, it may seem reasonable to specify occ such that only the fact stated explicitly in the post-condition can change when executing the action: occ consists of just one (unconditional) mapping $\mathsf{occ}_{\mathsf{taut}}$ which maps all concept and role literals with the exception of $\neg\mathsf{registered\_for}$ to $\bot$ and $\{(\bot, \bot)\}$, respectively. By setting

$$\mathsf{occ}_{\mathsf{taut}}(\neg\mathsf{registered\_for}) := \{(\{\mathsf{peter}\}, \{\mathsf{cs}\})\},$$

where $\{\mathsf{peter}\}$ and $\{\mathsf{cs}\}$ are nominals, we achieve the desired effect that only the pair $(\mathsf{peter}, \mathsf{cs})$ can be added to "$\mathsf{registered\_for}$", and nothing else can be changed. This shows why nominals are indispensible in the occlusion pattern: without them, we are not able to occlude only the change that is enforced by simple post-conditions such as the one in the example.

However, our choice of the occlusion pattern is too strict. Due to the presence of the TBox, the action is inconsistent in the sense that there is no model $\mathcal{I}$ of $\mathcal{A}$ and $\mathcal{T}$ such that $\mathcal{I} \Rightarrow^{\mathcal{T}}_{\alpha} \mathcal{I}'$ for some model $\mathcal{I}'$ of $\mathcal{T}$. The reason is that, due to $\mathcal{T}$,

Peter should have access to a library after execution of the action. Since he does not have access before the action and the occlusion pattern does not allow us to change $\mathsf{access\_to}$, there is no way to achieve this. This shows that admitting general TBoxes induces a ramification problem: there may be indirect effects of an action that are not directly stated in the post-conditions. Such ramifications, which led us to introducing occlusion patterns in the current paper, cannot occur in the more basic DL action formalism introduced in [1] where we admit only acyclic TBoxes and primitive concepts in post-conditions. An obvious way to regain consistency of the action α is to allow via the occlusion pattern that $\mathsf{access\_to}$ can change in the required way, i.e., setting

$$\mathsf{occ}_{\mathsf{taut}}(\neg\mathsf{access\_to}) := \{(\{\mathsf{peter}\}, \mathsf{Library})\}$$

and thus allow Peter to have access to a library after the action.

In the example above, for simplicity we did not use conditional post-conditions, i.e., conditions φ/ψ where φ is not taut. For this reason, it was sufficient that occ consists of a single (unconditional) mapping as well. Obviously, controlling the ramifications of conditional post-conditions requires occlusion patterns that are conditional as well.

The above example suggests that deciding consistency of an action is an important task because failure to specify the occlusion pattern in a proper way can result in inconsistent actions. In the following, we propose two notions of consistency.

Definition 3 (Consistency). Let $\alpha = \alpha_1, \ldots, \alpha_n$ be a composite action, let $\mathcal{T}$ be a TBox, and $\mathcal{A}$ an ABox. We say that

- α is *weakly consistent with* $\mathcal{T}$ *and* $\mathcal{A}$ iff there is a model $\mathcal{I}$ of $\mathcal{T}$ and $\mathcal{A}$, and a model $\mathcal{I}'$ of $\mathcal{T}$ such that $\mathcal{I} \Rightarrow_\alpha^\mathcal{T} \mathcal{I}'$.
- α is *strongly consistent with* $\mathcal{T}$ *and* $\mathcal{A}$ iff for all models $\mathcal{I}$ of $\mathcal{T}$ and $\mathcal{A}$, there is a model $\mathcal{I}'$ of $\mathcal{T}$ such that $\mathcal{I} \Rightarrow_\alpha^\mathcal{T} \mathcal{I}'$.

Ideally, the designer of an action $\alpha = (\mathsf{pre}, \mathsf{occ}, \mathsf{post})$ should establish strong consistency of α w.r.t. the relevant TBox and the ABox pre. Note that strong consistency of an action α w.r.t. $\mathcal{T}$ and pre implies strong consistency of α w.r.t. $\mathcal{T}$ and $\mathcal{A}$ for all ABoxes $\mathcal{A}$ that satisfy the preconditions pre. Unfortunately, we shall show later that strong consistency is undecidable even for $\mathcal{ALC}$ actions. In contrast, weak consistency will turn out to be decidable. As demonstrated by our example, in which the first attempt to specify occ yields an action that is not even weakly consistent, checking for weak consistency is helpful to detect severe ramfication problems. In the case of weak consistency, consistency w.r.t. pre does not imply consistency w.r.t. all ABoxes satisfying the pre-conditions. Hence, the designer of an action should not only consider the ABox pre when

checking weak consistency, but also additional ABoxes that he considers typical for the application domain.

To check whether an action can be applied in a given situation, the user wants to know whether it is (strongly consistent and) executable, where executable means that all pre-conditions are satisfied in the states of the world considered possible. If the action is executable, he wants to know whether applying it achieves the desired effect, i.e., whether an assertion that he wants to make true really holds after executing the action. This problem is usually called projection [8, 1].

Definition 4 (Executability and projection). Let $\alpha_1, \ldots, \alpha_n$ be a composite action with $\alpha_i = (\mathsf{pre}_i, \mathsf{occ}_i, \mathsf{post}_i)$ for $i = 1, \ldots, n$, let $\mathcal{T}$ be a TBox, and $\mathcal{A}$ an ABox.

- Executablity: $\alpha_1, \ldots, \alpha_n$ is *executable in $\mathcal{A}$ w.r.t. $\mathcal{T}$* iff the following conditions are true for all models $\mathcal{I}$ of $\mathcal{A}$ and $\mathcal{T}$:
  - $\mathcal{I} \models \mathsf{pre}_1$
  - for all i with $1 \leq i < n$ and all interpretations $\mathcal{I}'$ with $\mathcal{I} \Rightarrow^{\mathcal{T}}_{\alpha_1,\ldots,\alpha_i} \mathcal{I}'$, we have $\mathcal{I}' \models \mathsf{pre}_{i+1}$.
- Projection: The assertion φ is a *consequence of applying $\alpha_1, \ldots, \alpha_n$ in $\mathcal{A}$ w.r.t. $\mathcal{T}$* iff for all models $\mathcal{I}$ of $\mathcal{A}$ and $\mathcal{T}$ and for all $\mathcal{I}'$ with $\mathcal{I} \Rightarrow^{\mathcal{T}}_{\alpha_1,\ldots,\alpha_n} \mathcal{I}'$, we have $\mathcal{I}' \models \varphi$.

It is not too difficult to see that the action formalism just introduced is a generalization of the one introduced in [1], for details see [5]. As in [1], projection and executability are mutually reducible in polynomial time. Moreover, (i) an action α is weakly consistent with a TBox $\mathcal{T}$ and ABox $\mathcal{A}$ iff $\bot(a)$ is not a consequence of applying α in $\mathcal{A}$ w.r.t. $\mathcal{T}$; and (ii) φ is a consequence of applying $\alpha = (\mathsf{pre}, \mathsf{occ}, \mathsf{post})$ in $\mathcal{A}$ w.r.t. $\mathcal{T}$ iff the action $(\mathsf{pre}, \mathsf{occ}, \mathsf{post} \cup \{\top(a)/\neg\varphi\})$ is not weakly consistent with $\mathcal{T}$ and $\mathcal{A}$. Thus, since both executability and weak consistency can be reduced to (non-)projection and vice versa, we will concentrate on the latter throughout this paper.

3 Deciding Projection in $\mathcal{ALCQI}$ and $\mathcal{ALCQIO}$

We consider the prominent DLs $\mathcal{ALCQI}$ and $\mathcal{ALCQIO}$ and show that projection is co-NExpTime-complete. It is shown in [5] that Lemma 8 of [1] implies the following.

Theorem 5. *Projection and executability (weak consistency) in $\mathcal{ALCQI}$ are co-*NExpTime*-hard (*NExpTime*-hard) even if occlusions for roles are restricted to $\{(\bot, \bot)\}$ and only nominals are allowed in the occlusions of concept names.*

Note that projection in $\mathcal{ALCQI}$ is thus harder than subsumption in the same logic, which is ExpTime-complete [13]. In the case of $\mathcal{ALCQIO}$, the complexities of subsumption and projection coincide. Intuitively (and as shown by the proof of Theorem 5), projection in a logic $\mathcal{L}$ should be expected to be of the same complexity as subsumption in $\mathcal{L}$ extended with nominals.

In the following, we establish a matching co-NExpTime upper bound for projection in $\mathcal{ALCQIO}$ (and thus also $\mathcal{ALCQI}$). The proof proceeds by reducing projection in $\mathcal{ALCQIO}$ to ABox (in)consistency in $\mathcal{ALCQIO}^{\neg,\cap,\cup}$, i.e. the extension of $\mathcal{ALCQIO}$ with the Boolean role constructors.

Let $\alpha_1, \dots, \alpha_n$ be a composite action with $\alpha_i = (\mathsf{pre}_i, \mathsf{occ}_i, \mathsf{post}_i)$ for $i = 1, \dots, n$, and let $\mathcal{T}$ be a TBox, $\mathcal{A}_0$ an ABox and φ_0 an assertion. We are interested in deciding whether φ_0 is a consequence of applying $\alpha_1, \dots, \alpha_n$ in $\mathcal{A}_0$ w.r.t. $\mathcal{T}$. In what follows, we call $\alpha_1, \dots, \alpha_n$, $\mathcal{T}$, $\mathcal{A}_0$ and φ_0 the *input*. W.l.o.g., we make the following assumptions:

- φ_0 is of the form $\varphi_0 = C_0(a_0)$, where C_0 is a (possibly complex) concept.

  This assumption can be made because an assertion $r(a, b)$ can be replaced with $(\exists r.\{b\})(a)$, and $\neg r(a, b)$ with $(\neg\exists r.\{b\})(a)$.

- Each occlusion pattern occ_i contains exactly one occlusion pattern that is unconditional (i.e., indexed by taut) and formulated in $\mathcal{ALCQIO}^{\neg,\cap,\cup}$.

  An occlusion pattern $\{\mathsf{occ}_{\varphi_1}, \dots, \mathsf{occ}_{\varphi_n}\}$ can be converted into an occlusion pattern $\{\mathsf{occ}_{\mathsf{taut}}\}$ formulated in $\mathcal{ALCQIO}^{\neg,\cap,\cup}$ as follows. First, we may assume w.l.o.g. that φ_i is of the form $C_i(a_i)$ for $1 \leq i \leq n$ (see previous point). For $1 \leq i \leq n$, let P_i denote the concept $\forall U.(\{a_i\} \to C_i)$, where U denotes the universal role, i.e. $r \cup \neg r$ for some $r \in \mathsf{N_R}$. Then, define for each concept literal A

  $$\mathsf{occ}_{\mathsf{taut}}(A) := \bigsqcup_{1 \leq i \leq n} \big(P_i \sqcap \mathsf{occ}_{\varphi_i}(A)\big)$$

  Likewise, for each role literal r, define

  $$\mathsf{occ}_{\mathsf{taut}}(r) := \{(P_i \sqcap C, P_i \sqcap D) \mid (C, D) \in \mathsf{occ}_{\varphi_i}\}.$$

  Having the occlusion pattern formulated in $\mathcal{ALCQIO}^{\neg,\cap,\cup}$ is unproblematic since our reduction is to $\mathcal{ALCQIO}^{\neg,\cap,\cup}$ anyway. In the following, we slightly abuse notation and confuse the singleton set occ_i with the (unconditional) occlusion mapping contained in it.

The idea of the reduction is to define an ABox $\mathcal{A}_{\mathsf{red}}$ and a TBox $\mathcal{T}_{\mathsf{red}}$ such that a single model of them encodes a sequence of interpretations $\mathcal{I}_0, \dots, \mathcal{I}_n$ such that $\mathcal{I}_0 \models \mathcal{A}_0, \mathcal{T}$ and $\mathcal{I}_{i-1} \Rightarrow^{\mathcal{T}}_{\alpha_i} \mathcal{I}_i$ for $i = 1, \dots, n$. In the following, we use Sub to denote the set of subconcepts of the concepts which occur in the input. In the

reduction, we introduce concept names $A^{(i)}$ and role names $r^{(i)}$ for every concept name A and every role name r used in the input, and every $i \leq n$. Intuitively, $A^{(i)}$ and $r^{(i)}$ represent the extensions of A and r in the i-th interpretation. For a complex concept $C \in \mathsf{Sub}$, we use $C^{(i)}$, for $i \leq n$, to denote the concept obtained by replacing all concept names A and role names r occurring in C by $A^{(i)}$ and $r^{(i)}$ respectively.

We start by assembling the reduction ABox $\mathcal{A}_{\mathsf{red}}$. First, define a "copy" $\mathcal{A}_{\mathsf{ini}}$ of the input ABox $\mathcal{A}_0$ as:

$$\begin{array}{rl} \mathcal{A}_{\mathsf{ini}} := & \{C^{(0)}(a) \mid C(a) \in \mathcal{A}_0\} \cup \\ & \{r^{(0)}(a,b) \mid r(a,b) \in \mathcal{A}_0\} \cup \{\neg r^{(0)}(a,b) \mid \neg r(a,b) \in \mathcal{A}_0\} \end{array}$$

Then, introduce abbreviations, for $i \leq n$:

$$\begin{array}{rcl} \mathsf{p}_i(C(a)) & := & \forall U.(\{a\} \rightarrow C^{(i)}), \\ \mathsf{p}_i(r(a,b)) & := & \forall U.(\{a\} \rightarrow \exists r^{(i)}.\{b\}), \\ \mathsf{p}_i(\neg r(a,b)) & := & \forall U.(\{a\} \rightarrow .\forall r^{(i)}.\neg\{b\}), \end{array}$$

Now we can define the components of $\mathcal{A}_{\mathsf{red}}$ that take care of post-condition satisfaction. For $1 \leq i \leq n$, we define:

$$\mathcal{A}^{(i)}_{\mathsf{post}} := \{\big(\mathsf{p}_{i-1}(\varphi) \rightarrow \mathsf{p}_i(\psi)\big)(a_0) \mid \varphi/\psi \in \mathsf{post}_i\}$$

We assemble $\mathcal{A}_{\mathsf{red}}$ as

$$\mathcal{A}_{\mathsf{red}} := \mathcal{A}_{\mathsf{ini}} \cup \bigcup_{1 \leq i \leq n} \mathcal{A}^{(i)}_{\mathsf{post}}.$$

Next, we define the components of the TBox $\mathcal{T}_{\mathsf{red}}$. Since all interpretations $\mathcal{I}_0, \ldots, \mathcal{I}_n$ have to be models of the input TBox $\mathcal{T}$, we define for each $i \leq n$, a copy $\mathcal{T}^{(i)}$ of $\mathcal{T}$ in the obvious way:

$$\mathcal{T}^{(i)} = \{C^{(i)} \sqsubseteq D^{(i)} \mid C \sqsubseteq D \in \mathcal{T}\}.$$

To deal with occlusions, we introduce auxiliary role names $r^{(i)}_{\mathsf{Dom}(C)}$ and $r^{(i)}_{\mathsf{Ran}(D)}$ for $0 \leq i < n$ and all concepts C, D such that $(C, D) \in \mathsf{occ}_i(s)$ for some role literal s. The following TBox $\mathcal{T}^{(i)}_{\mathsf{aux}}$ ensures that $r^{(i)}_{\mathsf{Dom}(C)}$ and $r^{(i)}_{\mathsf{Ran}(D)}$ are interpreted as $C^{(i)} \times \top$ and $\top \times D^{(i)}$, respectively. It contains the following axioms, for all concepts C, D as above:

$$\begin{array}{rclcrcl} C^{(i)} & \sqsubseteq & \forall \neg r^{(i)}_{\mathsf{Dom}(C)}.\bot & \qquad & \top & \sqsubseteq & \forall r^{(i)}_{\mathsf{Ran}(D)}.D^{(i)} \\ \neg C^{(i)} & \sqsubseteq & \forall r^{(i)}_{\mathsf{Dom}(C)}.\bot & & \top & \sqsubseteq & \forall \neg r^{(i)}_{\mathsf{Ran}(D)}.\neg D^{(i)} \end{array}$$

The following TBox $\mathcal{T}^{(i)}_{\mathsf{fix}}$ ensures that concept and role names do not change unless this is allowed by the occlusion pattern:

- for every concept name A in the input,

$$\begin{array}{rcl} A^{(i)} \sqcap \neg A^{(i+1)} & \sqsubseteq & (\mathsf{occ}_{i+1}(A))^{(i)} \\ \neg A^{(i)} \sqcap A^{(i+1)} & \sqsubseteq & (\mathsf{occ}_{i+1}(\neg A))^{(i)} \end{array}$$

- for every role name r in the input,

$$\top \sqsubseteq \forall \neg \Big(\bigcup_{(C,D) \in \mathsf{occ}_{i+1}(r)} (r^{(i)}_{\mathsf{Dom}(C)} \cap r^{(i)}_{\mathsf{Ran}(D)}) \Big) \cap (r^{(i)} \cap \neg r^{(i+1)}).\bot$$

$$\top \sqsubseteq \forall \neg \Big(\bigcup_{(C,D) \in \mathsf{occ}_{i+1}(\neg r)} (r^{(i)}_{\mathsf{Dom}(C)} \cap r^{(i)}_{\mathsf{Ran}(D)}) \Big) \cap (\neg r^{(i)} \cap r^{(i+1)}).\bot$$

Finally, we can construct $\mathcal{T}_{\mathsf{red}}$ as

$$\mathcal{T}_{\mathsf{red}} := \bigcup_{0 \leq i \leq n} \mathcal{T}^{(i)} \cup \bigcup_{0 \leq i < n} \mathcal{T}^{(i)}_{\mathsf{aux}} \cup \bigcup_{0 \leq i < n} \mathcal{T}^{(i)}_{\mathsf{fix}}.$$

The following is shown in [5]:

Lemma 6. *$C_0(a_0)$ is a consequence of applying $\alpha_1, \ldots, \alpha_n$ in $\mathcal{A}_0$ w.r.t. $\mathcal{T}$ iff $\mathcal{A}_{\mathsf{red}} \cup \{\neg C_0^{(n)}(a_0)\}$ is inconsistent w.r.t. $\mathcal{T}_{\mathsf{red}}$.*

Since $\mathcal{ALCQIO}^{\neg,\cap,\cup}$ is a fragment of $\mathcal{C}^2$ (the 2-variable fragment of first-order logic with counting), ABox inconsistency in $\mathcal{ALCQIO}^{\neg,\cap,\cup}$ is in co-NExpTime even if numbers are coded in binary [7]. Since $\mathcal{A}_{\mathsf{red}}$ and $\mathcal{T}_{\mathsf{red}}$ are polynomial in the size of the input, Lemma 6 gives us a co-NExpTime upper bound for projection in $\mathcal{ALCQIO}$ and $\mathcal{ALCQI}$.

Theorem 7. *Projection and executability are co-*NExpTime*-complete, while weak consistency is* NExpTime*-complete in $\mathcal{ALCQIO}$ and $\mathcal{ALCQI}$.*

4 Undecidability of Strong Consistency

Here we show that strong consistency is undecidable. The proof consists of a reduction of the undecidable *semantic consequence problem* from modal logic. Before formulating the DL version of this problem, we need some preliminaries. We use concepts and interpretations with only one role name r, which we call $\mathcal{ALC}_r$-concepts. Accordingly, we also assume that interpretations interprete only concept names and the role name r. A *frame* is a structure $\mathcal{F} = (\Delta^{\mathcal{F}}, r^{\mathcal{F}})$ where $\Delta^{\mathcal{F}}$ is a non-empty set and $r^{\mathcal{F}} \subseteq \Delta^{\mathcal{F}} \times \Delta^{\mathcal{F}}$. An interpretation $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ is *based* on a frame $\mathcal{F}$ iff $\Delta^{\mathcal{I}} = \Delta^{\mathcal{F}}$ and $r^{\mathcal{I}} = r^{\mathcal{F}}$. We say that a concept C is *valid* on $\mathcal{F}$ (written $\mathcal{F} \models C$) iff $C^{\mathcal{I}} = \Delta^{\mathcal{I}}$ for every interpretation $\mathcal{I}$ based on $\mathcal{F}$.

Definition 8 (Semantic consequence problem). Let D and E be $\mathcal{ALC}_r$-concepts. We say that E is a *semantic consequence* of D iff for every frame $\mathcal{F} = (\Delta^{\mathcal{F}}, r^{\mathcal{F}})$ such that $\mathcal{F} \models D$, it holds that $\mathcal{F} \models E$.

In [12], it is proved that for $\mathcal{ALC}_r$-concepts D and E, the problem "Is E a semantic consequence of D?" is undecidable. We now show that the semantic consequence problem can be reduced to strong consistency. For $\mathcal{ALC}_r$-concepts D and E, we define the ABox $\mathcal{A}_E := \{\neg E(a)\}$ and the atomic action $\alpha_D = (\emptyset, \{\mathsf{occ}_{\mathsf{taut}}\}, \mathsf{post})$ with $\mathsf{post} := \{\top(a)/(\exists u.\neg D)(a)\}$ where u is an arbitrary role name and $\mathsf{occ}_{\mathsf{taut}}$ maps r and $\neg r$ to $\{(\bot, \bot)\}$, all other role literals to $\{(\top, \top)\}$, and all concept literals to $\top$. Then the action α_D is strongly consistent with the empty TBox and the ABox $\mathcal{A}_E$ iff E is a semantic consequence of D [5]. As an immediate consequence, we obtain the following theorem.

Theorem 9. *Strong consistency of $\mathcal{ALC}$-actions is undecidable, even with the empty TBox.*

5 Discussion

We have introduced an action formalism based on description logics that admits general TBoxes and complex post-conditions. To deal with ramifications induced by general TBoxes, the formalism includes powerful occlusion patterns that can be used to fine-tune the ramifications. We believe that undecidability of strong consistency is no serious obstacle for the feasibility of our approach in practice. Although deciding strong consistency would provide valuable support for the designer of an action, it could not replace manual inspection of the ramifications. For example, occluding all concept names with $\top$ and all role names with $\{(\top, \top)\}$ usually ensures strong consistency but does not lead to an intuitive behaviour of the action. With weak consistency, we offer at least some automatic support to the action designer for detecting ramification problems.

In [6], we investigate the complexity of reasoning with DL actions for other fragments of $\mathcal{ALCQIO}$, most notably $\mathcal{ALC}$, $\mathcal{ALCI}$, and $\mathcal{ALCIO}$. It turns out that, in these logics, deciding projection (as well as executability and weak consistency) is ExpTime-complete, which is in accordance with the observation that, usually, reasoning with $\mathcal{L}$-actions has the same complexity as subsumption in $\mathcal{L}$ extended with nominals. In [5], we also show how reasoning with a restricted version of $\mathcal{ALCQIO}$ actions can be reduced to reasoning in $\mathcal{ALCQIO}$ instead of $\mathcal{ALCQIO}^{\neg,\cap,\cup}$. This is relevant since reasoners for the former (but not for the latter) are readily available.

Acknowledgements. We would like to thank Giuseppe De Giacomo for ideas and discussions. The third author is supported by the DFG Graduiertenkolleg 334. The fourth author is partially supported by UK EPSRC grant no. GR/S63182/01.

References

[1] F. Baader, C. Lutz, M. Milicic, U. Sattler, and F. Wolter. Integrating description logics and action formalisms: First results. In *Proc. of (AAAI-05)*, Pittsburgh, PA, USA, 2005.

[2] F. Baader, D. L. McGuiness, D. Nardi, and P. Patel-Schneider. *The Description Logic Handbook: Theory, implementation and applications*. Cambridge University Press, 2003.

[3] H. J. Levesque, R. Reiter, Y. Lespérance, F. Lin, and R. B. Scherl. GOLOG: A logic programming language for dynamic domains. *J. Log. Program.*, 31(1-3):59–83, 1997.

[4] F. Lin. Embracing causality in specifying the indirect effects of actions. In *Proc. of IJCAI-95*, Montreal, Canada, 1995. Morgan Kaufmann.

[5] H. Liu, C. Lutz, M. Milicic, and F. Wolter. Description logic actions with general TBoxes: a pragmatic approach. LTCS-Report 06-03, TU Dresden, Germany, 2006. See http://lat.inf.tu-dresden.de/research/reports.html.

[6] H. Liu, C. Lutz, M. Milicic, and F. Wolter. Description logic actions with general TBoxes: a pragmatic approach. Submitted to JELIA'06, 2006.

[7] I. Pratt-Hartmann. Complexity of the two-variable fragment with counting quantifiers. *Journal of Logic, Language and Inf.*, 14(3):369–395, 2005.

[8] R. Reiter. *Knowledge in Action*. MIT Press, 2001.

[9] M. Thielscher. Ramification and causality. *Artificial Intelligence Journal*, 89(1–2):317–364, 1997.

[10] M. Thielscher. Introduction to the Fluent Calculus. *Electronic Transactions on Artificial Intelligence*, 2(3–4):179–192, 1998.

[11] M. Thielscher. FLUX: A logic programming method for reasoning agents. *TPLP*, 5(4-5):533–565, 2005.

[12] S. K. Thomason. The logical consequence relation of propositional tense logic. *Z. Math. Logik Grundl. Math.*, 21:29–40, 1975.

[13] S. Tobies. The complexity of reasoning with cardinality restrictions and nominals in expressive description logics. *Journal of Artificial Intelligence Research*, 12:199–217, 2000.

[14] M. Winslett. Reasoning about action using a possible models approach. In *AAAI*, pages 89–93, Saint Paul, MN, 1988.

Discrete Tableau Algorithms for $\mathcal{FSHI}$*

Yanhui Li[1], Baowen Xu[1,2], Jianjiang Lu[3] and Dazhou Kang[1]
[1] School of Computer Science and Engineering, Southeast University, Nanjing 210096, China
[2] Jiangsu Institute of Software Quality, Nanjing 210096, China
[3] Institute of Command Automation, PLA University of Science and Technology, Nanjing 210007, China
bwxu@seu.edu.cn

Abstract

A variety of fuzzy description logics are proposed to extend classical description logics with fuzzy capability. However, reasoning with general TBoxes is still an open problem in fuzzy description logics. In this paper, we present a novel discrete tableau algorithm for a given fuzzy description logic $\mathcal{FSHI}$ with general TBoxes, which tries to construct discrete tableaus of $\mathcal{FSHI}$ knowledge bases. We prove the equivalence of existence between discrete tableaus and models of $\mathcal{FSHI}$ knowledge bases, hence getting that the discrete tableau algorithm is a sound and complete decision procedure for $\mathcal{FSHI}$ reasoning problems with general TBoxes.

1 Introduction

Increasing demands for fuzzy knowledge representation have triggered a variety of fuzzy extensions of description logics (DLs) that make them convenient to express knowledge in fuzzy cases. Straccia adopted fuzzy interpretations to propose a representative fuzzy extension $\mathcal{FALC}$ of $\mathcal{ALC}$, and designed a tableau algorithm for acyclic TBoxes [9]. Based on $\mathcal{FALC}$, Höldobler et al proposed membership manipulator constructors to define new fuzzy concepts [2]. Fuzzy extensions of more expressive DLs like $\mathcal{ALCQ}$ and $\mathcal{SHOIN}$(D) were presented

*This work was supported in part by the NSFC (60373066, 60425206 and 90412003), National Grand Fundamental Research 973 Program of China (2002CB312000), Excellent Ph.D. Thesis Fund of Southeast University, Advanced Armament Research Project (51406020105JB8103), High Technology Research Project of Jiangsu Province (BG2005032) and Advanced Armament Research Project (51406020105JB8103).

in [6,11], but there were no reasoning algorithms for them. The concrete domains were also introduced into fuzzy DLs with an optimized reasoning technique in $\mathcal{FALC}$(D) [10]. Stoilos et al extended Straccia's fuzzy framework into OWL, hence getting a fuzzy ontology language: Fuzzy OWL [8]. They also gave a reasoning technique to deal with ABox consistency without TBoxes.

Though the fuzzy extension of DLs has done a lot, reasoning with general TBoxes is still an open problem in fuzzy DLs. In this paper, we will propose a novel discrete tableau algorithm for satisfiability of $\mathcal{FSHI}$ knowledge bases (KBs) with general TBoxes. The remaind of this paper is organized as follows. A brief introduction to $\mathcal{FSHI}$ KBs will be given in section 2. The main theoretical foundation of our discrete tableau algorithms is the discretization of fuzzy models, which will be discussed in section 3. Following that, we will present the definition of discrete tableaus and the expansion rules of discrete tableau algorithms, and propose a sketch proof of correctness and complexity of our algorithms in section 4. Finally section 5 will conclude this paper and discuss the further work.

2 A Brief Introduction to $\mathcal{FSHI}$

$\mathcal{FSHI}$ is a complex fuzzy description logic with role hierarchy, transitive and inverse role. Let $\mathrm{N_C}$ be a set of concept names and R be a set of role names with transitive role names $\mathrm{R}^+ \subseteq \mathrm{R}$. $\mathcal{FSHI}$ roles are either role names $R \in \mathrm{R}$ or their inverse role R^-. $\mathcal{FSHI}$ concepts are inductively defined as follows:

1. For any $A \in \mathrm{N_C}$, A is a concept;
2. The top concept $\top$ and the bottom concept $\bot$ are concepts;
3. If C and D are two concepts and R is a role, the $\neg C$, $C \sqcap D$, $C \sqcup D$, $\exists R.C$ and $\forall R.C$ are concepts.

One of the main differences between fuzzy DLs and classical DLs is that fuzzy DLs adopt fuzzy interpretations. A fuzzy interpretation $\mathcal{I} = \langle \Delta^{\mathcal{I}}, \cdot^{\mathcal{I}} \rangle$ of $\mathcal{FSHI}$ consists of a nonempty domain $\Delta^{\mathcal{I}}$ and an interpretation function $\cdot^{\mathcal{I}}$ mapping:

$$\begin{aligned} \text{any individual name } a \quad &\text{into} \quad a^{\mathcal{I}} \in \Delta^{\mathcal{I}} \\ \text{any concept name } A \quad &\text{into} \quad A^{\mathcal{I}} : \Delta^{\mathcal{I}} \to [0,1] \\ \text{any role name } R \quad &\text{into} \quad R^{\mathcal{I}} : \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}} \to [0,1] \end{aligned}$$

And for any transitive role name $R \in \mathrm{R}^+$, $\cdot^{\mathcal{I}}$ satisfies $\forall d, d' \in \Delta^{\mathcal{I}}$, $R^{\mathcal{I}}(d,d') \geq \sup_{x \in \Delta^{\mathcal{I}}}\{\min(R^{\mathcal{I}}(d,x), R^{\mathcal{I}}(x,d'))\}$. Intuitively, any concept name A is naturally interpreted as the membership degree function $A^{\mathcal{I}}$ w.r.t. $\Delta^{\mathcal{I}}$: for any element $d \in \Delta^{\mathcal{I}}$, $A^{\mathcal{I}}(d)$ shows the degree of d being an instance of the fuzzy concept A

$$
\begin{array}{rcl}
\top^{\mathcal{I}}(d) & = & 1 \\
\bot^{\mathcal{I}}(d) & = & 0 \\
(\neg C)^{\mathcal{I}}(d) & = & 1 - C^{\mathcal{I}}(d) \\
(C \sqcap D)^{\mathcal{I}}(d) & = & \min(C^{\mathcal{I}}(d), D^{\mathcal{I}}(d)) \\
(C \sqcup D)^{\mathcal{I}}(d) & = & \max(C^{\mathcal{I}}(d), D^{\mathcal{I}}(d)) \\
(\exists R.C)^{\mathcal{I}}(d) & = & \sup_{d' \in \Delta^{\mathcal{I}}}\{\min(R^{\mathcal{I}}(d,d'), C^{\mathcal{I}}(d'))\} \\
(\forall R.C)^{\mathcal{I}}(d) & = & \inf_{d' \in \Delta^{\mathcal{I}}}\{\max(1 - R^{\mathcal{I}}(d,d'), C^{\mathcal{I}}(d'))\} \\
(R^{-})^{\mathcal{I}}(d,d') & = & R^{\mathcal{I}}(d',d)
\end{array}
$$

Figure 1: The Semantics of $\mathcal{FSHI}$

under the interpretation $\mathcal{I}$. Similarly for role name R. For complex concepts and inverse roles, $\cdot^{\mathcal{I}}$ satisfies the following conditions (see figure 1):

A general TBox $\mathcal{T}$ is a finite set of fuzzy general concept inclusions $C \sqsubseteq D$, where C and D are $\mathcal{FSHI}$ concepts. An interpretation $\mathcal{I}$ satisfies $C \sqsubseteq D$ iff $\forall d \in \Delta^{\mathcal{I}}, C^{\mathcal{I}}(d) \leq D^{\mathcal{I}}(d)$. $\mathcal{I}$ satisfies (is a fuzzy model of) a TBox $\mathcal{T}$ (written $\mathcal{I} \models \mathcal{T}$) iff $\mathcal{I}$ satisfies every inclusion in $\mathcal{T}$.

A RBox $\mathcal{R}$ is a finite set of fuzzy role inclusions $R \sqsubseteq P$, where R and P are $\mathcal{FSHI}$ roles. An interpretation $\mathcal{I}$ satisfies $R \sqsubseteq P$ iff $\forall d,\ d' \in \Delta^{\mathcal{I}}, R^{\mathcal{I}}(d,d') \leq P^{\mathcal{I}}(d,d')$. $\mathcal{I}$ satisfies (is a fuzzy model of) a RBox $\mathcal{R}$ (written $\mathcal{I} \models \mathcal{R}$) iff $\mathcal{I}$ satisfies every inclusion in $\mathcal{R}$. Here we introduce $\sqsubseteq^*$ as the transitive-reflexive closure of $\sqsubseteq$ on $\mathcal{R} \cup \{R^- \sqsubseteq P^- | R \sqsubseteq P \in \mathcal{R}\}$.

An ABox $\mathcal{A}$ is a finite set of fuzzy assertions $\alpha \bowtie n$, where α is an assertion: $a : C$ or $\langle a, b\rangle : R$, $\bowtie \in \{\geq, >, \leq, <\}$ and $n \in [0, 1]$. $\mathcal{I}$ satisfies a fuzzy assertion $a{:}C \geq n$ iff $C^{\mathcal{I}}(a^{\mathcal{I}}) \geq n$. Similarly for other three cases: $<$, $\leq$ and $>$, and role assertions $\langle a, b\rangle : R \bowtie n$. $\mathcal{I}$ satisfies (is a fuzzy model of) an ABox $\mathcal{A}$ (written $\mathcal{I} \models \mathcal{A}$) iff $\mathcal{I}$ satisfies every fuzzy assertion in $\mathcal{A}$.

A $\mathcal{FSHI}$ KB $\mathcal{K} = \langle \mathcal{T}, \mathcal{R}, \mathcal{A}\rangle$ consists of its TBox $\mathcal{T}$, RBox $\mathcal{R}$ and ABox $\mathcal{A}$. An interpretation $\mathcal{I}$ is a fuzzy model of a KB $\mathcal{K}$ (written $\mathcal{I} \models \mathcal{K}$), iff $\mathcal{I}$ satisfies its TBox, RBox and ABox. $\mathcal{K}$ is satisfiable iff there is a fuzzy model of $\mathcal{K}$. In this paper, we will present a discrete tableau algorithm to decide satisfiability of $\mathcal{FSHI}$ KBs.

3 Discretization of Fuzzy Models

Before discussing discretization of fuzzy models, we analyze troubles of reasoning with general TBoxes in fuzzy DLs: the key question is "why reasoning technique for general TBoxes in classical DLs can not be applied in fuzzy DLs". To answer this question, we will compare the semantics of fuzzy DLs with classical DLs. In classical DL cases, an individual (a pair of individuals) completely belongs to a concept (a role) or not, that means in classical models any concept (role) will be

interpreted as two-value degree functions: $\Delta^{\mathcal{I}}(\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}) \rightarrow \{0, 1\}$. For any general concept inclusion $C \sqsubseteq D$ and any individual a, classical tableau algorithms nondeterministically "guess" the membership degrees n and m of a being an instance of C and D, for some $n, m \in \{0, 1\}$ and $n \leq m$ [1]. However, such "guess" technique cannot be directly applied in fuzzy DLs. The main difficulty is that in fuzzy models concepts and roles are interpreted as complex membership degree functions by extending $\{0, 1\}$ to $[0, 1]$, hence the value of membership degree functions is continuous but not discrete. To solve this problem, we try to design a discretization step to translate fuzzy model into corresponding discrete model, in which any membership degree value belongs to a special discrete degree set. This discretization can enable similar "guess" technique applied in fuzzy DLs.

The first issue in discretization of fuzzy models is to decide the special discrete degree set S. Let us now proceed formally in the creation of S. Consider a $\mathcal{FSHI}$ KB $\mathcal{K} = \langle \mathcal{T}, \mathcal{R}, \mathcal{A} \rangle$. Let $N_{\mathcal{K}}$ be the set of degrees appearing in $\mathcal{A}$: $N_{\mathcal{K}} = \{n | \, \alpha \bowtie n \in \mathcal{A}\}$. From $N_{\mathcal{K}}$, we define the degree closure $DS_{\mathcal{K}}$ of $\mathcal{K}$ as: $DS_{\mathcal{K}} = \{0, 0.5, 1\} \cup N_{\mathcal{K}} \cup \{1 - n | \, n \in N_{\mathcal{K}}\}$ and sort $DS_{\mathcal{K}}$ in ascending order: $DS_{\mathcal{K}} = \{n_0, n_1, \ldots, n_s\}$, where for any $0 \leq i < s$, $n_i < n_{i+1}$. It is easy to prove that $n_0 = 0$ and $n_s = 1$; s is even; and for any $0 \leq i \leq s$, $n_i + n_{n-i} = 1$.

Consider a constant vector $M = [\, c_1, c_2, \ldots, c_{s/2}]$, where for any c_i, $0 < c_i < 1$. We define the $\bigotimes$ operation: $NS_{\mathcal{K}} = DS_{\mathcal{K}} \bigotimes M = \{m_1, m_2, \ldots, m_s\}$, where if $i \leq s/2$, $m_i = c_i \times n_{i-1} + (1 - c_i) \times n_i$, otherwise $m_i = (1 - c_{s+1-i}) \times n_{i-1} + c_{s+1-i} \times n_i$. Obviously for any $1 \leq i \leq s$, $m_i + m_{s+1-i} = 1$ and $n_{i-1} < m_i < n_i$.

Let $S = DS_{\mathcal{K}} \cup NS_{\mathcal{K}}$, we call S a discrete degree set w.r.t $\mathcal{K}$. Note that $|S| = 2s + 1 = \mathrm{O}(|N_{\mathcal{K}}|) = \mathrm{O}(|\mathcal{A}|)$. We also sort degrees of S in ascending order: $S = \{n_0, m_1, n_1, \ldots, n_{s-1}, m_s, n_s\}$. For a fuzzy model $\mathcal{I}_c$ of $\mathcal{K}$, if every membership degree value of $C^{\mathcal{I}_c}(\ )$ or $R^{\mathcal{I}_c}(\ )$ belongs to a discrete degree set S, $\mathcal{I}_c$ is called a discrete model of $\mathcal{K}$ within S. Following theorem guarantees the equivalence between existence of fuzzy models and discrete models of $\mathcal{K}$.

Theorem 1 *For any $\mathcal{K} = \langle \mathcal{T}, \mathcal{R}, \mathcal{A} \rangle$ and any discrete degree set S w.r.t $\mathcal{K}$, $\mathcal{K}$ has a fuzzy model, iff it has a discrete model within S.*

Proof.⇒) Let $\mathcal{I} = \langle \Delta^{\mathcal{I}}, \cdot^{\mathcal{I}} \rangle$ be a fuzzy model of $\mathcal{K}$ and the degree set $S = \{n_0, m_1, n_1, \ldots, n_{s-1}, m_s, n_s\}$. Consider a translation function $\varphi(\ ) : [\, 0, 1] \rightarrow S$:

$$\varphi(x) = \begin{cases} n_i & \text{if} \quad x = n_i \\ m_i & \text{if} \quad n_{i-1} < x < n_i \end{cases}$$

Here we enumerate some properties of $\varphi(\ )$, which are useful for the following proof: for any $x \leq y$, $\varphi(x) \leq \varphi(y)$; for any $x < y$, if x or $y \in DS_{\mathcal{K}}$, $\varphi(x) < \varphi(y)$; and for any x and y, $\varphi(1 - x) = 1 - \varphi(x)$, $\varphi(\max(x, y)) = \max(\varphi(x), \varphi(y))$, and $\varphi(\min(x, y)) = \min(\varphi(x), \varphi(y))$.

Based on $\varphi(\ )$, we construct a discrete model $\mathcal{I}_c = \langle \Delta^{\mathcal{I}_c}, \cdot^{\mathcal{I}_c} \rangle$ within S from $\mathcal{I} = \langle \Delta^{\mathcal{I}}, \cdot^{\mathcal{I}} \rangle$:

- The interpretation domain $\Delta^{\mathcal{I}_c}$ is defined as: $\Delta^{\mathcal{I}_c} = \Delta^{\mathcal{I}}$;
- The interpretation function $\cdot^{\mathcal{I}_c}$ is defined as: for any individual name a, $a^{\mathcal{I}_c} = a^{\mathcal{I}}$; for any concept name A and any role name R: $A^{\mathcal{I}_c}(\ ) = \varphi(A^{\mathcal{I}}(\ ))$ and $R^{\mathcal{I}_c}(\ ) = \varphi(R^{\mathcal{I}}(\ ))$; and for complex concept C and inverse role R^-, their interpretation are recursively defined based on membership degree functions $A^{\mathcal{I}_c}(\ )$ and $R^{\mathcal{I}_c}(\ )$ of concept names and role names.

1. For any concept C and role R and any two elements $d, d' \in \Delta^{\mathcal{I}_c}$, we show, by induction on the structure of C and R, that $C^{\mathcal{I}_c}(d) = \varphi(C^{\mathcal{I}}(d))$ and $R^{\mathcal{I}_c}(d, d') = \varphi(R^{\mathcal{I}}(d, d'))$:

   - Case A: from the construction of $\mathcal{I}_c$, $A^{\mathcal{I}_c}(d) = \varphi(A^{\mathcal{I}}(d))$;
   - Case R: the proof is similar to case A;
   - Case R^-: from the semantics of R^- in $\mathcal{I}$ and $\mathcal{I}_c$,
   $$(R^-)^{\mathcal{I}_c}(d, d') = R^{\mathcal{I}_c}(d', d) = \varphi(R^{\mathcal{I}}(d', d)) = \varphi((R^-)^{\mathcal{I}}(d, d'))$$
   - Case $\top$: for $1 \in DS_{\mathcal{K}}$, $\top^{\mathcal{I}_c}(d) = 1 = \varphi(\top^{\mathcal{I}}(d))$;
   - Case $\bot$: the proof is similar to case $\top$;
   - Case $\neg C$: from induction, $C^{\mathcal{I}_c}(d) = \varphi(C^{\mathcal{I}}(d))$. And from $\varphi(1 - x) = 1 - \varphi(x)$, we have
   $$\begin{aligned}(\neg C)^{\mathcal{I}_c}(d) &= 1 - C^{\mathcal{I}_c}(d) = 1 - \varphi(C^{\mathcal{I}}(d)) \\ &= \varphi(1 - C^{\mathcal{I}}(d)) = \varphi((\neg C)^{\mathcal{I}}(d))\end{aligned}$$
   - Case $C \sqcap D$: from induction, $C^{\mathcal{I}_c}(d) = \varphi(C^{\mathcal{I}}(d))$ and $D^{\mathcal{I}_c}(d) = \varphi(D^{\mathcal{I}}(d))$. We can get that
   $$\begin{aligned}(C \sqcap D)^{\mathcal{I}_c}(d) &= \min(C^{\mathcal{I}_c}(d), D^{\mathcal{I}_c}(d)) \\ &= \min(\varphi(C^{\mathcal{I}}(d)), \varphi(D^{\mathcal{I}}(d))) \\ &= \varphi(\min(C^{\mathcal{I}}(d), D^{\mathcal{I}}(d))) \\ &= \varphi((C \sqcap D)^{\mathcal{I}}(d))\end{aligned}$$
   - Case $C \sqcup D$: the proof is similar to case $C \sqcap D$.
   - Case $\forall R.C$: let $f(d') = \max(1 - R^{\mathcal{I}}(d, d'), C^{\mathcal{I}}(d'))$. From definition, $(\forall R.C)^{\mathcal{I}}(d) = \inf_{d' \in \Delta^{\mathcal{I}}} f(d')$. Assume there is an element d'' with the minimal value of $f(\ )$: for any d' in $\Delta^{\mathcal{I}}$, $f(d'') \leq f(d')$[1]. Let
   $$\begin{aligned}f^*(d') &= \varphi(f(d')) \\ &= \varphi(\max(1 - R^{\mathcal{I}}(d, d'), C^{\mathcal{I}}(d'))) \\ &= \max(\varphi(1 - R^{\mathcal{I}}(d, d')), \varphi(C^{\mathcal{I}}(d'))) \\ &= \max(1 - \varphi(R^{\mathcal{I}}(d, d')), \varphi(C^{\mathcal{I}}(d'))) \\ &= \max(1 - R^{\mathcal{I}_c}(d, d'), C^{\mathcal{I}_c}(d'))\end{aligned}$$

[1]The detailed proof of this assumption is given in [5]

Obviously, for $\forall d'$ in $\Delta^{\mathcal{I}_c}$, $f^*(d'') = \varphi(f(d'')) \leq \varphi(f(d')) = f^*(d')$. Then we get

$$\begin{aligned}(\forall R.C)^{\mathcal{I}_c}(d) &= \inf_{d' \in \Delta^{\mathcal{I}_c}} f^*(d') = f^*(d'') \\ &= \varphi(f(d'')) = \varphi((\forall R.C)^{\mathcal{I}}(d))\end{aligned}$$

- Case $\exists R.C$: for $\neg(\exists R.C) = \forall R.\neg C$, we can get the proof from case $\neg C$ and $\forall R.C$.

2. We show $\mathcal{I}_c$ is a fuzzy model of $\mathcal{K}$.
   - Case $R \in \mathrm{R}^+$: for $\mathcal{I}$ is a fuzzy model of $\mathcal{K}$, $\forall d, d' \in \Delta^{\mathcal{I}}$, $R^{\mathcal{I}}(d, d') \geq \sup_{x \in \Delta^{\mathcal{I}}}\{R^{\mathcal{I}}(d, x), R^{\mathcal{I}}(x, d')\}$. Therefore,

$$\begin{aligned}R^{\mathcal{I}_c}(d, d') &= \varphi(R^{\mathcal{I}}(d, d')) \\ &\geq \sup_{x \in \Delta^{\mathcal{I}}}\{\min(\varphi(R^{\mathcal{I}}(d, x)), \varphi(R^{\mathcal{I}}(x, d')))\} \\ &= \sup_{x \in \Delta^{\mathcal{I}_c}}\{\min(R^{\mathcal{I}_c}(d, x), R^{\mathcal{I}_c}(x, d'))\}\end{aligned}$$

   - Case $C \sqsubseteq D \in \mathcal{T}$: for $\mathcal{I}$ is a fuzzy model of $\mathcal{K}$, $\forall d \in \Delta^{\mathcal{I}}$, $C^{\mathcal{I}}(d) \leq D^{\mathcal{I}}(d)$. And from 1, for any concept C, $C^{\mathcal{I}_c}(d) = \varphi(C^{\mathcal{I}}(d))$. Therefore, $\forall d \in \Delta^{\mathcal{I}_c}$, $C^{\mathcal{I}_c}(d) = \varphi(C^{\mathcal{I}}(d)) \leq \varphi(D^{\mathcal{I}}(d)) = D^{\mathcal{I}_c}(d)$;
   - Case $R \sqsubseteq S \in \mathcal{R}$: the proof is similar to case $C \sqsubseteq D$;
   - Case $\alpha \bowtie n \in \mathcal{A}$: here we only focus on $a{:}C \geq n$. For $C^{\mathcal{I}}(a^{\mathcal{I}}) \geq n$ and $n \in DS_{\mathcal{K}}$, we can get:

$$C^{\mathcal{I}_c}(a^{\mathcal{I}_c}) = \varphi(C^{\mathcal{I}}(a^{\mathcal{I}})) \geq \varphi(n) = n$$

From above two points, $\mathcal{I}_c$ is a discrete model of $\mathcal{K}$ within S.

$\Leftarrow$) Let $\mathcal{I}_c$ be a discrete model of $\mathcal{K}$ within S. It is also a fuzzy model of $\mathcal{K}$.□

4 Discrete Tableau Algorithms

This section will talk about discrete tableau algorithms, which try to decide the existence of discrete models of a $\mathcal{FSHI}$ KB $\mathcal{K}$ by constructing a discrete tableau. Before going into the definition of discrete tableaus, we first introduce some notations. It will be assumed that the concepts appearing in tableau algorithms are written in Negation Normal Form (NNF). And for any concept C, we use $\mathrm{nnf}(C)$ to denote its equivalent form in NNF. The set of subconcepts of a concept C is denoted as $\mathrm{sub}(C)$. For a KB $\mathcal{K}$, we define $\mathrm{sub}(\mathcal{K})$ as the union of all $\mathrm{sub}(C)$, when the concept C appears in $\mathcal{K}$. We also make two notions about roles to make the following consideration easier: we use $\mathrm{Inv}(R)$ to denote the inverse role of R and $\mathrm{Tran}(R)$ as a Boolean value to tell whether R is transitive. $\mathrm{Trans}(R)$ =True, iff R or $\mathrm{Inv}(R) \in \mathrm{R}^+$ or there is a role P with (1) $P \sqsubseteq^* R$ and $R \sqsubseteq^* P$; and (2) P or $\mathrm{Inv}(P) \in \mathrm{R}^+$. Moreover, we use the symbols $\rhd$ and $\lhd$ as two placeholders for the inequalities $\geq$, $>$ and $\leq$, $<$, and the symbols $\bowtie^-$, $\rhd^-$

Table 1: Conjugated pairs

	$\langle <, m\rangle$	$\langle \leq, m\rangle$
$\langle \geq, n\rangle$	$n \geq m$	$n > m$
$\langle >, n\rangle$	$\neg\exists n_1 \in S$ with $n < n_1 < m$	$n \geq m$

and $\lhd^-$ to denote the reflections of $\bowtie$, $\rhd$ and $\lhd$. For example, $\geq$ and $\leq$ are the reflections to each other. Finally, we define $\langle \bowtie, n\rangle$ as a degree pair. Two degree pairs $\langle \rhd, n\rangle$ and $\langle \lhd, m\rangle$ are called conjugated, iff they satisfy one of following conditions (see table 1).

Now we define the discrete tableau for $\mathcal{K}$. Let $\mathrm{R}_\mathcal{K}$ and $\mathrm{O}_\mathcal{K}$ be the sets of roles and individuals appearing in $\mathcal{K}$. A discrete tableau T for $\mathcal{K}$ within a degree set S is a quadruple: $\langle \mathcal{O}, \mathcal{L}, \mathcal{E}, \mathcal{V}\rangle$, where

- $\mathcal{O}$: a nonempty set of nodes;
- $\mathcal{L}$: $\mathcal{O} \to 2^M$, $M = \mathrm{sub}(\mathcal{K}) \times \{\geq, >, \leq, <\} \times S$;
- $\mathcal{E}$: $\mathrm{R}_\mathcal{K} \to 2^Q$, $Q = \{\mathcal{O} \times \mathcal{O}\} \times \{\geq, >, \leq, <\} \times S$;
- $\mathcal{V}$: $\mathrm{O}_\mathcal{K} \to \mathcal{O}$, maps any individual into a corresponding node in $\mathcal{O}$.

The discrete tableau has a forest-like structure, which is a collection of trees that correspond to individuals in the ABox $\mathcal{A}$. Every tree consists of nodes standing for the individuals, and edges representing the relations between two nodes (individuals). Each node d is labelled with a set $\mathcal{L}(d)$ of degree triples: $\langle C, \bowtie, n\rangle$, which denotes the membership degree of d being an instance of $C \bowtie n$. A pair of triple $\langle C, \bowtie, n\rangle$ and $\langle C, \bowtie', m\rangle$ are conjugated if $\langle \bowtie, n\rangle$ and $\langle \bowtie', m\rangle$ are conjugated. In a discrete tableau T, for any $d, d' \in \mathcal{O}$, $a, b \in \mathrm{O}_\mathcal{K}$, $C, D \in \mathrm{sub}(\mathcal{K})$ and $R \in \mathrm{R}_\mathcal{K}$, the following conditions must hold:

1. There are not two conjugated degree triples in $\mathcal{L}(d)$;
2. There are not inconsistent triples: $\langle \bot, \geq, n\rangle$ $(n > 0)$, $\langle \top, \leq, n\rangle$ $(n < 1)$, $\langle \bot, >, n\rangle$, $\langle \top, <, n\rangle$, $\langle C, >, 1\rangle$ and $\langle C, <, 0\rangle$ in $\mathcal{L}(d)$;
3. If $C \sqsubseteq D \in \mathcal{T}$, then there must be some $n \in S$ with $\langle C, \leq, n\rangle$ and $\langle D, \geq, n\rangle$ in $\mathcal{L}(d)$;
4. If $\langle C, \bowtie, n\rangle \in \mathcal{L}(d)$, then $\langle \mathrm{nnf}(\neg C), \bowtie^-, 1 - n\rangle \in \mathcal{L}(d)$;
5. If $\langle C \sqcap D, \rhd, n\rangle \in \mathcal{L}(d)$, then $\langle C, \rhd, n\rangle$ and $\langle D, \rhd, n\rangle \in \mathcal{L}(d)$;
6. If $\langle C \sqcap D, \lhd, n\rangle \in \mathcal{L}(d)$, then $\langle C, \lhd, n\rangle$ or $\langle D, \lhd, n\rangle \in \mathcal{L}(d)$;
7. If $\langle C \sqcup D, \rhd, n\rangle \in \mathcal{L}(d)$, then $\langle C, \rhd, n\rangle$ or $\langle D, \rhd, n\rangle \in \mathcal{L}(d)$;
8. If $\langle C \sqcup D, \lhd, n\rangle \in \mathcal{L}(d)$, then $\langle C, \lhd, n\rangle$ and $\langle D, \lhd, n\rangle \in \mathcal{L}(d)$;
9. If $\langle \forall R.C, \rhd, n\rangle \in \mathcal{L}(d)$,$\langle \langle d, d'\rangle, \rhd', m\rangle \in \mathcal{E}(R)$, and $\langle \rhd', m\rangle$ is conjugated with $\langle \rhd^-, 1 - n\rangle$, then $\langle C, \rhd, n\rangle \in \mathcal{L}(d')$;
10. If $\langle \forall R.C, \lhd, n\rangle \in \mathcal{L}(d)$, then there must be a node $d' \in \mathcal{O}$ with $\langle \langle d, d'\rangle, \lhd^-, 1-n\rangle \in \mathcal{E}(R)$ and $\langle C, \lhd, n\rangle \in \mathcal{L}(d')$;
11. If $\langle \exists R.C, \rhd, n\rangle \in \mathcal{L}(d)$, then there must be a node $d' \in \mathcal{O}$ with $\langle \langle d, d'\rangle, \rhd, n\rangle \in \mathcal{E}(R)$ and $\langle C, \rhd, n\rangle \in \mathcal{L}(d')$;

12. If $\langle \exists R.C, \lhd, n\rangle \in \mathcal{L}(d)$, $\langle\langle d, d'\rangle, \rhd', m\rangle \in \mathcal{E}(R)$, and $\langle \rhd', m\rangle$ is conjugated with $\langle \lhd, n\rangle$, then $\langle C, \lhd, n\rangle \in \mathcal{L}(d')$;
13. If $\langle \forall P.C, \rhd, n\rangle \in \mathcal{L}(d)$, $\langle\langle d, d'\rangle, \rhd', m\rangle \in \mathcal{E}(R)$ for some $R \sqsubseteq^* P$ with Trans(R) =True and $\langle \rhd', m\rangle$ is conjugated with $\langle \rhd^-, 1-n\rangle$, then $\langle \forall R.C, \rhd, n\rangle \in \mathcal{L}(d')$;
14. If $\langle \exists P.C, \lhd, n\rangle \in \mathcal{L}(d)$, $\langle\langle d, d'\rangle, \rhd', m\rangle \in \mathcal{E}(R)$ for some $R \sqsubseteq^* P$ with Trans(R) =True and $\langle \rhd', m\rangle$ is conjugated with $\langle \lhd, n\rangle$, then $\langle \exists R.C, \lhd, n\rangle \in \mathcal{L}(d')$;
15. If $\langle\langle d, d'\rangle, \bowtie, n\rangle \in \mathcal{E}(R)$, then $\langle\langle d', d\rangle, \bowtie, n\rangle \in \mathcal{E}(\text{Inv}(R))$;
16. If $\langle\langle d, d'\rangle, \rhd, n\rangle \in \mathcal{E}(R)$ and $R \sqsubseteq^* P$, then $\langle\langle d, d'\rangle, \rhd, n\rangle \in \mathcal{E}(P)$;
17. If $a : C \bowtie n \in \mathcal{A}$, then $\langle C, \bowtie, n\rangle \in \mathcal{L}(\mathcal{V}(a))$;
18. If $\langle a, b\rangle : R \bowtie n \in \mathcal{A}$, then $\langle\langle \mathcal{V}(a), \mathcal{V}(b)\rangle, \bowtie, n\rangle \in \mathcal{E}(R)$.

Discrete tableau is an extension of fuzzy tableau [7] with additional conditions (condition 3) to deal with general TBoxes. For any $C \sqsubseteq D \in \mathcal{T}$, since any membership degree in the discrete model belongs to S, for any individuals d, let $d : C = n_1$ and $d : D = n_2$, where $n_1, n_2 \in S$ and $n_1 \leq n_2$. Obviously, there must be some $n \in S$ satisfying $n_1 \leq n \leq n_2$. Then we add $\langle C, \leq, n\rangle$ and $\langle D, \geq, n\rangle$ in $\mathcal{L}(d)$. For other conditions, conditions 1 and 2 prevent tableau from containing any clash; condition 4-16 are necessary for the completeness of discrete tableaus; and condition 17-18 ensure the correctness of individual mapping function $\mathcal{V}(\ )$.

Theorem 2 *For any $\mathcal{K} =< \mathcal{T}, \mathcal{R}, \mathcal{A} >$ and any discrete degree set S w.r.t $\mathcal{K}$, $\mathcal{K}$ has a discrete model within S iff it has a discrete tableau* T *within S.*

Proof. $\Leftarrow$) Let $S = \{n_0, m_1, n_1, \ldots, n_{s-1}, m_s, n_s\}$ and T = $\langle \mathcal{O}, \mathcal{L}, \mathcal{E}, \mathcal{V}\rangle$ be a discrete tableau within S. We define a sign function $h(\ )$: $S \to \{1, 2, \ldots, 2s+1\}$. For any $0 \leq i \leq s$, $h(n_i) = 2i+1$ and $h(m_i) = 2i$. Obviously, for any $x \in S$, x is the $h(x)$-th minimal element in S. And we define $g(\ )$ as the inverse function of $h(\ )$. Based on T, we construct a discrete model $\mathcal{I}_c = \langle \Delta^{\mathcal{I}_c}, \cdot^{\mathcal{I}_c}\rangle$ of $\mathcal{K}$ within S:

- The interpretation domain $\Delta^{\mathcal{I}_c}$ is defined as follows: $\Delta^{\mathcal{I}_c} = \mathcal{O}$;
- The interpretation function $\cdot^{\mathcal{I}_c}$ is defined as follows: for any individual a, $a^{\mathcal{I}_c} = \mathcal{V}(a)$; for any concept name A any $d \in \Delta^{\mathcal{I}_c}$:

$$A^{\mathcal{I}_c}(d) = \max\{\ \ 0, \max\{n|\langle A, \geq, n\rangle \in \mathcal{L}(d)\}, \\ h(g(\max\{n|\langle A, >, n\rangle \in \mathcal{L}(d)\}) + 1)\ \ \}$$

And for any role name R and $d, d' \in \Delta^{\mathcal{I}_c}$, let

$$R^*(d, d') = \max\{\ \ 0, \max\{n|\langle\langle d, d'\rangle, \geq, n\rangle \in \mathcal{E}(R)\}, \\ h(g(\max\{n|\langle\langle d, d'\rangle, >, n\rangle \in \mathcal{E}(R)\}) + 1)\ \ \}$$

For any $k \geq 0$, let

$$R^*_k(d, d') = \sup_{x_1, \ldots, x_k \in \Delta^{\mathcal{I}_c}}\{\min(R^*(d, x_1), R^*(x_1, x_2), \ldots, \\ R^*(x_{k-1}, x_k), R^*(x_k, d'))\}$$

$$R^{\mathcal{I}_c}(d, d') = \begin{cases} \sup_{k\geq 0}\{R^*_k(d, d')\} & \text{if Tran}(R) = \text{True} \\ R^*(d, d') & \text{otherwise} \end{cases}$$

From above, $A^{\mathcal{I}_c}(d)$ are defined as the minimal value to satisfy constraints in both $\geq$ and $>$ cases. Note that, in order to be greater than all values in $S^* = \{n|\langle A, >, n\rangle \in \mathcal{L}(d)\}$, $A^{\mathcal{I}_c}(d)$ must be greater than or equal to the subsequence value $h(g(\max S^*) + 1)$ of the maximal value of S^* in S. And similarly for $R^{\mathcal{I}_c}(d, d')$. And for any complex concept C and inverse role R^-, their interpretation are recursively defined based on membership degree functions $A^{\mathcal{I}_c}(\ )$ and $R^{\mathcal{I}_c}(\ )$ of concept names and role names.

1. We show, for any C and d with $\langle C, \bowtie, n\rangle \in \mathcal{L}(d)$, $C^{\mathcal{I}_c}(d) \bowtie n$.
   - Case A: from the definition of $A^{\mathcal{I}_c}(\ )$, for any $\langle A, \rhd, n\rangle \in \mathcal{L}(d)$, obviously $A^{\mathcal{I}_c}(d) \rhd n$. And for any $\langle A, \lhd, n\rangle \in \mathcal{L}(d)$, here we only focus on $\leq$ cases. Assume $A^{\mathcal{I}_c}(d) > n$, (1) if $A^{\mathcal{I}_c}(d)$ is $\max\{n|\langle A, \geq, n\rangle \in \mathcal{L}(d)\}$ or $h(g(\max\{n|\langle A, >, n\rangle \in \mathcal{L}(d)\}) + 1)$, then there must be two conjugated triples in $\mathcal{L}(d)$, which is contrary to condition 1 of the discrete tableaus; or (2) if $A^{\mathcal{I}_c}(d)$ is 0, then $n < 0$ holds which is contrary to assumption $n \in [0, 1]$.
   - Case complex concepts: the proof is similar to case A.
2. Similarly, for any R and d, d' with $\langle\langle d, d'\rangle, \bowtie, n\rangle \in \mathcal{E}(R)$, $R^{\mathcal{I}_c}(d, d') \bowtie n$.
3. We show $\mathcal{I}_c$ is a fuzzy model of $\mathcal{K}$.
   - Case $R \in \mathrm{R}^+$: from the construction of $\mathcal{I}_c$, for any $d, d' \in \Delta^{\mathcal{I}_c}$,
   $$R^{\mathcal{I}_c}(d, d') \geq \sup_{x \in \Delta^{\mathcal{I}_c}}\{R^{\mathcal{I}_c}(d, x), R^{\mathcal{I}_c}(x, d')\}$$
   - Case $C \sqsubseteq D \in \mathcal{T}$: from condition 3 of discrete tableaus, for any $d \in \Delta^{\mathcal{I}_c}$ there must be some $n \in S$ with $\langle C, \leq, n\rangle$ and $\langle D, \geq, n\rangle$ in $\mathcal{L}(d)$. And from 1, $C^{\mathcal{I}_c}(d) \leq n$ and $D^{\mathcal{I}_c}(d) \geq n$ hold. Therefore, $\mathcal{I}_c$ satisfies $C \sqsubseteq D$
   - Case $R \sqsubseteq S \in \mathcal{R}$: the proof is similar to case $C \sqsubseteq D$;
   - Case $\alpha \bowtie n \in \mathcal{A}$: here we only focus on $a : C \bowtie n$. From condition 17 of discrete tableau, $\langle C, \bowtie, n\rangle \in \mathcal{L}(\mathcal{V}(a))$. And from $a^{\mathcal{I}_c} = \mathcal{V}(a)$ and 1, we get $C^{\mathcal{I}_c}(a^{\mathcal{I}_c}) \bowtie n$.

$\Rightarrow$) Let $\mathcal{I}_c = \langle\Delta^{\mathcal{I}_c}, \cdot^{\mathcal{I}_c}\rangle$ be a discrete model within S. And we construct a discrete tableau $\mathrm{T} = \langle\mathcal{O}, \mathcal{L}, \mathcal{E}, \mathcal{V}\rangle$ from $\mathcal{I}_c$:

- $\mathcal{O}$: $\mathcal{O} = \Delta^{\mathcal{I}_c}$;
- $\mathcal{L}$: for any $d \in \mathcal{O}$, $\mathcal{L}(d) = \{\langle C, \bowtie, n\rangle | C^{\mathcal{I}_c}(d) \bowtie n, n \in S\}$;
- for any $R \in \mathrm{R}_\mathcal{K}$, $\mathcal{E}(R) = \{\langle\langle d, d'\rangle, \bowtie, n\rangle | R^{\mathcal{I}_c}(d, d') \bowtie n, n \in S\}$;
- $\mathcal{V}$: for any $a \in \mathrm{O}_\mathcal{K}$, $\mathcal{V}(a) = a^{\mathcal{I}_c}$.

From definition, $\langle C, \bowtie, n\rangle \in \mathcal{L}(d) \Leftrightarrow C^{\mathcal{I}_c}(d) \bowtie n$ and $\langle\langle d, d'\rangle, \bowtie, n\rangle \in \mathcal{E}(R) \Leftrightarrow R^{\mathcal{I}_c}(d, d') \bowtie n$. We show T is a discrete tableau of $\mathcal{K}$ within S: here we give the proof of that T satisfies condtion 3 and 4:

3. if $C \sqsubseteq D \in \mathcal{T}$, for any d, $C^{\mathcal{I}_c}(d) \leq D^{\mathcal{I}_c}(d)$. Let $C^{\mathcal{I}_c}(d) = n$ and obviously $n \in S$. From the construction of T, $\langle C, \leq, n\rangle$ and $\langle D, \geq, n\rangle \in \mathcal{L}(d)$.
4. if $\langle C, \bowtie, n\rangle \in \mathcal{L}(d)$, then $C^{\mathcal{I}_c}(d) \bowtie n$. And for the semantics of negation, $\mathrm{nnf}(\neg C)^{\mathcal{I}_c}(d) \bowtie^{-} 1 - n$, then $\langle \mathrm{nnf}(\neg C), \bowtie^{-}, 1 - n\rangle \in \mathcal{L}(d)$.

From above, T is a discrete tableau of $\mathcal{K}$ within S. □

From theorem 1 and 2, an algorithm that constructs a discrete tableau of $\mathcal{K}$ within S can be considered as a decision procedure for the satisfiability of $\mathcal{K}$. The discrete tableau algorithm works on a completion forest $\mathcal{F}_\mathcal{K}$, where each node x is labelled with $\mathcal{L}(x) \subseteq M = \mathrm{sub}(\mathcal{K}) \times \{\geq, >, \leq, <\} \times S$; and each edge $\langle x, y\rangle$ is labelled $\mathcal{L}(\langle x, y\rangle)$=$\{\langle R, \bowtie, n\rangle\}$, for some $R \in \mathrm{R}_\mathcal{K}$ and $n \in S$.

The tableau algorithm initializes $\mathcal{F}_\mathcal{K}$ to contain a root node x_a for each individual a in $\mathrm{O}_\mathcal{K}$ and labels x_a with $\mathcal{L}(x_a) = \{\langle C, \bowtie, n\rangle | a : C \bowtie n \in \mathcal{A}\}$. Moreover, for any pair $\langle x_a, x_b\rangle$, $\mathcal{L}(\langle x_a, x_b\rangle) = \{\langle R, \bowtie, n\rangle | \langle a, b\rangle : R \bowtie n \in \mathcal{A}\}$. The algorithm expands the forest $\mathcal{F}_\mathcal{K}$ either by extending $\mathcal{L}(x)$ for the current node x or by adding new leaf node y with expansion rules in table 2.

In table 2, we adopt an optimized way to reduce "$\lhd$ rules": for any "$\lhd$" triple $\langle C, \lhd, n\rangle \in \mathcal{L}(x)$, we use $\neg^{\bowtie}$ rules to add its equivalence $\langle \mathrm{nnf}(C), \lhd^{-}, 1 - n\rangle$ to $\mathcal{L}(x)$, and then deal it with $\rhd$ rules.

Edges and nodes are added when expanding triples $\langle \exists R.C, \rhd, n\rangle$, $\langle \geq pR, \rhd, n\rangle$ in $\mathcal{L}(x)$. A node y is called a R-successor of another node x and x is called a R-predecessor of y, if $\langle R, \bowtie, n\rangle \in \mathcal{L}(\langle x, y\rangle)$. Ancestor is the transitive closure of predecessor. And for any two connected nodes x and y, we define $D_R(x, y)$=$\{\langle \rhd, n\rangle | P \sqsubseteq^{*} R, \langle P, \rhd, n\rangle \in \mathcal{L}(\langle x, y\rangle)$ or $\langle \mathrm{Inv}(P), \rhd, n\rangle \in \mathcal{L}(\langle y, x\rangle)\} \bigcup$ $\{\langle \lhd, n\rangle | R \sqsubseteq^{*} P, \langle P, \lhd, n\rangle \in \mathcal{L}(\langle x, y\rangle)$ or $\langle \mathrm{Inv}(P), \lhd, n\rangle \in \mathcal{L}(\langle y, x\rangle)\}$. If $D_R(x, y)$ $\neq \emptyset$, y is called a R-neighbor of x. As inverse role is allowed in $\mathcal{FSHI}$, we make use of dynamic blocking technique [3] to ensure the termination and correctness of our tableau algorithm. A node x is directly blocked by its ancestor y iff x is not a root node and $\mathcal{L}(x) = \mathcal{L}(y)$. A node x is indirectly blocked if its predecessor is blocked. A node x is blocked iff it is either directly or indirectly blocked.

A completion forest $\mathcal{F}_\mathcal{K}$ is said to contain a clash, if for a node x in $\mathcal{F}_\mathcal{K}$, $\mathcal{L}(x)$ contains two conjugated triples or an inconsistent triple (see condition 2 of discrete tableaus). A completion forest $\mathcal{F}_\mathcal{K}$ is clash-free if it does not contain any clash, and it is complete if none of the expansion rules are applicable.

From dynamic blocking technique, the worst-case complexity of our tableau algorithm is 2NEXPTIME [4]. And the soundness and completeness of our tableau algorithm are guaranteed by the following theorem.

Table 2: Expansion rules of discrete Tableau

Rule name	Description
KB rule:	if $C \sqsubseteq D \in \mathcal{T}$ and there is no n with $\langle C, \leq, n\rangle$ and $\langle D, \geq, n\rangle$ in $\mathcal{L}(x)$; then $\mathcal{L}(x) \rightarrow \mathcal{L}(x) \cup \{\langle C, \leq, n\rangle, \langle D, \geq, n\rangle\}$ for some $n \in S$.
The following rules are applied to nodes x which is not indirectly blocked.	
$\neg^{\bowtie}$ rule:	if $\langle C, \bowtie, n\rangle \in \mathcal{L}(x)$ and $\langle \text{nnf}(\neg C), \bowtie^{-}, n\rangle \notin \mathcal{L}(x)$; then $\mathcal{L}(x) \rightarrow \mathcal{L}(x) \cup \{\langle \text{nnf}(\neg C), \bowtie^{-}, n\rangle\}$.
$\sqcap^{\rhd}$ rule:	if $\langle C \sqcap D, \rhd, n\rangle \in \mathcal{L}(x)$, and $\langle C, \rhd, n\rangle$ or $\langle D, \rhd, n\rangle \notin \mathcal{L}(x)$; then $\mathcal{L}(x) \rightarrow \mathcal{L}(x) \cup \{\langle C, \rhd, n\rangle, \langle D, \rhd, n\rangle\}$.
$\sqcup^{\rhd}$ rule:	if $\langle C \sqcup D, \rhd, n\rangle \in \mathcal{L}(x)$, and $\langle C, \rhd, n\rangle, \langle D, \rhd, n\rangle \notin \mathcal{L}(x)$ then $\mathcal{L}(x) \rightarrow \mathcal{L}(x) \cup \{T\}$, for some $T \in \{\langle C, \rhd, n\rangle, \langle D, \rhd, n\rangle\}$
$\forall^{\rhd}$ rule:	if $\langle \forall R.C, \rhd, n\rangle \in \mathcal{L}(x)$, there is a R-neighbor y of x with $\langle \rhd', m\rangle \in D_R(x, y)$, which is conjugated with $\langle \rhd^{-}, 1-n\rangle$, and $\langle C, \rhd, n\rangle \notin \mathcal{L}(y)$; then $\mathcal{L}(y) \rightarrow \mathcal{L}(y) \cup \{\langle C, \rhd, n\rangle\}$.
$\forall^{+\rhd}$ rule:	if $\langle \forall P.C, \rhd, n\rangle \in \mathcal{L}(x)$, there is a R-neighbor y of x with $R \sqsubseteq^{*} P$, Trans(R)=True and $\langle \rhd', m\rangle \in D_R(x, y)$, $\langle \rhd', m\rangle$ is conjugated with $\langle \rhd^{-}, 1-n\rangle$ $\mathcal{L}(y) \rightarrow \mathcal{L}(y) \cup \{\langle \forall R.C, \rhd, n\rangle\}$.
The following rules are applied to nodes x which is not blocked.	
$\exists^{\rhd}$ rule:	if $\langle \exists R.C, \rhd, n\rangle \in \mathcal{L}(x)$; there is not a R-neighbor y of x with $\langle \rhd, n\rangle \in D_R(x, y)$ and $\langle C, \rhd, n\rangle \in \mathcal{L}(y)$. then add a new node z with $\langle R, \rhd, n\rangle \in \mathcal{L}(\langle x, z\rangle)$ and $\langle C, \rhd, n\rangle \in \mathcal{L}(z)$.

Theorem 3 *For any $\mathcal{K} = \langle \mathcal{T}, \mathcal{R}, \mathcal{A}\rangle$ and any discrete degree set S w.r.t $\mathcal{K}$, $\mathcal{K}$ has a discrete tableau within S iff the tableau algorithm can construct a complete and clash-free completion forest.*

Proof.(Sketch) Here we only focus on $\Leftarrow$). The proof of $\Rightarrow$) is similar to the one given in [3]. Let $\mathcal{F}_{\mathcal{K}}$ a complete and clash-free completion forest. We construct a discrete tableau T = $\langle \mathcal{O}, \mathcal{L}, \mathcal{E}, \mathcal{V}\rangle$ from $\mathcal{F}_{\mathcal{K}}$:

- $\mathcal{O}$: $\mathcal{O} = \{x \,|\, x$ is a node in $\mathcal{F}_{\mathcal{K}}$, and it is not blocked$\}$;
- $\mathcal{L}$: for any $x \in \mathcal{O}$, $\mathcal{L}(x)$ = the labelling set $\mathcal{L}(x)$ of nodes x in $\mathcal{F}_{\mathcal{K}}$;
- $\mathcal{E}$: for any $R \in \mathrm{R}_{\mathcal{K}}$,

$$\mathcal{E}(R) = \{\langle\langle x, y\rangle, \bowtie, n\rangle \,|\, 1.\ y \text{ is } R\text{-neighbor of } x \text{ and } \langle \bowtie, n\rangle \in D_R(x, y); \textbf{or}$$
$$2.\ y \text{ blocks } z, \text{ and } \langle \bowtie, n\rangle \in D_R(x, z); \textbf{or}$$
$$3.\ x \text{ blocks } z, \text{ and } \langle \bowtie, n\rangle \in D_{\text{Inv}(R)}(y, z) \quad \}$$

- $\mathcal{V}$: for any $a \in \mathrm{O}_{\mathcal{K}}$, $\mathcal{V}(a)$ = the initialized node x_a of a in $\mathcal{F}_{\mathcal{K}}$.

We can follow the similar steps in theorem 2 to prove that T is a discrete tableau of $\mathcal{K}$ within S.

5 Conclusion

This paper presents the discretization technique to reduce fuzzy models of $\mathcal{FSHI}$ and proposes a discrete tableau algorithm to solve satisfiability of $\mathcal{FSHI}$ KBs with general TBoxes. This discretization technique supports a new way to achieve reasoning with general TBoxes in fuzzy DLs. We will try to extend this discretization technique in more complex fuzzy DLs and design corresponding tableau algorithms for reasoning with them. Moreover, we plan to apply it in complexity research of reasoning problems in fuzzy DLs.

References

[1] F. Baader and U. Sattler. An overview of tableau algorithms for description logics. *Studia Logica*, 69(1):5–40, 2001.

[2] S. Höldobler, H.P. Sötr, T.D. Khang, and N.H. Nga. The subsumption problem in the fuzzy description logic alcfh. In *Proceedings of the Tenth International Conference of Information Processing and Managment of Uncertainty in Knowledge-Based Systems*, pages 243–250, Perugia, Italy, 2004.

[3] I. Horrocks and U. Sattler. A description logic with transitive and inverse roles and role hierarchies. *Journal of Logic and Computation*, 9:385–410, 1999.

[4] I. Horrocks, U. Sattler, and S. Tobies. Practical reasoning for expressive description logics. In *Proceedings of of LPAR99*, 1999.

[5] Y.H. Li, B.W. Xu, J.J. Lu, and D.Z. Kang. Witnessed models of fuzzy description logcis. Paper in preparation.

[6] D. Sanchez and G. Tettamanzi. Generalizing quantification in fuzzy description logic. In *Proceedings of the 8th Fuzzy Days*, Dortmund, Germany, 2004.

[7] G. Stoilos, G. Stamou, V. Tzouvaras, J.Z. Pan, and I. Horrock. A Fuzzy Description Logic for Multimedia Knowledge Representation. In *Proc. of the International Workshop on Multimedia and the Semantic Web*, 2005.

[8] G. Stoilos, G. Stamou, V. Tzouvaras, J.Z. Pan, and I. Horrocks. Fuzzy owl: Uncertainty and the semantic web. In *Proceedings of International Workshop of OWL: Experiences and Directions*, Galway, 2005.

[9] U. Straccia. Reasoning within fuzzy description logics. *Journal of Artificial Intelligence Researc*, 14:137–166, 2001.

[10] U. Straccia. Fuzzy alc with fuzzy concrete domains. In *Proceeedings of the International Workshop on Description Logics (DL-05)*, pages 96–103, Edinburgh, Scotland, 2005. CEUR.

[11] U. Straccia. Towards a fuzzy description logic for the semantic web. In *Proceedings of the 2nd European Semantic Web Conference*, Heraklion, Greece, 2005.

Epistemic First-Order Queries over Description Logic Knowledge Bases*

Diego Calvanese[1], Giuseppe De Giacomo[2], Domenico Lembo[2], Maurizio Lenzerini[2], Riccardo Rosati[2]

[1] Faculty of Computer Science
Free University of Bozen-Bolzano
Piazza Domenicani 3
39100 Bolzano, Italy
calvanese@inf.unibz.it

[2] Dipartimento di Informatica e Sistemistica
Università di Roma "La Sapienza"
Via Salaria 113
00198 Roma, Italy
lastname@dis.uniroma1.it

Abstract

Querying Description Logic knowledge bases has received great attention in the last years. The need of coping with incomplete information is the distinguishing feature with respect to querying databases. Due to this feature, we have to deal with two conflicting needs: on the one hand, we would like to query the knowledge base with sophisticated mechanisms provided by full first-order logic as in databases; on the other hand, the presence of incomplete information makes query answering a much more difficult task than in databases. In this paper we advocate the use of an epistemic first-order query language, which is able to incorporate closed-world reasoning on demand, as a means for expressing sophisticated queries over Description Logic knowledge bases. We show that through a subset of this language, called *EQL-Lite*, we are able to formulate full first-order queries over Description Logic knowledge bases, while keeping computational complexity of query answering under control. In particular, we show that *EQL-Lite* queries over *DL-Lite* knowledge bases are first-order reducible (i.e., can be compiled into SQL) and hence can be answered in LogSpace through standard database technologies.

1 Introduction

Querying Description Logic (DL) knowledge bases has received great attention in the last years. Indeed, the definition of suitable query languages, and the design of query answering algorithms is arguably one of the crucial issues in applying DLs to ontology management and to the Semantic Web [9].

Answering queries in DLs must take into account the open-world semantics of such logics, and is therefore much more difficult than in Databases. For example,

*This research has been partially supported by FET project TONES (Thinking ONtologiES), funded by the EU under contract number FP6-7603, by project HYPER, funded by IBM through a Shared University Research (SUR) Award grant, and by MIUR FIRB 2005 project "Tecnologie Orientate alla Conoscenza per Aggregazioni di Imprese in Internet" (TOCAI.IT).

while First Order Logic (FOL) is the basis of any query language (e.g., relational algebra and SQL) for relational databases [1], it is well-known that answering FOL queries posed to DL knowledge bases is undecidable[1]. More precisely, to the best of our knowledge, the most expressive class of queries that go beyond instance checking, and for which decidability of query answering has been proved in DLs, is the class of union of conjunctive queries [5, 14, 15]. This restriction on the query language may constitute a serious limitation to the adoption of DLs technology in information management tasks, such as those required in Semantic Web applications.

The open-world semantics of DLs on one hand is essential for representing incomplete information, but on the other hand may complicate the task of interpreting the answers by the users, or may call for the need of reasoning about the incompleteness of the knowledge base. For example, knowing that there are no parents with only female children, one might become interested in asking for all parents whose known chidren are all female. Note that querying mechanisms such as the one mentioned in the example go beyond FOL.

To summarize, due to the need of coping with incomplete information in DL knowledge bases, two conflicting requirements arise in querying: on one hand, we would like to query the knowledge base with powerful mechanisms that are able to reason about incompleteness, and on the other hand we aim at query languages that are both close in expressive power to FOL, and decidable (and, possibly, computationally tractable).

This paper presents the following contributions:

- We define a new query language for DL knowledge bases, called *EQL* (see Section 2), based on a variant of the well-known first-order modal logic of knowledge/belief [12, 17, 13]. The language incorporates a modal operator $\mathbf{K}$, that is used to formalize the epistemic state of the knowledge base. Informally, the formula $\mathbf{K}\phi$ is read as "ϕ is known to hold (by the knowledge base)". Using this operator, we are able to pose queries that reason about the incompleteness of information represented by the knowledge base. For instance, a user can express queries that are able to incorporate closed-world reasoning on demand.

- We show (see Section 3) that through a subset of this language, called *EQL-Lite*($\mathcal{Q}$), we are able to formulate queries that are interesting both from the expressive power point of view, and from the computational complexity perspective. Queries in *EQL-Lite*($\mathcal{Q}$) have atoms that are expressed using a specific query language $\mathcal{Q}$, and enjoy the property that they can be evaluated essentially with the same data complexity (i.e., measured wrt the size of the ABox only) as $\mathcal{Q}$.

- We investigate the properties of *EQL-Lite*($\mathcal{Q}$) for the cases of $\mathcal{ALCQI}$ (Section 4) and *DL-Lite* (Section 5) knowledge bases, under the assumption that $\mathcal{Q}$ is the language of unions of conjunctive queries. We study the data complexity of query answering for both cases. In particular, we show that answering such queries over *DL-Lite* knowledge bases is LogSpace, and, notably, can be reduced to

[1] Indeed, query answering can be easily reduced to validity checking in FOL.

evaluating FOL queries over the ABox, when considered as a database. It follows that query processing in this setting can be done through standard database technologies.

2 Epistemic query language

We make use of a variant of the well-known first-order modal logic of knowledge/belief [12, 16, 13] (see also [7, 11]), here called *EQL*. The language *EQL* is a first-order modal language with equality and with a single modal operator **K**, constructed from concepts, interpreted as unary predicates, and roles/relations, interpreted as binary/n-ary predicates, and an infinitely countable set of disjoint constants (a.k.a., *standard names* [13]) corresponding to elements of an infinite countable fixed domain Δ. In *EQL*, the modal operator is used to formalize the epistemic state of the knowledge base. Informally, the formula $\mathbf{K}\phi$ should be read as "ϕ is known to hold (by the knowledge base)".

Under this view, a DL knowledge base corresponds to a finite set of FOL sentences (i.e., closed FOL formulas), capturing what is known about the world. We query such information by using (possibly open) *EQL* formulas possibly involving **K**.

In the following, we use the symbol c (possibly with subscript) to denote a constant, the symbol x to denote a variable, and ϕ, ψ to denote arbitrary formulas.

A *world* is a first-order interpretation (over Δ). An *epistemic interpretation* is a pair E, w, where E is a (possibly infinite) set of worlds, and w is a world in E. We inductively define when a sentence (i.e., a closed formula) ϕ *is true in an interpretation* E, w (or, is true in w and E), written $E, w \models \phi$, as follows:[2]

$$
\begin{array}{lll}
E, w \models c_1 = c_2 & \text{iff} & c_1 = c_2 \\
E, w \models P(c_1, \ldots, c_n) & \text{iff} & w \models P(c_1, \ldots, c_n) \\
E, w \models \phi_1 \wedge \phi_2 & \text{iff} & E, w \models \phi_1 \text{ and } E, w \models \phi_2 \\
E, w \models \neg\phi & \text{iff} & E, w \not\models \phi \\
E, w \models \exists x.\psi & \text{iff} & E, w \models \psi_c^x \text{ for some constant } c \\
E, w \models \mathbf{K}\psi & \text{iff} & E, w' \models \psi \text{ for every } w' \in E
\end{array}
$$

Formulas without occurrences of **K** are said to be *objective* since they talk about what is true. Observe that in order to establish if $E, w \models \phi$, where ϕ is an objective formula, we have to look at w but not at E: we only need the FOL interpretation w. All assertions in the DL knowledge base are indeed objective sentences.

Instead, formulas where each occurrence of predicates and of the equality is in the scope of the **K** operator are said to be *subjective*, since they talk about what is known to be true. Observe that for a subjective sentence ϕ, in order to establish if $E, w \models \phi$ we do not have to look at w but only at E. We use such formulas to query what the knowledge base knows. In other words, through subjective sentences we do not query information about the world represented by the knowledge base; instead, we query the epistemic state of the knowledge base itself. Obviously there are formulas that

[2] For a formula ϕ with free variables $x_1, \ldots, x_n$, we use $\phi_{c_1,\ldots,c_n}^{x_1,\ldots,x_n}$ to denote the formula obtained from ϕ by substituting each free occurrence of the variable x_i with the constant c_i, for each $i \in \{1, \ldots, n\}$.

are neither objective nor subjective. For example $\exists x.P(x)$ is an objective sentence, $\mathbf{K}(\exists x.P(x) \wedge \neg \mathbf{K}P(x))$ is a subjective sentence, while $\exists x.P(x) \wedge \neg \mathbf{K}P(x)$ is neither objective nor subjective.

In our setting, among the various epistemic interpretations, we are interested in specific ones that guarantee *minimal knowledge* over a DL knowledge base. Namely: let Σ be a DL knowledge base (TBox and ABox), and let $Mod(\Sigma)$ be the set of all FOL-interpretations that are models of Σ. Then a Σ-*EQL-interpretation* is an epistemic interpretation E, w where $E = Mod(\Sigma)$. We say that a sentence ϕ is Σ-*EQL-satisfiable* if there exists a Σ-*EQL-model* for ϕ, i.e., a Σ-*EQL*-interpretation E, w such that $E, w \models \phi$. Otherwise, we say that ϕ is Σ-*EQL-unsatisfiable*. Observe that for objective formulas this notion of satisfiability becomes the standard notion of FOL-satisfiability (relative to Σ). A sentence ϕ is *EQL-logically implied* by Σ, written $\Sigma \models_{EQL} \phi$, if every Σ-EQL-interpretation is a Σ-EQL-model of ϕ.

It is worth mentioning some characterizing properties of *EQL*.

Proposition 1 *For every DL knowledge base Σ and every EQL sentence ϕ we have:*

$$\Sigma \models_{EQL} \mathbf{K}\phi \supset \phi$$
$$\Sigma \models_{EQL} \mathbf{K}\phi \supset \mathbf{K}\mathbf{K}\phi$$
$$\Sigma \models_{EQL} \neg\mathbf{K}\phi \supset \mathbf{K}\neg\mathbf{K}\phi$$

These are the standard S5 axioms of modal logic. The first one expresses that "what is known is true" (knowledge is accurate), and the latter two express that the knowledge base has "complete knowledge on what is known and not known".

Proposition 2 *For every DL knowledge base Σ and every objective EQL sentence ϕ we have:*

$$\Sigma \models \phi \quad \textit{iff} \quad \Sigma \models_{EQL} \mathbf{K}\phi$$
$$\Sigma \not\models \phi \quad \textit{iff} \quad \Sigma \models_{EQL} \neg\mathbf{K}\phi$$

The above proposition relates knowledge to FOL logical implication. It says that if an objective sentence ϕ is logically implied then it is known, and vice-versa, that if ϕ is not logically implied then it is not known. Notably, the latter property is a consequence of the minimal knowledge semantics that we are adopting. Observe also that, as a consequence of this, every objective sentence is either known or not known by a DL knowledge base.

Proposition 3 *For every subjective EQL formula ϕ with free variables $x_1, \ldots, x_n$ there is another subjective EQL formula ϕ', with free variables $x_1, \ldots, x_n$, such that: (i) every occurrence of a subformula of the form $\mathbf{K}\psi$ in ϕ' is such that ψ is a non-subjective formula and $\mathbf{K}\psi$ occurs in ϕ; (ii) for every epistemic interpretation E, w, we have that $E, w \models \forall x_1, \ldots, x_n.\phi \equiv \phi'$.*

The above proposition says that we do not gain expressive power by putting in the scope of the $\mathbf{K}$ operator a formula that is already subjective. In other words, if we start from formulas of the form $\mathbf{K}\psi$, where ψ is not subjective, as the basic building blocks of the language, then applying the full *EQL* constructs actually gives

the same expressive power as applying the first-order constructs only. By the way, to get the sentence ϕ' from ϕ we simply need to "push inward" the $\mathbf{K}$ operators through subjective subformulas, stopping when we get to subformulas that are not subjective, and simplifying $\mathbf{KK}\psi$ to $\mathbf{K}\psi$ whenever possible.

Finally we provide the definition of *EQL*-queries.

Definition 4 An *EQL-query* is an *EQL*-formula q with free variables $x_1, \ldots, x_n$, for $n \geq 0$, written $q[x_1, \ldots, x_n]$. ∎

Given a Σ-*EQL*-interpretation E, w, we say that an n-tuple $(c_1, \ldots, c_n)$ of constants in Δ *satisfies an EQL-query* $q[x_1, \ldots, x_n]$ *in* E, w, written $E, w \models q[c_1, \ldots, c_n]$, if $E, w \models q^{x_1,\ldots,x_n}_{c_1,\ldots,c_n}$. A tuple $(c_1, \ldots, c_n)$ of constants in Δ is a *certain answer* to q over Σ, denoted $\Sigma \models_{EQL} q[c_1, \ldots, c_n]$, if $E, w \models q[c_1, \ldots, c_n]$ for every Σ-EQL-interpretation E, w.

Given two *EQL*-queries $q[x_1, \ldots, x_n]$ and $q'[x_1, \ldots, x_n]$ we say that $q[x_1, \ldots, x_n]$ is *contained in* (resp., *equivalent to*) $q'[x_1, \ldots, x_n]$ if for every Σ-*EQL*-interpretation E, w and every n-tuple $(c_1, \ldots, c_n)$ of constants in Δ we have that $E, w \models q[c_1, \ldots, c_n]$ implies (resp., if and only if) $E, w \models q[c_1, \ldots, c_n]$.

Example 5 Consider the DL knowledge base Σ constituted by the following TBox $\mathcal{T}$ and ABox $\mathcal{A}$:

$$\begin{array}{rcl} \mathcal{T} & = & \{\ \mathsf{Male} \sqsubseteq \neg \mathsf{Female}\ \} \\ \mathcal{A} & = & \{\ \mathsf{Female}(\mathsf{mary}), \mathsf{Female}(\mathsf{ann}), \mathsf{Female}(\mathsf{jane}), \mathsf{Male}(\mathsf{bob}), \\ & & \ \ \mathsf{Male}(\mathsf{john}), \mathsf{Male}(\mathsf{paul}), \mathsf{PARENT}(\mathsf{bob}, \mathsf{mary}), \mathsf{PARENT}(\mathsf{bob}, \mathsf{ann}), \\ & & \ \ \mathsf{PARENT}(\mathsf{john}, \mathsf{paul}), \mathsf{PARENT}(\mathsf{mary}, \mathsf{jane})\ \} \end{array}$$

Suppose we want to know the set of males that have only female children. This corresponds to the following first-order query q_1:

$$q_1[x] \ = \ \mathsf{Male}(x) \wedge \forall y.\mathsf{PARENT}(x, y) \rightarrow \mathsf{Female}(y)$$

It is easy to verify that the set of certain answers of q_1 over Σ is empty. In particular, bob is not a certain answer to the above query, since (due to the open-world semantics of DLs) there are models of Σ in which the interpretation of PARENT contains pairs of elements of the form (bob, x) and the interpretation of Male contains the element x.

Suppose now that we want to know who are the persons all of whose *known* children are female. This can be expressed by the following *EQL* query q_2:

$$q_2[x] \ = \ \mathsf{Male}(x) \wedge \forall y.(\mathbf{K}\mathsf{PARENT}(x, y)) \rightarrow \mathsf{Female}(y)$$

It is immediate to verify that the certain answers over Σ of the query q_2 are bob and paul. In fact, for each Σ-*EQL*-interpretation E, w (we recall that $E = Mod(\Sigma)$), $(\mathsf{bob}, \mathsf{mary})$ and $(\mathsf{bob}, \mathsf{ann})$ are the only pairs (x, y) such that $\Sigma \models_{EQL} \mathbf{K}\mathsf{PARENT}(x, y)$; moreover, paul is a certain answer because he is male and has no known children. Analogously, it can be seen that no other constant is in the set of certain answers of q_2 over Σ. ∎

Example 6 Suppose now that we want to know who are the single children according to what is known, i.e., the known children who have no known sibling. This can be expressed by the following *EQL* query q_3:

$$q_3[x] \;=\; \exists y.(\mathbf{K}\mathsf{PARENT}(y,x)) \wedge \forall z.(\mathbf{K}\mathsf{PARENT}(y,z)) \rightarrow z = x$$

It is immediate to verify that the certain answers over Σ of the query q_3 are paul and jane. ∎

3 *EQL-Lite*($\mathcal{Q}$)

We introduce now the query language *EQL-Lite*($\mathcal{Q}$). Such a language is parameterized with respect to a *basic* query language $\mathcal{Q}$, which is a subset of *EQL*. Informally, *EQL-Lite*($\mathcal{Q}$) is the first-order query language with equality whose atoms are formulas of the form $\mathbf{K}q$ where q is a $\mathcal{Q}$-query, i.e., a query in $\mathcal{Q}$.

To define *EQL-Lite*($\mathcal{Q}$) formally, we first need to introduce the notion of domain independence for first-order queries, which is the semantical restriction on first-order logic that is needed to get the equivalence to relational algebra [1]. A first-order query q is *domain independent* if for each pair of FOL interpretations $\mathcal{I}_1$ and $\mathcal{I}_2$, respectively over domains $\Delta_{\mathcal{I}_1} \subseteq \Delta$ and $\Delta_{\mathcal{I}_2} \subseteq \Delta$, for which $P^{\mathcal{I}_1} = P^{\mathcal{I}_2}$ for all atomic relations P, we have that $q^{\mathcal{I}_1} = q^{\mathcal{I}_2}$ [3].

Given a subset $\mathcal{Q}$ of *EQL*, we call *epistemic atom in* $\mathcal{Q}$ an expression of the form $\mathbf{K}q[x_1,\ldots,x_n]$, where $q[x_1,\ldots,x_n]$ is a $\mathcal{Q}$-query.

Definition 7 An *EQL-Lite($\mathcal{Q}$) query* is a formula ψ that:

- is constructed according to the following syntax:

$$\psi \;::=\; a \mid x_1 = x_2 \mid \psi_1 \wedge \psi_2 \mid \neg\psi \mid \exists x.\psi,$$

  where a is an epistemic atom in $\mathcal{Q}$, and

- is *domain-independent*, when we consider epistemic atoms as atomic formulas. ∎

Observe that in *EQL-Lite*(Q) we do not allow for nesting of the $\mathbf{K}$ operator outside the expressions of the basic query language $\mathcal{Q}$. Indeed, we now show that allowing such nested occurrences of the epistemic operator does not actually increase the expressive power of *EQL-Lite*($\mathcal{Q}$).

Proposition 8 *Let EQL-Lite($\mathcal{Q}$)$^+$ be the extension of EQL-Lite($\mathcal{Q}$) obtained by adding to the abstract syntax for EQL-Lite($\mathcal{Q}$) formulas the rule*

$$\psi \;::=\; \mathbf{K}\psi$$

For each query $q \in$ EQL-Lite($\mathcal{Q}$)$^+$, there exists a query $q' \in$ EQL-Lite($\mathcal{Q}$) such that q is equivalent to q'.

[3] For an interpretation $\mathcal{I}$ over domain $\Delta_{\mathcal{I}}$ and a FOL query $q[x_1,\ldots,x_n]$, we use $q^{\mathcal{I}}$ to denote the result of the evaluation of q in $\mathcal{I}$, i.e., the set of tuples $(c_1,\ldots,c_n)$ of constants in $\Delta_{\mathcal{I}}$ such that $\phi^{x_1,\ldots,x_n}_{c_1,\ldots,c_n}$ is true in $\mathcal{I}$.

The above property is an immediate consequence of Proposition 3, since *EQL-Lite*($\mathcal{Q}$) queries are subjective *EQL* formulas, where the **K** operator is applied to non subjective (in fact objective) subformulas only, and hence each *EQL-Lite*($\mathcal{Q}$)$^+$ query can be reduced to an equivalent *EQL-Lite*($\mathcal{Q}$) query by pushing inward the **K** operator, stopping in front of the epistemic atoms, and simplifying $\mathbf{KK}\psi$ to $\mathbf{K}\psi$ whenever possible.

In spite of its expressive richness, *EQL-Lite*($\mathcal{Q}$) enjoys an interesting complexity characterization of query answering. In the rest of the paper, when we speak about the computational complexity of the query answering problem we actually refer to the computational complexity of the *recognition problem associated with query answering* [1]. Let $\mathcal{Q}$ be a query language and $\mathcal{L}$ a DL language, and let us assume that the query language $\mathcal{Q}$ over $\mathcal{L}$-knowledge bases has data complexity $C_{\mathcal{Q},\mathcal{L}}$, i.e., the complexity of answering queries in $\mathcal{Q}$ over $\mathcal{L}$-knowledge bases measured in the size of the data of the knowledge base is $C_{\mathcal{Q},\mathcal{L}}$.

Let us further consider the following restriction over queries and knowledge bases.

Definition 9 Given a knowledge base Σ in $\mathcal{L}$, a query q in $\mathcal{Q}$ is Σ-*range-restricted*, if the certain answers of q over Σ contain only elements of $adom(\Sigma)$, where $adom(\Sigma)$ denotes the set of constants occurring in Σ. By extension, an *EQL-Lite*($\mathcal{Q}$) query is Σ-range-restricted if each of its epistemic atoms involves a Σ-range-restricted query. ∎

The class of Σ-range-restricted queries is perfectly natural in this setting: indeed, it can be shown that if the set of certain answers of a query q on a knowledge base Σ contains elements that are not in $adom(\Sigma)$, then the set of certain answers is infinite.

Example 10 The following in an *EQL-Lite*($\mathcal{Q}$) query, where $\mathcal{Q}$ is the language of atomic queries:

$$q_4[x] \;=\; (\mathbf{K}\mathsf{Male}(x)) \wedge \forall y.(\mathbf{K}\mathsf{PARENT}(x,y)) \rightarrow (\mathbf{K}\mathsf{Female}(y))$$

It is easy to verify that for the knowledge base Σ given in Example 5, q_4 is Σ-range-restricted. ∎

Observe that, by Proposition 2, we have complete information on each instantiation of the epistemic atoms of an *EQL-Lite*($\mathcal{Q}$) query, i.e., either the instantiated epistemic atom is entailed by Σ or its negation is entailed by Σ. Now, answering a Σ-range-restricted *EQL-Lite*($\mathcal{Q}$) query amounts to evaluating a domain independent first-order query whose variables range over $adom(\Sigma)$ and whose instantiated epistemic atoms $\mathbf{K}q[c_1,\ldots,c_n]$ can be checked by verifying whether $(c_1,\ldots,c_n)$ is a certain answer of q over the knowledge base. We know that evaluating a first-order query over a given database is in LogSpace in data complexity [1], and, by our assumption, computing whether a tuple of elements of $adom(\Sigma)$ is in the relation corresponding to the extension of an epistemic atom, can be done in $C_{\mathcal{Q},\mathcal{L}}$ in data complexity. Hence, we immediately derive the following result on the data complexity of answering Σ-range-restricted *EQL-Lite*($\mathcal{Q}$)-queries, where we denote with $C_1^{C_2}$ the class of languages recognized by a C_1-Turing Machine that uses an oracle in C_2.

Theorem 11 *Let $\mathcal{Q}$ be a query language over $\mathcal{L}$-knowledge bases that is in $C_{\mathcal{Q},\mathcal{L}}$ with respect to data complexity. Let Σ be an $\mathcal{L}$-knowledge base, and q a Σ-range-restricted EQL-Lite($\mathcal{Q}$) query. Then, answering q over Σ is in* LogSpace$^{C_{\mathcal{Q},\mathcal{L}}}$ *with respect to data complexity.*

Among the various choices of the basic query language $\mathcal{Q}$ in *EQL-Lite*($\mathcal{Q}$), a prominent role is played by *unions of conjunctive queries (UCQs)*. In fact, the language of UCQs is currently the most expressive subset of first-order logic for which query answering over DL knowledge bases is known to be decidable [5, 15]. Consequently, in the following we will focus on *EQL-Lite*(UCQ), and will call such a language simply *EQL-Lite*.

4 Answering *EQL-Lite* queries over $\mathcal{ALCQI}$ knowledge bases

As a consequence of the properties shown in the previous section, we now provide a computational characterization of answering Σ-range-restricted *EQL-Lite* queries in $\mathcal{ALCQI}$. It is known that answering unions of conjunctive queries over $\mathcal{ALCQI}$ knowledge bases is coNP-complete with respect to data complexity [15]. Based on this characterization and on Theorem 11, we are able to show the following result.

Theorem 12 *Let Σ be an $\mathcal{ALCQI}$-knowledge base, and q a Σ-range-restricted EQL-Lite query. Then, answering q over Σ is in Θ_2^p with respect to data complexity.*

We recall that $\Theta_2^p = \Delta_2^p[O(\log n)] = P^{NP[O(\log n)]}$ [10, 8], i.e., Θ_2^p is the class of the decision problems that can be solved in polynomial time through a logarithmic number of calls to an NP-oracle. Such a class is considered as "mildly" harder than the class NP, since a problem in Θ_2^p can be solved by solving "few" (i.e., a logarithmic number of) instances of problems in NP. Consequently, answering *EQL-Lite* queries in $\mathcal{ALCQI}$ (and in all the DLs in which answering UCQs is a coNP-complete problem) is "mildly harder" than answering UCQs.

5 Answering *EQL-Lite* queries over *DL-Lite* knowledge bases

In this section we study *EQL-Lite* queries posed over *DL-Lite* knowledge bases. *DL-Lite* [6, 3] is a DL specifically tailored to capture basic ontology languages, while keeping low complexity of reasoning, in particular, polynomial in the size of the instances in the knowledge base. Answering UCQs in *DL-Lite* is in LogSpace with respect to data complexity[4]. Moreover all UCQs in *DL-Lite* are Σ-range-restricted. As a consequence of Theorem 11 we get that moving from UCQs to *EQL-Lite* does not blow up computational complexity of the query answering problem.

[4]It is easy to see that all results for CQs in [6, 3] can be extended to UCQs.

Theorem 13 *Answering EQL-Lite queries over DL-Lite knowledge bases is in* LogSpace *with respect to data complexity.*

We point out that membership in LogSpace for the problem of answering UCQs over *DL-Lite* knowledge bases follows from a notable property of such a DL language, namely, *FOL-reducibility* of query answering [4]. Intuitively, FOL-reducibility means that query answering can be reduced to evaluating queries over the database corresponding to the ABox of a DL knowledge base, which therefore can be maintained in secondary storage. More formally, given an ABox $\mathcal{A}$ involving membership assertions on atomic concept and roles only, we define $\mathcal{I}_\mathcal{A}$ as the interpretation constructed as follows:

- $a^{\mathcal{I}_\mathcal{A}} = a$ for each constant a,
- $A^{\mathcal{I}_\mathcal{A}} = \{a \mid A(a) \in \mathcal{A}\}$ for each atomic concept A, and
- $P^{\mathcal{I}_\mathcal{A}} = \{(a_1, a_2) \mid P(a_1, a_2) \in \mathcal{A}\}$ for each atomic role P.

Then, query answering in a DL $\mathcal{L}$ is *FOL-reducible* if for every query q (of a given language) and every TBox $\mathcal{T}$ expressed in $\mathcal{L}$, there exists a FOL query q_1 such that for every ABox $\mathcal{A}$, we have that $(\mathcal{T}, \mathcal{A}) \models_{EQL} q[c_1, \ldots, c_n]$ if and only if $(c_1, \ldots, c_n)^{\mathcal{I}_\mathcal{A}} \in q_1^{\mathcal{I}_\mathcal{A}}$. In other words, q_1 is evaluated over the ABox $\mathcal{A}$ considered as a database. Observe that FOL-reducibility is a very nice property from a practical point of view. Indeed, in all such cases in which query answering can be reduced to evaluation of a suitable *domain independent* FOL query q_1, then q_1 can be expressed in relational algebra, i.e., in SQL. Therefore, query answering can take advantage of optimization strategies provided by current DBMSs (which are in charge of properly managing ABoxes in secondary storage).

Now, it turns out that FOL-reducibility is also at the basis of the membership in LogSpace of answering *EQL-Lite* queries over *DL-Lite* knowledge bases, as the following theorem shows.

Theorem 14 *Answering EQL-Lite queries in DL-Lite is FOL-reducible. Furthermore, the resulting FOL-queries are domain independent.*

Proof (sketch). We make use of the algorithm for FOL-reducibility of UCQs over *DL-Lite* knowledge bases presented in [3]. More precisely, given a *DL-Lite* TBox $\mathcal{T}$ and an *EQL-Lite* query ψ over $\mathcal{T}$, we execute the algorithm of [3] for each epistemic atom $\mathbf{K}q[x_1, \ldots, x_n]$ of ψ, giving as inputs to each such execution the union of conjunctive queries $q[x_1, \ldots, x_n]$ and the *DL-Lite* TBox $\mathcal{T}$. Then, we substitute to each epistemic atom $\mathbf{K}q[x_1, \ldots, x_n]$ of ψ the union of conjunctive queries $q'[x_1, \ldots, x_n]$ produced by the corresponding execution of the rewriting algorithm, thus obtaining (provided some further syntactic transformations) a FOL query q_1. Now, it is possible to show that q_1 is domain independent, and that for every ABox $\mathcal{A}$, $(\mathcal{T}, \mathcal{A}) \models_{EQL} q[c_1, \ldots, c_n]$ iff $(c_1, \ldots, c_n)^{\mathcal{I}_\mathcal{A}} \in q_1^{\mathcal{I}_\mathcal{A}}$, thus proving the claim. □

As a consequence, to perform query answering of *EQL-Lite* queries in *DL-Lite*, we can rely on traditional relational DBMSs.

Recently, different versions of *DL-Lite* have been considered, and an entire family of "lite" DLs, namely, the *DL-Lite family*, has been defined [4]. Roughly speaking, DLs of such a family differ one another for the set of constructs allowed in the right-hand side and in the left-hand side of inclusion assertions between concepts and/or roles specified in the TBox (e.g., allowing in certain cases for the presence of existential qualified quantification on the right-hand side, or conjunctions of concepts in the left-hand side), as well as the possibility of specifying functionality assertions on roles, inclusion assertions between roles, and n-any relationships in addition to binary roles. Notably, the DLs of the *DL-Lite* family are the maximal logics allowing for FOL-reducibility of answering unions of conjunctive queries [4]. As for answering *EQL-Lite* queries, we point out that Theorem 14 also holds for all the DLs belonging to the *DL-Lite* family.

6 Conclusions

Motivated by various needs related to querying DL knowledge bases, we have proposed the query language *EQL*, based on a variant of the well-known first-order modal logic of knowledge/belief. Then, we have studied a subset of this language, called *EQL-Lite*($\mathcal{Q}$), arguing that it allows for formulating queries that are interesting both from the expressive power point of view, and from the computational complexity perspective. Finally, we have investigated the properties of *EQL-Lite*($\mathcal{Q}$) for the cases of $\mathcal{ALCQI}$ and *DL-Lite* knowledge bases, under the assumption that $\mathcal{Q}$ is the language of unions of conjuntive queries. In particular, we have shown that answering *EQL-Lite*($\mathcal{Q}$) in the latter setting is LogSpace in data complexity, and, notably, can be done through standard database technologies.

We are working specifically on *EQL-Lite*($\mathcal{Q}$) for *DL-Lite* knowledge bases, for the case where $\mathcal{Q}$ is the query language whose queries are either a UCQ or a comparison atom involving values taken from a set of given domains. We are currently implementing such an extended language with the goal of enhancing the querying capabilities of the QuOnto system [2].

References

[1] S. Abiteboul, R. Hull, and V. Vianu. *Foundations of Databases*. Addison Wesley Publ. Co., 1995.

[2] A. Acciarri, D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, M. Palmieri, and R. Rosati. QuOnto: QUerying ONTOlogies. In *Proc. of the 20th Nat. Conf. on Artificial Intelligence (AAAI 2005)*, pages 1670–1671, 2005.

[3] D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, and R. Rosati. DL-Lite: Tractable description logics for ontologies. In *Proc. of the 20th Nat. Conf. on Artificial Intelligence (AAAI 2005)*, pages 602–607, 2005.

[4] D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, and R. Rosati. Data complexity of query answering in description logics. In *Proc. of the 10th Int.*

Conf. on the Principles of Knowledge Representation and Reasoning (KR 2006), 2006.

[5] D. Calvanese, G. De Giacomo, and M. Lenzerini. On the decidability of query containment under constraints. In *Proc. of the 17th ACM SIGACT SIGMOD SIGART Symp. on Principles of Database Systems (PODS'98)*, pages 149–158, 1998.

[6] D. Calvanese, G. De Giacomo, M. Lenzerini, R. Rosati, and G. Vetere. DL-Lite: Practical reasoning for rich DLs. In *Proc. of the 2004 Description Logic Workshop (DL 2004)*. CEUR Electronic Workshop Proceedings, `http://ceur-ws.org/Vol-104/`, 2004.

[7] B. F. Chellas. *Modal Logic: An introduction*. Cambridge University Press, 1980.

[8] T. Eiter and G. Gottlob. The complexity class Θ_2^p: Recent results and applications in AI and modal logic. In *Proc. of the 11th Int. Symp. on Fundamentals of Computation Theory (FCT'97)*, volume 1279 of *Lecture Notes in Artificial Intelligence*, pages 1–18. Springer, 1997.

[9] R. Fikes, P. Hayes, and I. Horrocks. OWL-QL: A language for deductive query answering on the semantic web. *J. of Web Semantics*, 2(1), 2005.

[10] G. Gottlob. NP trees and Carnap's modal logic. *J. of the ACM*, 42(2):421–457, 1995.

[11] G. E. Hughes and M. J. Cresswell. *A Companion to Modal Logic*. Methuen, London (United Kingdom), 1984.

[12] H. J. Levesque. Foundations of a functional approach to knowledge representation. *Artificial Intelligence*, 23:155–212, 1984.

[13] H. J. Levesque and G. Lakemeyer. *The Logic of Knowledge Bases*. The MIT Press, 2001.

[14] M. M. Ortiz, D. Calvanese, and T. Eiter. Characterizing data complexity for conjunctive query answering in expressive description logics. In *Proc. of the 21st Nat. Conf. on Artificial Intelligence (AAAI 2006)*, 2006.

[15] M. M. Ortiz, D. Calvanese, and T. Eiter. Data complexity of answering unions of conjunctive queries in $\mathcal{SHIQ}$. In *Proc. of the 2006 Description Logic Workshop (DL 2006)*. CEUR Electronic Workshop Proceedings, `http://ceur-ws.org/`, 2006.

[16] R. Reiter. What should a database know? *J. of Logic Programming*, 14:127–153, 1990.

[17] R. Reiter. Narratives as programs. In *Proc. of the 7th Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR 2000)*, pages 99–108, 2000.

Data Complexity of Answering Unions of Conjunctive Queries in $\mathcal{SHIQ}^*$

Magdalena Ortiz[1,2], Diego Calvanese[1], Thomas Eiter[2]

[1] Faculty of Computer Science
Free University of Bozen-Bolzano
Piazza Domenicani 3, Bolzano, Italy
calvanese@inf.unibz.it,
magdalena.ortiz@stud-inf.unibz.it

[2] Institute of Information Systems
Vienna University of Technology
Favoritenstraße 9-11, Vienna, Austria
eiter@kr.tuwien.ac.at

Abstract

The novel context of accessing and querying large data repositories through ontologies that are formalized in terms of expressive DLs requires on the one hand to consider query answering as the primary inference technique, and on the other hand to optimize it with respect to the size of the data, which dominates the size of ontologies. While the complexity of DLs has been studied extensively, data complexity in expressive DLs has been characterized only for answering atomic queries, and was still open for more expressive query languages, such as unions of conjunctive queries (UCQs). In this paper we advocate the need for studying this problem, and provide a significant technical contribution in this direction. Specifically, we prove a tight coNP upper bound for answering UCQs over $\mathcal{SHIQ}$ knowledge bases, for the case where the queries do not contain transitive roles. We thus establish that for a whole range of DLs from AL to $\mathcal{SHIQ}$, answering such UCQs has coNP-complete data complexity. We obtain our result by a novel tableaux-based algorithm for checking query entailment, inspired by the one in [20], but which manages the technical challenges of simultaneous inverse roles and number restrictions (which leads to a DL lacking the finite model property).

1 Introduction

Description Logics (DLs) [2] provide the formal foundations for the standard Web ontology languages [11]. In fact, OWL-Lite and OWL-DL are syntactic variants of the DLs $\mathcal{SHIF}(D)$ and $\mathcal{SHOIN}(D)$, respectively [12, 22]. In the Semantic Web and domains such as Enterprise Application Integration and Data Integration [19], ontologies provide a high-level, conceptual view of the relevant information. However, they are increasingly seen also as a mechanism to access and query data repositories.

*This work was partially supported by the Austrian Science Funds (FWF) project P17212, by the European Commission project REWERSE (IST-2003-506779), and by the European Commission FET project TONES (FP6-7603).

This novel context poses an original combination of challenges both in DLs/ontologies and in related areas such as data modeling and querying in databases:

(*i*) On the one hand, data repositories can be very large and are usually much larger than the intensional level expressing constraints on the data. Therefore, the contribution of the extensional data to inference complexity must be singled out, and one must pay attention to optimizing inference techniques with respect to data size, as opposed to the overall size of the knowledge base. In databases, this is accounted for by *data complexity* of query answering [26], where the relevant parameter is the size of the data, as opposed to *combined complexity*, which additionally considers the size of the query and of the schema.

(*ii*) On the other hand, the data underlying an ontology should be accessed using well established and flexible mechanisms such as those provided by database query languages. This goes well beyond the traditional inference tasks involving objects in DL-based systems, like *instance checking* [10, 23]. Indeed, since explicit variables are missing, DL concepts have limited possibility for relating specific data items to each other. *Conjunctive queries (CQs)* and unions of CQs (UCQs) provide a good tradeoff between expressive power and nice computational properties, and thus are adopted as core query language in several contexts, such as data integration [19].

(*iii*) Finally, the considered DLs should have sufficient expressive power to capture common constructs used in data modeling [4]. This calls for *expressive DLs* [5], such as $\mathcal{SHIQ}$ [13, 15], featuring besides full Booleans also number restrictions, inverse and transitive roles, and role hierarchies.

As for data complexity of DLs, [10, 23] showed that instance checking is coNP-hard already in the rather weak DL $\mathcal{ALE}$, and [7] that CQ answering is coNP-hard in the yet weaker DL $\mathcal{AL}$. For suitably tailored DLs, answering UCQs is polynomial (actually LogSpace) in data complexity [6, 7]; see [7] for an investigation of the NLogSpace, PTime, and coNP boundaries.

For expressive DLs (with the features above, notably inverse roles), TBox+ABox reasoning has been studied extensively using techniques ranging from reductions to Propositional Dynamic Logic (PDL) (see, e.g., [8, 5]) over tableaux [3, 15] to automata on infinite trees [5, 25]. For many such DLs, the combined complexity of TBox+ABox reasoning is ExpTime-complete, including $\mathcal{ALCQI}$ [5, 25], $\mathcal{DLR}$ [8], and $\mathcal{SHIQ}$ [25]. However, until recently, little attention has been explicitly devoted to data complexity in expressive DLs. An ExpTime upper bound for data complexity of UCQ answering in $\mathcal{DLR}$ follows from the results on UCQ containment and view-based query answering in [8, 9]. They are based on a reduction to reasoning in PDL, which however prevents to single out the contribution to the complexity coming from the ABox. In [20] a tight coNP upper bound for CQ answering in $\mathcal{ALCNR}$ is shown. However, this DL lacks inverse roles and is thus not suited to capture semantic data models or UML. In [17, 18] a technique based on a reduction to Disjunctive Datalog is used for $\mathcal{ALCHIQ}$. For instance checking, it provides a (tight) coNP upper bound for data complexity, since it allows to single out the ABox contribution. The result can immediately be extended to tree shaped conjunctively queries, since these admit a representation as a description logic concept (e.g., by making use of the notion of tuple-graph of [8], or via rolling up [16]). However, this is not the case for general CQs, resulting in a non-tight

2ExpTime upper bound (matching also combined complexity).

Summing up, a precise characterization of data complexity for UCQ answering in expressive DLs was still open, with a gap between a coNP lower-bound and an ExpTime upper bound. We close this gap, thus simultaneously addressing the three challenges identified above. Specifically, we make the following contributions:

- Building on techniques of [20, 15], we devise a novel tableaux-based algorithm for answering UCQs where all roles are simple over $\mathcal{SHIQ}$ knowledge bases. Technically, to show its soundness and completeness, we have to deal both with a novel blocking condition (inspired by the one in [20], but taking into account inverse and transitive roles), and with the lack of the finite model property.

- This novel algorithm provides us with a characterization of data complexity for UCQ answering in expressive DLs. Specifically, we show that data complexity of answering such an UCQ over $\mathcal{SHIQ}$ knowledge bases is in coNP, and thus coNP-complete for all DLs ranging from $\mathcal{AL}$ to $\mathcal{SHIQ}$.

For space reasons, proofs are omitted here; they can be found in [21].

2 Preliminaries

We only briefly recall $\mathcal{SHIQ}$ and refer to the literature (e.g., [13, 15]) for details and background. We denote by $\mathbf{C}$, $\mathbf{R}$, $\mathbf{R}_+$ (where $\mathbf{R}_+ \subseteq \mathbf{R}$), and $\mathbf{I}$ the sets of *concept names*, *role names*, *transitive role names*, and *individuals* respectively. A *role* R is either an atomic role name P or its inverse P^-. A *concept* C is either an atomic concept name A or one of $C \sqcap D$, $C \sqcup D$, $\neg C$, $\forall R.C$, $\exists R.C$, $\geq n\, S.C$, or $\leq n\, S.C$, where C and D denote concepts, R a role, S a *simple* role (see later), and $n \geq 0$ an integer. A *knowledge base* is a triple $K = \langle \mathcal{T}, \mathcal{R}, \mathcal{A} \rangle$, where the *TBox* $\mathcal{T}$ is a set of *concept inclusion axioms* $C_1 \sqsubseteq C_2$; the *role hierarchy* $\mathcal{R}$ is a set of *role inclusion axioms* $R_1 \sqsubseteq R_2$; and the *ABox* $\mathcal{A}$ is a set of *assertions* $A(a)$, $P(a, b)$, and $a \neq b$, where A (resp., P) is an atomic concept (resp., role) and a and b are individuals.

A role S is *simple*, if for no role $R \in \mathbf{R}_+$ we have that $R \sqsubseteq^* S$, where $\sqsubseteq^*$ denotes the reflexive and transitive closure of the subrole relation $\sqsubseteq$ over $\mathcal{R} \cup \{\mathsf{Inv}(R_1) \sqsubseteq \mathsf{Inv}(R_2) \mid R_1 \sqsubseteq R_2 \in \mathcal{R}\}$. Without loss of expressivity, we assume that all concepts in K are in *negation normal form* (NNF). For a concept C, $\mathsf{clos}(C)$ is the smallest set of concepts containing C that is closed under subconcepts and their negation (in NNF); and $\mathsf{clos}(K)$ denotes the union of all $\mathsf{clos}(C)$ for each C occurring in K. We will use K to denote a knowledge base $\langle \mathcal{T}, \mathcal{R}, \mathcal{A} \rangle$, $\mathbf{R}_K$ the roles occurring in K and their inverses, and $\mathbf{I}_K$ the individuals occurring in $\mathcal{A}$.

The semantics of K is defined in terms of first-order *interpretations* $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$, where $\Delta^{\mathcal{I}}$ is the domain and $\cdot^{\mathcal{I}}$ the valuation function, as usual (without unique names assumption;[1] see [2]). $\mathcal{I}$ is a *model* of K, denoted $\mathcal{I} \models K$, if it satisfies $\mathcal{T}$, $\mathcal{R}$ and $\mathcal{A}$.

[1] The unique names assumption can be easily emulated using $\not\approx$.

Example 1 *As a running example, we use the knowledge base*

$$K = \langle \{A \sqsubseteq \exists P_1.A,\ A \sqsubseteq \exists P_2.\neg A\},\ \{\},\ \{A(a)\}\rangle.$$

We assume that K has an associated set of *distinguished concept names*, denoted $\mathcal{C}_q$, which are the concepts that can occur in queries.

Definition 1 (Union of Conjunctive Queries) *A* conjunctive query *(CQ) Q over a knowledge base K is a set of atoms of the form $\{p_1(\overline{Y_1}), \ldots, p_n(\overline{Y_n})\}$ where each p_i is either a role name in $\mathbf{R}_K$ or a concept in $\mathcal{C}_q$, and $\overline{Y_i}$ is a tuple of variables or individuals in $\mathbf{I}_K$ matching its arity. A* union of CQs (UCQ) *U is defined as an expression of the form $Q_1 \vee \cdots \vee Q_m$ where Q_i is a CQ for each $i \in \{1, \ldots, m\}$.*

For short, a *query* is either a CQ or a UCQ. We will restrict our attention to queries where each role occurring in some $p_i(\overline{Y_i})$ is a *simple* role, i.e., transitive roles and super-roles of transitive roles are disallowed in queries. We denote by $\mathsf{varind}(Q)$ the set of variables and individuals in query Q. An interpretation $\mathcal{I}$ is a model of a CQ Q, denoted $\mathcal{I} \models Q$, if there is a substitution $\sigma : \mathsf{varind}(Q) \to \Delta^{\mathcal{I}}$ such that $\sigma(a) = a^{\mathcal{I}}$ for each individual $a \in \mathsf{varind}(Q)$ and $\mathcal{I} \models p(\sigma(\overline{Y}))$, for each $p(\overline{Y})$ in Q. For a UCQ $U = Q_1 \vee \cdots \vee Q_m$, $\mathcal{I} \models U$ is defined as $\mathcal{I} \models Q_i$ for some $i \in \{1, \ldots, m\}$. We say that K *entails* query Q, denoted $K \models Q$, if $\mathcal{I} \models Q$ for each model $\mathcal{I}$ of K.

Example 2 *Let $\mathcal{C}_q = \{A\}$. We consider the CQs $Q_1 = \{P_1(x,y), P_2(x,z), A(y)\}$ and $Q_2 = \{P_2(x,y), P_2(y,z)\}$ and the UCQ $U = Q_1 \vee Q_2$. Note that $K \models Q_1$. Indeed, for an arbitrary model $\mathcal{I}$ of K, we can map x to $a^{\mathcal{I}}$, y to an object connected to $a^{\mathcal{I}}$ via role P_1 (which by the inclusion axiom $A \sqsubseteq \exists P_1.A$ exists and is an instance of A), and z to an object connected to $a^{\mathcal{I}}$ via role P_2 (which exists by the inclusion $A \sqsubseteq \exists P_2.\neg A$). Also, $K \not\models Q_2$. A model $\mathcal{I}$ of K that is not a model of Q_2 is the one with $\Delta^{\mathcal{I}} = \{o_1, o_2\}$, $a^{\mathcal{I}} = o_1$, $A^{\mathcal{I}} = \{o_1\}$, $P_1^{\mathcal{I}} = \{\langle o_1, o_1\rangle\}$, and $P_2^{\mathcal{I}} = \{\langle o_1, o_2\rangle\}$. Finally, since $K \models Q_1$, then also $K \models U$.*

Query answering for a certain DL $\mathcal{L}$ is in a complexity class $\mathcal{C}$, if given any knowledge base K in $\mathcal{L}$ and query Q, deciding $K \models Q$ is in $\mathcal{C}$; this is also called *combined complexity*. The *data complexity* of query answering is the complexity of deciding $K \models Q$ where Q and all of K except $\mathcal{A}$ is fixed.

We only consider Boolean queries (i.e., yes/no queries, with no free variables), to which queries with distinguished variables are reducible as usual. Note that query answering is not reducible to knowledge base satisfiability, since the negated query is not expressible within the knowledge base.

3 The Query Answering Algorithm

In this section, Q denotes a CQ and U a UCQ. As mentioned, we assume that all roles occurring in queries are simple. We first describe our method for deciding $K \models Q$, and then how it is extended to $K \models U$. Our technique builds on the $\mathcal{SHIQ}$ satisfiability algorithm in [15]. The adaption to query answering is inspired by [20], yet we deal with DLs that have no finite model property.

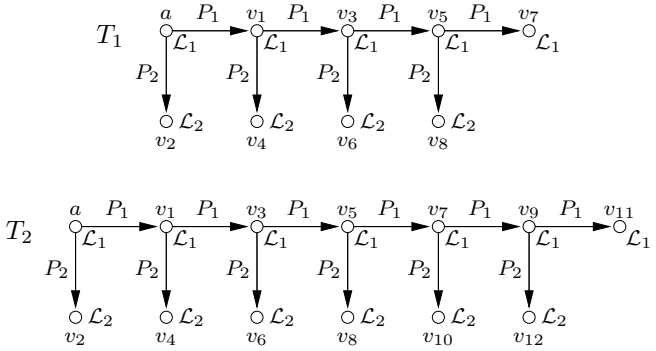


Figure 1: Trees and completion forests for the example knowledge base

We use *completion forests*, which are finite relational structures capturing sets of models of K. Roughly speaking, the models of K are represented by an initial completion forest $\mathcal{F}_K$. By applying tableaux-style *expansion rules* repeatedly, new completion forests are generated non-deterministically where also new individuals might be introduced. Each model of K is preserved in some of the resulting forests. Therefore, checking $K \models Q$ equals checking $\mathcal{F} \models Q$ for each completion forest $\mathcal{F}$. The blocking conditions on the rules, which ensure that expansion terminates, are more involved than those in [15]. They require a depth parameter n (depending on Q) that ensures a sufficient expansion of each forest $\mathcal{F}$, in such a way that semantical entailment of Q in $\mathcal{F}$ can be checked effectively via a syntactic mapping of the variables in Q to the nodes in $\mathcal{F}$. Thus, to witness that $K \not\models Q$, it is sufficient to (nondeterministically) construct a large enough forest $\mathcal{F}$ to which Q cannot be mapped.

A *variable tree* T is a tree whose nodes are variables except the root, which might be also an individual, and where each node v is labeled with a set of concepts $\mathcal{L}(v)$ and each arc $v \to w$ is labeled with a set of roles $\mathcal{L}(v \to w)$. For any integer $n \geq 0$, the *n-tree* of a node v in T, denoted T_v^n, is the subtree of T rooted at v that contains all descendants of v within distance n. Variables v and v' in T are *n-tree equivalent in T*, if T_v^n and $T_{v'}^n$ are isomorphic (i.e., there is a bijection from the nodes of T_v^n to those of $T_{v'}^n$ which preserves all labels). If, for such v and v', v' is an ancestor of v in T and v is not in $T_{v'}^n$, then we say that $T_{v'}^n$ *tree-blocks* T_v^n. A *completion forest* (cf. [15]) for K is constituted by (i) a set of variable trees whose roots are the individuals in $\mathbf{I}_K$ and can be arbitrarily connected by arcs; and (ii) a binary relation $\not\approx$ on the individuals in $\mathbf{I}_K$, implicitly assumed to be symmetric.

Example 3 *Consider the variable tree T_1 in Figure 1, with a as root, $\mathcal{L}_1 = \{A, \neg A \sqcup \exists P_1.A, \neg A \sqcup \exists P_2.\neg A, A \sqcup \neg A, \exists P_1.A, \exists P_2.\neg A\}$, and $\mathcal{L}_2 = \{\neg A, \neg A \sqcup \exists P_1.A, \neg A \sqcup \exists P_2.\neg A, A \sqcup \neg A\}$. Then, v_1 and v_5 are 1-tree equivalent in T_1 and $T_1{}^1_{v_1}$ tree-blocks $T_1{}^1_{v_5}$.*

Now we introduce the initial completion forest for K. We use a set of *global concepts* $\mathsf{gcon}(K, \mathcal{C}_q) = \{\neg C \sqcup D \mid C \sqsubseteq D \in \mathcal{T}\} \cup \{C \sqcup \neg C \mid C \in \mathcal{C}_q\}$. Informally, by requiring that each individual belongs to all global concepts, satisfaction of the TBox

is enforced and, by case splitting, each individual can be classified with respect to the distinguished concepts (i.e., those appearing in queries).

The *initial completion forest* for K, denoted $\mathcal{F}_K$, is defined as follows:[2]

- The nodes are the individuals $a \in \mathbf{I}_K$, and $\mathcal{L}(a) = \{A \mid A(a) \in \mathcal{A}\} \cup \mathsf{gcon}(K, \mathcal{C}_q)$.
- The arc $a \to a'$ is present iff $\mathcal{A}$ contains some assertion $P(a, a')$, and $\mathcal{L}(a \to a') = \{P \mid P(a, a') \in \mathcal{A}\}$.
- $a \not\approx a'$ iff $a \neq a' \in \mathcal{A}$.

Example 4 *In our running example, $\mathcal{F}_K$ contains only the node a which has the label $\mathcal{L}(a) := \{A,\ \neg A \sqcup \exists P_1.A,\ \neg A \sqcup \exists P_2.\neg A,\ A \sqcup \neg A\}$.*

In the expansion rules, we use a notion of blocking that depends on a depth parameter n: a node is *n-blocked* if it's a leaf of a tree-blocked n-tree in $\mathcal{F}$. Note that if v is n-blocked, then it is m-blocked for each $m \leq n$. For $n \geq 1$, n-blocking implies blocking as in [15], and for $n = 0$ amounts to blocking by equal node labels. Starting from $\mathcal{F}_K$, we can generate a set $\mathbb{F}_K$ of new completion forests by applying expansion rules. The rules are analogous to those in [15], but they use "n-blocking" and initialize newly introduced nodes with a label containing $\mathsf{gcon}(K, \mathcal{C}_q)$. When applying the expansion rules, a *clash* may arise in some $\mathcal{F}$, i.e., two complementary concepts C and $\neg C$ occur in a node label, or a node that should satisfy a number restriction $\leq n\, S.C$ has more than n distinct S successors marked C. If this is not the case, then $\mathcal{F}$ is *clash free*. We call a completion forest *n-complete*, if (under n-blocking) no rule can be applied to it. We denote by $\mathsf{ccf}_n(\mathbb{F}_K)$ the set of *n-complete* and clash free completion forests in $\mathbb{F}_K$. For the set of expansion rules and more details, see [21].

Example 5 *Consider the completion forest $\mathcal{F}_1$ with the variable tree T_1 from Example 3 and with empty $\not\approx$. $\mathcal{F}_1$ is 1-blocked. Analogously, consider $\mathcal{F}_2$ that has T_2 in Figure 1 and where $\not\approx$ is also empty. $\mathcal{F}_2$ is 2-blocked. Both $\mathcal{F}_1$ and $\mathcal{F}_2$ can be obtained from $\mathcal{F}_K$ by applying the expansion rules. They are both complete and clash-free, so $\mathcal{F}_1 \in \mathsf{ccf}_1(\mathbb{F}_K)$ and $\mathcal{F}_2 \in \mathsf{ccf}_2(\mathbb{F}_K)$.*

Models of a completion forest

Viewing variables in a completion forest $\mathcal{F}$ as individuals, we can define models of $\mathcal{F}$ as models of K (over an extended vocabulary). An interpretation $\mathcal{I}$ is a *model* of a completion forest $\mathcal{F}$ for K, denoted $\mathcal{I} \models \mathcal{F}$, if $\mathcal{I} \models K$ and for all nodes v, w in $\mathcal{F}$ it holds that (i) $v^{\mathcal{I}} \in C^{\mathcal{I}}$ if $C \in \mathcal{L}(v)$, (ii) $\langle v^{\mathcal{I}}, w^{\mathcal{I}} \rangle \in R^{\mathcal{I}}$ if $\mathcal{F}$ has an arc $v \to w$ and $R \in \mathcal{L}(\langle v, w \rangle)$, and (iii) $v^{\mathcal{I}} \neq w^{\mathcal{I}}$ if $v \not\approx w \in \mathcal{F}$.

Clearly, the models of the initial completion forest $\mathcal{F}_K$ and of K coincide, and thus $\mathcal{F}_K$ semantically represents K. Then, each time an expansion rule is applied, all models are preserved in some resulting forest. As a consequence, it holds that for each model $\mathcal{I}$ of K, there exists some $\mathcal{F} \in \mathsf{ccf}_n(\mathbb{F}_K)$ and a model of $\mathcal{F}$ which extends $\mathcal{I}$ (for any $n \geq 0$). Since the set of n-complete and clash-free forests for K semantically

[2] If $\mathcal{A} = \emptyset$, then $\mathcal{F}_K$ contains a single node a with $\mathcal{L}(a) = \mathsf{gcon}(K, \mathcal{C}_q)$.

captures K (modulo new individuals), we can transfer query entailment $K \models Q$ to logical consequence of Q from completion forests as follows. For any completion forest $\mathcal{F}$ and CQ Q, let $\mathcal{F} \models Q$ denote that $\mathcal{I} \models Q$ for every model $\mathcal{I}$ of $\mathcal{F}$.

Proposition 1 *Let Q be a CQ in which all roles are simple and let $n \geq 0$ be arbitrary. $K \models Q$ iff $\mathcal{F} \models Q$ for each $\mathcal{F} \in \mathsf{ccf}_n(\mathbb{F}_K)$.*

Now we will show that, if n is sufficiently large, we can decide $\mathcal{F} \models Q$ for an $\mathcal{F} \in \mathsf{ccf}_n(\mathbb{F}_K)$ by syntactically mapping the query Q into $\mathcal{F}$. We say that Q can be *mapped into* $\mathcal{F}$, denoted $Q \hookrightarrow \mathcal{F}$, if there is a mapping μ from the variables and individuals in $\mathsf{varind}(Q)$ into the nodes of $\mathcal{F}$, such that (i) for each individual a, $\mu(a) = a$; (ii) for each atom $C(x)$ in Q, $C \in \mathcal{L}(\mu(x))$; and (iii) for each atom $R(x, y)$ in Q, there is an arc $\mu(x) \to \mu(y)$ whose label contains R' or an arc $\mu(y) \to \mu(x)$ whose label contains the inverse of R' for some $R' \sqsubseteq^* R$.

Example 6 *$Q_1 \hookrightarrow \mathcal{F}_2$ holds, as witnessed by the mapping $\mu(x) = a$, $\mu(y) = v_2$, and $\mu(z) = v_1$. Note that there is no mapping of Q_2 into $\mathcal{F}_2$ satisfying the above conditions.*

It is clear that if Q can be mapped to $\mathcal{F}$ via μ, then Q is satisfied in each model $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ of $\mathcal{F}$ by assigning to each variable x in Q the value of its image $\mu(x)^{\mathcal{I}}$. Thus $Q \hookrightarrow \mathcal{F}$ implies $\mathcal{F} \models Q$. To prove that the converse also holds, we have to show that if n is large enough, a mapping of Q into $\mathcal{F} \in \mathsf{ccf}_n(\mathbb{F}_K)$ can be constructed from a distinguished canonical model of $\mathcal{F}$.

The *canonical model* $\mathcal{I}_{\mathcal{F}}$ of $\mathcal{F}$ is constructed by unraveling the forest $\mathcal{F}$ in the standard way, where the blocked nodes act like 'loops'. Its domain comprises the set of all paths from some root in $\mathcal{F}$ to some node of $\mathcal{F}$ (thus, it can be infinite). Note that in order for $\mathcal{I}_{\mathcal{F}}$ to be a model, $\mathcal{F}$ must be in $\mathsf{ccf}_n(\mathbb{F}_K)$ for some $n \geq 1$. The formal definition of $\mathcal{I}_{\mathcal{F}}$ is then straightforward yet complex, and we must refer to the extended report [21] for the details. Instead, we provide an example.

Example 7 *By unraveling $\mathcal{F}_2$, we obtain a model $\mathcal{I}_{\mathcal{F}_2}$ that has as domain the infinite set of paths from a to each v_i. Note that a path actually comprises a sequence of pairs of nodes, in order to witness the loops introduced by blocked variables. When a node is not blocked, like v_1, the pair $\frac{v_1}{v_1}$ is added to the path. Since $T^2_{v_1}$ tree-blocks $T^2_{v_7}$, every time a path reaches v_{11}, which is a leaf of a blocked tree, we add $\frac{v_5}{v_{11}}$ to the path and 'loop' back to the successors of v_5. In this way, we obtain the following infinite set of paths:*

$$
\begin{array}{ll}
p_0 = [\frac{a}{a}], & p_7 = [\frac{a}{a}, \frac{v_1}{v_1}, \frac{v_3}{v_3}, \frac{v_5}{v_5}, \frac{v_7}{v_7}], \\
p_1 = [\frac{a}{a}, \frac{v_1}{v_1}], & p_8 = [\frac{a}{a}, \frac{v_1}{v_1}, \frac{v_3}{v_3}, \frac{v_5}{v_5}, \frac{v_8}{v_8}], \\
p_2 = [\frac{a}{a}, \frac{v_2}{v_2}], & p_9 = [\frac{a}{a}, \frac{v_1}{v_1}, \frac{v_3}{v_3}, \frac{v_5}{v_5}, \frac{v_7}{v_7}, \frac{v_9}{v_9}], \\
p_3 = [\frac{a}{a}, \frac{v_1}{v_1}, \frac{v_3}{v_3}], & p_{10} = [\frac{a}{a}, \frac{v_1}{v_1}, \frac{v_3}{v_3}, \frac{v_5}{v_5}, \frac{v_7}{v_7}, \frac{v_{10}}{v_{10}}], \\
p_4 = [\frac{a}{a}, \frac{v_1}{v_1}, \frac{v_4}{v_4}], & p_{11} = [\frac{a}{a}, \frac{v_1}{v_1}, \frac{v_3}{v_3}, \frac{v_5}{v_5}, \frac{v_7}{v_7}, \frac{v_9}{v_9}, \frac{v_5}{v_{11}}], \\
p_5 = [\frac{a}{a}, \frac{v_1}{v_1}, \frac{v_3}{v_3}, \frac{v_5}{v_5}], & p_{12} = [\frac{a}{a}, \frac{v_1}{v_1}, \frac{v_3}{v_3}, \frac{v_5}{v_5}, \frac{v_7}{v_7}, \frac{v_9}{v_9}, \frac{v_{12}}{v_{12}}], \\
p_6 = [\frac{a}{a}, \frac{v_1}{v_1}, \frac{v_3}{v_3}, \frac{v_6}{v_6}], & p_{13} = [\frac{a}{a}, \frac{v_1}{v_1}, \frac{v_3}{v_3}, \frac{v_5}{v_5}, \frac{v_7}{v_7}, \frac{v_9}{v_9}, \frac{v_5}{v_{11}}, \frac{v_7}{v_7}], \\
 & p_{14} = [\frac{a}{a}, \frac{v_1}{v_1}, \frac{v_3}{v_3}, \frac{v_5}{v_5}, \frac{v_7}{v_7}, \frac{v_9}{v_9}, \frac{v_5}{v_{11}}, \frac{v_8}{v_8}], \\
 & \vdots
\end{array}
$$

This set of paths is the domain of $\mathcal{I}_{\mathcal{F}_2}$. The extension of each concept C is determined by the set all p_i such that C occurs in the label of the last node in p_i. For the extension of each role R, we consider the pairs $\langle p_i, p_j\rangle$ such that the last node in p_j is an R-successor of p_i (the extension of the roles in $\mathbf{R}_+$ are transitively expanded). Therefore p_0, p_1, p_3, ... are in $A^{\mathcal{I}_{\mathcal{F}_2}}$, and $\langle p_0, p_1\rangle$, $\langle p_1, p_3\rangle$, $\langle p_3, p_5\rangle$, $\langle p_5, p_7\rangle$, ... are all in $P_1^{\mathcal{I}_{\mathcal{F}_2}}$.

In the following, let n_Q denote the number of role atoms in Q, and let $n \geq n_Q$. For any forest $\mathcal{F}$ that is n-complete, a mapping μ of Q into $\mathcal{F}$ can be obtained from any mapping σ of $\mathsf{varind}(Q)$ into $\mathcal{I}_{\mathcal{F}}$ satisfying Q. If we see the image of Q under σ as a graph G (considering only the edges that correspond to simple roles), then the length of a path connecting the images $\sigma(x)$ and $\sigma(y)$ of any two variables x and y in Q will be at most n_Q. Since $\mathcal{F}$ contains at least two non-overlapping trees of size n, it is big enough to ensure that for each path in G there is an isomorphic one in $\mathcal{F}$.

Proposition 2 *Let Q be a CQ in which all roles are simple, let $\mathcal{F} \in \mathsf{ccf}_n(\mathbb{F}_K)$, with $n \geq n_Q$, and let $\mathcal{I}_{\mathcal{F}} \models Q$. Then $Q \hookrightarrow \mathcal{F}$.*

Example 8 *$\mathcal{I}_{\mathcal{F}_2}$ models Q_1, as witnessed by the substitution $\sigma(x) = p_7$, $\sigma(y) = p_9$ and $\sigma(z) = p_{10}$. From it we can obtain the mapping $\mu(x) = v_7$, $\mu(y) = v_9$ and $\mu(z) = v_{10}$, which shows that $Q_1 \hookrightarrow \mathcal{F}_2$. Note that $n_{Q_1} = 2$ and the image of Q_1 under σ has no paths of length > 2.*

The results given above can be extended straightforwardly to a UCQ U. As before, we will use $\mathcal{F} \models U$ to denote that $\mathcal{F}$ semantically entails U (i.e., every model of $\mathcal{F}$ is a model of U), and $U \hookrightarrow \mathcal{F}$ to denote syntactical mappability, which is defined as $Q_i \hookrightarrow \mathcal{F}$ for some Q_i in U. We already know that to decide $K \models U$ it is sufficient to verify whether $\mathcal{F} \models U$ for every $\mathcal{F}$ in $\mathsf{ccf}_n(\mathbb{F}_K)$ for an arbitrary $n \geq 0$ (in fact, Proposition 1 holds for any kind of query). It is only left to prove that for a suitable n, $\mathcal{F} \models U$ can be effectively reduced to $U \hookrightarrow \mathcal{F}$.

Again, one direction is trivial. If $U \hookrightarrow \mathcal{F}$, then by definition there is some Q_i in U such that $Q_i \hookrightarrow \mathcal{F}$, and as this implies $\mathcal{F} \models Q_i$, we also have that $\mathcal{F} \models U$.

Example 9 *Since $Q_1 \hookrightarrow \mathcal{F}_2$ (see Examples 6 and 8), we have $U \hookrightarrow \mathcal{F}_2$. $\mathcal{F}_2 \models Q_1$ implies $\mathcal{F}_2 \models U$.*

The other direction is also quite straightforward. Consider n_U as the maximal n_{Q_i} for all Q_i in U. For each $\mathcal{F} \in \mathsf{ccf}_n(\mathbb{F}_K)$, with $n \geq n_U$, if $\mathcal{I}_{\mathcal{F}} \models U$, then $\mathcal{I}_{\mathcal{F}} \models Q_i$ for some Q_i in U. Since $n \geq n_U \geq n_{Q_i}$, by Proposition 2 we know that $Q_i \hookrightarrow \mathcal{F}$ and then $U \hookrightarrow \mathcal{F}$. Thus $\mathcal{I}_{\mathcal{F}} \models U$ implies $U \hookrightarrow \mathcal{F}$ as well.

Finally, we establish our key result: answering $K \models U$ for a UCQ U reduces to finding a mapping of U into every $\mathcal{F} \in \mathsf{ccf}_n(\mathbb{F}_K)$ for any $n \geq n_U$.

Theorem 3 *Let U be a UCQ in which all roles are simple and let $n \geq n_U$. Then $K \models U$ iff $U \hookrightarrow \mathcal{F}$ holds for each $\mathcal{F} \in \mathsf{ccf}_n(\mathbb{F}_K)$.*

4 Complexity of Query Answering

In the following, $\|K, Q\|$ denotes the total size of the string encoding a given knowledge base K and a query Q. As shown in [21], for a CQ Q, branching in each variable tree in a completion forest $\mathcal{F} \in \mathsf{ccf}_n(\mathbb{F}_K)$ is polynomially bounded in $\|K, Q\|$, and the maximal depth of a variable is double exponential in $\|K, Q\|$ if n is polynomial in $\|K, Q\|$. Therefore, $\mathcal{F}$ has at most triple exponentially many nodes. Since each rule can be applied only polynomially often to a node, the expansion of the initial completion forest $\mathcal{F}_K$ into some $\mathcal{F} \in \mathsf{ccf}_n(\mathbb{F}_K)$ terminates in nondeterministic triple exponential time in $\|K, Q\|$ for $n = n_Q$.

Theorem 4 *Given a $\mathcal{SHIQ}$ knowledge base K and a union of conjunctive queries U in which all roles are simple, deciding whether $K \models U$ is in* co-3NExpTime.

Proof (Sketch). It is sufficient to check for every $\mathcal{F} \in \mathsf{ccf}_{n_U}(\mathbb{F}_K)$ whether $U \hookrightarrow \mathcal{F}$, i.e., $Q_i \hookrightarrow \mathcal{F}$ for some CQ Q_i in U. The size of $\mathcal{F}$ is at most triple exponential in $\|K, Q_{i^*}\|$, where Q_{i^*} is such that $n_{Q_{i^*}} = n_U$, thus also in $\|K, U\|$. Furthermore, $Q_i \hookrightarrow \mathcal{F}$ can be checked by naive methods in triple exponential time in $\|K, Q_{i^*}\|$ and thus in $\|K, U\|$ as well. (We stress that this test is NP-hard even for fixed $\mathcal{F}$.) □

Notice that the result holds for binary encoding of number restrictions in K. An exponential drop results for unary encoding if Q is fixed.

Under data complexity, U and all components of $K = \langle \mathcal{T}, \mathcal{R}, \mathcal{A} \rangle$ except for the ABox $\mathcal{A}$ are fixed. Therefore, n_U is constant. Thus every completion forest $\mathcal{F} \in \mathsf{ccf}_n(\mathbb{F}_K)$ has *linearly* many nodes in $|\mathcal{A}|$, and any expansion of $\mathcal{F}_K$ terminates in polynomial time. Furthermore, deciding whether $U \hookrightarrow \mathcal{F}$ is polynomial in the size of $\mathcal{F}$ by simple methods. As a consequence,

Theorem 5 *For a knowledge base K in $\mathcal{SHIQ}$ and a union of conjunctive queries U in which all roles are simple, deciding $K \models U$ is in* coNP *w.r.t. data complexity.*

Matching coNP-hardness follows from the respective result for $\mathcal{ALE}$ [24], which has been extended later to DLs even less expressive than $\mathcal{AL}$ [7]. Thus we obtain the following main result.

Theorem 6 *On knowledge bases in any DL from $\mathcal{AL}$ to $\mathcal{SHIQ}$, answering unions of conjunctive queries in which all roles are simple is* coNP*-complete w.r.t. data complexity.*

This result not only exactly characterizes the data complexity of UCQs for a range of DLs, but also extends two previous coNP-completeness results w.r.t. data complexity which are not obvious: the result on CQs over $\mathcal{ALCNR}$ given in [20] to $\mathcal{SHIQ}$, and the result in [18] for atomic queries in $\mathcal{SHIQ}$ to UCQs.

5 Discussion and Conclusion

By the correspondence between query containment and query answering [1], our algorithm can also be applied to decide containment of two UCQs over a $\mathcal{SHIQ}$ knowledge

base. Also, the technique is applicable to the DL $\mathcal{SHOIQ}$, which extends $\mathcal{SHIQ}$ with *nominals*, i.e., concepts denoting single individuals, by tuning of the $\mathcal{SHOIQ}$ tableaux rules [14]. It remains open whether the proposed technique can be applied to even more expressive logics, for example, containing reflexive-transitive closure in the TBox (in the style of PDL), or to more expressive query languages.

Several issues remain for further work. The restriction of the query languages to allow only simple roles is also adopted in [18]. To the best of our knowledge, it is yet unknown whether conjunctive query answering remains decidable in $\mathcal{SHIQ}$ when this constraint is relaxed. It remains unclear whether the algorithm which we have presented here can be exploited, since the presence of transitive roles imposes difficulties in establishing a bound on the depth of completion forests which need to be considered for answering a given query. Also the combined complexity of UCQs remains for further study. As follows from [17], it is in 2ExpTime, and thus Theorem 4 gives not a tight bound. However, we can use a more relaxed blocking condition in our algorithm, where the root of the blocked tree is not necessarily an ancestor of the blocked one. This lowers the worst-case size of a forest exponentially, leading to a co-2NExpTime upper bound, which is not optimal either. We want to point out that such blocking could be similarly used in the standard tableau algorithms for satisfiability checking, but might be inconvenient from an implementation perspective.

References

[1] S. Abiteboul and O. Duschka. Complexity of answering queries using materialized views. In *Proc. of the 17th ACM SIGACT SIGMOD SIGART Symp. on Principles of Database Systems (PODS'98)*, pages 254–265, 1998.

[2] F. Baader, D. Calvanese, D. McGuinness, D. Nardi, and P. F. Patel-Schneider, editors. *The Description Logic Handbook: Theory, Implementation and Applications*. Cambridge University Press, 2003.

[3] F. Baader and U. Sattler. An overview of tableau algorithms for description logics. *Studia Logica*, 69(1):5–40, 2001.

[4] A. Borgida and R. J. Brachman. Conceptual modeling with description logics. In Baader et al. [2], chapter 10, pages 349–372.

[5] D. Calvanese and G. De Giacomo. Expressive description logics. In Baader et al. [2], chapter 5, pages 178–218.

[6] D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, and R. Rosati. DL-Lite: Tractable description logics for ontologies. In *Proc. of the 20th Nat. Conf. on Artificial Intelligence (AAAI 2005)*, pages 602–607, 2005.

[7] D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, and R. Rosati. Data complexity of query answering in description logics. In *Proc. of the 10th Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR 2006)*, 2006.

[8] D. Calvanese, G. De Giacomo, and M. Lenzerini. On the decidability of query containment under constraints. In *Proc. of the 17th ACM SIGACT SIGMOD SIGART Symp. on Principles of Database Systems (PODS'98)*, pages 149–158, 1998.

[9] D. Calvanese, G. De Giacomo, and M. Lenzerini. Answering queries using views over description logics knowledge bases. In *Proc. of the 17th Nat. Conf. on Artificial Intelligence (AAAI 2000)*, pages 386–391, 2000.

[10] F. M. Donini, M. Lenzerini, D. Nardi, and A. Schaerf. Deduction in concept languages: From subsumption to instance checking. *J. of Logic and Computation*, 4(4):423–452, 1994.

[11] J. Heflin and J. Hendler. A portrait of the Semantic Web in action. *IEEE Intelligent Systems*, 16(2):54–59, 2001.

[12] I. Horrocks, P. F. Patel-Schneider, and F. van Harmelen. From SHIQ and RDF to OWL: The making of a web ontology language. *J. of Web Semantics*, 1(1):7–26, 2003.

[13] I. Horrocks and U. Sattler. A description logic with transitive and inverse roles and role hierarchies. *J. of Logic and Computation*, 9(3):385–410, 1999.

[14] I. Horrocks and U. Sattler. A tableaux decision procedure for $\mathcal{SHOIQ}$. In *Proc. of the 19th Int. Joint Conf. on Artificial Intelligence (IJCAI 2005)*, pages 448–453, 2005.

[15] I. Horrocks, U. Sattler, and S. Tobies. Reasoning with individuals for the description logic $\mathcal{SHIQ}$. In D. McAllester, editor, *Proc. of the 17th Int. Conf. on Automated Deduction (CADE 2000)*, volume 1831 of *Lecture Notes in Computer Science*, pages 482–496. Springer, 2000.

[16] I. Horrocks and S. Tessaris. A conjunctive query language for description logic ABoxes. In *Proc. of the 17th Nat. Conf. on Artificial Intelligence (AAAI 2000)*, pages 399–404, 2000.

[17] U. Hustadt, B. Motik, and U. Sattler. A decomposition rule for decision procedures by resolution-based calculi. In *Proc. of the 11th Int. Conf. on Logic for Programming, Artificial Intelligence and Reasoning (LPAR 2004)*, pages 21–35, 2004.

[18] U. Hustadt, B. Motik, and U. Sattler. Data complexity of reasoning in very expressive description logics. In *Proc. of the 19th Int. Joint Conf. on Artificial Intelligence (IJCAI 2005)*, pages 466–471, 2005.

[19] M. Lenzerini. Data integration: A theoretical perspective. In *Proc. of the 21st ACM SIGACT SIGMOD SIGART Symp. on Principles of Database Systems (PODS 2002)*, pages 233–246, 2002.

[20] A. Y. Levy and M.-C. Rousset. Combining Horn rules and description logics in CARIN. *Artificial Intelligence*, 104(1–2):165–209, 1998.

[21] M. M. Ortiz de la Fuente, D. Calvanese, and T. Eiter. Data complexity of answering unions of conjunctive queries in $\mathcal{SHIQ}$. Technical report, Faculty of Computer Science, Free University of Bozen-Bolzano, Mar. 2006. Available at `http://www.inf.unibz.it/~calvanese/papers/orti-calv-eite-TR-2006-03.pdf%`.

[22] P. Patel-Schneider, P. Hayes, and I. Horrocks. OWL Web Ontology Language semantics and abstract syntax – W3C recommendation. Technical report, World Wide Web Consortium, Feb. 2004. Available at `http://www.w3.org/TR/owl-semantics/`.

[23] A. Schaerf. On the complexity of the instance checking problem in concept languages with existential quantification. *J. of Intelligent Information Systems*, 2:265–278, 1993.

[24] A. Schaerf. *Query Answering in Concept-Based Knowledge Representation Systems: Algorithms, Complexity, and Semantic Issues*. PhD thesis, Dipartimento di Informatica e Sistemistica, Università di Roma "La Sapienza", 1994.

[25] S. Tobies. *Complexity Results and Practical Algorithms for Logics in Knowledge Representation*. PhD thesis, LuFG Theoretical Computer Science, RWTH-Aachen, Germany, 2001.

[26] M. Y. Vardi. The complexity of relational query languages. In *Proc. of the 14th ACM SIGACT Symp. on Theory of Computing (STOC'82)*, pages 137–146, 1982.

PSpace Automata for Description Logics

Jan Hladik*
Rafael Peñaloza†

Abstract

Tree automata are often used for satisfiability testing in the area of description logics, which usually yields ExpTime complexity results. We examine conditions under which this result can be improved, and we define two classes of automata, called *segmentable* and *weakly-segmentable*, for which emptiness can be decided using space logarithmic in the size of the automaton (and thus polynomial in the size of the input). The usefulness of segmentable automata is demonstrated by reproving the known PSpace result for satisfiability of $\mathcal{ALC}$ concepts with respect to acyclic TBoxes.

1 Introduction

Tableau- and automata-based algorithms are two mechanisms which are widely used for testing satisfiability in description logics (DLs). Aside from considerations regarding implementation (most of the efficient implementations are tableau-based) and elegance (tableaus for expressive logics require a blocking condition to ensure termination), there is also a difference between the complexity results which can be obtained "naturally", i.e. without using techniques with the sole purpose of remaining in a specific complexity class. With tableaus, the natural complexity class is usually NExpTime, e.g. for $\mathcal{SHIQ}$ [HST00, BS01] or $\mathcal{SHIQ}(\mathcal{D})$ [Lut04], although this result is not optimal in the former case. With automata, one usually obtains an ExpTime result, e.g. for $\mathcal{ALC}$ with general TBoxes [Sch94].

Previously, we examined which properties of a NExpTime tableau algorithm make sure that the logic is decidable in ExpTime, and we defined a class of tableau algorithms for which an ExpTime automata algorithm can be automatically derived [BHLW03]. For ExpTime automata, a frequently used

*Automata Theory, TU Dresden (jan.hladik@tu-dresden.de)
†Intelligent Systems, Uni Leipzig (rpenalozan@yahoo.com)

method to obtain a lower complexity class is testing emptiness of the language accepted by the automaton *on the fly*, i.e. without keeping the entire automaton in the memory at the same time. Examples for DLs can be found in [HST00] for $\mathcal{SI}$ and in [BN03] for $\mathcal{ALCN}$. Furthermore, the *inverse method* [Vor01] for the modal logic **K** can be regarded as an optimised emptiness test of the corresponding automaton [BT01], and also the "bottom up" version of the *binary decision diagram* based satisfiability test presented in [PSV02] can be considered as an on-the-fly emptiness test.

In this paper, our aim is to generalise these results by finding properties of ExpTime automata algorithms which guarantee that the corresponding logic can be decided in PSpace, and to develop a framework which can be used to automatically obtain PSpace results for new logics.

2 Preliminaries

We will first define the automata which in the following will be used to decide the satisfiability problem for DLs. These automata operate on infinite k-ary trees, for which the root node is identified by ε and the i-th successor of a node n is identified by $n \cdot i$ for $1 \leq i \leq k$. Thus, the set of all nodes is $\{1, \ldots, k\}^*$, and in the case of labelled trees, we will refer to the labelling of the node n in the tree t by $t(n)$.

The following definition of automata does not include an alphabet for labelling the tree nodes, because in order to decide the emptiness problem, we are only interested in the *existence* of a model and not in the labelling of its nodes.

Definition 1 (Automaton, run, accepted language.) A *Büchi automaton* over k-ary trees is a tuple (Q, Δ, I, F), where Q is a finite set of states, $\Delta \subseteq Q^{k+1}$ is the transition relation, $I \subseteq Q$ is the set of initial states, and $F \subseteq Q$ is the set of final states.

A *looping automaton* is a Büchi automaton where all states are accepting, i.e. $F = Q$. For simplicity, it will be written (Q, Δ, I).

A *run* of an automaton $\mathcal{A} = (Q, \Delta, I, F)$ on a k-ary tree t is a labelled k-ary tree r such that $r(\varepsilon) \in I$ and, for all $w \in \{1, \ldots, k\}^*$, it holds that $(r(w), r(w \cdot 1), \ldots, r(w \cdot k)) \in \Delta$. A run r is *accepting* if every path of r contains a final state infinitely often.

The *language accepted by* $\mathcal{A}$, $\mathcal{L}(\mathcal{A})$, is the set of all trees t such that there exists an accepting run of $\mathcal{A}$ on t.

For the DL $\mathcal{ALC}$ [SS91], it is well-known that looping automata can be used to decide satisfiability of a concept C: we define a looping automaton $\mathcal{A}_C = (Q, \Delta, I)$, where the set of states Q consists of (propositionally expanded and clash-free) sets of subformulas of C, and the transition relation Δ ensures

that the successor states of a state q contain those concepts which are required by the universal and existential concepts in q. Since $\#Q$, the cardinality of Q, is exponential in the size of C and $\mathcal{T}$ (because it is bounded by the number of sets of subformulas of C and $\mathcal{T}$) and the emptiness test is linear [BT01, VW86] in the cardinality of Q, this yields an ExpTime algorithm. This result is optimal for $\mathcal{ALC}$ with general TBoxes, but not for $\mathcal{ALC}$ with acyclic (or empty) TBoxes.

In the following, we will define this algorithm in detail. We assume that *acyclic* TBoxes are sets of concept definitions $A_i \doteq C_i$, for concept names A_i and concept terms C_i, where there is at most one definition for a concept name and there is no sequence of concept definitions $A_1 \doteq C_1, \ldots, A_n \doteq C_n$ such that C_i contains A_{i+1} for $1 \leq i < n$ and C_n contains A_1. In contrast, *general* TBoxes can additionally contain general concept inclusion axioms (GCIs) of the kind $C_i \sqsubseteq D_i$ for arbitrary concept terms C_i and D_i. For the sake of simplicity, we will assume that all concepts are in *negation normal form (NNF)*, i.e. negation appears only directly before concept names. All $\mathcal{ALC}$ concepts can be transformed into NNF in linear time using de Morgan's laws and their analogues for universal and existential formulas. We will denote the NNF of a concept C by $\mathsf{nnf}(C)$ and $\mathsf{nnf}(\neg C)$ by $\backsim C$.

The data structures that will serve as models for $\mathcal{ALC}$ concepts are *Hintikka trees*, a special kind of trees whose nodes are labelled with propositionally expanded and clash-free sets of $\mathcal{ALC}$ concepts.

Definition 2 (Sub-concept, Hintikka set, Hintikka tree.) Let C be an $\mathcal{ALC}$ concept term. The set of *sub-concepts of C,* $\mathsf{sub}(C)$*,* is the minimal set S which contains C and has the following properties: if S contains $\neg A$ for a concept name A, then $A \in S$; if S contains $D \sqcup E$ or $D \sqcap E$, then $\{D, E\} \subseteq S$; if S contains $\exists r.D$ or $\forall r.D$, then $D \in S$.

For a TBox $\mathcal{T}$, $\mathsf{sub}(C, \mathcal{T})$ is defined as follows:

$$\mathsf{sub}(C) \quad \cup \bigcup_{A \doteq D \in \mathcal{T}} (A \cup \mathsf{sub}(D) \cup \mathsf{sub}(\backsim D)) \quad \cup \bigcup_{C \sqsubseteq D \in \mathcal{T}} \mathsf{sub}(\backsim C \sqcup D)$$

A set $H \subseteq \mathsf{sub}(C, \mathcal{T})$ is called a *Hintikka set for C* if the following three conditions are satisfied: if $D \sqcap E \in H$, then $\{D, E\} \subseteq H$; if $D \sqcup E \in H$, then $\{D, E\} \cap H \neq \emptyset$; there is no concept name A with $\{A, \neg A\} \subseteq H$.

For a TBox $\mathcal{T}$, a Hintikka set S is called $\mathcal{T}$*-expanded* if for every GCI $C \sqsubseteq D \in \mathcal{T}$, it holds that $\backsim C \sqcup D \in S$, and for every concept definition $A \doteq C \in \mathcal{T}$, it holds that if $A \in S$ then $C \in S$ and if $\neg A \in S$ then $\backsim C \in S$. We will refer to this technique of handling definitions as *lazy unfolding* because, in contrast to GCIs, we use the definition only if A or $\neg A$ is explicitly present in a node label.

For a concept term C and TBox $\mathcal{T}$, fix an ordering of the existential concepts in $\mathsf{sub}(C, \mathcal{T})$ and let $\varphi : \{\exists r.D \in \mathsf{sub}(C, \mathcal{T})\} \to \{1, \ldots, k\}$ be the corresponding

ordering function. Then the tuple $(S, S_1, \ldots, S_k)$ is called $C, \mathcal{T}$*-compatible* if, for every existential formula $\exists r.D \in \mathsf{sub}(C, \mathcal{T})$, it holds that if $\exists r.D \in S$, then $S_{\varphi(\exists r.D)}$ contains D and every concept E_i for which there is a universal formula $\forall r.E_i \in S$.

A k-ary tree t is called a *Hintikka tree for C and $\mathcal{T}$* if, for every node $n \in \{1, \ldots, k\}^*$, $t(n)$ is a $\mathcal{T}$-expanded Hintikka set and the tuple $(t(n), t(n{\cdot}1), \ldots, t(n{\cdot}k))$ is $C, \mathcal{T}$-compatible.

Note that in a Hintikka tree t, the k-th successor of a node n stands for the individual satisfying the k-th existential formula D if $D \in t(n)$. We can use Hintikka trees to test the satisfiability in $\mathcal{ALC}$:

Lemma 3 An $\mathcal{ALC}$ concept term C is satisfiable w.r.t. a TBox $\mathcal{T}$ iff there is a $C, \mathcal{T}$-compatible Hintikka tree t with $C \in t(\varepsilon)$.

Proof sketch. The proof is similar to the one presented for $\mathcal{ALC}^{\neg}$ in [LS00] where the "if" direction is shown by defining a model $(\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ from a Hintikka tree in the following way:

- $\Delta^{\mathcal{I}} := \{n \in \{1, \ldots, k\}^* \mid n = \varepsilon$ or $n = m \cdot \varphi(\exists r.D)$ for some m with $\exists r.D \in t(m)\}$;
- for a role name r, $r^{\mathcal{I}} := \{(n, n \cdot i) \mid \exists r.C \in t(n)$ and $\varphi(\exists r.C) = i\}$;
- for a concept name A, $A^{\mathcal{I}} := \{n \mid A \in t(n)\}$;
- the function $\cdot^{I}$ is extended to concept terms in the natural way.

For our Hintikka trees, we have to modify the definition of $A^{\mathcal{I}}$ for concept names A in order to deal with TBoxes. For a GCI $C \sqsubseteq D$, it follows immediately from the definition of $\mathcal{T}$-expanded that a node whose label contains C also contains D. However, concept definitions need special consideration because due to the lazy unfolding of $A \doteq C$, a node label might contain C but not A, thus we have to modify the definition of $A^{\mathcal{I}}$. To this end, we define a hierarchy $\prec$ on concept names in such a way that if a concept name A appears in the definition of B, then $A \prec B$. As the concept definitions are acyclic, this hierarchy is well-founded. When we define the interpretation of concept names, we start with the lowest ones in the hierarchy, i.e. the primitive concepts, and define $A^{\mathcal{I}} := \{n \in \Delta^{\mathcal{I}} \mid A \in t(n)\}$. Then we move gradually up in the hierarchy and define, for a defined concept $B \doteq C$, $B^{\mathcal{I}} = \{n \in \Delta^{\mathcal{I}} \mid B \in t(n)\} \cup \{n \in \Delta^{\mathcal{I}} \mid n \in C^{\mathcal{I}}\}$. This is well-defined because the interpretation of all concept names appearing in C has already been defined.

Our Hintikka trees differ from those in [LS00] in that, if an existential formula $\exists r.D$ is not present in a node n, we do not require that $n \cdot \varphi(\exists r.D)$ is labelled with $\emptyset$. However, it can still be shown that if a concept term C is contained in

the label of a node n of the Hintikka tree, then $n \in C^{\mathcal{I}}$, because in the definition of the interpretation for the role names, we consider only the successors that will satisfy the existential restrictions of a node, and pay no attention to any other possible successors. The universal restrictions are then immediately satisfied by the definition of C,$\mathcal{T}$-compatible.

The "only-if" direction does not require significant modifications, because if there is a model for a concept C and a TBox $\mathcal{T}$, the node labels of the Hintikka tree constructed as in [LS00] can easily be extended to reflect the constraints imposed by the TBox. ■

With this result, we can use automata operating on Hintikka trees to test for the existence of models. As mentioned before, we can omit the labelling of the tree, since we are only interested in the existence of a model and all relevant information to answer this question is kept in the transition relation.

Definition 4 (Automaton $\mathcal{A}_{C,\mathcal{T}}$.) For a concept C and a TBox $\mathcal{T}$, let k be the number of existential formulas in $\mathsf{sub}(C, \mathcal{T})$. Then the looping automaton $\mathcal{A}_{C,\mathcal{T}} = (Q, \Delta, I)$ is defined as follows: $Q = \{S \subseteq \mathsf{sub}(C, \mathcal{T}) \mid S \text{ is a } \mathcal{T}\text{-expanded Hintikka set}\}$; $\Delta = \{(S, S_1, \ldots, S_k) \mid (S, S_1, \ldots, S_k) \text{ is } C, \mathcal{T}\text{-compatible}\}$; $I = \{S \in Q \mid C \in S\}$.

Using $\mathcal{A}_{C,\mathcal{T}}$, we can reduce the satisfiability problem for $\mathcal{ALC}$ to the (non-) emptiness problem of $\mathcal{L}(\mathcal{A}_{C,\mathcal{T}})$. Since these results are well known, they will not be formally proved.

Theorem 5 The language accepted by the automaton $\mathcal{A}_{C,\mathcal{T}}$ is empty iff C is unsatisfiable w.r.t. $\mathcal{T}$.

Corollary 6 Satisfiability of $\mathcal{ALC}$ concepts w.r.t. general TBoxes is decidable in ExpTime.

For general TBoxes, this complexity bound is tight [Spa93], but in the special case of an empty or acyclic TBoxes, it can be improved to PSpace. Usually, this is proved using a tableau algorithm, but in the next section we will show how the special properties of acyclic TBoxes can be used to perform the emptiness test of the automaton with logarithmic space.

3 Segmentable automata

In this section we will show how the space efficiency for the construction and the emptiness test of the automaton can be improved under specific conditions. The idea is to define a hierarchy of states and ensure that the level of the state decreases with every transition. In Section 4 we will then show how the role depth of concepts can be used to define this hierarchy.

```
guess an initial state q ∈ I
if there is a transition from q then
   guess a transition (q, q_1, ..., q_k) ∈ Δ
else
   return "empty"
end if
push (SQ, (q_1, ..., q_k)); push (SN, 0)
while SN is not empty do
   (q_1, ..., q_k) := pop (SQ)
   n := pop (SN) + 1
   if n ≤ k then
      push(SQ,(q_1, ..., q_k))
      push(SN,n)
      if q_n ∉ Q_0 then
         if there is a transition from q_n then
            guess a transition (q_n, q'_1, ..., q'_k)
         else
            return "empty"
         end if
         push(SQ,(q'_1, ..., q'_k))
         push(SN,0)
      end if
   end if
end while
return "not empty"
```

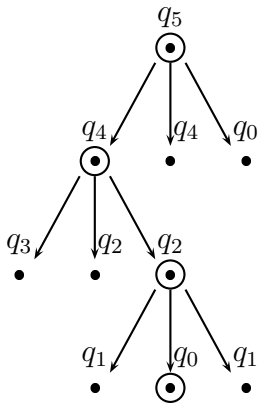
Figure 1: Emptiness test for segmentable automata

Definition 7 (Q_0-looping, m-segmentable.) Let $\mathcal{A} = (Q, \Delta, I, F)$ be a Büchi automaton over k-ary trees and $Q_0 \subseteq F$. We call $\mathcal{A}$ Q_0-*looping* if for every $q \in Q_0$ there exists a set of states $\{q_1, \ldots, q_k\} \subseteq Q_0$ such that $(q, q_1, \ldots, q_k) \in \Delta$.

An automaton $\mathcal{A} = (Q, \Delta, I, F)$ is called m-*segmentable* if there exists a partition $Q_0, Q_1, \ldots, Q_m$ of Q such that $\mathcal{A}$ is Q_0-looping and, for every $(q, q_1, \ldots, q_k) \in \Delta$, it holds that if $q \in Q_n$, then $q_i \in Q_{<n}$ for $1 \leq i \leq k$, where $Q_{<n}$ denotes $Q_0 \cup \bigcup_{j=1}^{n-1} Q_j$.

Note that it follows immediately from this definition that for every element q of Q_0 there exists an infinite tree with q as root which is accepted by $\mathcal{A}$. The hierarchy $Q_m, \ldots, Q_0$ ensures that Q_0 is reached eventually.

Our algorithm performing the emptiness test for m-segmentable Büchi automata is shown in Figure 1. Essentially, we perform a depth-first traversal of a run. Since $\mathcal{A}$ is m-segmentable, we do not have to go to a depth larger than



SQ	SN
(q_4, q_4, q_0)	1
(q_3, q_2, q_2)	3
(q_1, q_0, q_1)	1

Figure 2: state of SQ, SN and the associated run, at an iteration of the algorithm

m. Moreover, since the different branches of the tree are independent, we only have to keep one of them in memory at a time. Note that the construction of the automaton is interleaved with the emptiness test, so we also never keep the whole automaton in memory, but only the states which are relevant for the current branch.

In order to remember the backtracking information for the depth first traversal, we use two data structures: SQ is a stack storing, for every predecessor of the current node, the transition which led to that node, and thus it contains the required node labels for the nodes of the current branch and their siblings. SN is another stack recording the current path by storing, for every level of the tree, the number of the node on the current path. If we refer to the elements in SN by $\mathsf{SN}(1)$(the bottom element), $\ldots$, $\mathsf{SN}(d)$(the top element), the next node to be checked is $\mathsf{SN}(1) \cdot \mathsf{SN}(2) \cdot \ldots \cdot \mathsf{SN}(d) + 1$ (d is the depth of SN). Thus, $\mathsf{SQ} \in (Q^k)^*$, because every transition is a k-ary tuple, and $\mathsf{SN} \in \{1, \ldots, k\}^*$.

Figure 2 shows the values stored in each of the stacks SQ and SN at the beginning of an iteration, and their relation with the traversal of the run. The circled nodes represent the path followed to reach the node about to be checked. The values of the elements of the stack are shown next to the depth in the run to which they correspond. For this reason, the stacks appear backwards, with their bottom element at the top of the figure, and *vice versa*.

After starting the algorithm, we first guess an initial transition. If we can find one, we push the labels of the nodes $1, \ldots, k$ onto SQ and the number 0 onto SN. Then we enter the while loop. As long as the stacks are not empty, we take the top elements of both stacks. If $n > k$ in line 11, this indicates that we have checked all nodes on this level, and we backtrack without pushing anything on the stacks, which means that we will continue at the next upper level in the next loop. Otherwise, we store the information that we have to check our next sibling

by pushing the same tuple of states onto SQ and the incremented number n onto SN. If the current node belongs to Q_0 (line 14), we backtrack, which means that we will continue with the next sibling. Otherwise, we try to guess a transition from this node, and if we can find one, we push the required node labels for the children of the current node onto SQ and the value 0 onto SN (line 20), which means that we will descend to the first child of the current node in the next loop.

Theorem 8 The emptiness problem of the language accepted by an m-segmentable Büchi automaton $\mathcal{A} = (Q, \Delta, I, F)$ over k-ary trees can be decided by a non-deterministic algorithm using space $O(\log(\#Q) \cdot m \cdot k)$.

Proof. In order to show soundness, we will prove the claim "if the algorithm processes a node n or backtracks without descending into n, then there is a run r in which n is labelled with the same state as in the algorithm" by induction over the iterations of the while loop. Initially, if the algorithm does not answer "empty", there is a transition $(q_0, q_1, \ldots, q_k)$ from an initial state, which can serve as root of the run r, and for which the states $1, \ldots, k$ of r can be labelled with $q_1, \ldots, q_k$.

If the algorithm has reached a node $n = n_0 \cdot n_1 \cdot \ldots \cdot n_\ell$ without failing, it follows by induction hypothesis that each of the previously visited nodes corresponds to a node in r. Now there are two possibilities: firstly, if $r(n) \in Q_0$, then, since $\mathcal{A}$ is Q_0-looping, there exists a k-ary subtree rooted at n all of whose states are accepting. Otherwise, since the algorithm does not answer "empty", there is a transition $(r(n), q'_1, \ldots, q'_k)$, and we can use the same transition in the construction of a run.

In order to show completeness, we will prove the claim "if there exists a run, the algorithm can reach or skip every node in $\{1, \ldots, k\}^*$ without failing" by induction over the structure of the run r. Since there is a run, we can guess an initial transition, and the nodes of the first level have the same labels in the algorithm as in r. If we have reached a node n which corresponds to the node n in r with $r(n) = q$, there are again two possibilities: if $q \in Q_0$, the algorithm will backtrack and skip over all successor nodes of n. Otherwise, since r is a run, there exists a transition $(q, q'_1, \ldots, q'_k)$, which the algorithm can guess, and therefore it will not fail.

Regarding memory consumption, observe that the SQ stack contains, for every level, k states, each of which can be represented using space logarithmic in the number of states, e.g. by using binary coding. Since $\mathcal{A}$ is m-segmentable, there can be at most m tuples before the current state q_n belongs to Q_0, thus the size of SQ is bounded by $\log(\#Q) \cdot m \cdot k$. SN stores at most m numbers between 0 and k, so the algorithm uses space logarithmic in the size of $\mathcal{A}$. ■

The condition that an automaton $\mathcal{A}$ is m-segmentable is rather strong since it requires the transition relation to reduce the class with *every* possible transition, and thus e.g. the automaton $\mathcal{A}_{C,\mathcal{T}}$ in Definition 4 cannot easily be proved to be segmentable even if the TBox is empty. The reason for this is that $\mathcal{A}_{C,\mathcal{T}}$ does not require the Hintikka sets of the successor states to use only a lower quantification depth. However, in order to test emptiness, we only need the *existence* of such a transition. This is the idea behind the generalisation in the following definition.

Definition 9 (Weakly-m-segmentable, reduced.) A Büchi automaton $\mathcal{A} = (Q, \Delta, I, F)$ is called *weakly-m-segmentable* if there exists a partition $Q_0, Q_1, \ldots, Q_m$ of Q such that $\mathcal{A}$ is Q_0-looping and for every $q \in Q$ there exists a function $f_q : Q \to Q$ which satisfies the following conditions:

1. if $(q, q_1, \ldots, q_k) \in \Delta$, then $(q, f_q(q_1), \ldots, f_q(q_k)) \in \Delta$, and if $q \in Q_n$, then $f_q(q_i) \in Q_{<n}$ for all $1 \leq i \leq k$;
2. if $(q', q_1, \ldots, q_k) \in \Delta$, then $(f_q(q'), f_q(q_1), \ldots, f_q(q_k)) \in \Delta$.

If $\mathcal{A} = (Q, \Delta, I, F)$ is a weakly-m-segmentable automaton, then $\mathcal{A}^{\mathrm{r}}$, the *reduced automaton of* $\mathcal{A}$, is defined as follows: $\mathcal{A}^{\mathrm{r}} = (Q, \Delta', I, F)$ with $\Delta' = \{(q, q_1, \ldots, q_m) \in \Delta \mid \text{ if } q \in Q_n \text{ then } q_i \in Q_{<m} \text{ for } 1 \leq i \leq k\}$.

Note that the reduced automaton $\mathcal{A}^{\mathrm{r}}$ is m-segmentable by definition. Intuitively, condition 1 ensures that that the class decreases for the first transition, and condition 2 ensures that there are still transitions for *all* nodes after modifying the node labels according to f_q.

We can transfer the complexity result from segmentable to weakly-segmentable automata:

Theorem 10 Let $\mathcal{A} = (Q, \Delta, I, F)$ be a weakly-m-segmentable automaton. Then $\mathcal{L}(\mathcal{A})$ is empty iff $\mathcal{L}(\mathcal{A}^{\mathrm{r}})$ is empty.

Proof. Since every run of $\mathcal{A}^{\mathrm{r}}$ is also a run of $\mathcal{A}$, $\mathcal{L}(\mathcal{A})$ can only be empty if $\mathcal{L}(\mathcal{A}^{\mathrm{r}})$ is empty, thus the "only if" direction is obvious. For the "if" direction, we will show how to transform an accepting run r of $\mathcal{A}$ into an accepting run s of $\mathcal{A}^{\mathrm{r}}$. To do this, we traverse r breadth-first, creating an intermediate run $\hat{r}$, which initially is equal to r. At every node $n \in \{1, \ldots, k\}^*$, we replace the labels of the direct and indirect successors of n with their respective f_n values (see Definition 9). More formally, at node n, we replace $\hat{r}(p)$ with $f_n(\hat{r}(p))$ for all $p \in \{n \cdot q \mid q \in \{1, \ldots, k\}^+\}$, where for a set S, S^+ denotes $S^* \setminus \{\varepsilon\}$. By definition 9, $\hat{r}$ is still a run after the replacement, and all direct successors of n are in a lower class than n (or Q_0 if $\hat{r}(n) \in Q_0$). Note that the labels of n's successors are not modified anymore after n has been processed. We can

therefore define $s(n)$ as the value of $\hat{r}(n)$ after n has been processed. As argued before, s is a run in which every node is in a lower class than its father node (or both are in class 0). Consequently, all transitions used in s belong to the transition relation of $\mathcal{A}^{\mathrm{r}}$. ■

Corollary 11 The emptiness problem for weakly-m-segmentable automata is in NLogSpace.

4 An application to $\mathcal{ALC}$ with acyclic TBoxes

In order to apply our framework to $\mathcal{ALC}$ with acyclic TBoxes, we will use the role depth to define the different classes. The following definition of role depth considers concept definitions:

Definition 12 (Expanded role depth.) For an $\mathcal{ALC}$ concept C and an acyclic TBox $\mathcal{T}$, the *expanded role depth* $\mathsf{rd}_{\mathcal{T}}(C)$ is inductively defined as follows: for a primitive role name A, $\mathsf{rd}_{\mathcal{T}}(A) = 0$; for a concept definition $A \doteq C$, $\mathsf{rd}_{\mathcal{T}}(A) = \mathsf{rd}_{\mathcal{T}}(C)$; $\mathsf{rd}_{\mathcal{T}}(\neg A) = \mathsf{rd}_{\mathcal{T}}(A)$; $\mathsf{rd}_{\mathcal{T}}(D \sqcap E) = \mathsf{rd}_{\mathcal{T}}(D \sqcup E) = \max\{\mathsf{rd}_{\mathcal{T}}(D), \mathsf{rd}_{\mathcal{T}}(E)\}$; $\mathsf{rd}_{\mathcal{T}}(\forall r.D) = \mathsf{rd}_{\mathcal{T}}(\exists r.D) = \mathsf{rd}_{\mathcal{T}}(D) + 1$. For a set of concepts S, $\mathsf{rd}_{\mathcal{T}}(S)$ is defined as $\max\{\mathsf{rd}_{\mathcal{T}}(D) \mid D \in S\}$.

The set $\mathsf{sub}_{<n}(C, \mathcal{T})$ is defined as $\{D \in \mathsf{sub}(C, \mathcal{T}) \mid \mathsf{rd}_{\mathcal{T}}(D) \leq \max\{0, n-1\}\}$.

Again, note that $\mathsf{rd}_{\mathcal{T}}$ is well-defined because $\mathcal{T}$ is acyclic.

The intuition behind using the role depth is that, for a node q in a Hintikka tree, any concept having a higher role depth than q is superfluous in successors of q and thus the tuple without these formulas is also in the transition relation.

Lemma 13 Let C be an $\mathcal{ALC}$ concept, $\mathcal{T}$ an acyclic TBox, $(S, S_1, \ldots, S_k)$ a C,$\mathcal{T}$-compatible tuple, and $n = \mathsf{rd}_{\mathcal{T}}(S)$. Then $(S, \mathsf{sub}_{<n}(C, \mathcal{T}) \cap S_1, \ldots, \mathsf{sub}_{<n}(C, \mathcal{T}) \cap S_k)$ is C,$\mathcal{T}$-compatible, and for every $m \geq 0$, the tuple $(\mathsf{sub}_{<m}(C, \mathcal{T}) \cap S, \mathsf{sub}_{<m}(C, \mathcal{T}) \cap S_1, \ldots, \mathsf{sub}_{<m}(C, \mathcal{T}) \cap S_k)$ is C,$\mathcal{T}$-compatible.

Proof. We need to show that the conditions in Definition 2 are satisfied for both tuples. In the case of the first tuple suppose that $\exists r.D \in S$. Then, $S_{\varphi(\exists r.D)}$ contains D and every concept E_i for which there is a universal formula $\forall r.E_i \in S$. But since $\mathsf{rd}_{\mathcal{T}}(D) < \mathsf{rd}_{\mathcal{T}}(\exists.rD) \leq n$ and $\mathsf{rd}_{\mathcal{T}}(E_i) < \mathsf{rd}\mathcal{T}(\forall r.E_i) \leq n$, it holds that $\mathsf{sub}_{<n}(C, \mathcal{T}) \cap S_{\varphi(\exists r.D)}$ contains D and each of the E_i concepts.

For the second tuple, if $\exists r.D \in \mathsf{sub}_{<m}(C, \mathcal{T}) \cap S$, then $\mathsf{rd}_{\mathcal{T}}(\exists r.D) < m$ and $D \in S_{\varphi(\exists r.D)}$. If additionally there is a concept term E_i such that the universal formula $\forall r.E_i \in \mathsf{sub}_{<m}(C, \mathcal{T}) \cap S$, then again $\mathsf{rd}_{\mathcal{T}}(E_i) < m$ and $E_i \in S_{\varphi(\exists r.D)}$. Hence, $\mathsf{sub}_{<m}(C, \mathcal{T}) \cap S_{\varphi(\exists r.D)}$ contains D and each such concept E_i. ■

Theorem 14 Let C be an $\mathcal{ALC}$ concept, $\mathcal{T}$ an acyclic TBox and $m = \max\{\mathsf{rd}_{\mathcal{T}}(D) \mid D \in \mathsf{sub}(C, \mathcal{T})\}$. Then $\mathcal{A}_{C,\mathcal{T}}$ is weakly-m-segmentable.

Proof. We have to give the segmentation of Q and the functions f_q and show that they satisfy the conditions in Definition 9. Define the classes $Q_i := \{S \in Q \mid \mathsf{rd}_{\mathcal{T}}(S) = i\}, 0 \leq i \leq m$ and $f_q(q') := q' \cap \mathsf{sub}_{<n}(C, \mathcal{T})$, where $n = \mathsf{rd}_{\mathcal{T}}(q)$ for every $q, q' \in Q$. By this definition, it is obvious that for every q', $f_q(q')$ is in a lower class than q (or in Q_0 if $q \in Q_0$). Lemma 13 shows that conditions 1 and 2 of Definition 9 are satisfied. It remains to show that $\mathcal{A}_{C,\mathcal{T}}$ is Q_0-looping. If $q \in Q_0$, there are no existential formulas in q, and therefore $(q, \emptyset, \ldots, \emptyset) \in \Delta$. ■

Although our algorithm in Figure 1 is non-deterministic, we can obtain a deterministic complexity class by using Savitch's theorem [Sav70].

Corollary 15 Satisfiability of $\mathcal{ALC}$ concepts with respect to acyclic TBoxes is in PSpace.

5 Conclusion

We have introduced segmentable and weakly-segmentable Büchi automata, two classes of automata for which the emptiness problem of the accepted language is decidable in NLogSpace, whereas in general the complexity class of this problem for Büchi automata is P. The complexity bound is proved by testing the possibility of a model on the fly using depth-first search. This generalises previous results of on-the-fly emptiness tests for several modal and description logics. As an example, we showed how our framework can be used to obtain the PSpace upper complexity bound for $\mathcal{ALC}$ with acyclic TBoxes in an easy way. We hope that this framework will make it easier to prove a PSpace upper bound also for new logics.

Acknowledgements

We would like to thank Franz Baader for fruitful discussions. We also thank the reviewers for suggesting improvements of this paper.

References

[BHLW03] F. Baader, J. Hladik, C. Lutz, and F. Wolter. From tableaux to automata for description logics. *Fundamenta Informaticae*, 57:1–33, 2003.

[BN03] F. Baader and W. Nutt. *The Description Logic Handbook*, chapter 2: Basic Description Logics. Cambridge University Press, 2003.

[BS01] F. Baader and U. Sattler. An overview of tableau algorithms for description logics. *Studia Logica*, 69, 2001.

[BT01] F. Baader and S. Tobies. The inverse method implements the automata approach for modal satisfiability. In R. Goré, A. Leitsch, and T. Nipkow, editors, *Proceedings of IJCAR-01*, volume 2083 of *LNAI*. Springer-Verlag, 2001.

[HST00] I. Horrocks, U. Sattler, and S. Tobies. Practical reasoning for very expressive description logics. *Logic Journal of the IGPL*, 8(3):239–264, 2000.

[LS00] C. Lutz and U. Sattler. Mary likes all cats. In F. Baader and U. Sattler, editors, *Proceedings of DL 2000*, CEUR Proceedings, 2000.

[Lut04] C. Lutz. NExpTime-complete description logics with concrete domains. *ACM Transactions on Computational Logic*, 5(4):669–705, 2004.

[PSV02] G. Pan, U. Sattler, and M. Y. Vardi. BDD-based decision procedures for K. In *Proceedings of the Conference on Automated Deduction*, volume 2392 of *Lecture Notes in Artificial Intelligence*, 2002.

[Sav70] W. J. Savitch. Relationships between nondeterministic and deterministic tape complexities. *Journal of computer and system sciences*, 4(2):177–192, 1970.

[Sch94] K. Schild. Terminological cycles and the propositional μ-calculus. In J. Doyle, E. Sandewall, and P. Torasso, editors, *Proceedings of KR-94*. Morgan Kaufmann, 1994.

[Spa93] E. Spaan. *Complexity of Modal Logics*. PhD thesis, University of Amsterdam, 1993.

[SS91] M. Schmidt-Schauß and G. Smolka. Attributive concept descriptions with complements. *Artificial Intelligence*, 48(1):1–26, 1991.

[Vor01] A. Voronkov. How to optimize proof-search in modal logics: new methods of proving reduncancy criteria for sequent calculi. *ACM transactions on computational logic*, 2(2), 2001.

[VW86] M. Y. Vardi and P. Wolper. Automata-theoretic techniques for modal logics of programs. *J. of Computer and System Science*, 32:183–221, 1986.

Binary Absorption in Tableaux-Based Reasoning for Description Logics

Alexander K. Hudek and Grant Weddell
David R. Cheriton School of Computer Science
University of Waterloo
{akhudek,gweddell}@cs.uwaterloo.ca

1 Introduction

A fundamental problem in Description Logics (DLs) is *satisfiability*, the problem of checking if a given DL *terminology* $\mathcal{T}$ remains sufficiently unconstrained to enable at least one instance of a given DL *concept* C to exist. It has been known for some time that *lazy unfolding* is an important optimization technique in model building algorithms for satisfiability [2]. It is also imperative for large terminologies to be manipulated by an *absorption generation* process to maximize the benefits of lazy unfolding in such algorithms, thereby reducing the combinatorial effects of disjunction in underlying chase procedures [5]. In this paper, we propose a generalization of the absorption theory and algorithms developed by Horrocks and Tobies [6, 7]. The generalization, called *binary absorption*, makes it possible for lazy unfolding to be used for parts of terminologies not handled by current absorption algorithms and theory.

The basic idea of binary absorption is to avoid the need to *internalize* (at least some of the) terminological axioms of the form

$$(A_1 \sqcap A_2) \sqsubseteq C,$$

where the A_i denote *primitive concepts* and C a general concept. This idea, coupled with equivalences and another idea relating to "role absorptions" developed by Tsarkov and Horrocks [8], makes it possible for an algorithm to absorb, e.g., the definition

$$\begin{array}{lcl} \text{SPECIALCLIENT} & \doteq & \text{CLIENT} \sqcap \\ & & (\exists \mathit{Buy}.(\text{EXPENSIVE} \sqcup \text{PROFITABLE})) \sqcap \\ & & (\exists \mathit{Recommend}^{-}.\text{TRUSTEDCLIENT}) \end{array}$$

as the set of axioms

$$
\begin{array}{rcl}
\text{SPECIALCLIENT} & \sqsubseteq & \text{CLIENT} \sqcap \\
 & & (\exists Buy.(\text{EXPENSIVE} \sqcup \text{PROFITABLE})) \sqcap \\
 & & (\exists Recommend^{-}.\text{TRUSTEDCLIENT}) \\
\text{EXPENSIVE} & \sqsubseteq & \text{A1} \\
\text{PROFITABLE} & \sqsubseteq & \text{A1} \\
\text{A1} & \sqsubseteq & \forall Buy^{-}.\text{A2} \\
\text{CLIENT} \sqcap \text{A2} & \sqsubseteq & \text{A3} \\
\text{TRUSTEDCLIENT} & \sqsubseteq & \forall Recommend.\text{A4} \\
\text{A3} \sqcap \text{A4} & \sqsubseteq & \text{SPECIALCLIENT}
\end{array}
$$

in which the primitive concepts A1, A2, A3 and A4 are new internal primitive concepts introduced by the absorption algorithm.

There is another reason that binary absorption proves useful, beyond the well-documented advantages of reducing the need for internalization of general terminological axioms. In particular, Palacios has explored the possibility of a tighter integration of conjunctive query optimization and DL consistency checking in which a query is mapped to a concept in such a way that view integration becomes an automatic consequence of an initial *chase phase* for optimization [4, 9]. When used for such purposes, it becomes crucial for DL reasoners to support special case (and likely incomplete) *deterministic* modes of model building. Binary absorption enables such modes to incorporate reasoning about *materialized views* in a terminology, such as SPECIALCLIENT above.

The organization of the paper is as follows. The next section is a review of the basic definitions introduced by Horrocks and Tobies [6] for the notion of a DL, of terminologies and satisfiability, and an important abstraction that relates to model building algorithms for satisfiability: the notion of a *witness*. In Section 3, we define binary absorptions, and give a related lemma that establishes an additional sufficiency condition for correct binary absorptions. (Note that existing sufficiency conditions given by Horrocks and Tobies [6, 7] for correct non-binary absorptions are inherited in our formulation.) In Section 4, we present an absorption algorithm for binary absorptions that derives from earlier procedures [1, 6, 7]. Our summary comments follow in Section 5.

2 Preliminaries

The definitions and lemmas in this section are largely reproduced from Horrocks and Tobies [7]. The main difference is in our formulation of a description logic immediately following in which we have an additional stipulation that the logic includes the concept forming operations of $\mathcal{ALCI}$.

Definition 2.1. (Description Logic) *Let* L *be a DL based on infinite sets of atomic concepts* NC *and atomic roles* NR*. We identify* L *with the set of its well-formed concepts and require* L *to be closed under sub-concepts and the concept forming operations of dialect $\mathcal{ALCI}$: we require that if* NR *contains R and* L *contains C_1 and C_2, then* L *also contains $\neg C_1$, $C_1 \sqcap C_2$, $C_1 \sqcup C_2$, $\top$, $\bot$, $\exists R.C_1$, $\exists R^{-}.C_1$, $\forall R.C_1$ and $\forall R^{-}.C_1$.*

An interpretation $\mathcal{I}$ *is a pair* $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$*, where* $\Delta^{\mathcal{I}}$ *is a non-empty set and* $\cdot^{\mathcal{I}}$ *is a function mapping* NC *to subsets of* $\Delta^{\mathcal{I}}$ *and* NR *to subsets of* $\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$*. Each* L *is associated with a set* Int(L) *of* admissible *interpretations that (1) must be closed under isomorphisms, and that (2) satisfies* $\mathcal{I} \in \text{Int(L)} \Leftrightarrow \mathcal{I}' \in \text{Int(L)}$ *for any two interpretations* $\mathcal{I}$ *and* $\mathcal{I}'$ *that agree on* NR*. Each* L *must also come with a semantics that allows any* $\mathcal{I} \in \text{Int(L)}$ *to be extended to each concept* $C \in \text{L}$ *in a way that satisfies the following conditions:*

(I1) the concept forming operations of $\mathcal{ALCI}$ *are mapped in the standard way, and*

(I2) the interpretation $C^{\mathcal{I}}$ *of a compound concept* $C \in \text{L}$ *depends only on the interpretation of those atomic concepts and roles that appear syntactically in* C*.*

Definition 2.2. (TBox, Satisfiability) *A* TBox $\mathcal{T}$ *for* L *is a finite set of axioms of the form* $C_1 \sqsubseteq C_2$ *or* $C_1 \doteq C_2$ *where* $C_i \in \text{L}$*. If* $\mathcal{T}$ *contains* $A \sqsubseteq C$ *or* $A \doteq C$ *for some* $A \in \text{NC}$*, then we say that* A is defined in $\mathcal{T}$*.*

Let L *be a DL and* $\mathcal{T}$ *a TBox. An interpretation* $\mathcal{I} \in \text{Int(L)}$ *is a* model *of* $\mathcal{T}$*, written* $\mathcal{I} \models \mathcal{T}$*, iff* $C_1^{\mathcal{I}} \subseteq C_2^{\mathcal{I}}$ *holds for each* $C_1 \sqsubseteq C_2 \in \mathcal{T}$*, and* $C_1^{\mathcal{I}} = C_2^{\mathcal{I}}$ *holds for each* $C_1 \doteq C_2 \in \mathcal{T}$*. A concept* $C \in \text{L}$ *is* satisfiable *with respect to a TBox* $\mathcal{T}$ *iff there is an* $\mathcal{I} \in \text{Int(L)}$ *such that* $\mathcal{I} \models \mathcal{T}$ *and such that* $C^{\mathcal{I}} \neq \emptyset$*.*

A TBox $\mathcal{T}$ *is called* primitive *iff it consists entirely of axioms of the form* $A \doteq C$ *with* $A \in \text{NC}$*, each* $A \in \text{NC}$ *appears in at most one left hand side of an axiom, and* $\mathcal{T}$ *is* acyclic*. Acyclicity is defined as follows:* $A_1 \in \text{NC}$ directly uses $A_2 \in \text{NC}$ *if* $A_1 \doteq C \in \mathcal{T}$ *and* A_2 *occurs in* C*;* uses *is the transitive closure of "directly uses". Then* $\mathcal{T}$ *is* acyclic *if there is no* $A \in \text{NC}$ *that uses itself.*

Model building algorithms for checking the satisfaction of a concept C operate by manipulating an internal data structure (e.g., in the form of a node and edge labelled rooted tree with "back edges"). The data structure "encodes" a *partial description* of (eventual) interpretations $\mathcal{I}$ for which $C^{\mathcal{I}}$ will be non-empty. Such a partial description will almost always abstract details on class membership for hypothetical elements of $\Delta^{\mathcal{I}}$ and on details relating to the interpretation of roles. To talk formally about absorption and lazy evaluation, it is necessary to codify the idea of a partial description. Horrocks and Tobies have done this by introducing the following notion of a *witness*, of an interpretation that *stems* from a witness, and of what it means for a witness to be *admissible* with respect to a given terminology.

Definition 2.3. (Witness) *Let* L *be a DL and* $C \in \text{L}$ *a concept. A* witness $\mathcal{W} = (\Delta^{\mathcal{W}}, \cdot^{\mathcal{W}}, \mathcal{L}^{\mathcal{W}})$ *for* C *consists of a non-empty set* $\Delta^{\mathcal{W}}$*, a function* $\cdot^{\mathcal{W}}$ *that maps* NR *to subsets of* $\Delta^{\mathcal{W}} \times \Delta^{\mathcal{W}}$*, and a function* $\mathcal{L}^{\mathcal{W}}$ *that maps* $\Delta^{\mathcal{W}}$ *to subsets of* L *such that:*

(W1) there is some $x \in \Delta^{\mathcal{W}}$ *with* $C \in \mathcal{L}^{\mathcal{W}}(x)$*,*

(W2) there is an interpretation $\mathcal{I} \in \text{Int(L)}$ *that* stems *from* $\mathcal{W}$*, and*

(W3) for each interpretation $\mathcal{I} \in \text{Int(L)}$ *that* stems *from* $\mathcal{W}$*,* $x \in C^{\mathcal{I}}$ *if* $C \in \mathcal{L}^{\mathcal{W}}(x)$*.*

An interpretation $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ *is said to* stem *from* $\mathcal{W}$ *if* $\Delta^{\mathcal{I}} = \Delta^{\mathcal{W}}$, $\cdot^{\mathcal{I}}|_{\mathrm{NR}} = \cdot^{\mathcal{W}}$, *and for each* $A \in \mathrm{NC}$, $A \in \mathcal{L}^{\mathcal{W}}(x)$ *implies* $x \in A^{\mathcal{I}}$ *and* $\neg A \in \mathcal{L}^{\mathcal{W}}(x)$ *implies* $x \notin A^{\mathcal{I}}$.

A witness $\mathcal{W}$ *is called* admissible *with respect to a TBox* $\mathcal{T}$ *if there is an interpretation* $\mathcal{I} \in \mathrm{Int(L)}$ *that stems from* $\mathcal{W}$ *with* $\mathcal{I} \models \mathcal{T}$.

The important properties satisfied by a witness are captured by the following lemmas by Horrocks and Tobies [7].

Lemma 2.1. *Let* L *be a DL. A concept* $C \in \mathrm{L}$ *is satisfiable w.r.t. a TBox* $\mathcal{T}$ *iff it has a witness that is admissible w.r.t.* $\mathcal{T}$.

Lemma 2.2. *Let* L, C, $\mathcal{T}$ *and* $\mathcal{W}$ *be a DL, a concept in* L, *a TBox for* L *and a witness for* C, *respectively. Then* $\mathcal{W}$ *is admissible w.r.t.* $\mathcal{T}$ *if, for each* $x \in \Delta^{\mathcal{W}}$:

$$\begin{array}{lll} C_1 \sqsubseteq C_2 \in \mathcal{T} & \textit{implies} & \neg C_1 \sqcup C_2 \in \mathcal{L}^{\mathcal{W}}(x), \\ C_1 \doteq C_2 \in \mathcal{T} & \textit{implies} & \neg C_1 \sqcup C_2 \in \mathcal{L}^{\mathcal{W}}(x) \textit{ and} \\ C_1 \doteq C_2 \in \mathcal{T} & \textit{implies} & C_1 \sqcup \neg C_2 \in \mathcal{L}^{\mathcal{W}}(x). \end{array}$$

3 Binary Absorptions

Our generalization of the notion of an *absorption* developed by Horrocks and Tobies [6, 7] is given as follows.

Definition 3.1. (Binary Absorption) *Let* L *and* $\mathcal{T}$ *be a DL and a TBox, respectively. A* binary absorption *of* $\mathcal{T}$ *is a pair of TBoxes* $(\mathcal{T}_u, \mathcal{T}_g)$ *such that* $\mathcal{T} \equiv \mathcal{T}_u \cup \mathcal{T}_g$ *and* $\mathcal{T}_u$ *contains only axioms of the form* $A_1 \sqsubseteq C$, *the form* $\neg A_1 \sqsubseteq C$ *or the form* $(A_1 \sqcap A_2) \sqsubseteq C$, *where* A_1 *and* A_2 *are primitive concepts occurring in* NC.

A binary absorption $(\mathcal{T}_u, \mathcal{T}_g)$ *of* $\mathcal{T}$ *is called* correct *if it satisfies the following condition. For each witness* $\mathcal{W}$ *and* $x \in \Delta^{\mathcal{W}}$, *if*

$$\begin{array}{rlll} (A_1 \sqcap A_2) \sqsubseteq C \in \mathcal{T}_u \textit{ and } \{A_1, A_2\} \subseteq \mathcal{L}^{\mathcal{W}}(x) & \textit{implies} & C \in \mathcal{L}^{\mathcal{W}}(x), & \\ A \sqsubseteq C \in \mathcal{T}_u \textit{ and } A \in \mathcal{L}^{\mathcal{W}}(x) & \textit{implies} & C \in \mathcal{L}^{\mathcal{W}}(x), & \\ \neg A \sqsubseteq C \in \mathcal{T}_u \textit{ and } \neg A \in \mathcal{L}^{\mathcal{W}}(x) & \textit{implies} & C \in \mathcal{L}^{\mathcal{W}}(x), & \\ C_1 \sqsubseteq C_2 \in \mathcal{T}_g & \textit{implies} & \neg C_1 \sqcup C_2 \in \mathcal{L}^{\mathcal{W}}(x), & (1) \\ C_1 \doteq C_2 \in \mathcal{T}_g & \textit{implies} & \neg C_1 \sqcup C_2 \in \mathcal{L}^{\mathcal{W}}(x) \textit{ and} & \\ C_1 \doteq C_2 \in \mathcal{T}_g & \textit{implies} & C_1 \sqcup \neg C_2 \in \mathcal{L}^{\mathcal{W}}(x), & \end{array}$$

then $\mathcal{W}$ *is admissible w.r.t.* $\mathcal{T}$. *A witness that satisfies* (1) *will be called* unfolded.

The distinguishing feature of binary absorption is the addition of the first implication in condition (1). This allows additional axioms in $\mathcal{T}_u$ to be dealt with in a deterministic manner, as we illustrate in our introductory example. If a label of a node contains neither primitive concept A_1 nor A_2, then *nothing* further is required. There is, however, a new requirement for computing the intersection of sets of primitive concepts during model building when a new primitive concept is added to a node. (For this problem, we refer the reader to algorithms developed by Demaine, Lopez-Ortiz and Munro [3].)

The next three lemmas by Horrocks and Tobies [7] hold without modification.

Lemma 3.1. *Let $(\mathcal{T}_u, \mathcal{T}_g)$ be a correct binary absorption of $\mathcal{T}$. For any $C \in \mathrm{L}$, C has a witness that is admissible w.r.t. $\mathcal{T}$ iff C has an unfolded witness.*

Lemma 3.2. *Let $\mathcal{T}$ be a primitive TBox and $\mathcal{T}_u$ defined as*

$$\{A \sqsubseteq C, \neg A \sqsubseteq \neg C \mid A \doteq C \in \mathcal{T}\}.$$

Then $(\mathcal{T}_u, \emptyset)$ is a correct absorption of $\mathcal{T}$.

Lemma 3.3. *Let $(\mathcal{T}_u, \mathcal{T}_g)$ be a correct absorption of a TBox $\mathcal{T}$.*

1. *If $\mathcal{T}'$ is an arbitrary TBox, then $(\mathcal{T}_u, \mathcal{T}_g \cup \mathcal{T}')$ is a correct absorption of $\mathcal{T} \cup \mathcal{T}'$.*

2. *If $\mathcal{T}'$ is a TBox that consists entirely of axioms of the form $A \sqsubseteq C$, where $A \in \mathrm{NC}$ and A is not defined in $\mathcal{T}_u$, then $(\mathcal{T}_u \cup \mathcal{T}', \mathcal{T}_g)$ is a correct absorption of $\mathcal{T} \cup \mathcal{T}'$.*

The main benefit of binary absorptions is that they allow the following additional sufficiency condition for correct absorptions.

Lemma 3.4. *Let $(\mathcal{T}_u, \mathcal{T}_g)$ be a correct absorption of a TBox $\mathcal{T}$. If $\mathcal{T}'$ is a TBox that consists entirely of axioms of the form $(A_1 \sqcap A_2) \sqsubseteq C$, where $\{A_1, A_2\} \subseteq \mathrm{NC}$ and where neither A_1 nor A_2 is defined in $\mathcal{T}_u$, then $(\mathcal{T}_u \cup \mathcal{T}', \mathcal{T}_g)$ is a correct absorption of $\mathcal{T} \cup \mathcal{T}'$.*

Proof. Observe that $\mathcal{T}_u \cup \mathcal{T}_g \cup \mathcal{T}' \equiv \mathcal{T} \cup \mathcal{T}'$ holds trivially. Let $C \in \mathrm{L}$ be a concept and $\mathcal{W}$ be an unfolded witness for C w.r.t. the absorption $(\mathcal{T}_u \cup \mathcal{T}', \mathcal{T}_g)$. From $\mathcal{W}$, define a new witness $\mathcal{W}'$ for C by setting $\Delta^{\mathcal{W}'} = \Delta^{\mathcal{W}}$, $\cdot^{\mathcal{W}'} = \cdot^{\mathcal{W}}$, and defining $\mathcal{L}^{\mathcal{W}'}$ to be the function that, for every $x \in \Delta^{\mathcal{W}'}$, maps x to the set

$$\begin{array}{rcl} \mathcal{L}^{\mathcal{W}}(x) & \cup & \{\neg A_1, \neg A_2 \mid (A_1 \sqcap A_2) \sqsubseteq C' \in \mathcal{T}', \{A_1, A_2\} \cap \mathcal{L}^{\mathcal{W}}(x) = \emptyset\} \\ & \cup & \{\neg A_1 \mid (A_1 \sqcap A_2) \sqsubseteq C' \in \mathcal{T}', A_1 \notin \mathcal{L}^{\mathcal{W}}(x), A_2 \in \mathcal{L}^{\mathcal{W}}(x)\} \\ & \cup & \{\neg A_2 \mid (A_1 \sqcap A_2) \sqsubseteq C' \in \mathcal{T}', A_1 \in \mathcal{L}^{\mathcal{W}}(x), A_2 \notin \mathcal{L}^{\mathcal{W}}(x)\}. \end{array}$$

It is easy to see that $\mathcal{W}'$ is also unfolded w.r.t. the absorption $(\mathcal{T}_u \cup \mathcal{T}', \mathcal{T}_g)$. This implies that $\mathcal{W}'$ is also unfolded w.r.t. the (smaller) absorption $(\mathcal{T}_u, \mathcal{T}_g)$. Since $(\mathcal{T}_u, \mathcal{T}_g)$ is a correct absorption of $\mathcal{T}$, there exists an interpretation $\mathcal{I}$ stemming from $\mathcal{W}'$ such that $\mathcal{I} \models \mathcal{T}$. We show that $\mathcal{I} \models \mathcal{T}'$ also holds. Assume $\mathcal{I} \not\models \mathcal{T}'$. Then there is an axiom $(A_1 \sqcap A_2) \sqsubseteq C_1 \in \mathcal{T}'$ and an $x \in \Delta^{\mathcal{I}}$ such that $x \in (A_1 \sqcap A_2)^{\mathcal{I}}$ but $x \notin C_1^{\mathcal{I}}$. By construction of $\mathcal{W}'$, $x \in (A_1 \sqcap A_2)^{\mathcal{I}}$ implies $\{A_1, A_2\} \subseteq \mathcal{L}^{\mathcal{W}'}(x)$ because otherwise $\{\neg A_1, \neg A_2\} \cap \mathcal{L}^{\mathcal{W}'}(x) \neq \emptyset$ would hold in contradiction to (W3). Then, since $\mathcal{W}'$ is unfolded, $C_1 \in \mathcal{L}^{\mathcal{W}'}(x)$, which, again, by (W3), implies $x \in C_1^{\mathcal{I}}$, a contradiction.

Hence, we have shown that there exists an interpretation $\mathcal{I}$ stemming from $\mathcal{W}'$ such that $\mathcal{I} \models \mathcal{T}_u \cup \mathcal{T}' \cup \mathcal{T}_g$. By construction of $\mathcal{W}'$, any interpretation stemming from $\mathcal{W}'$ also stems from $\mathcal{W}$, hence $\mathcal{W}$ is admissible w.r.t. $\mathcal{T} \cup \mathcal{T}'$. □

4 A Binary Absorption Algorithm

In this section, we present a two-phase algorithm for generating binary absorptions that derives from the absorption algorithm for the FaCT system outlined in earlier work [1, 6, 7]. Our algorithm also incorporates role absorption, similar to the work of Tsarkov and Horrocks [8]. The notable differences in our algorithm happen in the second phase during which

- an opportunity for unary absorption now has a very low priority,
- binary absorption replaces unary absorption in becoming a high priority, and
- primitive concepts are introduced by the procedure to enable further binary absorptions and to absorb existential role restrictions.

The algorithm is given a TBox $\mathcal{T}$ containing arbitrary axioms, and proceeds by constructing four TBoxes, $\mathcal{T}_g, \mathcal{T}_{prim}, \mathcal{T}_{uinc}$, and $\mathcal{T}_{binc}$, such that: $\mathcal{T} \equiv \mathcal{T}_g \cup \mathcal{T}_{prim} \cup \mathcal{T}_{uinc} \cup \mathcal{T}_{binc}$, $\mathcal{T}_{prim}$ is primitive, $\mathcal{T}_{uinc}$ consists only of axioms of the form $A_1 \sqsubseteq C$, and $\mathcal{T}_{binc}$ consists only of axioms of the form $(A_1 \sqcap A_2) \sqsubseteq C$, where $\{A_1, A_2\} \subseteq \mathrm{NC}$ and neither A_1 nor A_2 are defined in $\mathcal{T}_{prim}$. Here, $\mathcal{T}_{uinc}$ contains unary absorptions and $\mathcal{T}_{binc}$ contains binary absorptions.

In the first phase, we move as many axioms as possible from $\mathcal{T}$ into $\mathcal{T}_{prim}$. We initialize $\mathcal{T}_{prim} = \emptyset$ and process each axiom $X \in \mathcal{T}$ as follows.

1. If X is of the form $A \doteq C$, A is not defined in $\mathcal{T}_{prim}$, and $\mathcal{T}_{prim} \cup \{X\}$ is primitive, then move X to $\mathcal{T}_{prim}$.
2. If X is of the form $A \doteq C$, then remove X from $\mathcal{T}$ and replace it with axioms $A \sqsubseteq C$ and $\neg A \sqsubseteq \neg C$.
3. Otherwise, leave X in $\mathcal{T}$.

In the second phase, we process axioms in $\mathcal{T}$, either by simplifying them or by placing absorbed components in either $\mathcal{T}_{uinc}$ or $\mathcal{T}_{binc}$. We place components that cannot be absorbed in $\mathcal{T}_g$. To ease axiom manipulation, we introduce a set representation. We let $\mathbf{G} = \{C_1, \ldots, C_n\}$ represent the axiom $\top \sqsubseteq (C_1 \sqcup \ldots \sqcup C_n)$. When removing an axiom from $\mathcal{T}$, we automatically convert it to a set $\mathbf{G}$. Similarly, when adding $\mathbf{G}$ to $\mathcal{T}$, we automatically convert it out of set notation. Details are as follows.

1. If $\mathcal{T}$ is empty, then return the binary absorption

   $$(\{A \sqsubseteq C, \neg A \sqsubseteq \neg C \mid A \doteq C \in \mathcal{T}_{prim}\} \cup \mathcal{T}_{uinc} \cup \mathcal{T}_{binc}, \mathcal{T}_g).$$

   Otherwise, remove an axiom $\mathbf{G}$ from $\mathcal{T}$.

2. Try to simplify $\mathbf{G}$.

   (a) If there is some $\neg C \in \mathbf{G}$ such that C is not a primitive concept, then add $(\mathbf{G} \cup \mathrm{NNF}(\neg C) \setminus \{\neg C\}$ to $\mathcal{T}$, where the function $\mathrm{NNF}(\cdot)$ converts concepts to negation normal form. Return to Step 1.

(b) If there is some $C \in \mathbf{G}$ such that C is of the form $(C_1 \sqcap C_2)$, then add both $(\mathbf{G} \cup \{C_1\}) \setminus \{C\}$ and $(\mathbf{G} \cup \{C_2\}) \setminus \{C\}$ to $\mathcal{T}$. Return to Step 1.

(c) If there is some $C \in \mathbf{G}$ such that C is of the form $C_1 \sqcup C_2$, then apply associativity by adding $(\mathbf{G} \cup \{C_1, C_2\}) \setminus \{C_1 \sqcup C_2\}$ to $\mathcal{T}$. Return to Step 1.

3. Try to partially absorb $\mathbf{G}$.

(a) If $\{\neg A_1, \neg A_2\} \subset \mathbf{G}$, $(A_3 \sqcap A_4) \sqsubseteq A_1 \in \mathcal{T}_{binc}$, and A_2 is not defined in $\mathcal{T}_{prim}$, then do the following. If there is an axiom of the form $(A_1 \sqcap A_2) \sqsubseteq A'$ in $\mathcal{T}_{binc}$, add $\mathbf{G} \cup \{\neg A'\} \setminus \{\neg A_1, \neg A_2\}$ to $\mathcal{T}$. Otherwise, introduce a new internal primitive concept A', add $(\mathbf{G} \cup \{\neg A'\}) \setminus \{\neg A_1, \neg A_2\}$ to $\mathcal{T}$, and add $(A_1 \sqcap A_2) \sqsubseteq A'$ to $\mathcal{T}_{binc}$. Return to Step 1.

(b) If $\{\neg A_1, \neg A_2\} \subset \mathbf{G}$, and neither A_1 nor A_2 are defined in $\mathcal{T}_{prim}$, then do the following. If there is an axiom of the form $(A_1 \sqcap A_2) \sqsubseteq A'$ in $\mathcal{T}_{binc}$, add $\mathbf{G} \cup \{\neg A'\} \setminus \{\neg A_1, \neg A_2\}$ to $\mathcal{T}$. Otherwise, introduce a new internal primitive concept A', add $(\mathbf{G} \cup \{\neg A'\}) \setminus \{\neg A_1, \neg A_2\}$ to $\mathcal{T}$, and add $(A_1 \sqcap A_2) \sqsubseteq A'$ to $\mathcal{T}_{binc}$. Return to Step 1.

(c) If $\forall R.C \in \mathbf{G}$, then do the following. Introduce a new internal primitive concept A' and add both $\neg C \sqsubseteq \forall R^-.A'$ and $(\mathbf{G} \cup \{\neg A'\}) \setminus \{\forall R.C\}$ to $\mathcal{T}$. Return to Step 1.

(d) If $\forall R^-.C \in \mathbf{G}$, then do the following. Introduce a new internal primitive concept A' and add both $\neg C \sqsubseteq \forall R.A'$ and $(\mathbf{G} \cup \{\neg A'\}) \setminus \{\forall R^-.C\}$ to $\mathcal{T}$. Return to Step 1.

4. Try to unfold $\mathbf{G}$. If, for some $A \in \mathbf{G}$ (resp. $\neg A \in \mathbf{G}$), there is an axiom $A \doteq C$ in $\mathcal{T}_{prim}$, then substitute $A \in \mathbf{G}$ (resp. $\neg A \in \mathbf{G}$) with C (resp. $\neg C$), and add $\mathbf{G}$ to $\mathcal{T}$. Return to Step 1.

5. Try to absorb $\mathbf{G}$. If $\neg A \in \mathbf{G}$ and A is not defined in $\mathcal{T}_{prim}$, add $A \sqsubseteq C$ to $\mathcal{T}_{uinc}$ where C is the disjunction of all the concepts in $\mathbf{G} \setminus \{\neg A\}$. Return to Step 1.

6. If none of the above are possible ($\mathbf{G}$ cannot be absorbed), add $\mathbf{G}$ to $\mathcal{T}_g$. Return to Step 1.

Termination of our procedure can be established by a straightforward counting argument involving concept constructors in $\mathcal{T}$. We now prove the correctness of our algorithm using induction. We use the following four lemmas in our induction step. The first two lemmas prove, in combination, that both Step 3(a) and Step 3(b) of our algorithm is correct. The last two lemmas prove Step 3(c) and Step 3(d) correct respectively.

Lemma 4.1. *Let $\mathcal{T}_1, \mathcal{T}_2$, and $\mathcal{T}$ denote TBoxes, $C \in \mathrm{L}$ an arbitrary concept, and A a primitive concept not used in C or $\mathcal{T}$. If $\mathcal{T}_1$ is of the form*

$$\mathcal{T}_1 = \mathcal{T} \cup \{(C_1 \sqcap C_2 \sqcap C_3) \sqsubseteq C_4\},$$

then C is satisfiable with respect to $\mathcal{T}_1$ iff C is satisfiable with respect to

$$\mathcal{T}_2 = \mathcal{T} \cup \{(C_1 \sqcap C_2) \sqsubseteq A, (A \sqcap C_3) \sqsubseteq C_4\}.$$

Proof. First we prove the if direction. Assume C is satisfiable with respect to $\mathcal{T}_1$. For each interpretation $\mathcal{I}$ such that $\mathcal{I} \models \mathcal{T}_1$ and $C^{\mathcal{I}} \neq \emptyset$, we extend to an interpretation $\mathcal{I}'$ such that $\mathcal{I}' \models \mathcal{T}_2$ and $C^{\mathcal{I}'} \neq \emptyset$. First, set $\mathcal{I}' = \mathcal{I}$. For each $x \in \Delta^{\mathcal{I}}$ such that $x \in C_1^{\mathcal{I}}$ and $x \in C_2^{\mathcal{I}}$, add x to $A^{\mathcal{I}'}$. Then, $\mathcal{I}' \models \mathcal{T}_2$.

For the only if direction, assume C is satisfiable with respect to $\mathcal{T}_2$. For each interpretation $\mathcal{I} \in \text{Int(L)}$ such that $\mathcal{I} \models \mathcal{T}_2$ and $C^{\mathcal{I}} \neq \emptyset$, we show that $\mathcal{I} \models \mathcal{T}_1$. The proof is by contradiction. Assume $\mathcal{I} \not\models \mathcal{T}_1$. It must be the case that $(C_1 \sqcap C_2 \sqcap C_3) \sqsubseteq C_4 \in \mathcal{T}_1$ does not hold, since the rest of $\mathcal{T}_1$ is a subset of $\mathcal{T}_2$. Therefore, there exists $x \in \Delta^{\mathcal{I}}$ such that $x \in C_1^{\mathcal{I}} \cap C_2^{\mathcal{I}} \cap C_3^{\mathcal{I}}$, and $x \notin C_4^{\mathcal{I}}$. But, in this case either $(C_1 \sqcap C_2) \sqsubseteq A \in \mathcal{T}_2$ or $(A \sqcap C_3) \sqsubseteq C_4 \in \mathcal{T}_2$ must not hold. A contradiction. □

The following lemma proves that instead of introducing a new primitive concept every time we execute Step 3(a) (or Step 3(b)) of our algorithm, we may instead reuse a previously introduced primitive concept. We use **H** to denote an arbitrary axiom.

Lemma 4.2. *Let $\mathcal{T}_1, \mathcal{T}_2$, and $\mathcal{T}$ denote TBoxes, $C \in$ L an arbitrary concept, A a primitive concept not used in C or $\mathcal{T}$, and A_1, A_2, primitive concepts introduced by Step 3(a) (or Step 3(b)) of our algorithm modified such that a new primitive is always introduced. If $\mathcal{T}_1$ is of the form*

$$\mathcal{T}_1 = \mathcal{T} \cup \{(C_1 \sqcap C_2) \sqsubseteq A_1, (C_1 \sqcap C_2) \sqsubseteq A_2\},$$

then C is satisfiable with respect to $\mathcal{T}_1$ iff C is satisfiable with respect to

$$\mathcal{T}_2 = \{(C_1 \sqcap C_2) \sqsubseteq A\} \cup \{\mathbf{H} \textit{ where } A \textit{ is substituted for } A_1 \textit{ and } A_2 \mid \mathbf{H} \in \mathcal{T}\}.$$

Proof. First we prove the if direction. We have two cases.

- Let $\mathcal{I}$ be an interpretation such that $\mathcal{I} \models \mathcal{T}_1$, $C^{\mathcal{I}} \neq \emptyset$, and $A_1^{\mathcal{I}} = A_2^{\mathcal{I}}$. We construct an interpretation $\mathcal{I}'$ from $\mathcal{I}$ such that $\mathcal{I}' \models \mathcal{T}_2$ and $C^{\mathcal{I}'} \neq \emptyset$. First, set $\mathcal{I}' = \mathcal{I}$. Then, set $A^{\mathcal{I}'} = A_1^{\mathcal{I}}$ and remove any references to A_1 and A_2 in $\mathcal{I}'$.
- Let $\mathcal{I}$ be an interpretation such that $\mathcal{I} \models \mathcal{T}_1$, $C^{\mathcal{I}} \neq \emptyset$, and $A_1^{\mathcal{I}} \neq A_2^{\mathcal{I}}$. We construct an interpretation $\mathcal{I}'$ from $\mathcal{I}$ such that $\mathcal{I}' \models \mathcal{T}_1$, $C^{\mathcal{I}'} \neq \emptyset$, and $A_1^{\mathcal{I}} = A_2^{\mathcal{I}}$. For $x \in \Delta^{\mathcal{I}}$ such that $x \in A_1^{\mathcal{I}} \cup A_2^{\mathcal{I}}$ and $x \notin A_1^{\mathcal{I}} \cap A_2^{\mathcal{I}}$, we show that we can remove x from either $A_1^{\mathcal{I}}$ or $A_2^{\mathcal{I}}$ so that $x \notin A_1^{\mathcal{I}} \cup A_2^{\mathcal{I}}$ without causing any axiom in $\mathcal{T}_1$ to fail to hold. Without loss of generality, assume $x \in A_1^{\mathcal{I}}$ and $x \notin A_2^{\mathcal{I}}$. If $x \in C_1$ and $x \in C_2$, then we have a contradiction. Otherwise, we remove x from $A_1^{\mathcal{I}}$. Since either $x \notin C_1^{\mathcal{I}}$ or $x \notin C_2^{\mathcal{I}}$, the axiom $(C_1 \sqcap C_2) \sqsubseteq A_1$ holds. No other axiom in $\mathcal{T}_1$ has A_1 on the right hand side, therefore removing x from $A_1^{\mathcal{I}}$ does not cause any other axiom to fail to hold. Since the above is true for all x such that x is in only one of $A_1^{\mathcal{I}}$ and $A_2^{\mathcal{I}}$, we may remove individuals from $A_1^{\mathcal{I}}$ and $A_2^{\mathcal{I}}$ until $A_1^{\mathcal{I}} = A_2^{\mathcal{I}}$. Then the first case applies.

Now we prove the only if direction. Let $\mathcal{I}$ be an interpretation such that $\mathcal{I} \models \mathcal{T}_2$ and $C^{\mathcal{I}} \neq \emptyset$. We construct an interpretation $\mathcal{I}'$ from $\mathcal{I}$ such that $\mathcal{I}' \models \mathcal{T}_1$ and $C^{\mathcal{I}'} \neq \emptyset$. First set $\mathcal{I}' = \mathcal{I}$. Then, set $A_1^{\mathcal{I}'} = A_2^{\mathcal{I}'} = A^{\mathcal{I}}$. Due to the construction of $\mathcal{T}_2$, $\mathcal{I}' \models \mathcal{T}_1$ and $C^{\mathcal{I}'} \neq \emptyset$. □

Lemma 4.3. *Let $\mathcal{T}_1, \mathcal{T}_2$, and $\mathcal{T}$ denote TBoxes, $C \in$ L an arbitrary concept, A a primitive concept not used in C or $\mathcal{T}$, and R a role. If $\mathcal{T}_1$ is of the form*

$$\mathcal{T}_1 = \mathcal{T} \cup \{\exists R.C_1 \sqsubseteq C_2\},$$

then C is satisfiable with respect to $\mathcal{T}_1$ iff C is satisfiable with respect to TBox

$$\mathcal{T}_2 = \mathcal{T} \cup \{C_1 \sqsubseteq \forall R^-.A, A \sqsubseteq C_2\}.$$

Proof. First we prove the if direction. Assume C is satisfiable with respect to $\mathcal{T}_1$. For an interpretation $\mathcal{I} \in$ Int(L) such that $\mathcal{I} \models \mathcal{T}_1$ and $C^{\mathcal{I}} \neq \emptyset$, we extend $\mathcal{I}$ to an interpretation $\mathcal{I}'$ such that $\mathcal{I}' \models \mathcal{T}_2$ and $C^{\mathcal{I}'} \neq \emptyset$. First set $\mathcal{I}' = \mathcal{I}$. For each $x \in \Delta^{\mathcal{I}}$ such that $x \in (\exists R.C_1)^{\mathcal{I}} \cap C_2^{\mathcal{I}}$, we add x to $A^{\mathcal{I}'}$. Then, $\mathcal{I}' \models \mathcal{T}_2$ and $C^{\mathcal{I}'} \neq \emptyset$.

Now we prove the only if direction. Assume C is satisfiable with respect to $\mathcal{T}_2$. For each interpretation $\mathcal{I} \in$ Int(L) such that $\mathcal{I} \models \mathcal{T}_2$ and $C^{\mathcal{I}} \neq \emptyset$, it is also the case that $\mathcal{I} \models \mathcal{T}_1$. The proof is by contradiction. Assume $\mathcal{I} \not\models \mathcal{T}_1$. It must be the case that axiom $\exists R.C_1 \sqsubseteq C_2$ does not hold as all other axioms in $\mathcal{T}_1$ are also in $\mathcal{T}_2$. Then there exists $x \in \Delta^{\mathcal{I}}$ such that $x \in (\exists R.C_1)^{\mathcal{I}}$ and $x \notin C_2^{\mathcal{I}}$. However, this implies that there exists $y \in \Delta^{\mathcal{I}}$ such that $(x, y) \in R^{\mathcal{I}}$ and $y \in C_1$. From axiom $C_1 \sqsubseteq \forall R^-.A$, it must be the case that $x \in A^{\mathcal{I}}$. From axiom $A \sqsubseteq C_2$, it must be the case that $x \in C_2^{\mathcal{I}}$. A contradiction. □

Lemma 4.4. *Let $\mathcal{T}_1, \mathcal{T}_2$, and $\mathcal{T}$ denote TBoxes, $C \in$ L an arbitrary concept, A a primitive concept not used in C or $\mathcal{T}$, and R a role. If $\mathcal{T}_1$ is of the form*

$$\mathcal{T}_1 = \mathcal{T} \cup \{\exists R^-.C_1 \sqsubseteq C_2\},$$

then C is satisfiable with respect to $\mathcal{T}_1$ iff C is satisfiable with respect to TBox

$$\mathcal{T}_2 = \mathcal{T} \cup \{C_1 \sqsubseteq \forall R.A, A \sqsubseteq C_2\}.$$

The proof of this lemma is similar to that of Lemma 4.3.

Theorem 4.1. *For any TBox $\mathcal{T}$, the binary absorption algorithm computes a correct absorption of $\mathcal{T}$.*

Proof. The proof is by induction on the number of iterations of our algorithm. We define an iteration to end when we return to Step 1. We abbreviate the pair

$$(\{A \sqsubseteq C, \neg A \sqsubseteq \neg C \mid A \doteq C \in \mathcal{T}_{prim}\} \cup \mathcal{T}_{uinc} \cup \mathcal{T}_{binc}, \mathcal{T}_g \cup \mathcal{T})$$

as $\mathcal{T}$ and claim that this pair is always a correct binary absorption.

Initially, $\mathcal{T}_{uinc}$, $\mathcal{T}_{binc}$, and $\mathcal{T}_g$ are empty, primitive axioms are in $\mathcal{T}_{prim}$, and the remaining axioms are in $\mathcal{T}$. By Lemma 3.1, Lemma 3.2, Lemma 3.3, and Lemma 3.4, $\mathcal{T}$ is a correct binary absorption at the start of our algorithm.

Assume we just finish iteration i and now perform iteration $i+1$. By our induction hypothesis, $\mathcal{T}$ is a correct binary absorption. We have a number of possible cases.

- If we perform Step 3(a) or Step 3(b) then iteration $i+1$ is finished. Due to the ordering of Step 3(a) and Step 3(b), newly introduced primitive concepts are absorbed first. Therefore, a newly introduced primitive concept only appears on the right hand side of an axiom once and Lemma 4.1 and Lemma 4.2 apply. We conclude that $\mathcal{T}$ is a correct binary absorption.
- If we perform Step 3(c), then iteration $i+1$ is finished and by Lemma 4.3, $\mathcal{T}$ is a correct binary absorption.
- If we perform Step 3(d), then iteration $i+1$ is finished and by Lemma 4.4, $\mathcal{T}$ is a correct binary absorption.
- If we perform any of Steps 1, 2, 5, or 6, then $\mathcal{T}$ is a correct binary absorption at the end of iteration $i+1$. This is because Steps 1, 2, 5, and 6 use only equivalence preserving operations.

After the final iteration of our algorithm, $\mathcal{T}$ is a correct binary absorption by mathematical induction. □

5 Summary

We have proposed a simple and straightforward generalization of the absorption theory and algorithms pioneered by Horrocks and Tobies [6, 7]. Called *binary absorption*, the basic idea is to allow terminological axioms of the form

$$(A_1 \sqcap A_2) \sqsubseteq C$$

to qualify for lazy unfolding in model building satisfaction procedures for description logics. An important issue for future work is to evaluate the efficacy of our binary absorption algorithm on real-world problems. Our immediate plans in this direction are to measure the reductions in the number of disjunctions in comparison to (basic) absorption for publicly accessible OWL DL ontologies.

References

[1] F. Baader, D. Calvanese, D. McGuinness, D. Nardi, and P. Patel-Schneider, editors. *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge Univserity Press, 2003.

[2] F. Baader, E. Franconi, B. Hollunder, B. Nebel, and H.-J. Profitlich. An empirical analysis of optimization techniques for terminological representation systems, or: Making KRIS get a move on. *Applied Artificial Intelligence*, 4:109–132, 1994.

[3] E. D. Demaine, A. Lopez-Ortiz, and I. Munro. Adaptive set intersections, unions, and differences. In *SODA '00: Proceedings of the eleventh annual ACM-SIAM symposium on Discrete algorithms*, pages 743–752, Philadelphia, PA, USA, 2000. Society for Industrial and Applied Mathematics.

[4] A. Deutsch, L.Popa, and V. Tannen. Query reformulation with constraints. *ACM SIGMOD Record*, 35(1):65–73, 2006.

[5] I. Horrocks. Using an Expressive Description Logic: FaCT or Fiction? In A. G. Cohn, L. Schubert, and S. C. Shapiro, editors, *Principles of Knowledge Representation and Reasoning: Proceedings of the Sixth International Conference (KR'98)*, pages 636–647. Morgan Kaufmann Publishers, San Francisco, California, 1998.

[6] I. Horrocks and S. Tobies. Optimisation of terminological reasoning. In *Proc. of the 2000 Description Logic Workshop (DL 2000)*, pages 183–192, 2000.

[7] I. Horrocks and S. Tobies. Reasoning with axioms: Theory and practice. In *Proc. of the 7th Int. Conf. on Principles of Knowledge Representation and Reasoning (KR 2000)*, pages 285–296, 2000.

[8] D. Tsarkov and I. Horrocks. Efficient reasoning with range and domain constraints. In Volker Haarslev and Ralf Möller, editors, *Description Logics*, volume 104 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2004.

[9] J. A. P. Villa. CGU: A common graph utility for DL reasoning and conjunctive query optimization. Master's thesis, David R. Cheriton School of Computer Science, University of Waterloo, 2005.

A Description Logic of Change

Alessandro Artale	University of Bolzano, IT	artale@inf.unibz.it
Carsten Lutz	University of Dresden, DE	clu@tcs.inf.tu-dresden.de
David Toman	University of Waterloo, CA	david@uwaterloo.ca

Abstract

We combine the modal logic S5 with the description logic (DL) $\mathcal{ALCQI}$. In this way, we obtain a multi-dimensional DL, $\mathsf{S5}_{\mathcal{ALCQI}}$, whose purpose is reasoning about change. $\mathsf{S5}_{\mathcal{ALCQI}}$ is capable of expressing that concepts and roles change over time, but cannot discriminate between changes in the past and in the future. Our main technical result is that $\mathsf{S5}_{\mathcal{ALCQI}}$ concept satisfiability with respect to terminologies of general concepts inclusions (GCIs) is decidable and 2-ExpTime-hard. We also provide a scenario based on temporal conceptual models with timestamping constraints in which the logic can be used.

1 Introduction

An important application of Temporal Description Logics (TDLs) is the representation of and reasoning about temporal conceptual models [1, 3, 4]. The general idea is that such models are translated into an appropriate TDL, and that TDL reasoning is then used to detect inconsistencies and implicit IS-A relations in the temporal model [2, 4, 6]. However, a serious obstacle for putting this general idea to work is posed by the fact that, for many natural temporal conceptual formalisms and their associated TDLs, reasoning turns out to be undecidable.

The most prominent witness of this problem is constituted by temporal entity-relationship (TER) models used to design temporal databases [8]. TERs are classical ER data models extended with two classes of constraints that model the temporal behavior of an application domain [13], i.e., *evolution constraints*—to govern *object migration*, i.e., the inability, possibility, and necessity of objects to change membership from one entity to another—and *timestamping*—to distinguish between temporal and atemporal components of a TER model.

In this paper, we are interested in devising a logic able to fully capture TER with timestamping constraints and study its computational properties. Indeed,

timestamping are temporal constructs supported by almost all temporal conceptual models proposed in the literature [10, 11, 13, 14]. It has be implemented either by marking entities (i.e., classes), relationships and attributes as *snapshot* or *temporary*, or leaving them un-marked. Then, objects belong to a snapshot entity either never or at all times, no object may belong to a temporary entity at all times, and there are no (temporal) assumptions about instances of unmarked entities. The meaning of timestamps for relationships and attributes is analogous.

It has been observed in [4] that TER models with timestamping and evolution constraints can be translated into the TDL $\mathsf{DLR}_{\mathcal{US}}$. In the same paper, undecidability of $\mathsf{DLR}_{\mathcal{US}}$ was established. Later, it was shown in [1] that these computational problems are not due to the translation to description logics: even direct reasoning with the (less powerful) TER models is undecidable. There appear to be two ways around this problem. First, one can restrict timestamping. Indeed, it has been shown in [4] that giving up timestamping of both relationships and attributes (in the sense that all relationships and attributes are left un-marked) re-establishes decidability of $\mathsf{DLR}_{\mathcal{US}}$, and thus also of TER models with both restricted timestamping and evolution constraints. Second, one can allow for full timestamping but avoiding evolution constraints.

This second alternative is pursued in the current paper. We devise a multi-dimensional description logic $\mathsf{S5}_{\mathcal{ALCQI}}$ that is obtained by combining modal $\mathsf{S5}$ with the standard DL $\mathcal{ALCQI}$. The $\mathsf{S5}$ modality can be applied to both concepts and roles but not in front of axioms—terminological axioms are interpreted globally. This logic may be viewed as a *description logic of change*, as it allows to state that concept and role memberships change in time, but does not allow to discriminate between changes in the past and changes in the future. We show that TER models with full timestamping (i.e., timestamping on entities, relationships and attributes) but without evolution constraints can be captured by $\mathsf{S5}_{\mathcal{ALCQI}}$ terminologies. The main result of this paper is to show that reasoning in $\mathsf{S5}_{\mathcal{ALCQI}}$ is decidable and 2-ExpTime-hard. While a decidability result was obtained for the simpler modal DL, $\mathsf{S5}$-$\mathcal{ALC}$ [9], this is the first time that the more complex $\mathsf{S5}_{\mathcal{ALCQI}}$ is showed to have a decidable reasoning problem. Thus, dropping evolution constraints indeed recovers decidability. However, it is surprising that adding change, a rather weak form of temporality, pushes the complexity of $\mathcal{ALCQI}$ from ExpTime-complete to 2-ExpTime-hard.

2 The Logic $\mathsf{S5}_{\mathcal{ALCQI}}$

The logic $\mathsf{S5}_{\mathcal{ALCQI}}$ is a combination of the epistemic modal logic $\mathsf{S5}$ and the description logic $\mathcal{ALCQI}$. It is similar in spirit to the multi-dimensional description logics proposed, e.g., in [9, 16]. Let $\mathsf{N_C}$ and $\mathsf{N_R}$ be disjoint and countably

infinite sets of *concept names* and *role names*. We assume that $\mathsf{N_R}$ is partitioned into two countably infinite sets $\mathsf{N_{glo}}$ and $\mathsf{N_{loc}}$ of *global role names* and *local role names*. The set ROL of *roles* is defined as $\{r, r^-, \Diamond r, \Diamond r^-, \Box r, \Box r^-\}$, with $r \in \mathsf{N_R}$. The set of concepts CON is defined inductively: $\mathsf{N_C} \subseteq \mathsf{CON}$; if $C, D \in \mathsf{CON}$, $r \in \mathsf{ROL}$, and $n \in \mathbb{N}$, then the following are also in CON: $\neg C$, $C \sqcap D$, $(\geqslant n\ r\ C)$, and $\Diamond C$. A *TBox* is a finite set of *general concept inclusions (GCIs)* $C \sqsubseteq D$ with $C, D \in \mathsf{CON}$.

The concept constructors $C \sqcup D$, $\exists r.C$, $(\leqslant n\ r\ C)$, $(= n\ r\ C)$, $\forall r.C$, $\Box C$, $\top$, and $\bot$ are defined as abbreviations in the usual way. Concerning roles, note that we allow only single applications of boxes and diamonds, while inverse is applicable only to role names. It is easily seen that any role obtained by nesting modal operators and inverse in an arbitrary way can be converted into an equivalent role in this restricted form: multiple temporal operators are absorbed and inverse commutes over temporal operators [5].

An $\mathsf{S5}_{\mathcal{ALCQI}}$*-interpretation* $\mathfrak{I}$ is a pair $(W, \mathcal{I})$ with W a non-empty set of *worlds* and $\mathcal{I}$ a function assigning to each $w \in W$ an $\mathcal{ALCQI}$*-interpretation* $\mathcal{I}(w) = (\Delta, \cdot^{\mathcal{I},w})$, where the *domain* Δ is a non-empty set and $\cdot^{\mathcal{I},w}$ is a function mapping each $A \in \mathsf{N_C}$ to a subset $A^{\mathcal{I},w} \subseteq \Delta$ and each $r \in \mathsf{N_R}$ to a relation $r^{\mathcal{I},w} \subseteq \Delta \times \Delta$, such that if $r \in \mathsf{N_{glo}}$, then $r^{\mathcal{I},w} = r^{\mathcal{I},v}$ for all $w, v \in W$. We can extend the mapping $\cdot^{\mathcal{I},w}$ to complex roles and concepts as follows:

$$
\begin{aligned}
(r^-)^{\mathcal{I},w} &:= \{(y,x) \in \Delta \times \Delta \mid (x,y) \in r^{\mathcal{I},w}\} \\
(\Diamond r)^{\mathcal{I},w} &:= \{(x,y) \in \Delta \times \Delta \mid \exists v \in W : (x,y) \in r^{\mathcal{I},v}\} \\
(\Box r)^{\mathcal{I},w} &:= \{(x,y) \in \Delta \times \Delta \mid \forall v \in W : (x,y) \in r^{\mathcal{I},v}\} \\
(\neg C)^{\mathcal{I},w} &:= \Delta \setminus C^{\mathcal{I},w} \\
(C \sqcap D)^{\mathcal{I},w} &:= C^{\mathcal{I},w} \cap D^{\mathcal{I},w} \\
(\geqslant n\ r\ C)^{\mathcal{I},w} &:= \{x \in \Delta \mid \sharp\{y \in \Delta \mid (x,y) \in r^{\mathcal{I},w} \text{ and } y \in C^{\mathcal{I},w}\} \geq n\} \\
(\Diamond C)^{\mathcal{I},w} &:= \{x \in \Delta \mid \exists v \in W : x \in C^{\mathcal{I},v}\}
\end{aligned}
$$

An $\mathsf{S5}_{\mathcal{ALCQI}}$-interpretation $\mathfrak{I} = (W, \mathcal{I})$ is a *model* of a TBox $\mathcal{T}$ iff it satisfies $C^{\mathcal{I},w} \subseteq D^{\mathcal{I},w}$ for all $C \sqsubseteq D \in \mathcal{T}$ and $w \in W$. It is a *model* of a concept C if $C^{\mathcal{I},w} \neq \emptyset$ for some $w \in W$.

Note that $\mathsf{S5}_{\mathcal{ALCQI}}$ does not have the finite model property: there are concepts and TBoxes that are only satisfiable in models with both an infinite set of worlds and an infinite domain. For example, this is true for the concept $\neg C$ w.r.t. the TBox $\{\neg C \sqsubseteq \Diamond C,\ C \sqsubseteq \exists r.\neg C,\ \neg C \sqsubseteq \forall r.\neg C\}$, where $r \in \mathsf{N_{glo}}$.

3 Capturing Conceptual Schemas

It is known that the TDL $\mathcal{ALCQI}_{\mathcal{US}}$ is able to capture the temporal conceptual model $\mathcal{ER}_{VT}$, a TER model that supports timestamping and evolution

constraints, IS-A links, disjointness and covering constraints, participation constraints [3]. In $\mathcal{ER}_{VT}$, timestamping is implemented using a marking approach as sketched in the introduction. Since the translation of atemporal constructs is similar to the one using $\mathcal{ALCQI}_{\mathcal{US}}$ the reader must refer to [3] for full details and examples. In the following, after recalling the translation of atemporal constructs, we show that $\mathsf{S5}_{\mathcal{ALCQI}}$ is sufficient to capture the fragment of $\mathcal{ER}_{VT}$ that has timestamping as the only temporal construct.

When translating $\mathcal{ER}_{VT}$ to TDLs, entities E—denoting sets of abstract objects—are mapped into concept names A_E while attributes P—denoting functions associating mandatory concrete properties to entities—are mapped into roles names r_P that are enforced to be interpreted as total functions using GCIs: $\top \sqsubseteq (= 1\ r_P\ \top)$. In $\mathsf{S5}_{\mathcal{ALCQI}}$, un-marked entities and attributes need no special treatment, while entities and attributes being of type snapshot or temporary can be expressed as follows:

$A_E \sqsubseteq \Box A_E$	*snapshot entity*
$A_E \sqsubseteq \Diamond \neg A_E$	*temporary entity*
$A_E \sqsubseteq \exists \Box r_P.\top$	*snapshot attribute*
$A_E \sqsubseteq \forall \Box r_P.\bot$	*temporary attribute*

Relationships—denoting n-ary relations between abstract objects—are translated by reification, i.e., each n-ary relationship R is translated into a concept name A_R with n *global* role names $r_1, \ldots, r_n$. Intuitively, for each instance $x \in A_R^{\mathcal{I},w}$, the tuple $(y_1, \ldots, y_n)$ with $(x, y_i) \in r_i^{\mathcal{I},w}$ is a tuple in the relationship R at time point w. To ensure that every instance of A_R gives rise to a unique tuple in R, we use GCIs $\top \sqsubseteq (= 1\ r_i\ \top)$, for $1 \leq i \leq n$. Now, to capture *snapshot relationships*, we simply put $A_R \sqsubseteq \Box A_R$, while for *temporary relationships*, we put $A_R \sqsubseteq \Diamond \neg A_R$.

Note that, the latter GCI does not fully capture temporary relationships. As an example, consider the interpretation $\mathfrak{I} = (\{w_1, w_2\}, \mathcal{I})$, with $\Delta = \{a, a', b, c\}$, $A_R^{\mathcal{I},w_1} = \{a\}$, $A_R^{\mathcal{I},w_2} = \{a'\}$, $r_1^{\mathcal{I},w_1} = \{(a, b)\}$, $r_2^{\mathcal{I},w_1} = \{(a, c)\}$, $r_1^{\mathcal{I},w_2} = \{(a', b)\}$, and $r_2^{\mathcal{I},w_2} = \{(a', c)\}$. Although the GCI $A_R \sqsubseteq \Diamond \neg A_R$ (expressing temporary relationships) is satisfied, (b, c) is constantly in the temporary relationship R. This is due to a mismatch between the models of an $\mathcal{ER}_{VT}$ schema and the models of its translation into $\mathsf{S5}_{\mathcal{ALCQI}}$. In particulars, in models of $\mathcal{ER}_{VT}$, tuples belonging to relationships are unique while in models of the reified translation there may be two *distinct* objects connected through the r_i global roles to the *same* objects, thus representing the same tuple, e.g., the distinct objects a, a' in the above interpretation. Then, $\mathsf{S5}_{\mathcal{ALCQI}}$ models verifying the above situation do not correspond directly to an $\mathcal{ER}_{VT}$ model. However, similarly to [7], it is possible to show that: *(i)* there are so called *safe* models of $\mathsf{S5}_{\mathcal{ALCQI}}$ that are in one-to-one correspondence with $\mathcal{ER}_{VT}$ models, *(ii)* every satisfiable $\mathsf{S5}_{\mathcal{ALCQI}}$ concept is also satisfied in a safe model. Since we are interested in reasoning

about $\mathcal{ER}_{VT}$ schemas we can thus forget about non-safe models. An $\mathsf{S5}_{\mathcal{ALCQI}}$ interpretation $\mathfrak{I} = (W, \mathcal{I})$ is *safe* for an $\mathcal{ER}_{VT}$ schema if, for every n-ary relationship R reified with the global functional roles r_i, and every $w \in W$ we have the following:

$$\forall x, y, x_1, \ldots, x_n \in \Delta : \neg((x, x_1) \in r_1^{\mathcal{I},w} \wedge (y, x_1) \in r_1^{\mathcal{I},w} \wedge \ldots \wedge (x, x_n) \in r_n^{\mathcal{I},w} \wedge (y, x_n) \in r_n^{\mathcal{I},w}).$$

It is not hard to see that: *(1)* the model in the example above is not safe, *(2)* given a safe model, the above GCI for temporary relationships correctly capture the safety property.

4 Decidability of Reasoning in $\mathsf{S5}_{\mathcal{ALCQI}}$

We show that the satisfiability problem is decidable for $\mathsf{S5}_{\mathcal{ALCQI}}$. For simplicity, throughout this section we assume that only local role names are used. This can be done w.l.o.g. since global role names can be simulated by $\Box r$, where r is a fresh local role name. To prove decidability, we start with devising tree abstractions of $\mathsf{S5}_{\mathcal{ALCQI}}$ models. Then we show how, given a concept C_0 and TBox $\mathcal{T}$, we can construct a looping tree automaton accepting exactly the tree abstractions of models of C_0 and $\mathcal{T}$.

Let C_0 and $\mathcal{T}$ be a concept and a TBox whose satisfiability is to be decided. We first introduce the following notation. For roles r, we use $\mathsf{Inv}(r)$ to denote r^- if $r \in \mathsf{N_R}$, s if $r = s^-$, $\Diamond\mathsf{Inv}(s)$ if $r = \Diamond s$, and $\Box\mathsf{Inv}(s)$ if $r = \Box s$. We use $\mathsf{rol}(C_0, \mathcal{T})$ to denote the smallest set that contains all roles used in C_0 and $\mathcal{T}$, and that is closed under Inv. We use $\mathsf{cl}(C_0, \mathcal{T})$ to denote the smallest set containing all sub-concepts appearing in C_0 and $\mathcal{T}$ closed under negation, if $C \in \mathsf{cl}(C_0, \mathcal{T})$ and "$\neg$" is not the top level operator in C, then $\neg C \in \mathsf{cl}(C_0, \mathcal{T})$.

4.1 Tree Abstractions of $\mathsf{S5}_{\mathcal{ALCQI}}$ models

The goal of this section is to develop *tree abstractions* of general $\mathsf{S5}_{\mathcal{ALCQI}}$ models: for a given model $\mathfrak{I}$ of C_0 and $\mathcal{T}$, we develop a *tree abstraction* called $(C_0, \mathcal{T})$-tree. The root node corresponds to the object that realizes C_0 in $\mathfrak{I}$, descendants of the root correspond to further objects in $\mathfrak{I}$ that can be reached by traversing roles in *some* S5 world staring from the *root* object. For the abstractions to capture the essence of the $\mathsf{S5}_{\mathcal{ALCQI}}$ models, we need to attach additional information to the nodes of the tree and to constrain the parent-child relationships. To this end we develop the notions of an *extended quasistate* and *matching successor*. In the rest of this section we formalize the above intuition.

We first introduce types and quasistates. Intuitively, a type describes the concept memberships of a domain element $x \in \Delta$ in a single $\mathsf{S5}$ world.

Definition 1 (Type). A *type t for $C_0, \mathcal{T}$* is a subset of $\mathsf{cl}(C_0, \mathcal{T})$ such that

$$\begin{array}{rlll} \neg C \in t & \text{iff} & C \notin t & \text{for all } \neg C \in \mathsf{cl}(C_0, \mathcal{T}) \\ C \sqcap D \in t & \text{iff} & C \in t \text{ and } D \in t & \text{for all } C \sqcap D \in \mathsf{cl}(C_0, \mathcal{T}) \\ C \in t & \text{implies} & D \in t & \text{for all } C \sqsubseteq D \in \mathcal{T} \end{array}$$

We use $\mathsf{tp}(C_0, \mathcal{T})$ to denote the set of all types for $C_0, \mathcal{T}$. To describe the concept memberships of a domain element in *all* $\mathsf{S5}$ worlds, we use quasistates:

Definition 2 (Quasistate). Let W be a set and $f : W \to \mathsf{tp}(C_0, \mathcal{T})$ a function such that for all $w \in W$ we have:

$$\Diamond C \in f(w) \text{ iff } C \in f(v) \text{ for some } v \in W.$$

We call the pair (W, f) a *quasistate witness* and the set $\{f(v) \mid v \in W\}$ a *quasistate*.

In particular, quasistates capture constraints implied by the S5 modalities on concept membership of an object in all worlds. To check whether a set of types, $\{t_1, \ldots, t_n\}$, is a quasistate we simply check whether the pair (W, f), with $W = \{t_1, \ldots, t_n\}$ and f the identity function is a quasistate witness. Note, however, that each quasistate has many witnesses.

To abstract the role structure of a model we define the notion of an *extended quasistate*. This abstraction is realized by a pair of quasistates in a $(C_0, \mathcal{T})$-*trees* and captures how two objects are related by a particular role.

Definition 3 (Extended Quasistate). Let W be a set, $f, g : W \to \mathsf{tp}(C_0, \mathcal{T})$, and $h : W \to \mathsf{rol}(C_0, \mathcal{T}) \cup \{\epsilon\}$ for $\epsilon \notin \mathsf{rol}(C_0, \mathcal{T})$ such that

1. (W, f) and (W, g) are quasistate witnesses;
2. if $\Diamond r = h(w)$ for some $w \in W$, then $r = h(v)$ for some $v \in W$;
3. if $r = h(w)$ for some $w \in W$, then $\Diamond r = h(v)$ or $r = h(v)$ for all $v \in W$;
4. it is not the case that $r = h(w)$ for all $w \in W$;
5. if $\Box r = h(w)$ for some $w \in W$, then $\Box r = h(v)$ for all $v \in W$.

We call (W, f, g, h) an *extended quasistate witness* and the set of triples

$$Q(W, f, g, h) = \{(f(v), g(v), h(v)) \mid v \in W\}$$

an *extended quasistate*.

We say that $Q(W, f, g, h)$ *realizes a concept* C if: $C \in f(w)$ for some $w \in W$; we say that $Q(W, f, g, h)$ *is root* if: $h(w) = \Box\epsilon$ for all $w \in W$.

The extended quasistates therefore enforce S5 modalities when applied to roles. In thee tree-shaped model abstraction we also use extended quasistates capture the idea that each object has exactly one parent: by convention we assume that the object abstracted by f has a parent abstracted by g. The two objects are connected by a role r that is abstracted by h in those worlds w in which $h(w) \in \{r, \Box r\}$. For uniformity, we use the ϵ dummy role for the root object[1]. We define an ordering between modalities of a basic role r as $\Diamond r \leq r \leq \Box r$, this arrangement allows us to use a single role in the extended quasistate to capture all the *implied* modalities. It is again immediate to verify whether a set of triples forms an extended quasistate.

The last ingredient needed in the tree abstractions is the ability to properly capture the effects of *number restrictions*. These constraints, given an object in a model, impact what and how many other objects can be connected to this object via roles (and in which worlds). In the tree abstraction this yields restrictions on which extended quasistates can possibly be children of a given quasistate. This intuition is captured by the notion of a matching successor:

Definition 4 (Matching Successor). Let W and O be sets, o an element such that $o \notin O$; $f, g : (O \cup \{o\}) \to W \to \mathsf{tp}(C_0, \mathcal{T})$, and $h : (O \cup \{o\}) \to W \to \mathsf{rol}(C_0, \mathcal{T})$ functions such that $(W, f(p), g(p), h(p))$ is an extended quasistate witness for all $p \in O \cup \{o\}$ and $f(o) = g(p)$ for all $p \in O$. We call (W, O, o, f, g, h) a *matching successor witness* if for all $w \in W$:

1. if $(\geqslant n\ r\ C) \in f(o)(w)$ and $C \notin g(o)(w)$ or $\mathsf{Inv}(r) \not\leq h(o)(w)$ then $|\{p \in O \mid r \leq h(p)(w), C \in f(p)(w)\}| \geq n$,

2. if $(\geqslant n\ r\ C) \in f(o)(w)$, then $|\{p \in O \mid r \leq h(p)(w), C \in f(p)(w)\}| \geq n-1$,

3. if $(\geqslant n\ r\ C) \in \mathsf{cl}(\mathcal{T}, C_0)$, $C \in g(o)(w)$, and $\mathsf{Inv}(r) \leq h(o)(w)$, and $|\{p \in O \mid r \leq h(p)(w), C \in f(p)(w)\}| \geq n-1$ then $(\geqslant n\ r\ C) \in f(o)(w)$,

4. if $(\geqslant n\ r\ C) \in \mathsf{cl}(\mathcal{T}, C_0)$ and $|\{p \in O \mid r \leq h(p)(w), C \in f(p)(w)\}| \geq n$ then $(\geqslant n\ r\ C) \in f(o)(w)$,

We call the pair $(Q(W, f(o), g(o), h(o)), \{Q(W, f(p), g(p), h(p)) \mid p \in O\})$ a *matching successor*. We say that two matching successor witnesses are *equivalent* if they define the same matching successor.

[1] Note that each node in the abstraction is labeled by an extended quasistate describing the corresponding object, its parent, and the role that connects them. This arrangement is needed in order to keep track of how inverses interact with number restrictions and is similar to the so-called *double-blocking* technique.

The intuition behind this definition is as follows: the object o stands for the parent node and the set of objects O for all its children in the tree abstraction. Each of these objects is labeled by an extended quasistate in a consistent way (i.e., the parent parts of the extended quasistates labeling the children match the quasistate attached to the parent). A *matching successor witness* is then a witness that the extended quasistates attached to o and to all elements of O can potentially be used to build a part of a model of C_0 and $\mathcal{T}$ without violating any constraints in $\mathcal{T}$. Thus we can use matching successors to define the sought after tree shaped abstraction of models of C_0 and $\mathcal{T}$.

Definition 5 ($(C_0, \mathcal{T})$-tree). Let $T = (n_0, N, E)$ be an ordinal tree with root $n_0 \in N$ and G a mapping of T's nodes to extended quasistates. We say that T is a $(C_0, \mathcal{T})$-tree if:

1. C_0 is realized in $G(n_0)$,
2. $G(n_0)$ is root,
3. for all $n \in N$ the pair $(G(n), \{G(m) \mid (n, m) \in E\})$ is a matching successor.

To be able to eventually construct a model from our abstraction we use the following lemma that allows us to *concatenate* matching successor witnesses along branches of such a tree abstraction:

Lemma 6. *Let (W, O, o, f, g, h) be a matching successor witness for (q, Q) and α an infinite cardinal. Then there is an equivalent matching successor witness (W', O, o, f', g', h') such that*

$$|\{w \in W' \mid (f'(p)(w), g'(p)(w), h'(p)(w)) \text{ is a constant triple}\}| = \alpha$$

for all $p \in O \cup \{o\}$.

This can always be achieved by replicating elements of W sufficiently many times. These witnesses are convenient as whenever the associated extended quasistate match they can be *plugged* one into the bottom of another just by permuting the set W.

The intuitions behind Definitions 4 and 5 is as follows: when given a model $\mathfrak{I}$ of C_0 and $\mathcal{T}$, we use the object $o \in \Delta$ that realizes C_0 in $\mathfrak{I}$ to define the extended quasistate for the root of our tree abstraction; we then collect all the extended quasistates for all successors of o in $\mathfrak{I}$ and make them children of the root quasistate. Repeating this construction yields a $(C_0, \mathcal{T})$-tree: the children of every node have been extracted from $\mathfrak{I}$ and thus must satisfy the conditions on matching successors (the witness comes from the model). Conversely, given a $(C_0, \mathcal{T})$-tree, we use the fact that for each matching successor, there must be a witness. We use these witnesses, with the help of Lemma 6, to construct a model by traversing the $(C_0, \mathcal{T})$-tree top down while concatenating the matching successor witnesses found along the branches. Thus the existence of a $(C_0, \mathcal{T})$-tree is equivalent to the existence of a $(C_0, \mathcal{T})$ model:

Theorem 7. *C_0 is satisfiable w.r.t. $\mathcal{T}$ iff there exists a $(C_0, \mathcal{T})$-tree.*

4.1.1 Decidability and Looping Tree Automata

To show decidability, we need to show existence of at least one $(C_0, \mathcal{T})$-tree. We proceed in two steps:

1. We need to determine whether (q, Q) is a *matching successor* for each extended quasistate q and set of extended quasistates Q.

2. We need to show that the matching successors can be arranged into a tree rooted by a node labeled by an extended quasistate realizing C_0.

Thus, we first need to define a procedure that constructs all matching successors given C_0 and $\mathcal{T}$. To show this, we need the following lemma, where $\max(C_0, \mathcal{T}) := \sum_{(\geqslant m\ r\ C) \in \mathsf{cl}(C_0, \mathcal{T})} m$, and $n = |\mathsf{cl}(C_0, \mathcal{T})|$.

Lemma 8. *Let (W, O, o, f, g, h) be a matching successor witness for a matching successor (q, Q). Then there is an equivalent matching successor witness (W', O', o, f', g', h') such that $|O'| \leq (\max(C_0, \mathcal{T})+1) \cdot 2^{n2^{2n}}$ and $|W'| \leq 2^{|O'| \cdot (n+1)}$.*

Figure 1 illustrates why the bounded matching successor witness required by

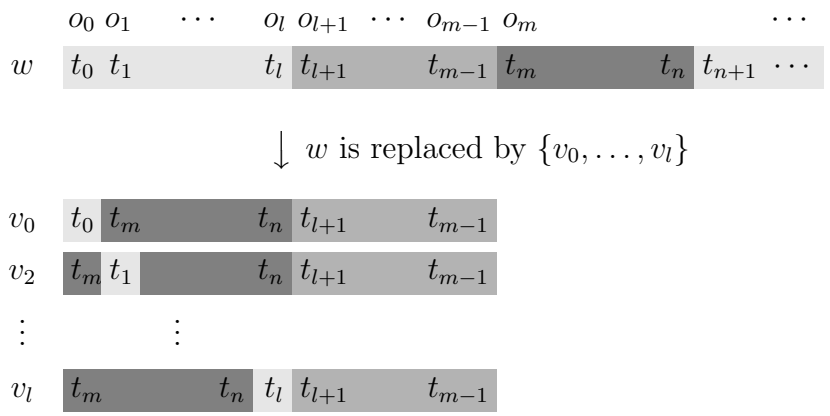


Figure 1: Reducing the size of a Matching Successor Witness.

Lemma 8 must exist: consider that the objects $o_0, \ldots, o_{n+1} \ldots \in O$ have been assigned the same extended quasistate by the current witness and that there are more than $\max(C_0, \mathcal{T}) + 1$ of such objects. Now consider a world w depicted on the top of the Figure. Assume that the (types of the) objects $o_{l+1}, \ldots, o_n$ are those needed to fulfill a number restriction in the (type of the) parent node in w. These must be *preserved* in the new witness after removing superfluous

objects. The transformation is depicted in the lower half of the figure: we create a *set of worlds* $v_0, \ldots, v_l$ with the same parent label and such that all the types $t_{l+1}, \ldots, t_n$ are present in *every* of the new worlds. Hence the number restriction in the parent's type still holds. Moreover, the original type assigned to the objects $o_0, \ldots, o_l$ is preserved in *at least one* of the new worlds. This guarantees that, in the new witness, all the objects are still labeled by the same extended quasiworld. Thus the new witness is equivalent to the original one. Applying this procedure to every $w \in W$ and every set of objects labeled with the same extended quasiworld by the original witness removes all the superfluous objects that were assigned the same extended quasistate; hence the bound holds for the new witness. Thus we can test all matching successor candidates for witnesses up to the size defined in Lemma 8—in 4-ExpTime we can check whether a pair (q, Q) is a matching successor and there are at most 3-Exp of such candidates to test.

What remains is to check whether a $(C_0, \mathcal{T})$-tree can be constructed using these matching successors: to this end we define a looping tree automaton $\mathcal{A}_{C_0,\mathcal{T}}$ that accepts exactly the $C_0, \mathcal{T}$-trees. To check satisfiability of C_0 w.r.t. $\mathcal{T}$, it then suffices to check whether this looping automaton accepts at least one tree. We show that this provides us with a decision procedure for satisfiability in $\mathsf{S5}_{\mathcal{ALCQI}}$ as the emptiness problem for looping tree automata is decidable in time linear in the size of the automaton [15].

Intuitively, we use the *matching successors* to define the transition relation of $\mathcal{A}_{C_0,\mathcal{T}}$. Since in our case the trees do not have a constant branching degree, we adopt a variant of a looping automaton on amorphous trees [12] (except in our case we know the branching degree is bounded and thus the transition relation can be represented finitely in a trivial way).

Definition 9 (Looping Tree Automaton). A *looping tree automaton* $\mathcal{A} = (Q, M, I, \delta)$ for an M-labeled tree is defined by a set Q of states, an alphabet M, a set $I \subseteq Q$ of initial states, and a transition relation $\delta \subseteq Q \times M \times 2^Q$.

A *run* of $\mathcal{A}$ on an M-labeled tree $\mathfrak{T} = (r_\mathfrak{T}, N_\mathfrak{T}, E_\mathfrak{T})$ with a root $r_\mathfrak{T}$ is a mapping $\tau : N_\mathfrak{T} \to Q$ such that $\tau(r_\mathfrak{T}) \in I$ and $(\tau(\alpha), \mathfrak{T}(\alpha), \{\tau(\beta) \mid (\alpha, \beta) \in E_\mathfrak{T}\}) \in \delta$ for all $\alpha \in N_\mathfrak{T}$. A looping automaton $\mathcal{A}$ *accepts* those M-labeled trees $\mathfrak{T}$ for which there exists a run of $\mathcal{A}$ on $\mathfrak{T}$.

We construct an automaton from C_0 and $\mathcal{T}$ as follows:

Definition 10. Let C_0 be a concept and $\mathcal{T}$ a $\mathsf{S5}_{\mathcal{ALCQI}}$ TBox. We denote with $\mathsf{nl}(C_0, \mathcal{T})$ the set of all extended quasistates for C_0 and $\mathcal{T}$. A looping automaton $\mathcal{A}_{C_0,\mathcal{T}} = (Q, M, I, \delta)$ is defined by setting $M = Q = \mathsf{nl}(C_0, \mathcal{T})$, $I := \{q \in Q \mid q \text{ realizes } C_0,\ q \text{ is root}\}$, and δ to the set of those tuples $(q, q, \overline{Q})$ such that $\overline{Q} \in 2^Q$ and $(q, \overline{Q})$ is a matching successor for C_0 and $\mathcal{T}$.

The following lemma states that the obtained looping automata behaves as expected.

Lemma 11. *$\mathfrak{T}$ is a $C_0, \mathcal{T}$-tree iff $\mathfrak{T}$ is accepted by $\mathcal{A}_{C_0,\mathcal{T}}$.*

The size of $\mathcal{A}_{C_0,\mathcal{T}}$ depends on the size of C_0 and $\mathcal{T}$ as there are only finitely many matching successors as shown in Lemma 8. To construct the transition function of the automaton, we also need to verify that the pair $(q, \overline{Q})$ is a matching successor. From Lemma 8, this can be done in 4-ExpTime. , we obtain that the proposed algorithm for satisfiability in $\mathsf{S5}_{\mathcal{ALCQI}}$ runs in 4-ExpTime. A lower bound can be established by reducing the word problem of exponentially space-bounded, alternating Turing machines [5].

Theorem 12. *Satisfiability in $\mathsf{S5}_{\mathcal{ALCQI}}$ is decidable and 2-ExpTime-hard.*

This result holds regardless of whether numbers inside number restrictions are coded in unary or in binary.

5 Conclusions

This work introduces the modal description logic $\mathsf{S5}_{\mathcal{ALCQI}}$ as a logic for representing and reasoning in temporal conceptual models with timestamping constraints only. $\mathsf{S5}_{\mathcal{ALCQI}}$ was shown to be decidable and 2-ExpTime-hard. This is the first decidability result for reasoning in temporal schemas with full timestamping—i.e., timestamping for entities, relationships, and attributes.

This paper leaves out few interesting open problems for further investigation. First, a tight complexity bound is not known. The gap between the 2-ExpTime-hardness and the 4-ExpTime algorithm shown in this paper needs to be closed. Second, we believe that the converse translation, from TER with full timestamping to $\mathsf{S5}_{\mathcal{ALCQI}}$, is also possible. Once formally proved, this result would allow us to characterize the complexity of reasoning over TER with timestamping. Finally, we are interested in checking the limits of expressive power of $\mathsf{S5}_{\mathcal{ALCQI}}$ w.r.t. various constraints have appeared in literature on temporal models other than timestamping.

References

[1] A. Artale. Reasoning on temporal conceptual schemas with dynamic constraints. In *11^{th} Int. Symposium on Temporal Representation and Reasoning (TIME04)*. IEEE Computer Society, 2004. Also in Proc. of DL'04.

[2] A. Artale and E. Franconi. Temporal ER modeling with description logics. In *Proc. of the Int. Conference on Conceptual Modeling (ER'99)*, volume 1728 of *Lecture Notes in Computer Science*. Springer-Verlag, 1999.

[3] A. Artale, E. Franconi, and F. Mandreoli. Description logics for modelling dynamic information. In Jan Chomicki, Ron van der Meyden, and Gunter Saake, editors, *Logics for Emerging Applications of Databases*. LNCS, Springer-Verlag, 2003.

[4] A. Artale, E. Franconi, F. Wolter, and M. Zakharyaschev. A temporal description logic for reasoning about conceptual schemas and queries. In *Proc. of the 8^{th} Joint European Conference on Logics in Artificial Intelligence (JELIA-02)*, volume 2424 of *LNAI*, pages 98–110. Springer, 2002.

[5] A. Artale, C. Lutz, and D. Toman. A description logic of change. Technical report, LTCS-Report 05-06, Technical University Dresden, 2002. see http://lat.inf.tu-dresden.de/research/reports.html.

[6] D. Calvanese, M. Lenzerini, and D. Nardi. Description logics for conceptual data modeling. In J. Chomicki and G. Saake, editors, *Logics for Databases and Information Systems*, pages 229–263. Kluwer, 1998.

[7] D. Calvanese, M. Lenzerini, and D. Nardi. Unifying class-based representation formalisms. *J. of Artificial Intelligence Research*, 11:199–240, 1999.

[8] J. Chomicki and D. Toman. Temporal Databases. In M. Fischer, D. Gabbay, and L. Villa, editors, *Handbook of Temporal Reasoning in Artificial Intelligence*, pages 429–467. Elsevier *Foundations of Artificial Intelligence*, 2005.

[9] D. Gabbay, A. Kurucz, F. Wolter, and M. Zakharyaschev. *Many-dimensional modal logics: theory and applications*. Studies in Logic. Elsevier, 2003.

[10] H. Gregersen and J.S. Jensen. Conceptual modeling of time-varying information. Technical Report TimeCenter TR-35, Aalborg University, Denmark, 1998.

[11] C. S. Jensen and R. T. Snodgrass. Temporal data management. *IEEE Transactions on Knowledge and Data Engineering*, 111(1):36–44, 1999.

[12] O. Kupferman and M. Y. Vardi. On bounded specifications. In *Proc. of the Int. Conference on Logic for Programming and Automated Reasoning (LPAR'01)*, LNAI, pages 24–38. Springer-Verlag, 2001.

[13] S. Spaccapietra, C. Parent, and E. Zimanyi. Modeling time from a conceptual perspective. In *Int. Conf. on Information and Knowledge Management (CIKM98)*, 1998.

[14] C. Theodoulidis, P. Loucopoulos, and B. Wangler. A conceptual modelling formalism for temporal database applications. *Information Systems*, 16(3):401–416, 1991.

[15] M. Y. Vardi and P. Wolper. Automata-theoretic techniques for modal logic of programs. *Journal of Computer and System Sciences*, 32:183–221, 1986.

[16] F. Wolter and M. Zakharyaschev. Modal descriptiopn logics: modalizing roles. *Fundamentae Informaticae*, 39:411–438, 1999.

Part II
Regular Papers

Reasoning for A Fuzzy Description Logic with Comparison Expressions *

Dazhou Kang[1], Baowen Xu[1,2], Jianjiang Lu[3] and Yanhui Li[1]

[1]School of Computer Science and Engineering, Southeast University, Nanjing 210096, China

[2]Jiangsu Institute of Software Quality, Nanjing, 210096, China

[3]Institute of Command Automation, PLA University of Science and Technology, Nanjing 210007, China

Abstract

The fuzzy extensions of description logics support representation and reasoning for fuzzy knowledge. But the current fuzzy description logics do not support the expression of comparisons between fuzzy membership degrees. The paper proposes $\mathcal{ALC}_{fc}$, a fuzzy extension of description logic $\mathcal{ALC}$ with comparison expressions. $\mathcal{ALC}_{fc}$ defines comparison cut concepts to express complex comparisons, and integrate them with fuzzy $\mathcal{ALC}$. The challenges of reasoning in $\mathcal{ALC}_{fc}$ is discussed, and a tableau algorithm is presented. It enables representation and reasoning for expressive fuzzy knowledge about comparisons.

1 Introduction

Description logics (DLs) [1] are widely used in the semantic web. Fuzzy extensions of description logics import the fuzzy theory to enable the capability of dealing with fuzzy knowledge [5]. In fuzzy DLs, an individual is an instance of a fuzzy concept only to a certain degree (called *fuzzy membership degree*); e.g. "Tom is tall to a degree greater than 0.8," where *Tall* is a fuzzy concept. It is also a familiar description that "Tom is taller than Mike," which can be seemed as a comparison between two degrees without any restrictions on their exact

*This work was supported by NSFC (60373066, 60425206 and 90412003), National Grand Fundamental Research 973 Program of China (2002CB312000), Innovation Plan for Jiangsu High School Graduate Student, Advanced Armament Research Project (51406020105JB8103), Advanced Research Fund of Southeast University (XJ0609233), and High Technology Research Project of Jiangsu Province (BG2005032). Email:swws@seu.edu.cn

values. Such comparison is important in many fuzzy information systems. For example, people are often interested in "the cheapest goods with the highest quality," disregarding the exact prices and qualities. There are even more complex comparison expressions, e.g. "no close friend of Tom is taller or stronger than him." However, the current fuzzy DLs do not support the expression of comparisons between fuzzy membership degrees. This paper defines comparison cut concepts to express complex comparisons, then presents $\mathcal{ALC}_{fc}$, and provides its reasoning algorithm. It enables representation and reasoning for expressive comparisons between fuzzy membership degrees.

2 Comparison expressions

In DLs, an assertion $a : C$ is either true or false (0 or 1). But in fuzzy DLs, any assertion is true only to a certain degree in $[0, 1]$. A *fuzzy assertion* is of the form $\langle a : C \bowtie n\rangle$, where $n \in [0, 1]$ is a constant and $\bowtie \in \{=, \neq, <, \leq, \geq, >\}$. In order to express a comparison that "Tom is taller than Mike," the most intuitionistic solution is $\langle \mathit{Tom} : \mathit{Tall} > \mathit{Mike} : \mathit{Tall}\rangle$. However, such fuzzy assertion is not allowed in current fuzzy DLs [3, 5].

There are more complex comparisons, like "Tom is either taller *or* stronger than Mike." We define *comparison cut concepts* (or *cuts* for short) to express such complex comparisons. The idea is as follows: individuals can be divided into different classes based on basic comparisons; the classes can be represented in basic cuts; and we can use constructors with such cuts to express more complex comparisons. Three kinds of basic cuts are defined:

numerical cuts $[C \bowtie n]$ represents a set of individuals s such that $\langle s : C \bowtie n\rangle$. For example, $[\mathit{Tall} < 0.9]$ refers to any person that is not very tall.

comparative cuts $[C \bowtie D]$ represents s such that $\langle s : C \bowtie s : D\rangle$. $[\mathit{Absolutist} < \mathit{Liberalist}]$ refers to people who prefer liberalism to absolutism.

relative cuts $[C \bowtie D^{\uparrow}]$ represents any s such that $\langle s : C \bowtie t : D\rangle$ with respect to an individual t. $[C \bowtie]$ is an abbreviation of $[C \bowtie C^{\uparrow}]$. They describe the comparisons between individuals.

The idea of cuts is not completely innovate. [3] and [6] defined concepts similar to the numerical cuts. But they do not consider other cuts.

Complex cuts are constructed from basic cuts: if P, Q are cuts, then $\neg P$, $P \sqcap Q$ and $P \sqcup Q$ are also cuts. More expressive fuzzy concepts can be defined by importing cuts: for any cuts P, Q and a fuzzy role R, $\exists R.P$ and $\forall R.P$ are fuzzy concepts. For example, $\langle \mathit{Tom} : \forall \mathit{friend}.[\mathit{Tall} \leq] \geq 0.8\rangle$ means Tom is tallest among his close friends. Such fuzzy concepts can be used in cuts again. We use $\langle a : P(b)\rangle$ to assert that a is in a cut P with respect to b. Then $\langle \mathit{Tom} : \mathit{Tall} \geq 0.8\rangle$ and $\langle \mathit{Tom} : \mathit{Tall} > \mathit{Mike} : \mathit{Tall}\rangle$ can be seemed as aliases of $\langle \mathit{Tom} : [\mathit{Tall} \geq 0.8]\rangle$ and $\langle \mathit{Tom} : [\mathit{Tall} >](\mathit{Mike})\rangle$.

3 The fuzzy description logic $\mathcal{ALC}_{fc}$

Definition 1. *Let N_I, N_C and N_R be three disjoint sets of names of individuals, fuzzy concepts and fuzzy roles respectively.* $\mathcal{ALC}_{fc}$-concepts *are defined as*

- $\top$, $\bot$ *and* A *are concepts, where* $A \in N_C$*;*
- *if* C, D *are concepts,* $R \in N_R$*, and* P *is a cut, then* $\neg C$*,* $C \sqcup D$*,* $C \sqcap D$*,* $\exists R.C$*,* $\forall R.C$*,* $\exists R.P$ *and* $\forall R.P$ *are concepts.*

A fuzzy interpretation $\mathcal{I} = \langle \Delta^{\mathcal{I}}, \cdot^{\mathcal{I}} \rangle$ *consists a nonempty set* $\Delta^{\mathcal{I}}$*, and a function* $\cdot^{\mathcal{I}}$ *maps every* $a \in N_I$ *to an element* $a^{\mathcal{I}} \in \Delta^{\mathcal{I}}$*, maps every* $A \in N_C$ *to a function* $A^{\mathcal{I}} : \Delta^{\mathcal{I}} \to [0,1]$*, and maps every* $R \in N_R$ *to a function* $R^{\mathcal{I}} : \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}} \to [0,1]$*. For* $\mathcal{ALC}_{fc}$*,* $\cdot^{\mathcal{I}}$ *also maps every concept* C *to a function* $C^{\mathcal{I}} : \Delta^{\mathcal{I}} \to [0,1]$ *such that for any* $s \in \Delta^{\mathcal{I}}$*,*

$$\top^{\mathcal{I}}(s) = 1; \bot^{\mathcal{I}}(s) = 0; \qquad (\exists R.C)^{\mathcal{I}}(s) = \sup_{t \in \Delta^{\mathcal{I}}} \{\min(R^{\mathcal{I}}(s,t), C^{\mathcal{I}}(t))\};$$
$$(\neg C)^{\mathcal{I}}(s) = 1 - C^{\mathcal{I}}(s); \qquad (\forall R.C)^{\mathcal{I}}(s) = \inf_{t \in \Delta^{\mathcal{I}}} \{\max(1 - R^{\mathcal{I}}(s,t), C^{\mathcal{I}}(t))\};$$
$$(C \sqcap D)^{\mathcal{I}}(s) = \max(C^{\mathcal{I}}(s), D^{\mathcal{I}}(s)); \qquad (\exists R.P)^{\mathcal{I}}(s) = \sup_{t \in P^{\mathcal{I}}(s)} R^{\mathcal{I}}(s,t);$$
$$(C \sqcup D)^{\mathcal{I}}(s) = \min(C^{\mathcal{I}}(s), D^{\mathcal{I}}(s)); \qquad (\forall R.P)^{\mathcal{I}}(s) = \inf_{t \in (\neg P)^{\mathcal{I}}(s)} (1 - R^{\mathcal{I}}(s,t)).$$

Definition 2. *The* comparison cut concepts *are defined as:*

- *if* C, D *are concepts,* $n \in [0,1]$ *and* $\bowtie \in \{=, \neq, >, \geq, <, \leq\}$*, then* $[C \bowtie n]$*,* $[C \bowtie D]$ *and* $[C \bowtie D^{\uparrow}]$ *are cuts;*
- *if* P, Q *are cuts, then* $\neg P$*,* $P \sqcap Q$ *and* $P \sqcup Q$ *are cuts.*

The interpretation function $\cdot^{\mathcal{I}}$ *of a fuzzy interpretation* $\mathcal{I} = \langle \Delta^{\mathcal{I}}, \cdot^{\mathcal{I}} \rangle$ *maps, additionally, every cut* P *into a function* $P^{\mathcal{I}} : \Delta^{\mathcal{I}} \to 2^{\Delta^{\mathcal{I}}}$*:*

$$[C \bowtie n]^{\mathcal{I}}(s) = \{t | C^{\mathcal{I}}(t) \bowtie n\}; \qquad (\neg P)^{\mathcal{I}}(s) = \Delta^{\mathcal{I}} \backslash P^{\mathcal{I}}(s);$$
$$[C \bowtie D]^{\mathcal{I}}(s) = \{t | C^{\mathcal{I}}(t) \bowtie D^{\mathcal{I}}(t)\}; \qquad (P \sqcap Q)^{\mathcal{I}}(s) = P^{\mathcal{I}}(s) \cap Q^{\mathcal{I}}(s);$$
$$[C \bowtie D^{\uparrow}]^{\mathcal{I}}_{\bowtie}(s) = \{t | C^{\mathcal{I}}(t) \bowtie D^{\mathcal{I}}(s)\}; \qquad (P \sqcup Q)^{\mathcal{I}}(s) = P^{\mathcal{I}}(s) \cup Q^{\mathcal{I}}(s).$$

For any cut P and $a \in N_I$, $P(a)$ is called an absolute cut, and $(P(a))^{\mathcal{I}} = P^{\mathcal{I}}(a^{\mathcal{I}})$. If a cut P contains no $\uparrow$, then P itself is an absolute cut, and for any a, we do not distinguish $P(a)$ and P, since they are equivalent in semantics.

Definition 3. *An* $\mathcal{ALC}_{fc}$ knowledge base *is composed of a TBox and an ABox:*

A TBox *is a finite set of axioms of the form* $C \sqsubseteq D$ *or* $C \sqsubset D$*. An interpretation* $\mathcal{I}$ *satisfies* $C \sqsubseteq D$ *or* $C \sqsubset D$ *iff for any* $s \in \Delta^{\mathcal{I}}$*,* $C^{\mathcal{I}}(s) \leq D^{\mathcal{I}}(s)$ *or* $C^{\mathcal{I}}(s) < D^{\mathcal{I}}(s)$*.* $\mathcal{I}$ *is a model of a TBox* $\mathcal{T}$ *iff* $\mathcal{I}$ *satisfies every axiom in* $\mathcal{T}$*.*

An ABox *is a finite set of assertions of the form* $\langle (a,b) : R \bowtie n \rangle$ *or* $\langle a : P(b) \rangle$*, where* $n \in [0,1]$*,* C *is a concept,* R *is a role,* $a, b \in N_I$*, and* P *is a cut. An interpretation* $\mathcal{I}$ *satisfies* $\langle (a,b) : R \bowtie n \rangle$ *iff* $R^{\mathcal{I}}(a^{\mathcal{I}}, b^{\mathcal{I}}) \bowtie n$*, satisfies* $\langle a : P(b) \rangle$ *iff* $a^{\mathcal{I}} \in P^{\mathcal{I}}(b^{\mathcal{I}})$*.* $\mathcal{I}$ *is a model of an ABox* $\mathcal{A}$ *iff* $\mathcal{I}$ *satisfies every assertion in* $\mathcal{A}$*.*

The basic inference problem of $\mathcal{ALC}_{fc}$ *is* consistency of ABoxes*: an ABox* $\mathcal{A}$ *is consistent w.r.t. a TBox* $\mathcal{T}$*, iff there exists a model* $\mathcal{I}$ *of both* $\mathcal{A}$ *and* $\mathcal{T}$*.*

4 Reasoning issues

4.1 Challenges

In the current reasoning algorithms for fuzzy DLs [4, 5], they always assume that for an element s in a model $\mathcal{I}$ with $(\exists R.C)^{\mathcal{I}}(s) \geq n$, there is t such that $\min(R^{\mathcal{I}}(s,t), C^{\mathcal{I}}(t)) \geq n$. In [2], the author pointed out that this is an unessential error. However, it yields a serious problem in $\mathcal{ALC}_{fc}$.

Example 4. *Let* $\mathcal{A} = \{\langle a : \exists R.C \geq 1\rangle, \langle a : \exists R.[c \geq 1] \leq 0\rangle\}$. *For any element* s *in a model* $\mathcal{I}$ *with* $(\exists R.[c \geq 1])^{\mathcal{I}}(s) \leq 0$, *there cannot be any* t *such that* $\min(R^{\mathcal{I}}(s,t), C^{\mathcal{I}}(t)) \geq 1$. *Nevertheless,* $\mathcal{A}$ *is consistent: there can be infinite elements* $t_1, t_2, \ldots$ *such that* $\sup_{i=1,2,\ldots}\{\min(R^{\mathcal{I}}(s,t_i), C^{\mathcal{I}}(t_i))\} = 1$, *and for any* $i = 1, 2, \ldots$, $\min(R^{\mathcal{I}}(s,t_i), C^{\mathcal{I}}(t_i)) < 1$.

Another challenge is that current algorithms only consider a finite set of constants in $[0, 1]$; but there can be infinite different degrees in $\mathcal{ALC}_{fc}$.

Example 5. *Assume* $\mathcal{T} = \{C \sqsubset \exists R.C\}$, $\mathcal{A} = \{s_0 : C > n\}$. *In a model* $\mathcal{I}$ *of* $\mathcal{T}$ *and* $\mathcal{A}$, *it holds* $n < C^{\mathcal{I}}(s_0) < (\exists R.C)^{\mathcal{I}}(s_0)$. *For any element* s_i, $C^{\mathcal{I}}(s_i) < (\exists R.C)^{\mathcal{I}}(s_i)$ *and there is* s_{i+1} *with* $C^{\mathcal{I}}(s_{i+1}) > C^{\mathcal{I}}(s_i)$. *It also follows* $C^{\mathcal{I}}(s_{i+1}) < (\exists R.C)^{\mathcal{I}}(s_{i+1})$. *So there must be an infinite path of elements* $s_0, s_1, s_2, \ldots$ *such that* $n < C^{\mathcal{I}}(s_0) < C^{\mathcal{I}}(s_1) < C^{\mathcal{I}}(s_2) < \cdots$.

Because of the above challenges, the current reasoning algorithms for fuzzy DLs are not capable for $\mathcal{ALC}_{fc}$. The following parts firstly define a novel fuzzy tableau for $\mathcal{ALC}_{fc}$. The essential difference from the tableau in [4] is **P4** and **P6**. Then presents a tableau algorithm to overcome the challenges.

Definition 6. *Let* $\mathcal{A}$ *be an ABox,* $\mathcal{T}$ *be a TBox, a* fuzzy tableau $\boldsymbol{T}$ *of* $\mathcal{A}$ *w.r.t.* $\mathcal{T}$ *is a quadruple* $\boldsymbol{T} = \langle \mathbf{S}, \mathcal{L}, \mathcal{E}, \mathcal{V}\rangle$ *such that*

- $\mathbf{S}$ *is a non-empty set of elements;*
- $\mathcal{L} : \mathbf{S} \to \mathbf{AC}$ *maps each element of* $\mathbf{S}$ *to a set of absolute cuts;*
- $\mathcal{E} : N_R \times [0, 1] \to 2^{\mathbf{S}\times\mathbf{S}}$ *maps each pair of a role name and a number in* $[0, 1]$ *to a set of pairs of elements;*
- $\mathcal{V} : I_{\mathcal{A}} \to \mathbf{S}$ *maps individual names occurring in* $\mathcal{A}$ *to elements in* $\mathbf{S}$.

P1 *If* $[\neg C = n] \in \mathcal{L}(s)$, *then* $[C = 1 - n] \in \mathcal{L}(s)$.

P2 *If* $[C \sqcap D = n] \in \mathcal{L}(s)$, *then* $\{[C = m_1], [D = m_2]\} \subseteq \mathcal{L}(s)$ *and* $\min(m_1, m_2) = n$.

P3 *If* $[C \sqcup D = n] \in \mathcal{L}(s)$, *then* $[\neg C \sqcap \neg D = 1 - n] \in \mathcal{L}(s)$

P4 *If* $[\exists R.C = n] \in \mathcal{L}(s)$, *then for any* t *with* $(s,t) \in \mathcal{E}(R, m_1)$ *and* $m_1 > n$, *it holds* $[C = m_2] \in \mathcal{L}(t)$ *and* $m_2 \leq n$, *and there are two possible cases:*

- *there is* $t \in \mathbf{S}$ *with* $(s,t) \in \mathcal{E}(R, m_1)$, $[C = m_2] \in \mathcal{L}(t)$, $\min(m_1, m_2) = n$;
- *there are* $t_1, t_2, \cdots \in \mathbf{S}$ *with for any* $i = 1, 2, \ldots$, $(s, t_i) \in \mathcal{E}(R, n_i)$, $[C = m_i] \in \mathcal{L}(t_i)$, $\min(n_i, m_i) < n$, *and* $\sup_{i=1,2,\ldots}\{\min(n_i, m_i)\} = n$.

P5 *If* $[\forall R.C = n] \in \mathcal{L}(s)$*, then* $[\exists R.\neg C = 1 - n] \in \mathcal{L}(s)$.
P6 *If* $[\exists R.P = n] \in \mathcal{L}(s)$*, then for any* t *with* $(s,t) \in \mathcal{E}(R,m)$ *that* $m > n$*, it holds* $\neg P(s) \in \mathcal{L}(t)$*, and there are two possible cases:*
- *there is* $t \in \mathbf{S}$ *with* $(s,t) \in \mathcal{E}(R,n)$, $P(s) \in \mathcal{L}(t)$;
- *there are* $t_1, t_2, \cdots \in \mathbf{S}$ *with for any* $i = 1, 2, \ldots$, $(s,t_i) \in \mathcal{E}(R,n_i)$, $P(s) \in \mathcal{L}(t_i)$, $n_i < n$, *and* $\sup_{i=1,2,\ldots}\{n_i\} = n$.

P7 *If* $[\forall R.P = n] \in \mathcal{L}(s)$*, then* $[\exists R.\neg P = 1 - n] \in \mathcal{L}(s)$.
P8 *If* $[C \bowtie n] \in \mathcal{L}(s)$*, then* $[C = m] \in \mathcal{L}(s)$ *and* $m \bowtie n$.
P9 *If* $[C \bowtie D] \in \mathcal{L}(s)$*, then* $\{[C = m], [D = n]\} \subseteq \mathcal{L}(s)$ *and* $m \bowtie n$.
P10 *If* $[C \bowtie D^{\uparrow}](t) \in \mathcal{L}(s)$*, then* $[C = m] \in \mathcal{L}(s)$, $[D = n] \in \mathcal{L}(t)$ *and* $m \bowtie n$.
P11 *If* $(P \sqcap Q)(t) \in \mathcal{L}(s)$*, then* $\{P(t), Q(t)\} \subseteq \mathcal{L}(s)$.
P12 *If* $(P \sqcup Q)(t) \in \mathcal{L}(s)$*, then* $P(t) \in \mathcal{L}(s)$ *or* $Q(t) \in \mathcal{L}(s)$.
P13 *If* $\langle (a,b) : R \bowtie n \rangle \in \mathcal{A}$*, then* $(\mathcal{V}(a), \mathcal{V}(b)) \in \mathcal{E}(R,m)$ *and* $m \bowtie n$.
P14 *If* $\langle a : P(b) \rangle \in \mathcal{A}$*, then* $P(\mathcal{V}(b)) \in \mathcal{L}(\mathcal{V}(a))$.
P15 *If* $C \sqsubseteq D \in \mathcal{T}$*, then for any* $s \in \mathbf{S}$, $\{[C = m], [D = n]\} \subseteq \mathcal{L}(s)$ *and* $m \leq n$.
P16 *If* $C \sqsubset D \in \mathcal{T}$*, then for any* $s \in \mathbf{S}$, $\{[C = m], [D = n]\} \subseteq \mathcal{L}(s)$ *and* $m < n$.
P17 *If* $[c = n] \in \mathcal{L}(s)$*, then there is no* $[c = m] \in \mathcal{L}(s)$ *such that* $n \neq m$.
P18 *If* $(s,t) \in \mathcal{E}(R,n)$*, then there is no* $(s,t) \in \mathcal{E}(R,m)$ *such that* $n \neq m$.

All concepts and cuts are transformed into *negation normal form*, where $\neg$ can only occur in front of a concept name, by pushing negations inwards: $\neg[C \bowtie W] \rightarrow [C \bowtie^{\ominus} W]$; $\neg(X \sqcup Y) \rightarrow \neg X \sqcap \neg Y$; $\neg(X \sqcap Y) \rightarrow \neg X \sqcup \neg Y$; $\neg(\exists R.X) \rightarrow \forall R.\neg X$; $\neg(\forall R.X) \rightarrow \exists R.\neg X$; where $W \in \{n, D, D^{\uparrow}\}$, X, Y are concepts or cuts, and $>^{\ominus} = \leq$, $<^{\ominus} = \geq$, $\geq^{\ominus} = <$, $\leq^{\ominus} = >$. All elements satisfy $[\top = 1]$ and $[\bot = 0]$. So $\top, \bot, \neg P$ are not considered here. It is easy to prove that the existence of a fuzzy tableau is equivalent with the existence of a model: $\Delta^{\mathcal{I}} = \mathcal{S}$, $A^{\mathcal{I}}(s) = n$ iff $[A = n] \in \mathcal{L}(s)$, and $R^{\mathcal{I}}(s,t) = n$ iff $(s,t) \in \mathcal{E}(R,n)$.

Theorem 7. *An* $\mathcal{ALC}_{fc}$ *ABox* $\mathcal{A}$ *is consistent w.r.t. a TBox* $\mathcal{T}$*, iff there is a fuzzy tableau* $\boldsymbol{T}$ *of* $\mathcal{A}$ *w.r.t.* $\mathcal{T}$.

4.2 A tableau algorithm

Here presents an algorithm to decide the existence of a fuzzy tableau by constructing a completion graph. There are three key points in the algorithm:

1. All degrees such as $C(x)$ and $R(x,y)$ are not constants but variables. The relations between degrees are recorded in a set δ, which is a partially ordered set $(V, \leq, \neq)$. The set of degrees V can be easily mapping to $[0,1]$.
2. In **P4** and **P6**, there can be infinite elements. We just generate one node to represent them with a new relation $\lessdot$ added into δ by **R5** and **R8**.
3. It is clear that the construction of a completion graph may not terminate if $\mathcal{T} \neq \emptyset$. A cycle detection technique called *blocking* is employed to unravel a finite completion graph into an infinite fuzzy tableau.

Definition 8. *A* completion graph *is* $T = \langle S, E, L, \delta \rangle$*, where*

- S *is a set of nodes in the graph.*
- E *is a set of edges (pairs of nodes) in the graph.*
- L *is a function: for every node* $x \in S$*,* $L(x)$ *is a set of concepts or absolute cuts; for every edge* $(x, y) \in E$*,* $L(x, y)$ *is a set of roles.*
- δ *is a set of formulas of the form* $X \leq Y$*,* $X \neq Y$ *or* $X \lessdot Y$*, where* $X, Y ::= n|C(x)|R(x, y)|1 - X$ *such that* $n \in [0, 1]$*,* C *is a concept,* R *is a role,* $x, y \in S$*, and for any* X*,* $1 - (1 - X) = X$*.*

Several abbreviations are defined below:

$$X \leq_\delta Y \ =_{def}\ X \leq Y \in \delta, \text{ or } X \leq_\delta Y, Y \leq_\delta Z, \text{ or } 1 - Y \leq_\delta 1 - X$$
$$\min(X, Y) =_\delta Z \ =_{def}\ Z \leq_\delta X, Z \leq_\delta Y, W \leq_\delta Z \text{ for some } W \in \{X, Y\};$$
$$X \neq_\delta Y \ =_{def}\ X \neq Y \in \delta; \qquad X \lessdot_\delta Y \ =_{def}\ X \lessdot Y \in \delta;$$
$$X \geq_\delta Y \ =_{def}\ Y \leq_\delta X; \qquad X =_\delta Y \ =_{def}\ X \leq_\delta Y, Y \leq_\delta X;$$
$$X <_\delta Y \ =_{def}\ X \leq_\delta Y, X \neq_\delta Y; \quad X_\delta > Y \ =_{def}\ Y \leq_\delta X, X \neq_\delta Y.$$

The completion graph T of an ABox $\mathcal{A}$ w.r.t. a TBox $\mathcal{T}$ initializes with: $S = \{a \in N_I | a \text{ occurs in } \mathcal{A}\}$; for any $\langle a : P(b) \rangle \in \mathcal{A}$, $P(b) \in L(a)$; for any $\langle (a, b) : R \bowtie n \rangle \in \mathcal{A}$, $R \in L(a, b)$ and $R \bowtie_\delta n$. Let $V_0 = \{v_1, v_2, \ldots, v_k\} = \{0, 1, 0.5\} \cup \{n \in [0, 1] | n \text{ or } 1 - n \text{ occurs in } \mathcal{A} \text{ or } \mathcal{T}\}$, where $0 = v_1 < v_2 < \cdots < v_k = 1$. For any $v_i < v_j$, let $v_i <_\delta v_j$.

Then the graph grows up by applying the *expansion rules* showed in Fig. 1. If a rule applied to x creates a new node y, then y is a *successor* of x. Let *descendant* be transitive closure of successor.

The $\lessdot_\delta$ relation is used to simulate the infinite supreme. For any $a \in N_I$, $\text{lev}(a) = 1$. If $\text{lev}(x) = i$, y is a successor of x by updating $\lessdot_\delta$, then $\text{lev}(y) = i+1$. For any X of the form $C(x)$, $1 - C(x)$, $R(y, x)$, or $1 - R(y, x)$, if $\text{lev}(x) = i$, then $X \in V_i$. If $X \lessdot_\delta Y$ and $Y \in V_i$, then for any $Z \in V_j$ such that $j \leq i$, $Z <_\delta Y \rightarrow Z <_\delta X$ and $Z >_\delta X \rightarrow Z \geq_\delta Y$. So $X \lessdot_\delta Y$ means X is greater than any $Z < Y$ such that $Z \in V_0 \cup V_1 \cup \cdots \cup V_i$ and $Y \in V_i$. It ensures that for any given constant ε, we can assign values to the variables in V such that $X - Y < \varepsilon$ without inducing any conflict.

Since there are variables, the blocking condition in $\mathcal{ALC}_{fc}$ is different from classical DLs. It has to consider the comparisons between degrees.

Definition 9. *For any* x*, let* $\delta(x) = \{X \bowtie Y | X \bowtie_\delta Y$*,* X, Y *are of the form* $C(x)$*,* $1 - C(x)$*, or* v_i*. A node* x *is* blocked *by* y*, iff* x *is an descendant of* y*, and* $\delta(x) = [x/y]\delta(y)$*, where* $[x/y]\delta(y)$ *means to replace any* y *in* $\delta(y)$ *by* x*. Then we call* y blocks x*. When* x *is blocked, all descendants of* x *is also blocked.*

No rules in Fig. 1 can be applied to blocked nodes. T is said to contain a *clash* if $\{X \neq_\delta Y, X =_\delta Y\} \subseteq \delta$, or $X >_\delta 1$, or $X <_\delta 0$. T is said to be *clash-free*

R1	if	$\neg C \in L(x)$, and not $C(x) =_\delta 1 - (\neg C)(x)$
	then	$L(x) \rightarrow L(x) \cup \{C\}$, and $C(x) =_\delta 1 - (\neg C)(x)$
R2	if	$C \sqcap D \in L(x)$, and not $\min(C(x), D(x)) =_\delta (C \sqcap D)(x)$
	then	$L(x) \rightarrow L(x) \cup \{C, D\}$, and $\min(C(x), D(x)) =_\delta (C \sqcap D)(x)$
R3	if	$C \sqcup D \in L(x)$, and not $(C \sqcup D)(x) =_\delta 1 - (\neg C \sqcap \neg D)(x)$
	then	$L(x) \rightarrow L(x) \cup \{C \sqcup D\}$, and $(C \sqcup D)(x) =_\delta 1 - (\neg C \sqcap \neg D)(x)$
R4	if	$\exists R.C \in L(x)$, and there is y with $R \in L(x, y)$ but not $X \leq_\delta (\exists R.C)(x)$ for some $X \in \{R(x, y), C(x)\}$
	then	$L(y) \rightarrow L(y) \cup \{C\}$, and $X \leq_\delta (\exists R.C)(x)$
R5	if	$\exists R.C \in L(x)$, and there is no y with $X =_\delta (\exists R.C)(x)$ or $X \lessdot_\delta (\exists R.C)(x)$, for some $X \in \{R(x, y), C(x)\}$
	then	add a new node y with $L(x, y) = \{R\}, L(y) = \{C\}$, and $X =_\delta (\exists R.C)(x)$ or $X \lessdot_\delta (\exists R.C)(x)$
R6	if	$\forall R.C \in L(x)$, and not $(\forall R.C)(x)_\delta = 1 - (\exists R.\neg C)(x)$
	then	$L(x) \rightarrow L(x) \cup \{\exists R.\neg C\}$, and $(\forall R.C)(x) =_\delta 1 - (\exists R.\neg C)(x)$
R7	if	$\exists R.P \in L(x)$, and there is y with $R \in L(x, y)$ but not $R(x, y)_\delta \leq (\exists R.C)(x)$ nor $\neg P(x)_\delta \in L(y)$
	then	$R(x, y) \leq_\delta (\exists R.C)(x)$, or $L(y) \rightarrow L(y) \cup \{\neg P(x)\}$
R8	if	$\exists R.P \in L(x)$, and there is no y with $P(x) \in L(y)$, $R(x, y) =_\delta (\exists R.C)(x)$ or $R(x, y) \lessdot_\delta (\exists R.C)(x)$
	then	add a new node y with $L(x, y) = \{R\}, L(y) = \{P(x)\}$, and $R(x, y) =_\delta (\exists R.C)(x)$ or $R(x, y) \lessdot_\delta (\exists R.C)(x)$
R9	if	$\forall R.P \in L(x)$, and not $(\forall R.P)(x) =_\delta 1 - (\exists R.\neg P)(x)$
	then	$L(x) \rightarrow L(x) \cup \{\exists R.\neg P\}$, and $(\forall R.P)(x) =_\delta 1 - (\exists R.\neg P)(x)$
R10	if	$[C \bowtie n] \in L(x)$, and not $C(x) \bowtie_\delta n$
	then	$L(x) \rightarrow L(x) \cup \{C\}$, and $C(x) \bowtie_\delta n$
R11	if	$[C \bowtie D] \in L(x)$, and not $C(x) \bowtie_\delta D(x)$
	then	$L(x) \rightarrow L(x) \cup \{C, D\}$, and $C(x) \bowtie_\delta D(x)$
R12	if	$[C \bowtie D^{\uparrow}](y) \in L(x)$, and not $C(x) \bowtie_\delta D(y)$
	then	$L(x) \rightarrow L(x) \cup \{C\}$, $L(y) \rightarrow L(y) \cup \{D\}$, and $C(x) \bowtie_\delta D(y)$
R13	if	$(P \sqcap Q)(y) \in L(x)$, and not $\{P(y), Q(y)\} \subseteq L(x)$
	then	$L(x) \rightarrow L(x) \cup \{P(y), Q(y)\}$
R14	if	$(P \sqcup Q)(y) \in L(x)$, and $\{P(y), Q(y)\} \cap L(x) = \emptyset$
	then	$L(x) \rightarrow L(x) \cup \{X\}$ for some $X \in \{P(y), Q(y)\}$
R15	if	$C \sqsubseteq D \in \mathcal{T}$, and there is x with no $C(x) \leq_\delta D(x)$
	then	$L(x) \rightarrow L(x) \cup \{C, D\}$, and $C(x) \leq_\delta D(x)$
R16	if	$C \sqsubset D \in \mathcal{T}$, and there is x with no $C(x) <_\delta D(x)$
	then	$L(x) \rightarrow L(x) \cup \{C, D\}$, and $C(x) <_\delta D(x)$
R17	if	$C \in L(x)$ or $R \in L(x, y)$, and let $X = C(x)$ or $R(x, y)$ there is no i such that $v_i <_\delta X <_\delta v_{i+1}$, or $X =_\delta v_i$
	then	$v_i <_\delta X <_\delta v_{i+1}$ for some v_i, v_{i+1}, or $X =_\delta v_i$ for some v_i

Figure 1: Expansion rules for $\mathcal{ALC}_{fc}$

if it contains no clash. If none of the expansion rules can be applied to T, then T is said to be *complete*.

From the blocking condition and the number of concepts in any $L(x)$ is finite, the algorithm terminates. There is a fuzzy tableau $\boldsymbol{T}$ of $\mathcal{A}$ w.r.t. $\mathcal{T}$, iff a complete and clash-free completion graph can be constructed from $\mathcal{A}$ w.r.t. $\mathcal{T}$. So the above algorithm is a decision procedure for consistency of $\mathcal{ALC}_{fc}$ ABoxes w.r.t. empty TBoxes.

5 Conclusion

This paper presents $\mathcal{ALC}_{fc}$, a fuzzy extension of description logic $\mathcal{ALC}$ with comparison expressions. New challenges of reasoning within $\mathcal{ALC}_{fc}$ are discussed and a reasoning algorithm for $\mathcal{ALC}_{fc}$ is proposed to overcome the challenges. It enables representation and reasoning for expressive fuzzy knowledge about comparisons. The future work is to extend comparison expressions in more expressive fuzzy description logics and design their reasoning algorithms.

References

[1] Franz Baader, Diego Calvanese, Deborah McGuinness, Daniele Nardi, and Peter F. Patel-Schneider, editors. *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press, 2003.

[2] Petr Hajek. Making fuzzy description logic more general. 2005.

[3] Yanhui Li, Baowen Xu, Jianjiang Lu, Dazhou Kang, and Peng Wang. A family of extended fuzzy description logics. In *Proceedings of the IEEE 29th Annual International Computer Software and Applications Conference*, pages 221–226, Edinburgh, Scotland, 2005.

[4] Giorgos Stoilos, Giorgos Stamou, Vassilis Tzouvaras, Jeff Z. Pan, and Ian Horrocks. The fuzzy description logic f-shin. In *Proceedings of the International Workshop on Uncertainty Reasoning for the Semantic Web*, Galway, Ireland, 2005.

[5] Umberto Straccia. Reasoning within fuzzy description logics. *Journal of Artificial Intelligence Research*, 14:137–166, 2001.

[6] Umberto Straccia. Transforming fuzzy description logics into classical description logics. In *Proceedings of the 9th European Conference on Logics in Artificial Intelligence*, number 3229, pages 385–399, Lisbon, Portugal, 2004.

Handling Imprecise Knowledge with Fuzzy Description Logic

Giorgos Stoilos and Giorgos Stamou
National Technical University of Athens, Greece
gstoil@image.ntua.gr
gstam@softlab.ntua.gr
Jeff Z. Pan
University of Aberdeen, UK
jpan@csd.abdn.ac.uk

Abstract

Fuzzy Description Logics have been proposed in the literature as a way to represent and reason with vague and imprecise knowledge. Their decidability, the empirically tractable and efficient reasoning algorithms, that carry over to fuzzy Description Logics, have attracted the attention of many research communities and domains that deal with a wealth of imprecise knowledge and information. In the current paper we present the syntax and semantics of fuzzy $\mathcal{SHOIQ}$, investigating several properties of the semantics of transitivity, qualified cardinality restrictions and reasoning capabilities.

1 Introduction

Although Description Logics (DLs) provide considerable expressive power, they feature expressive limitations regarding their ability to represent vague and imprecise knowledge. Consider for example an image processing application. Such applications can be assisted by the aid of a knowledge base that contains definitions of the objects that can be found within an image. For example there could be a definition of the form,

$$\mathsf{Body} \sqcap \exists \mathsf{hasPart}.\mathsf{Tail} \sqsubseteq \mathsf{Animal}.$$

saying that if an object has a body and a part that is a tail then this object is an animal. Now suppose that we run an image analysis algorithm. Such algorithms usually segment and label objects that they identify in images. Since

the algorithms cannot be certain about the membership or non-membership of an object to a concept it usually assigns degrees of truth to these labellings. For example, we could have that the object o_1 hasPart o_2 to a degree of 0.8, that o_1 is a Body to a degree of 0.6 and that o_2 is a Tail to a degree of 0.7. From this knowledge we can deduce that o_1 is an Animal to a degree, at-least equal to 0.6. For that purpose *Fuzzy Description Logics* (f-DLs), have been proposed in the literature as a way to represent and reason with vague and imprecise knowledge. In the current paper we present the f-DL, f-$\mathcal{SHOIQ}$. f-$\mathcal{SHOIQ}$ extends f-$\mathcal{SHOIN}$ [10] with qualified cardinality restrictions (QCRs). Furthermore, we investigate the semantics of f-$\mathcal{SHOIQ}$, showing that it is a sound extension of $\mathcal{SHOIQ}$, we investigate properties of fuzzy QCRs and transitivity, we provide the inference problems and investigate reasoning capabilities in f-$\mathcal{SHOIQ}$.

2 Fuzzy Set Theory

Fuzzy set theory and fuzzy logic are widely used for capturing imprecise knowledge [3]. While in classical set theory an element either belongs to a set or not, in fuzzy set theory elements belong only to a certain degree. More formally, let X be a set of elements. A fuzzy subset A of X, is defined by a *membership function* $\mu_A(x)$, or simply $A(x)$ [3]. This function assigns any $x \in X$ to a value between 0 and 1 that represents the degree in which this element belongs to X. In this new framework the classical set theoretic and logical operations are performed by special mathematical functions. More precisely *fuzzy complement* is a unary operation of the form $c : [0,1] \to [0,1]$, *fuzzy intersection* and *union* are performed by two binary functions of the form $t : [0,1] \times [0,1] \to [0,1]$ and $u : [0,1] \times [0,1] \to [0,1]$, called *t-norm* and *t-conorm* operations [3], respectively, and *fuzzy implication* also by a binary function, $\mathcal{J} : [0,1] \times [0,1] \to [0,1]$. In order to produce meaningfull fuzzy complements, conjunctions, disjunctions and implications, these functions must satisfy certain mathematical properties. For example the operators must satisfy the following boundary properties, $c(0) = 1$, $c(1) = 0$, $t(1,a) = a$ and $u(0,a) = a$. Due to space limitations we cannot present all the properties that these functions should satisfy. The reader is referred to [3] for a comprehensive introduction. Nevertheless, it worths noting here that there exist two distinct classes of fuzzy implications, those of S-implications, given by the equation $\mathcal{J}(a,b) = u(c(a),b)$, and those of R-implications, given by $\mathcal{J}(a,b) = \sup\{x \in [0,1] \mid t(a,x) \leq b\}$. Examples of fuzzy operators are the Lukasiewicz negation, $c_L(a) = 1 - a$, t-norm, $t_L(a,b) = \max(0, a + b - 1)$, t-conorm $u_L(a,b) = \min(1, a + b)$, and implication, $\mathcal{J}_L(a,b) = \min(1, 1 - a + b)$, the Gödel norms $t_G(a,b) = \min(a,b)$, $u_G(a,b) = \max(a,b)$, and implication $\mathcal{J}_G(a,b) = b$ if $a > b$, 1 otherwise, and the Kleene-Dienes implication (KD-implication), $\mathcal{J}_{KD}(a,b) = \max(1 - a, b)$.

3 Fuzzy Description Logics

3.1 Syntax and Semantics

In this section we introduce the DL f-$\mathcal{SHOIQ}$. As usual we have an alphabet of distinct concept names ($\mathbf{C}$), role names ($\mathbf{R}$) and individual names ($\mathbf{I}$). f-$\mathcal{SHOIQ}$-roles and f-$\mathcal{SHOIQ}$-concepts are defined as follows:

Definition 3.1 *Let $RN \in \mathbf{R}$ be a role name and R an f-$\mathcal{SHOIQ}$-role. f-$\mathcal{SHOIQ}$-roles are defined by the abstract syntax: $R ::= RN \mid R^-$. The inverse relation of roles is symmetric, and to avoid considering roles such as R^{--}, we define a function* Inv*, which returns the inverse of a role, more precisely $\mathsf{Inv}(R) := R^-$ and $\mathsf{Inv}(R^-) := R$. Let $A \in \mathbf{C}$ be a concept name, C and D f-$\mathcal{SHOIQ}$-concepts, $p \in \mathbb{N}$, S a* simple[1] *f-$\mathcal{SHOIQ}$-role, $o \in \mathbf{I}$ and R an f-$\mathcal{SHOIQ}$-role. f-$\mathcal{SHOIQ}$ concepts are defined by the following abstract syntax:*

$$C, D \longrightarrow \bot \mid \top \mid A \mid \neg C \mid C \sqcup D \mid C \sqcap D \mid \exists R.C \mid \forall R.C \mid \leq pS.C \mid \geq pS.C \mid \{o\}$$

The semantics of f-DLs are provided by a *fuzzy interpretation* [9]. A fuzzy interpretation is a pair $\mathcal{I} = \langle \Delta^{\mathcal{I}}, \cdot^{\mathcal{I}} \rangle$ where the domain $\Delta^{\mathcal{I}}$ is a non-empty set of objects, called the *domain of interpretation*, and $\cdot^{\mathcal{I}}$ is a *fuzzy interpretation function* which maps an individual $\mathsf{a} \in \mathbf{I}$ to an element $\mathsf{a}^{\mathcal{I}} \in \Delta^{\mathcal{I}}$, a concept name $\mathsf{A} \in \mathbf{C}$ to a membership function $\mathsf{A}^{\mathcal{I}} : \Delta^{\mathcal{I}} \to [0, 1]$ and a role name $R \in \mathbf{R}$ to a membership function $R^{\mathcal{I}} : \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}} \to [0, 1]$. The semantics of f-$\mathcal{SHOIQ}$-concepts and roles are depicted in Table 1. Most of these semantics have been presented elsewhere [9, 10, 1, 4, 7]. Nevertheless, the semantics of fuzzy QCRs presented here are a revision of the definition provided in [10]. First, we extend the semantics of number restriction to qualified number restrictions. Second, we use arbitrary fuzzy implications to give semantics to at-most QCRs, while in [10] only S-implications were considered. Extending the definition has the effect that if $\forall R.C$ is interpreted by means of an R-implication and the fuzzy complement is the *involutive* ($c(c(a)) = a$) *precomplement* of the R-implication, then the equivalence $\forall R.C \equiv \leq 0R.\neg C$, holds. Since we are using arbitrary fuzzy implications we have to extend the definition in [10] to consider the equalities ($=$) and inequalities ($\neq$) of objects. Please note that this equality and inequality is usually considered crisp, i.e. either 0 or 1, in the f-DL literature [5]. Finally, as it is argued in [7] we choose not to fuzzify nominal concepts. The reason for this choice is that a concept of the form {o} intends to refer to a specific object of $\Delta^{\mathcal{I}}$, i.e. $o^{\mathcal{I}}$ and not some real life concept with an arbitrary number of members.

An f-$\mathcal{SHOIQ}$ knowledge base Σ consists of a TBox, an RBox and an ABox. Let C and D be f-$\mathcal{SHOIQ}$ concepts. As with the classical case, an f-$\mathcal{SHOIQ}$

[1] A role is called *simple* if it is neither transitive nor has any transitive sub-roles.

Table 1: Semantics of f-$\mathcal{SHOIQ}$-concepts and roles

Constructor	Syntax	Semantics
top	$\top$	$\top^{\mathcal{I}}(a) = 1$
bottom	$\bot$	$\bot^{\mathcal{I}}(a) = 0$
general negation	$\neg C$	$(\neg C)^{\mathcal{I}}(a) = c(C^{\mathcal{I}}(a))$
conjunction	$C \sqcap D$	$(C \sqcap D)^{\mathcal{I}}(a) = t(C^{\mathcal{I}}(a), D^{\mathcal{I}}(a))$
disjunction	$C \sqcup D$	$(C \sqcup D)^{\mathcal{I}}(a) = u(C^{\mathcal{I}}(a), D^{\mathcal{I}}(a))$
exists restriction	$\exists R.C$	$(\exists R.C)^{\mathcal{I}}(a) = \sup_{b \in \Delta^{\mathcal{I}}}\{t(R^{\mathcal{I}}(a,b), C^{\mathcal{I}}(b))\}$
value restriction	$\forall R.C$	$(\forall R.C)^{\mathcal{I}}(a) = \inf_{b \in \Delta^{\mathcal{I}}}\{\mathcal{J}(R^{\mathcal{I}}(a,b), C^{\mathcal{I}}(b))\}$
nominal	$\{\mathtt{o}\}$	$\{\mathtt{o}\}^{\mathcal{I}}(a) = 1$ if $a \in \{\mathtt{o}^{\mathcal{I}}\}$, otherwise $\{\mathtt{o}\}^{\mathcal{I}}(a) = 0$
at-most QCR	$\leq pR.C$	$\inf_{b_1,\ldots,b_{p+1} \in \Delta^{\mathcal{I}}} \mathcal{J}(t_{i=1}^{p+1}\{t(R^{\mathcal{I}}(a,b_i), C^{\mathcal{I}}(b_i))\}, u_{i<j}\{b_i = b_j\})$
at-least QCR	$\geq pR.C$	$\sup_{b_1,\ldots,b_p \in \Delta^{\mathcal{I}}} t(t_{i=1}^{p}\{t(R^{\mathcal{I}}(a,b_i), C^{\mathcal{I}}(b_i))\}, t_{i<j}\{b_i \neq b_j\})$
inverse roles	R^-	$(R^-)^{\mathcal{I}}(b,a) = R^{\mathcal{I}}(a,b)$

TBox is a finite set of axioms of the form $C \sqsubseteq D$, which are called, *fuzzy inclusion axioms*. A fuzzy interpretation $\mathcal{I}$ satisfies a fuzzy TBox $\mathcal{T}$ if $\forall o \in \Delta^{\mathcal{I}}, C^{\mathcal{I}}(o) \leq D^{\mathcal{I}}(o)$, for each $C \sqsubseteq D \in \mathcal{T}$; in this case, we say that $\mathcal{I}$ is a *model* of $\mathcal{T}$. Similarly, an f-$\mathcal{SHOIQ}$ RBox is a finite set of *fuzzy transitive role* axioms, $\mathsf{Trans}(R)$, and *fuzzy role inclusion* axioms, $R \sqsubseteq S$. $\mathcal{I}$ satisfies a fuzzy RBox $\mathcal{R}$ if $\forall a, c \in \Delta^{\mathcal{I}}$, $R^{\mathcal{I}}(a,c) \geq \sup_{b \in \Delta^{\mathcal{I}}}\{t(R^{\mathcal{I}}(a,b), R^{\mathcal{I}}(b,c))\}$ for each $\mathsf{Trans}(R) \in \mathcal{R}$ and $\forall a, b \in \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}, R^{\mathcal{I}}(a,b) \leq S^{\mathcal{I}}(a,b)$, for each $R \sqsubseteq S \in \mathcal{T}$; in this case, we say that $\mathcal{I}$ is a model of $\mathcal{R}$. An f-$\mathcal{SHOIQ}$ ABox is a finite set of fuzzy assertions [9] of the form $(\mathtt{a} : C) \bowtie n$ or $(\langle \mathtt{a}, \mathtt{b} \rangle : R) \bowtie n$, where $\bowtie$ stands for $\geq, >, \leq$ and $<$. Formally, $\mathcal{I}$ satisfies a fuzzy ABox $\mathcal{A}$, if $C^{\mathcal{I}}(\mathtt{a}^{\mathcal{I}}) \geq n$ ($R^{\mathcal{I}}(\mathtt{a}^{\mathcal{I}}, \mathtt{b}^{\mathcal{I}}) \geq n$) for each $(\mathtt{a} : C) \geq n$ ($(\langle \mathtt{a}, \mathtt{b} \rangle : R) \geq n$) in $\mathcal{A}$; in this case, we say that $\mathcal{I}$ is a model of $\mathcal{A}$. The satisfiability of fuzzy assertions with $\leq, >, <$ is defined analogously. Observe that we can also simulate assertions of the form $(\mathtt{a} : C) = n$ by considering the assertions $(\mathtt{a} : C) \geq n$ and $(\mathtt{a} : C) \leq n$.

As it has been argued in the literature, fuzzy set theory is an extension of classical set theory. Hence, the following lemma,

Lemma 3.2 *Fuzzy interpretations coincide with crisp interpretations if we restrict to the membership degrees of 0 and 1.*

In other words in the extreme limits of 0 and 1 the fuzzy operations provide the results of Boolean algebra. We call such an extention a *sound* extension.

Since we have defined fuzzy QCRs it is possible that $(a : (\geq p_1 R.C)) \geq n_1$ and $(a : (\leq p_2 R.C)) \geq n_2$, with $p_1 > p_2$ simultaneously hold, without forming a contradiction. More precisely if t is the Gödel t-norm and $\mathcal{J}$ the KD-implication we have,

Lemma 3.3 *Let $\mathcal{A} = \{(a : (\geq p_1 R.C)) \geq n_1, (a : (\leq p_2 R.C)) \geq n_2\}$ be a fuzzy ABox, with $n_1, n_2 \in [0,1]$, $p_1, p_2 \in \mathbb{N}$, and $p_2 < p_1$. Then $\mathcal{A}$ is satisfiable iff $n_1 + n_2 \leq 1$.*

In classical DLs, since $n_1, n_2 \in \{0, 1\}$, the inequality $n_1 + n_2 \leq 1$ is satisfied if and only if either $n_1 = 0$ or $n_2 = 0$. Indeed in crisp DLs an individual cannot simultaneously belong to both such concepts. Please note that investigating this property when other norm operations are used is an open research issue.

3.2 Logical Properties of Fuzzy DLs

As it is obvious different fuzzy operators specify different f-DLs. For example the f$_{KD}$-$\mathcal{SHOIQ}$ is obtained from f-$\mathcal{SHOIQ}$ when the Lukasiewicz negation, the Gödel t-norm and t-conorm and the Kleene-Dienes fuzzy implication are used, while f$_L$-$\mathcal{SHOIQ}$ is obtained if we use the Lukasiewicz negation, t-norm, t-conorm and fuzzy implication. The choice of the operations has an immediate impact on the logical properties of the resulting f-DL.

For any triple $\langle c, t, u\rangle$, due to the standard properties of the fuzzy operators [3], the following concept equivalences hold: $\neg\top \equiv \bot$, $\neg\bot \equiv \top$, $C \sqcap \top \equiv C$, $C \sqcup \bot \equiv C$, $C \sqcup \top \equiv \top$ and $C \sqcap \bot \equiv \bot$. If the complement is involutive it also holds that $\neg\neg C \equiv C$. Now if the fuzzy triple satisfies the De Morgan laws (called *dual* triple), we additionally have, $\neg(C \sqcup D) \equiv \neg C \sqcap \neg D$ and $\neg(C \sqcap D) \equiv \neg C \sqcup \neg D$. For example the fuzzy triples, $\langle c_L, t_L, u_L\rangle$ and $\langle c_L, t_G, u_G\rangle$, are dual triples. Moreover, for any dual triple $\langle c, t, u\rangle$ and S-implication $\mathcal{J}_S$ it holds that, $\forall R.C = \neg(\exists R.\neg C)$. For example the quadruple $\langle c_L, t_G, u_G, \mathcal{J}_{KD}\rangle$, satisfies this equivalence. Furthermore, if the fuzzy triple satisfies the laws of contradiction and excluded middle, then the following properties of boolean logic hold: $(C \sqcap \neg C \equiv \bot)$ and $(C \sqcup \neg C \equiv \top)$. For example, the triple $\langle c_L, t_L, u_L\rangle$, satisfies these laws.

It is important to notice that the classical properties of Boolean algebra, like the De Morgan the excluded middle and the contradiction laws, do not always hold in fuzzy set theory and logic. For example for the triple $\langle c_L, t_G, u_G\rangle$ we have $\max(0.6, 1 - 0.6) = 0.6 \neq 1$. The consequences of this property is that many well-known techniques in DLs, like *internalization* do not carry over to f-DLs. Fortunately, the laws of excluded middle and contradiction can be simulated.

Lemma 3.4 *For all $a \in \Delta^{\mathcal{I}}$, $n \in [0, 1]$, and interpretations $\mathcal{I}$*

1. *either $C^{\mathcal{I}}(a) < n$, or $C^{\mathcal{I}}(a) \geq n$ [8] [2], and*
2. *if $C^{\mathcal{I}}(a) \geq n$ and $(\neg C)^{\mathcal{I}}(a) \geq n$, then $\bot^{\mathcal{I}}(a) \geq \max(0, n - \epsilon)$, where ϵ is the* equilibrium *[3] of a fuzzy complement.*

Point 1 simulates the DL axiom $\top \sqsubseteq C \sqcup \neg C$, while point 2 the axiom $C \sqcap \neg C \sqsubseteq \bot$. For example for $\langle c_L, t_G, u_G\rangle$, where $\epsilon = 0.5$ if $n = 0.7$, then $0 = \bot^{\mathcal{I}}(a) \geq 0.2$, which is a contradiction, while for $n = 0.3$, $\bot^{\mathcal{I}}(a) \geq 0$, which is valid. Indeed it is possible that $C^{\mathcal{I}}(a) \geq 0.3$ and $(\neg C)^{\mathcal{I}}(a) \geq 0.3 = C^{\mathcal{I}}(a) \leq 0.7$.

[2] Similarly either $C^{\mathcal{I}}(a) \leq n$ or $C^{\mathcal{I}}(a) > n$ for all $n \in [0, 1]$

3.3 Inference Services

In the current section we will present the inference problems of f-DLs. An f-$\mathcal{SHOIQ}$ knowledge base Σ is *satisfiable* (*unsatisfiable*) iff there exists (does not exist) a fuzzy interpretation $\mathcal{I}$ which satisfies all axioms in Σ. An f-$\mathcal{SHOIQ}$-concept C is *n-satisfiable* w.r.t. Σ iff there exists a model $\mathcal{I}$ of Σ for which there is some $a \in \Delta^{\mathcal{I}}$ such that $C^{\mathcal{I}}(a) = n$, and $n \in (0, 1]$; C subsumes D w.r.t. Σ iff for every model $\mathcal{I}$ of Σ we have $\forall d \in \Delta^{\mathcal{I}}, C^{\mathcal{I}}(d) \leq D^{\mathcal{I}}(d)$; a fuzzy ABox $\mathcal{A}$ is *consistent* (*inconsistent*) w.r.t. a fuzzy TBox $\mathcal{T}$ and RBox $\mathcal{R}$ if there exists (does not exist) a model $\mathcal{I}$ of $\mathcal{T}$ and $\mathcal{R}$ that satisfies each assertion in $\mathcal{A}$. Given a fuzzy concept axiom, a fuzzy role axiom or a fuzzy assertion ϕ, Σ *entails* ϕ, written $\Sigma \models \phi$, iff for all models $\mathcal{I}$ of Σ, $\mathcal{I}$ satisfies ϕ.

Let $\Sigma = \langle \mathcal{T}, \mathcal{R}, \mathcal{A} \rangle$, be a fuzzy knowledge base. It has been proved that all inference problems of f-DLs can be reduced to ABox consistency w.r.t. $\mathcal{T}$ and $\mathcal{R}$. More precisely, C is n-satisfiable w.r.t. Σ iff $\langle \mathcal{T}, \mathcal{R}, \{(\mathtt{a} : C) \geq n\} \rangle$ is satisfiable, $\Sigma \models \phi \bowtie n$ iff $\Sigma = \langle \mathcal{T}, \mathcal{R}, \mathcal{A} \cup \{\phi \neg \bowtie n\} \rangle$ is unsatisfiable (where $\neg \bowtie$ represents the *negation* of inequalities, e.g. $\neg \geq = <$), and $\Sigma \models C \sqsubseteq D$ iff $\Sigma = \langle \mathcal{T}, \mathcal{R}, \mathcal{A} \cup \{(\mathtt{a} : C) \geq n, (\mathtt{a} : D) < n\} \rangle$, for both $n \in \{n_1, n_2\}$, $n_1 \in (0, 0.5]$ and $n_2 \in (0.5, 1]$, is unsatisfiable [9]. In the past, the consistency problem in f-DLs has been considered w.r.t. to a simple and acyclic TBox. Only recently a procedure for deciding fuzzy ABox consistency w.r.t. general and/or cyclic TBoxes has been developed [8]. In classical DLs general and cyclic TBoxes were handled by a process called *internalization* [2]. As we mentioned previously, internalization is based on the law of excluded middle, which is not always satisfied in f-DLs. In [8] the authors use the case analysis of lemma 3.4, providing the following result,

Lemma 3.5 *[8] A fuzzy interpretation $\mathcal{I}$ satisfies $C \sqsubseteq D$ iff for all $n \in [0, 1]$ and $a \in \Delta^{\mathcal{I}}$, either $C^{\mathcal{I}}(a) < n$ or $D^{\mathcal{I}}(a) \geq n$.*

Hence, the semantic restrictions of a TBox $\mathcal{T}$ can be encoded in mutually exclusive fuzzy assertions.

4 Reasoning in Fuzzy Description Logics

As we have seen f-DLs constitute a sound extension of classical DLs. Hence, the techniques used to perform reasoning in classical DLs could be extended to provide reasoning support for f-DLs. Since all inference problems can be reduced to the problem of ABox consistency w.r.t. an RBox $\mathcal{R}$, a procedure that decides this problem should be constructed. In classical DLs this is done with the aid of tableaux algorithms that given an ABox $\mathcal{A}$ they try to construct a *tableau* for $\mathcal{A}$ [2], i.e., an abstraction of a model of $\mathcal{A}$ which has a tree or forest-like

shape. In such trees, nodes correspond to objects in the model, and edges to certain relations that connect two nodes. Each node x is labelled with the set of concepts that it belongs to ($\mathcal{L}(x)$), and each edge $\langle x, y\rangle$ with a set of roles that connect two nodes x, y ($\mathcal{E}(\langle x, y\rangle)$). In the fuzzy case, since now we have fuzzy assertions, we extend these mappings to also include the membership degree that a node belongs to a concept. More formally we have the following definition.

Definition 4.1 *If $\mathcal{A}$ is a fuzzy ABox, $\mathcal{R}$ a fuzzy RBox, $\boldsymbol{R}_\mathcal{A}$ is the set of roles occurring in $\mathcal{A}$ and $\mathcal{R}$ together with their inverses, sub($\mathcal{A}$) is the set of sub-concepts that exist in $\mathcal{A}$ and $\mathbf{I}_\mathcal{A}$ is the set of individuals in $\mathcal{A}$, a fuzzy tableau T for $\mathcal{A}$ w.r.t. $\mathcal{R}$, is defined to be a quadruple ($\boldsymbol{S}$, $\mathcal{L}$, $\mathcal{E}$, $\mathcal{V}$) such that: $\boldsymbol{S}$ is a set of elements, $\mathcal{L} : \boldsymbol{S} \times sub(\mathcal{A}) \rightarrow [0, 1]$ maps each element and concept to the membership degree of that element to the concept, $\mathcal{E} : \boldsymbol{R}_\mathcal{A} \times \boldsymbol{S} \times \boldsymbol{S} \rightarrow [0, 1]$ maps each role and pair of elements to the membership degree of the pair to the role, and $\mathcal{V} : \mathbf{I}_\mathcal{A} \rightarrow \boldsymbol{S}$ maps individuals occurring in $\mathcal{A}$ to elements in $\boldsymbol{S}$.*

Additionally, a fuzzy tableau should satisfy certain properties of the semantics of the f-DL language [6, 5]. For example, if concept conjunction is performed by the Gödel t-norm, then for $a \in \mathbf{S}$, $C, D \in sub(\mathcal{A})$, $n \in [0, 1]$, and $\mathcal{L}(s, C \sqcap D) \geq n$, it follows that both $\mathcal{L}(s, C) \geq n$ and $\mathcal{L}(s, D) \geq n$ must hold. Currently, we have tableau definitions for the languages f_{KD}-$\mathcal{SI}$ and f_{KD}-$\mathcal{SHIN}$ [6, 5], while it is an open research issue to define a tableau structure for f_{KD}-$\mathcal{SHOIQ}$ as well as for f-DLs with other norm operations. One difficult point in these definitions is to handle transitivity in the new framework. In classical DLs the tree-like structure is preserved by pushing concepts of the form $\forall R.C$ from a node s to a node t if $R(s, t)$ exists. This is based on the observation that if $\mathsf{Trans}(R)$ then the axiom $\forall R.C \sqsubseteq \forall R.(\forall R.C)$, holds [2]. The following lemma characterizes transitivity in f_{KD}-DLs.

Lemma 4.2 *If $(\forall R.C)^\mathcal{I}(a) \geq n$, $R^\mathcal{I}(a, b) \geq r_1$ and $\mathsf{Trans}(R)$ then, in an f_{KD}-DL, $(\forall R.(\forall R.C))^\mathcal{I}(a) \geq n$ holds.*

Thus, as a tableau property we have that, if $\mathcal{L}(s, \forall R.C) \geq n$ and $\mathsf{Trans}(R)$, then either $c(\mathcal{E}(R, \langle s, t\rangle)) \geq n$ or $\mathcal{L}(t, \forall R.C) \geq n$. Another major problem towards our goal in constructing a tableaux reasoning algorithm is to determine if the appropriate blocking techniques can be applied. While this is quite straightforward in f_{KD}-DLs [6, 5, 8], this is very hard when other norms are used.

5 Conclusions

Fuzzy Description Logics are applicable in a number of research and industrial applications that face a vast amount of imprecise and vague information. The last couple of years the work on fuzzy DLs has provided with impressive results

such as the extension of very expressive DL languages, like $\mathcal{SHOIN}$ [10] and $\mathcal{SHOIQ}$, the development of reasoning procedures for fuzzy DLs like f$_{KD}$-$\mathcal{SI}$ [6] and f$_{KD}$-$\mathcal{SHIN}$ [5] and reasoning w.r.t. general inclusion axioms [8], which was considered an open problem for fuzzy DLs for many years. Currently the work on fuzzy DLs is focused on the reasoning problem for f$_{KD}$-$\mathcal{SHOIQ}$, on reasoning with other norms than the ones used in f$_{KD}$-$\mathcal{SI}$ and f$_{KD}$-$\mathcal{SHIN}$, on data-type support and on implementations.

References

[1] Steffen Hölldobler, Tran Dinh Khang, and Hans-Peter Störr. A fuzzy description logic with hedges as concept modifiers. In *Proceedings InTech/VJFuzzy'2002*, pages 25–34, 2002.

[2] I. Horrocks and U. Sattler. A description logic with transitive and inverse roles and role hierarchies. *Journal of Logic and Computation*, 9:385–410, 1999.

[3] G. J. Klir and B. Yuan. *Fuzzy Sets and Fuzzy Logic: Theory and Applications*. Prentice-Hall, 1995.

[4] D. Sánchez and G.B. Tettamanzi. Generalizing quantification in fuzzy description logic. In *Proceedings 8th Fuzzy Days in Dortmund*, 2004.

[5] G. Stoilos, G. Stamou, V. Tzouvaras, J.Z. Pan, and I. Horrocks. The fuzzy description logic f-$\mathcal{SHIN}$. Proc. of the International Workshop on Uncertainty Reasoning for the Semantic Web, 2005.

[6] G. Stoilos, G. Stamou, V. Tzouvaras, J.Z. Pan, and I. Horrocks. A fuzzy description logic for multimedia knowledge representation. Proc. of the International Workshop on Multimedia and the Semantic Web, 2005.

[7] G. Stoilos, G. Stamou, V. Tzouvaras, J.Z. Pan, and I. Horrocks. Fuzzy owl: Uncertainty and the semantic web. Proc. of the International Workshop on OWL: Experiences and Directions, 2005.

[8] G. Stoilos, U. Straccia, G. Stamou, and J.Z. Pan. General concept inclusions in fuzzy description logics. to appear in ECAI, 2006.

[9] U. Straccia. Reasoning within fuzzy description logics. *Journal of Artificial Intelligence Research*, 14:137–166, 2001.

[10] U. Straccia. Towards a fuzzy description logic for the semantic web. In *Proceedings of the 2nd European Semantic Web Conference*, 2005.

Finding Subsumers for Natural Language Presentation

Chris Mellish and Jeff Z. Pan
Department of Computing Science
University of Aberdeen

Abstract

This work is motivated by the task of describing in natural language a concept defined in an OWL DL ontology. However, rather than focussing on linguistic issues, we address the question of how to support natural language presentation with inference. We introduce a new non-standard DL reasoning problem, that of finding subsumers of a concept that are suitable for natural language presentation. We present a solution that works by enumerating successively more complex concepts in the limited language $\mathcal{ALEN}$. Although the search space is formidable, specific optimisations that take into account characteristics of natural language enable it to be tamed. Our initial experiments show that the approach may be quite feasible in practice.

1 Introduction

Knowledge engineers, domain experts and also casual users need better ways to understand ontologies. As the logical structure of ontologies becomes richer, it becomes harder to devise appropriate graphical means of presentation that do not require special training on the part of the users. In this scenario, presentation in natural language is becoming increasingly attractive. Natural language has developed good ways of conveying some complex logical structures and requires no special training.

The work described in this paper takes as its starting point the task of answering in natural language a question *What is A?*, where A (the *target*) is an atomic concept mentioned in some given OWL DL ontology. This may take place as a part of a longer dialogue between person and machine where, for instance, subsequently the person asks *What is B?*, for some atomic B mentioned in the answer to the first question. Rather than considering detailed linguistic aspects of this task, however, we focus on how to support it with appropriate reasoning.

A first attempt at the task of answering *What is A?* might somehow render in natural language the set of ontology axioms that mention A. However, an ontology axiom is not necessarily of appropriate complexity to be expressed as a natural language sentence. Also, it could be misleading to present true, but incomplete information. Finally, important information about the target may arise from logical consequences of the axioms, not only

from explicitly stated axioms. Elsewhere we have used these arguments to argue the need for new kinds of inference, *natural language directed inference* (NLDI), which are capable of deriving those logical consequences suitable for natural language presentation [8].

DL-based Ontologies have available powerful reasoning services, such as classification, subsumption and satisfiability checking. Standard reasoning services, however, require detailed specification of the reasoning goals (e.g. the subsumption service needs to be told exactly which two concepts are to be tested). Since NLDI is a kind of data-driven reasoning with goals that cannot be stated precisely in logical terms, standard DL reasoning services cannot be used immediately to implement NLDI. The challenge is to exploit these efficiently implemented services in more complex ways to make NLDI possible.

What kind of information is needed to answer a question *What is A?* McKeown [7] discusses a number of kinds of facts present in human descriptions of a target A, which include **identification**: *An aircraft carrier is a surface ship*, **attributive**: *A torpedo has an underwater target location*, and **equivalent**: *Wines described as 'great' are fine wines from an especially good village*. In DL terms, these correspond to concept subsumptions $A \sqsubseteq C$ w.r.t. a TBox $\mathcal{T}$, with A (A= $AircraftCarrier$, $Torpedo$, $GreatWine$) being the subsumee. Unfortunately, not all (or even any) of these concept descriptions C need necessarily appear explicitly directly in axioms of the form "$A \sqsubseteq C$" in the ontology.

In this paper we present a procedure for discovering subsumers C of a concept A that might be worth presenting in natural language. Although we allow the ontology $\mathcal{T}$ to be expressed in full OWL DL, nevertheless the subsumers C are in the more limited language $\mathcal{ALEN}$. In a sense, therefore, our real goal is to produce the most specific $\mathcal{ALEN}$ subsumer, which contains all the information known about the target. In general, however, this will be a conjunction, within which the conjuncts could be generated in many possible orders. Instead of generating the single conjunction, therefore, we generate a set of most specific non-conjunctive subsumers, which could then be combined together by conjunction if this was wished. We require these individual conjuncts to be the sort of things that could be presented in individual natural language sentences.

The closest related work is that on non-standard inferences in DLs. On the one hand, our task could be characterised as looking for a least subsumer (other than A itself) of A using a less expressive DL [3]. On the other hand, the task can be regarded as a "matching" problem, "$A \sqsubseteq^? P$", where P is a concept pattern [2]. Unfortunately, existing approaches to both computing least subsumers and matching only apply to less expressive DLs and assume TBoxes to be unfoldable. In this work, apart from assuming the existence of standard reasoning services, we do not assume unfoldability of the axioms.

2 Discovering Subsumers

Answering questions about an ontology is a form of *communication*, and formal theories of communication standardly make reference to models of belief [1]. Here we need to distinguish between two different sets of beliefs – the system's beliefs and the user's beliefs (as in the system's *user model*). The first of these is represented by the original ontology $\mathcal{T}$, whilst the second requires a separate *user knowledge base* $\mathcal{U}$. The user KB is likely to be different from the system KB because otherwise the user would not have sought information

about the target. Hence two notions of subsumption arise:[1]

1. C_1 **system-subsumes** C_2 iff $\mathcal{T} \models C_2 \sqsubseteq C_1$;
2. C_1 **user-subsumes** C_2 iff $\mathcal{U} \models C_2 \sqsubseteq C_1$.

We make the assumption that the user KB has the same vocabulary as the system KB and is an approximation to the system KB:

For all C_1, C_2, if $\mathcal{U} \models C_1 \sqsubseteq C_2$ then $\mathcal{T} \models C_1 \sqsubseteq C_2$.

I.e. everything the user knows is also known by the system. Our framework allows for any $\mathcal{U}$ satisfying the above constraints; in particular, one could have $\mathcal{U} = \mathcal{T}$ (ignoring this particular distinction) or have $\mathcal{U}$ be something that is built up over the dialogue as a result of the information that the user is told. Our current implementation considers just the first question of a dialogue, where the user has no initial domain knowledge, i.e. $\mathcal{U} = \phi$.

We can now present the task more formally:[2]

> **The natural language subsumer enumeration problem:** Given a system-satisfiable target named concept A, find the most specific (w.r.t. user-subsumption) non-conjunctive concepts C which system-subsume A, do not user-subsume A and are appropriate for natural language presentation.

Here we use the user KB to decide which of these are worth presenting, because the extra knowledge of the system may obscure certain user-relevant distinctions. For example, consider an ontology $\mathcal{T}$ which includes the axiom $C_1 \equiv C_2$ not in $\mathcal{U}$. Given this knowledge, if C_1 subsumes a target A, so does C_2. As C_1 and C_2 are system-equivalent, from the system's point of view the choice of which to present is arbitrary. From the point of view of the user who is not in possession of all the knowledge in $\mathcal{T}$, however, the descriptions C_1 and C_2 provide distinct information, and so it is worth considering presenting *both* of them.

Our approach is to enumerate concepts C subsuming A via a search through all possible concepts in $\mathcal{ALEN}$. At each stage we can test whether an enumerated concept C system-subsumes A using the subsumption reasoning service. Some such concepts are returned as *candidates* which may be user-least non-conjunctive subsumers of A. The set of candidates is then further filtered, in order to obtain a set, no element of which user-subsumes another element or user-subsumes the target (exactly one of a set of user-equivalent concepts is returned). In practice, the enumeration of candidates is organised in such a way as to avoid many candidates that would otherwise be filtered out by the second step.

3 The Refinement Relation

The search space is expressed in terms of a *refinement* relation $\searrow$, where $C_1 \searrow C_2$ indicates that C_2 results from a minimal change to C_1 that makes it more syntactically complex. The

[1] In the following, we will also sometimes mention corresponding varients of other logical tests (e.g. system- vs user-*equivalence*).

[2] We consider extra requirements for natural language presentation in Section 5.

search starts from the most general concept $\top$, working to candidates C_1 such that $\top \searrow C_1$, then to candidates C_2 such that $C_1 \searrow C_2$, and so on. Thus we are exploring the set of concepts α such that $\top \searrow^* \alpha$, where $\searrow^*$ is the transitive closure of $\searrow$.

To limit the amount of redundancy in the search space, concepts are assumed to be in a normal form, so that conjunctions are only allowed inside $\exists$ constructs, and conjunctive information at the top level or just inside $\forall$ constructs must be expanded out to yield multiple candidates. Nested conjunctions are flattened. Within conjunctions, conjuncts (if present) occur in the following order: negations (in lexicographic order) before role restrictions (with properties in a fixed order) before atomic concepts (in lexicographic order). Within the role restrictions, all the restrictions for a given role occur together, in the order: number restrictions before $\forall$ before $\exists$. For any role P, there are at most two number restrictions: either a single $=$ restriction or at most one of each of $\leq$ and $\geq$, in this order. For any role P, at most one $\forall P$ restriction can occur within any allowed conjunction.

The following exhaustive definition can also be read as the basis of an algorithm for enumerating, for a given concept α, the concepts β such that $\alpha \searrow \beta$. This gives us the original theoretical search space. In the actual implementation, we make a number of optimisations compared to using the basic refinement relation, as detailed below.

$\top \searrow A_i$ if A_i is a named concept
$\top \searrow \neg A_i$ if A_i is a named concept
$\top \searrow (\exists P.\top)$ if P is a role name
$\top \searrow (\forall P.\alpha)$ if P is a role name and $\top \searrow \alpha$
$\top \searrow (\geq nP)$ if P is a simple role and $0 \leq n \leq \pi$, where π a large number (1000000)
$\top \searrow (\leq nP)$ if $(0 \leq n \leq \pi)$, and P is a simple role
$\top \searrow (= nP)$ if $(0 \leq n \leq \pi)$, and P is a simple role
$(\exists P.\alpha) \searrow (\exists P.\beta)$ if $\alpha \searrow \beta$
$(\forall P.\alpha) \searrow (\forall P.\beta)$ if $\alpha \searrow \beta$
$\alpha \searrow (\beta \sqcap \alpha)$ where α is not a conjunction, $\top \searrow \beta$, this is within the scope of an $\exists$ and β is of a type allowed before α by the conjunction ordering rules.
$\alpha_1 \sqcap \alpha_2 \sqcap \ldots \alpha_n \searrow \beta \sqcap \alpha_2 \sqcap \ldots \alpha_n$ if $\alpha_1 \searrow \beta$
$\alpha_1 \sqcap \alpha_2 \sqcap \ldots \alpha_n \searrow \beta \sqcap \alpha_1 \sqcap \alpha_2 \sqcap \ldots \alpha_n$ if $\top \searrow \beta$ and β is of a type allowed before the α_i by the conjunction ordering rules.

The above relation $\searrow$ has the following properties. These can be shown by induction on the number of symbols (other than $\top$) occurring in C_1.

Lemma 1. If C_1 is a satisfiable concept in $\mathcal{ALEN}$ expressed in terms of the vocabulary of the ontology then $\top \searrow^* \beta$ for some concept β logically equivalent to C_1.

Lemma 2. If $C_1 \searrow C_2$ then C_2 is strictly more syntactically complex then C_1 (for a range of possible complexity metrics)

Lemma 3. If $C_1 \searrow C_2$ then C_1 subsumes C_2 (hence C_1 system- and user-subsumes C_2)

The first of these means that we can reach all possible concepts through $\searrow$ (notice that we do not need to allow $\bot$ in formulae, because of equivalences such as $(\forall P.\bot) \equiv (= 0P)$). Because of Lemmas 2 and 3, a search following the transitive closure of $\searrow$ is both a search in terms of increasing syntactic complexity and also a (perhaps rather slow) search in terms of increasing logical specificity.

4 The Search Strategy

The search algorithm works with a derived relation $\searrow\!\searrow$, defined in terms of $\searrow$, which produces results strictly user-subsumed by the original concept. Again, this definition can be thought of as an algorithm to enumerate the relevant refinements:

$C_1 \searrow\!\searrow C_2$ iff $C_1 \searrow C_2$ and C_2 does not user-subsume C_1
or $\exists C_3.C_1 \searrow C_3$, C_3 user-subsumes C_1 and $C_3 \searrow\!\searrow C_2$

Our search is organised in a depth-first manner. If a point in the tree is reached where the concept C_i does not system-subsume the target, there is no point in considering further refinements of this concept. By Lemma 3, such further refinements will be subsumed, and hence also system-subsumed, by C_i. One of these cannot system-subsume the target, because if so then, by transitivity of system-subsumption, C_i would have to as well. If such a C_i is reached, search down that path of the tree is terminated. Lemma 2 means that we can achieve termination by terminating the search path when it reaches a concept with a complexity equal to or exceeding a preset limit. A generated concept is returned as a candidate exactly when none of the first $\searrow\!\searrow$ descendents both are of acceptable size (see Section 5) and also system-subsume the target. This tends to lead to only user-least solutions being returned.

The above properties guarantee that the above search is complete, in that all user-least non-conjunctive concepts (or concepts logically equivalent to them) that system-subsume the target and have a size below the limit are enumerated (as well as possibly some other concepts). Logically equivalent solutions could, however, be generated many times. The search is partially correct, in that all candidates system-subsume the target. User-minimality (and so the rest of correctness) is then ensured by the subsequent filtering process.

Apart from the natural language based optimisations discussed in the next section, space does not permit us to describe in detail a number of other optimisations used to enhance the basic search approach. Firstly, the relevance filter of [9] is used to limit the vocabulary of atomic concept and role names used in the enumerated concepts. Secondly, we disallow the addition of elements to conjunctions which are either user-subsumed by or user-subsume existing conjuncts. Thirdly, a focussed search is used to ensure that any introduced number restriction is in fact the most specific such restriction such that the candidate with this restriction in it subsumes the target.

5 Natural Language Direction

Natural language easily expresses conjunctive information and often produces scope ambiguities in complex examples involving disjunction and negation. $\mathcal{ALEN}$ allows no disjunctions or complex negations and thus is a natural DL to act as the target for natural language based approximation.

The following summarises other ways in which we incorporate natural language direction into the algorithm, sometimes at the expense of *logical* completeness.

Concept Complexity. Because there is a limit to the complexity of a concept that can be presented in a sentence, we impose a size limit on concepts. Each negation or conjunction in a concept counts 1 towards the "size" of a formula, and each quantifier counts $n + 1$ towards the calculation, where n is the number of enclosing quantifiers. A complexity limit of around 4 or 5 seems roughly plausible for what can give rise to a comprehensible natural language sentence.

Specificity and Complexity. Because we use $\searrow\!\!\searrow$, rather than $\searrow$ in the algorithm, refinements of a concept that are user-equivalent to it are not returned. Given that these refinements are more complex than the original, the effect is that (apart from where equivalent concepts are reached by different paths through the search space) only one of the smallest of a set of user-equivalent concepts is ever returned. The consequence is that, roughly speaking, the simplest way of saying some particular content is chosen (c.f. Grice's principle of brevity [4]).

Introducing new Terminology. Since we are interested in finding just the most specific concepts that subsume the target, whenever $\top$ is refined to a named concept it can be refined to a most user-specific named concept such that the whole candidate, with this concept substituted, system-subsumes the target. Similarly, when a concept $\neg A_i$ is introduced, it can be done with a most user-general A_i such that the candidate still system-subsumes the target. Unfortunately, if $\mathcal{U} = \phi$, then there are no non-trivial user-subsumption relationships between named concepts, and so this measure has no effect. As a result, for instance, if a target is system-subsumed by $(\exists hasPet.Poodle)$ then concepts like $(\exists hasPet.Animate)$ will also appear as candidates (none of these is user-subsumed by any other). All of these convey new information to the imagined user, but they are not all equally good for natural language presentation. Given that the user has the opportunity in the dialogue to ask followup questions about mentioned atomic concepts, it is actually complete in a dialogue sense simply to return just the concept including the *system* most specific concepts in these cases. In contrast, if all the above concepts are presented then the impression may be given that there are no noteworthy system-subsumption relationships between them.

Negations. In natural languages, negation is used primarily to *deny* an explicitly or implicitly available proposition. This means that (in the absence of some specific context) it would be strange to answer the question *What is a mammal?* with something like *Every mammal is not a mushroom.* Our interpretation of this requirement is the condition that a negation $\neg\alpha$ can only be generated if it is within a conjunction where there is another conjunct (which must be a positive atomic concept) β such that $\alpha \sqcap \beta$ is satisfiable. This is a kind of "β but not α" negation. For instance, it is perfectly reasonable to say "... a pizza but not a vegetarian dish", because it is possible to be both a pizza and a vegetarian dish.

Trivial Universals. If instances of a given concept cannot possibly have values for a role P then all $\forall$ restrictions on this role trivially subsume the concept. Such restrictions are however not appropriate to be expressed. This means that it would be strange to answer the question *What is a SpicyTopping?* with something like *A SpicyTopping can only have*

a pizza as a topping. In this example, pizza toppings cannot themselves have toppings. This means that, for instance, $\forall topping.Pizza$ system-subsumes $SpicyTopping$. In this situation, *any* concept of the form $(\forall P.\alpha)$ system-subsumes the target. Although logically each of these is a subsumer, in natural language terms each of them is trivial and not worth expressing. The same problem can arise at any nesting within a candidate. Our solution to this problem is that if at any point we are planning to insert the concept $(\forall P.\top)$ at some point in a candidate then first of all we look to see whether the concept with $(\forall P.\bot)$ in this position system-subsumes the target. If so, then we judge that any universal would be trivial and refrain from introducing one.

6 Discussion

Our prototype considers just the first question of a question answering dialogue, where $\mathcal{U} = \emptyset$. It is implemented in SWI Prolog (version 5.4.7), using a DIG interface for Prolog [5]. RacerPro (version 1.9.0) is used via DIG to provide all reasoning services w.r.t. the system KB (e.g. system-subsumption) and results are cached. As $\mathcal{U} = \phi$, the checking of user-subsumption is implemented structurally in Prolog with simple subset of the algorithm of [6], to avoid overheads introduced by using the DIG interface.

The following table shows statistics about some examples. The two ontologies used are a food ontology from `http://www.w3.org/TR/owl-guide/food.rdf` and a pizza ontology from `http://www.co-ode.org/ontologies/pizza/pizza_20041007.owl`. For each example, we give the complexity limit, the search space size[3], the target, the number of calls to RacerPro needed, the number of subsumers found by the initial search and the number that this is filtered to.

Ontology	Limit	Search Space	Target	Calls	Subsumers	Filtered
Food	4	2.8E8	$FruitCourse$	2354	87	63
Food	5	1.8E10	$FruitCourse$	4706	124	93
Food	6	1.2E12	$FruitCourse$	13756	385	324
Pizza	4	2.1E8	$AmericanHot$	1386	80	76
Pizza	5	1.8E10	$AmericanHot$	2116	80	76
Pizza	6	1.5E12	$AmericanHot$	3970	99	95

The concepts found subsuming $AmericanHot$, assuming complexity limit 4, include:

$\neg\, VegetarianPizza \sqcap NamedPizza$
$(\forall hasTopping.\neg\, ArtichokeTopping \sqcap PizzaTopping)$
$(\forall hasTopping.(\leq 1 hasSpiciness))$
$(\exists hasTopping.(\forall hasSpiciness.Spiciness) \sqcap MozzarellaTopping)$
$(\geq 5 hasTopping)$

Our results so far show that it can be possible, given an appropriate search strategy and natural language direction, to solve the natural language subsumer enumeration problem

[3]The size of the basic refinement search space, without any optimisations, ignoring lexicographic ordering of conjuncts and assuming a maximum cardinality of 3. The implementation allows cardinalities up to 1 million, but using this figure in the calculation of the raw search space size would be misleading.

in spite of the huge search space involved. Actual runtimes are still somewhat problematic (the above calculation for *AmericanHot* with limit 5 takes about 31 seconds elapsed time on a 1695MHz PC), but further optimisations could be introduced if the reasoner and the rest of the system were in the same process.

7 Acknowledgements

Thanks to Racer Systems GmbH for free use of RacerPro, Jan Wielemaker for SWI Prolog, and Zhisheng Huang and Cees Visser for their DIG interface for Prolog. Thanks to Holger Wache and Frank van Harmelen for useful discussions. Chris Mellish's contribution to this work is funded by EPSRC grant GR/S62932/01. Jeff Z. Pan's contribution is partially funded by the FP6 Network of Excellence EU project Knowledge Web (IST-2004-507842).

References

[1] J. Allen. *Natural Language Understanding*. Benjamin Cummings, 1995.

[2] F. Baader, R. Küsters, A. Borgida, and D. McGuinness. Matching in description logics. *Journal of Logic and Computation*, 9(3):411–447, 1999.

[3] S. Brandt, R. Küsters, and A. Turhan. Approximation and difference in description logics. In *Proc. of the 8th Int. Conf. on Principles of Knowledge Representation and Reasoning (KR'2002)*, pages 203–214, 2002.

[4] H. P. Grice. Logic and conversation. In P. Cole and J. Morgan, editors, *Syntax and Semantics: Vol 3, Speech Acts*. Academic Press, 1975.

[5] Zhisheng Huang and Cees Visser. An Extended DIG Description Logic Interface for Prolog. Technical Report SEKT/2004/D3.4.1.2/v1.0, Dept of Artificial Intelligence, Vrije Universiteit Amsterdam, 2004.

[6] R. Küsters and R. Molitor. Structural subsumption and least common subsumers in a description logic with existential and number restrictions. *Studia Logica*, 81(2):227–259, 2005.

[7] K. McKeown. *Text Generation: Using Discourse Strategies and Focus Constraints to Generate Natural Language Text*. Cambridge University Press, 1985.

[8] C. Mellish and X. Sun. Natural language directed inference in the presentation of ontologies. In *Procs of the Tenth European Workshop on Natural Language Geeration*, Aberdeen, Scotland, 2005.

[9] Dmitry Tsarkov, Alexandre Riazanov, Sean Bechhofer, and Ian Horrocks. Using vampire to reason with owl. In Sheila A. McIlraith, Dimitris Plexousakis, and Frank van Harmelen, editors, *Procs of the 2004 International Semantic Web Conference (ISWC 2004)*, pages 471–485. Springer LNCS 3298, 2004.

SHIN ABox Reduction

Achille Fokoue, Aaron Kershenbaum, Li Ma,
Edith Schonberg, Kavitha Srinivas, Rose Williams
IBM Research, Hawthorne, NY 10532
achille, aaronk, malli, ediths, ksrinivs, rosemw@us.ibm.com

May 12, 2006

Abstract

We propose a technique to make consistency detection scalable for large Aboxes in secondary storage. We use static analysis of knowledge representation with summarization techniques to produce a dramatically reduced *proxy Abox*. We show that the *proxy Abox* is consistent only if the original Abox is also consistent. We also show that, in practice, our techniques dramatically reduce the time and space requirements for consistency detection.

1 Introduction

All common reasoning tasks in expressive DL ontologies reduce to consistency detection, which is well known to be intractable in the worst-case [2]. Given that the size of an Abox may be in the order of millions of assertions, this complexity poses a serious challenge for the practical use of DL ontologies.

We propose new techniques that make consistency detection scalable for SHIN Aboxes with millions of assertions. Ontology Aboxes often reside in transactional databases, so we do not assume that an Abox will fit into memory. With these new techniques, we are able to efficiently extract a small set of assertions from the database that represents the entire Abox, and reason over this small set in memory.

We first use static analysis to isolate a subportion of the Abox that captures all the *global* effects that can occur in reasoning, henceforth referred to as the global-effects Abox. By *global* effects, we mean effects that propagate through the Abox to affect an individual's membership in a given concept. In practice, isolating *global effects* results in substantial reductions in the size of the Abox, because most roles participate in local rather than global effects.

Once we've isolated *global* effects in the Abox, we use summarization techniques to dramatically reduce this subportion of the Abox further to produce a *proxy Abox*. For example, in the largest of the 4 ontologies that we studied, we reduced an Abox of 874K individuals and 3.5 million assertions to a *proxy Abox* with 18 individuals and 49 role assertions.

The power of creating such a *proxy* is that we can replace the consistency check on the global-effects Abox $\mathcal{A}'$ with a consistency check on a dramatically reduced *proxy*. Specifically, if the *proxy* is consistent, we are guaranteed that $\mathcal{A}'$ is consistent. If the *proxy* is inconsistent, it can still be used to partition $\mathcal{A}'$.

Our key contributions in this paper are as follows: (a) We present a technique to use static analysis of knowledge representation to isolate a portion of the Abox where *global* effects are possible. (b) We construct a dramatically reduced *proxy* of this portion using summarization techniques. (c) We present a method to further identify local effects in the global-effects Abox, based on the proxy Abox, to handle cases where the summarization may have been too conservative in building the *proxy*. (d) We show the efficiency of these techniques applied to 4 real ontologies.

2 Local/Global Effect Partitioning

We present several criteria for safely removing role assertions from an Abox $\mathcal{A}$. The reduced Abox is consistent iff $\mathcal{A}$ is consistent. In practice, applying these criteria results in both significantly shrinking the Abox and in partitioning it into many disconnected Aboxes, many of which consist of a single individual, which can be checked for consistency using concept satisfiability.

Our criteria for role assertion removal is based on the assumption that any concept in the $clos(\mathcal{A})$ can reach the concept set of any individual in $\mathcal{A}$ from the application of tableau expansion rules for SHIN. We define $clos(\mathcal{A})$ as $\bigcup_{a:C \in \mathcal{A}} clos(C)$ where $clos(C)$ is a set that contains C and all its sub-concepts (where C is in NNF). Note that our formal definition of $clos(\mathcal{A})$ differs from [6] in that we do not include $\neg C$ in $clos(C)$, due to lesser expressiveness of SHIN compared to SHIQ.

We assume that a and b are named individuals in the original $\mathcal{A}$, x is a new unnamed individual introduced as a result of tableau expansion rules, C is a concept in $clos(\mathcal{A})$, and R is a role. Let $\mathcal{L}$ be a mapping from each individual in $\mathcal{A}$ to a set of concepts in $clos(\mathcal{A})$, such that $a:C \in \mathcal{A}$ iff $C \in \mathcal{L}(a)$. $\mathcal{L}(a)$ is the *concept set* of a. An individual b is said to be an R-neighbor of a iff there is an assertion $Q(a,b)$ or $Q^-(b,a)$ in $\mathcal{A}$ where $Q \in \underline{R}$ (where $\underline{\mathrm{R}} = \{\ Q |\ Q \sqsubseteq^* R\}$ and $\sqsubseteq^*$ is the reflexive transitive closure of the sub-role relation). The SHIN tableau expansion rules can merge individuals, add membership assertions of the form $a:C$ where $C \in clos(\mathcal{A})$, add unnamed individuals, and add new role assertions

of the form $R(a, x)$ or $R(a, b)$ to $\mathcal{A}$.

We say that an expansion rule has a *global effect* if it uses an existing role assertion to add new assertions to $\mathcal{A}$ or to detect a clash. For example, a concept C will be propagated to the concept set $\mathcal{L}(b)$ of a named individual b if $a : \forall R.C$ and $R(a, b) \in \mathcal{A}$. The $\forall$-rule, $\leq$-rule, and $\forall_+$-rule can have global effects. In contrast, the $\exists$-rule and $\geq$-rule do not use any existing role assertions to alter the Abox, and are hence local effect rules; but these rules can generate new role assertions, so we call concepts of the form $(\exists R.C)$ and $(\geq nR)$ *R-generators*.

A role assertion $R(a, b)$ is said to be a global-effect role assertion iff there is at least one execution of the tableau algorithm in which it is used by a global-effect rule, or it is part of an explicit clash involving both a and b. Otherwise, it is a local-effect role assertion and cannot affect the outcome of a consistency check. Our criteria are designed to detect and remove local effect role assertions.

2.1 Role-based Local Effect Detection

We make the simple observation that if roles R and R^- are never used in any universal or maximum cardinality restrictions, then a role assertion $R(a, b)$ can never be used in a global-effect rule, so it can safely be ignored.

Definition 1: A role R is *part of a universal restriction* $\forall P.C$ iff $R \in \underline{P}$. (Similarly for maximum cardinality restriction). A role R is *part of the universal restrictions of an Abox* $\mathcal{A}$ iff there is a universal restriction $\forall P.C \in clos(\mathcal{A})$ such that $R \in \underline{P}$. (Similarly for maximum cardinality restrictions).

Theorem 2: A role assertion $R(a, b)$ can safely be removed from an Abox $\mathcal{A}$ if neither R nor its inverse R^- is part of the universal restrictions or maximum cardinality restrictions of $\mathcal{A}$.

Proof(*sketch*): Direct consequence of more the general Theorem 9. ∎

2.2 Assertion-based Local Effect Detection

We specify criteria for local-effect role assertion removal even in the presence of universal and maximum cardinality restrictions. The criteria for $\forall$-rule and $\forall_+$-rule is as follows: Let role R be part of a universal restriction $\forall P.C$. A role assertion $R(a, b)$ is *removable with respect to* $\forall P.C$ iff $b : C \in \mathcal{A}$ and R has no transitive superroles. $R(a, b)$ is *removable with respect to universal restrictions in an Abox* $\mathcal{A}$ iff it is removable with respect to all universal restrictions $\forall P.C$, where R or R^- is part of $\forall P.C$.

Next, we note that for the $\leq$-rule to have a global effect from merging there must be a maximum cardinality restriction $(\leq nP)$ in $clos(\mathcal{A})$, and an individual a with more than n P-neighbors. To ensure that an individual a has no more than n P-neighbors, we need to be able to compute an upper bound on P-neighbors of a safely. In particular, when counting $a's$ P-neighbors, it is

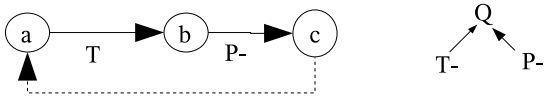


Figure 1: Effect of mergers on *a's* neighbors

important to include named individuals and unnamed individuals that can become P-neighbors through P-generators and mergers. One example of a merger is shown in Figure 1, where c can be merged with a because P^- is attracted to T^- through a shared super role Q which is part of a maximum cardinality restriction. We define the conditions under which the upper bound of $a's$ P-neighbors can be computed safely and efficiently below. First, we introduce the notion of an an *attractant* for P, to prevent new P-neighbors from mergers as shown in Figure 1:

Definition 3: For a given role P, we define $attractant(P)$ as follows: $T \in attractant(P)$ iff there is a role Q such that $P \sqsubseteq^* Q, T \sqsubseteq^* Q, \leq nQ \in clos(\mathcal{A})$.

We say that P is *safe* in $\mathcal{A}$ iff one of the following conditions is satisfied for the role P:

(a) the $attractant(P) \subseteq \{P\}$ and $attractant(P^-) \subseteq \{P^-\}$, or

(b) for all roles R such that either R or R^- is in $\underline{P}$, there are no R-generators in $clos(\mathcal{A})$.

Definition 4: An individual a in $\mathcal{A}$ is *mergeable* iff at some step of any execution of the tableau algorithm on $\mathcal{A}$, a is merged with a *named* individual b.

Lemma 5: Let P be a role that is safe in $\mathcal{A}$. During an execution of the tableau algorithm on $\mathcal{A}$ the following holds: if there is a unnamed individual x such that P or P^- is in the $\mathcal{L}(< parent(x), x >)$, then $|\mathcal{L}(< parent(x), x >)| = 1$, where $parent(x)$ denotes the parent node of x in the completion forest.

Proof: Proved by induction [3] on the iterations of the tableau algorithm.

Theorem 6: An individual a in $\mathcal{A}$ is *not mergeable* in $\mathcal{A}$ if, for any role P and any individual b in $\mathcal{A}$, the following conditions hold:

(1) if a is a P-neighbor of b there is no concept $(\leq nP)$ in $clos(\mathcal{A})$, and

(2) if b is a P-neighbor of a then P is safe in $\mathcal{A}$.

Proof *sketch*: By induction using Lemma 5 [3].■

Definition 7: Let role R be part of a maximum cardinality restriction $(\leq nP) \in clos(\mathcal{A})$. A role assertion $R(a, b)$ is *removable with respect to* $(\leq nP)$ iff

(1) if a is a Q-neighbor of a named individual c, there is no concept of the form $(\leq nQ)$ in $clos(\mathcal{A})$, and

(2) if a named individual c is a Q-neighbor of a then Q is safe in $\mathcal{A}$, and

(3) P is safe in $\mathcal{A}$ and its only mininum cardinality is of the form $\geq 1P$ $(\exists P.\top)$, and

(4) $|P(a)| + |Some(P, a)| \leq n$, where $Some(P, a) = \{\exists P.C \in clos(\mathcal{A}) \mid$ there is no P-neighbor c of a such that $c : C \in \mathcal{A}\}$.

$R(a,b)$ is *removable with respect to maximum cardinality restrictions in an Abox* $\mathcal{A}$ iff it is removable with respect to all maximum cardinality restrictions $\leq nR$, where R or R^- is part of $\leq nR$. Note that, by Theorem 6, (1) and (2) imply that a is not mergeable in $\mathcal{A}$.

Definition 8: A role assertion $R(a,b)$ is *removable with respect to maximum cardinality restrictions in an Abox* $\mathcal{A}$ iff the following holds: if R (resp. R^-) is part of a maximum cardinality restriction $(\leq nP) \in clos(\mathcal{A})$, then $R(a,b)$ (resp. $R^-(b,a)$) is removable with respect to $(\leq nP)$. $R(a,b)$ is *removable with respect to maximum cardinality restrictions in an Abox* $\mathcal{A}$ iff it is removable with respect to all maximum cardinality restrictions $\leq nP$, where R or R^- is part of $\leq nP$.

Theorem 9: A role assertion $R(a,b)$ can safely be removed from an Abox $\mathcal{A}$ if it is removable with respect to universal restrictions and removable with respect to maximum cardinality restrictions.

Proof Sketch: Let $R(a,b)$ be a role assertion removable w.r.t. maximum cardinality and universal restrictions in an Abox $\mathcal{A}$. Let $\mathcal{A}'$ be the Abox defined as $\mathcal{A}' = \mathcal{A} - \{R(a,b), R^-(b,a)\}$. If $\mathcal{A}$ is consistent, $\mathcal{A}'$ is obviously consistent. We show that if $\mathcal{A}'$ is consistent, a model of $\mathcal{A}$ can be constructed by applying the tableau algorithm rules in a particular way. [1]

First, for a root node c in the completion forest F, the root node $\alpha(c)$ is defined as follows: if $\mathcal{L}(c) \neq \emptyset$ then $\alpha(c) = c$; otherwise, $\alpha(c) = d$, where d is the unique root node in F with $\mathcal{L}(d) \neq \emptyset$ and $d \doteq c$. Since $\mathcal{A}'$ is consistent, we can apply the tableau expansion rules on $\mathcal{A}'$ without creating a clash in such a way that: (1) $\exists$-rule is never triggered to satisfy a constraint $\exists P.C \in \mathcal{L}(\alpha(a))$ (resp. $\mathcal{L}(\alpha(b))$) where $\leq nP \in clos(\mathcal{A})$, R (resp. R^-) is part of $\leq nP$, and $b : C \in \mathcal{A}$ (resp. $a : C \in \mathcal{A}$), and (2) $\geq$-rule is never triggered to satisfy a constraint $\geq nP \in \mathcal{L}(\alpha(a))$ (resp. $\mathcal{L}(\alpha(b))$) where $\leq nP \in clos(\mathcal{A})$, R (resp. R^-) is part of $\leq nP$, and, in the Abox $\mathcal{A}$, b (resp. a) is one of the n R-neighbors of a (resp. R^--neighbors of b) explicitly asserted to be distinct.

Such a rule application yields a clash-free completion forest F, and the only nodes on which expansion rules may be applicable are $\alpha(a)$ and $\alpha(b)$ (the only applicable rules are $\exists$-rule and $\geq$-rule). Next, we modify F to create a completion forest F' by adding to F the edge $< \alpha(a), \alpha(b) >$ if it was not already in F, and by adding R to $\mathcal{L}(< \alpha(a), \alpha(b) >)$, if it was not already there. We show that F' is complete (i.e. no rules are applicable) and clash-free.

The fact that, in F', $R \in \mathcal{L}(< \alpha(a), \alpha(b) >)$ ensures that the $\exists$ and $\geq$ rules, which may have been applicable on $\alpha(a)$ or $\alpha(b)$ in F, are not applicable on $\alpha(a)$ and $\alpha(b)$ in F'. However, the same fact may now make the $\forall$, $\forall_+$, $\leq$, and $\leq_r$ rules applicable on $\alpha(a)$ or $\alpha(b)$ in F'. We show that this cannot be the case.

The definition of removable w.r.t. universal restrictions obviously ensures that $\forall$ and $\forall_+$ rules are not applicable on $\alpha(a)$ or $\alpha(b)$ in F'. It can be shown

[1] A direct model-theoretic proof cannot easily provided here, see [3] for details.

[3] that $\leq$, and $\leq_r$ rules cannot be applicable on $\alpha(a)$ or $\alpha(b)$ in F' and that F' is still clash free. Thus, a tableau for $\mathcal{A}$ can be built from F' as in [6], which establishes that $\mathcal{A}$ has a model. ∎

3 Proxy Abox

Intuitively, the Abox contains many redundant assertions from the point of view of consistency checking that can be collapsed to create a reduced *proxy Abox*. As an example, if the Abox contains assertions of the form $R(m, c)$ and $R(j, y)$, where m and j are both members of W and c and y are both members of U, we can replace m and j by a proxy individual $w : W$ that is connected by a R relation to a proxy individual $u : U$. Reasoning over the resulting proxy Abox corresponds to reasoning over the original Abox, as shown formally below.

Definition 10: A proxy Abox is an Abox $\mathcal{A}''$ that is generated from any SHIN Abox $\mathcal{A}$ using a mapping function $\mathbf{f}$ that satisfies the following constraints, where $\mathcal{R}$ is the set of all roles and their inverses in an Abox:

(1) if $a : C \in \mathcal{A}$ then $\mathbf{f}(a) : C \in \mathcal{A}''$ (2) if $R(a, b) \in \mathcal{A}$ then $R(\mathbf{f}(a), \mathbf{f}(b)) \in \mathcal{A}''$

(3) if $a \neq b \in \mathcal{A}$ then $\mathbf{f}(a) \neq \mathbf{f}(b) \in \mathcal{A}''$

Theorem 11: If the proxy Abox $\mathcal{A}''$ obtained by applying the mapping function $\mathbf{f}$ to $\mathcal{A}$ is consistent then $\mathcal{A}$ is consistent. However, the converse of Theorem 11 does not hold.

Proof: Let us assume that $\mathcal{A}'$ is consistent w.r.t. $\mathcal{T}$ and $\mathcal{R}$. Therefore there is a model $\mathcal{I}' = (\Delta^{\mathcal{I}'}, .^{\mathcal{I}'})$ of $\mathcal{A}'$ w.r.t. $\mathcal{T}$ and $\mathcal{R}$. A model of $\mathcal{A}$ can easily be built from $\mathcal{I}'$ by interpreting an individual a in $\mathcal{A}$ in the same way as $\mathbf{f}(a)$ is interpreted by $\mathcal{I}'$. Formally, let $\mathcal{I} = (\Delta^{\mathcal{I}}, .^{\mathcal{I}})$ be the interpretation of the $\mathcal{A}$ w.r.t. $\mathcal{T}$ and $\mathcal{R}$ defined as follows: $\Delta^{\mathcal{I}} = \Delta^{\mathcal{I}'}$; for a concept $C \in \mathcal{T}$, $C^{\mathcal{I}} = C^{\mathcal{I}'}$; for a role R in $\mathcal{R}$, $R^{\mathcal{I}} = R^{\mathcal{I}'}$; for an individual a in $\mathcal{A}$, $a^{\mathcal{I}} = \mathbf{f}(a)^{\mathcal{I}'}$. $\mathcal{I}$ is a model of $\mathcal{A}$ w.r.t. $\mathcal{T}$ and $\mathcal{R}$ as a direct consequence of the fact that $\mathcal{I}'$ is a model of $\mathcal{A}'$ and $\mathcal{A}'$ satisfies the 3 conditions stated in definition 1 (see [3] for more details). ∎

The mapping function $\mathbf{f}$ that we use to create a proxy Abox from a global-effects Abox is defined such that if $\mathcal{L}(a) = \mathcal{L}(b)$ and $a \neq b \notin \mathcal{A}$, then $\mathbf{f}(a) = \mathbf{f}(b)$. That is, all individuals in the Abox $\mathcal{A}$ which have the same concept set and are not asserted to be distinct map to the same individual in the proxy Abox $\mathcal{A}''$. If a proxy Abox $\mathcal{A}''$ is not consistent, either there is a real inconsistency in $\mathcal{A}$ or the process of collapsing individuals to create $\mathcal{A}''$ caused an artificial inconsistency. To determine whether an inconsistency is real, we consider the global-effects Abox $\mathcal{A}'$. Like $\mathcal{A}'$, $\mathcal{A}''$ may consist of disconnected Aboxes and, since $\mathcal{A}''$ is typically small, it is not expensive to identify these partitions. Furthermore, we know from the consistency check which partitions in $\mathcal{A}''$ are inconsistent. For each inconsistent partition $\mathcal{A}_i{}''$ in $\mathcal{A}''$, we test for consistency the assertions in $\mathcal{A}'$ that map into it, which form a distinct partition $\mathcal{A}_i{}'$ in $\mathcal{A}'$. If any partition in $\mathcal{A}'$ is inconsistent, then $\mathcal{A}$ is inconsistent.

Table 1: Characteristics of $\mathcal{A}$, $\mathcal{A}$', and $\mathcal{A}$"

KB	Classes			Roles			Instances			Role Assertions		
	$\mathcal{A}$	$\mathcal{A}$'	$\mathcal{A}$"	$\mathcal{A}$	$\mathcal{A}$'	$\mathcal{A}$"	$\mathcal{A}$	$\mathcal{A}$'	$\mathcal{A}$"	$\mathcal{A}$	$\mathcal{A}$'	$\mathcal{A}$"
BioPax	31	14	14	40	2	2	261K	17K	39	582K	14K	106
LUBM	91	25	19	27	5	3	142K	44K	481	736K	45K	352
NIMD	19	2	2	28	1	1	1,278K	429K	2	2,000K	286K	1
ST	16	15	15	11	2	2	874K	547K	18	3,595K	580K	49

4 Computational Experience

We tested our approach on the four actual ontologies shown in Table 1. $\mathcal{A}$ corresponds to the original Abox, $\mathcal{A}'$ is the Abox obtained from $\mathcal{A}$ after removing local effects, and $\mathcal{A}''$ is the proxy Abox. Biopax includes the data for 11 organisms available at http://biocyc.org. We used a version of LUBM that was modified to SHIN [7] . The Abox of the NIMD ontology was generated from text analysis programs run over a large number of unstructured documents. The semantic traceability (ST) ontology Abox was generated from a program that extracted relationships between software artifacts of a large middleware application.

As can be seen in Table 1, in practice, $\mathcal{A}'$ is significantly smaller than $\mathcal{A}$, and $\mathcal{A}''$ is a substantial reduction over $\mathcal{A}'$. In most cases, $\mathcal{A}'$ was sufficient for the consistency check. For ST, a real inconsistency was detected in the local-effects consistency check. The Biopax and NIMD ontologies were consistent in $\mathcal{A}''$. For LUBM, we needed to check the consistency of $\mathcal{A}'$. The running time for our algorithm took from 12.6 seconds to 283 seconds, which included the time for building the proxy and checking it for consistency.

5 Related Work and Conclusion

There are many highly optimized reasoners such as Pellet [8], Racer [4], InstanceStore [1], and Kaon2 [10] designed for consistency checking, but only InstanceStore and Kaon2 can be extended to Aboxes in secondary storage. Kaon2 applies to deductive databases, whereas our techniques work with relational databases. InstanceStore is limited to role-free Aboxes. In theory, Instance Store can handle Aboxes with role assertions through a technique called precompletion [9], but this may not be practical for Aboxes stored in databases. Our techniques can be contrasted with optimization techniques such as model caching and Abox contraction [5], but again, it is unclear how such techniques can be applied to Aboxes in databases.

We have demonstrated a technique to scale consistency detection to large Aboxes in secondary storage by extracting a small representative Abox. Further,

we have shown that, in practice, this technique works efficiently on four large ontologies. Our plan is to extend this approach to apply more accurate static analysis techniques, extend its applicability to more expressive languages, and to apply these techniques to query processing.

References

[1] Sean Bechhofer, Ian Horrocks, and Daniele Turi. The owl instance store: System description. *Proc. of 20th Int.Conf. on Automated Deduction*, pages 177–181, 2005.

[2] F. Donini. Complexity of reasoning. In F. Baader, D. Calvanese, D.L. McGuinness, D. Nardi, and P.F. Patel-Schneider, editors, *Description Logic Handbook*, pages 101–141. Cambridge University Press, 2002.

[3] Achille Fokoue, Aaron Kershenbaum, Li Ma, Edith Schonberg, and Kavitha Srinivas. Scalable reasoning:cutting ontologies down to size. In *http://www.research.ibm.com/iaaa/aaaiSubmission.ps*, 2006.

[4] V. Haarslev and R. Moller. Racer system description. *Conf. on Automated Reasoning (IJCAR 2001)*, pages 701–705, 2001.

[5] Volker Haarslev and Ralf Moller. An empirical evaluation of optimization strategies for abox reasoning in expressive description logics. *Proc. of the International Workshop on Description Logics*, pages 115–199, 1999.

[6] Ian Horrocks, Ulrike Sattler, and Stephan Tobies. Reasoning with individuals for the description logic SHIQ*. *Proc. of 17th Int.Conf. on Automated Deduction*, pages 482–496, 2000.

[7] Li Ma, Yang Yang, Zhaomin Qiu, Guotong Xie, and Yue Pan. Towards a complete owl ontology benchmark. In *Proc. of the third European Semantic Web Conf.(ESWC 2006)*, 2006.

[8] Evren Sirin and Bijan Parsia. Pellet: An owl dl reasoner. In *Description Logics*, 2004.

[9] Sergio Tessaris and Ian Horrocks. Abox satisfiability reduced to terminological reasoning in expressive description logics. In *LPAR*, pages 435–449, 2002.

[10] U.Hustadt, B. Motik, and U. Sattler. Reducing shiq description logic to disjunctive datalog programs. *Proc. of 9th Intl. Conf. on Knowledge Representation and Reasoning (KR2004)*, pages 152–162.

Tableau Caching for Description Logics with Inverse and Transitive Roles

Yu Ding and Volker Haarslev

Concordia University, Montreal, Quebec, Canada
{ding_yu|haarslev}@cse.concordia.ca

Abstract. Modern description logic (DL) reasoners are known to be less efficient for DLs with inverse roles. The current loss of performance is largely due to the missing applicability of some well-known optimization techniques, especially the one for caching the satisfiability status of modal successors. In this paper, we present a rule synthesis technique from which an estimation of the potential back-propagation of constraints can be made. This estimation can be applied to both the concept classifier and the satisfiability tester. This paper presents a tableau caching technique for $\mathcal{SHI}$ as a first step to improving the performance of tableau-based DL reasoners for logics offering the use of inverse roles. The proposed techniques underwent a first empirical evaluation with a prototype DL reasoner for $\mathcal{SHI}$ using a set of synthetically generated knowledge bases. The initial results indicate an significant improvement in runtime performance once caching is effectively enabled.

1 Motivation

Description logics (DLs) are a family of logic based formalisms for terminological knowledge representation and reasoning [15]. Descending from KL-ONE [1], they now enjoy a wide spectrum of applications. Behind this success is a long line of research and implementation efforts in DL reasoners. Most modern DL systems are based on tableau algorithms. Such algorithms are first introduced in [2] by Schmidt-Schauss and Smolka. In spite of the high worst case complexity of the satisfiability problem [15] for most expressive DLs (typically ExpTime-complete), highly optimized implementations have been shown to work well in many realistic applications [3, 9, 14].

DLs express unary relations in terms of concepts and binary relations in terms of roles. To describe inverse binary relationships, inverse roles are a necessary language element. One of the first approaches to allow the declaration of inverse roles was in the early nineties when some DL systems started to support very expressive DLs with inverse roles [5]. Later, the theoretical result on elimination of inverse roles based on equi-satisfiability was established [7].

Tableau-based DL systems typically employ a wide range of optimization techniques, most of which were designed and established unfortunately without appropriate consideration of inverse roles (which would be reasonable if the elimination of inverse roles had become realistic in applications). If directly applied to DLs with inverse roles, some well-known optimizations become less efficient or even invalid (due to the two-way propagation of universal restrictions introduced by inverse roles). The caching technique is one such example that suddenly turns unsound when inverse roles are considered. The current technique of inverse role elimination could introduce an overwhelming number of GCIs (General Concept Inclusions) that are difficult for tableau algorithms. Other attempts were tried, e.g., the *dynamic blocking* technique [8] which can be considered as a successful attempt in adapting the *blocking* [4] technique to DLs with inverse roles.

The literature addressing the issues caused by inverse role is not plentiful, and a systematic treatment is still to be done. This paper will discuss the soundness problem of common caching techniques for DLs with inverse roles. We extend our previous study [16] to a more expressive description logic, i.e., $\mathcal{SHI}$, a logic with expressive role constructs (i.e., transitive roles, inverse roles, and role hierarchies). We assume the reader's familiarity with the DL $\mathcal{SHI}$ and tableau algorithms. We introduce the rule synthesis approach, discuss the use of heuristics for concept classification, and show how to apply the same technique to obtain sound tableau caching.

2 Synthesizing Rules

In this section, we present our rule synthesis approach in terms of a reachability analysis of the underlying unfolding rules. Our reachability analysis is based on a multi-graph $G = (V, E)$ constructed from the syntactic structure of the given unfolding rules of $\mathcal{T}$. The initial reachability relation is represented as ϵ-edges and its transitive closure can be computed according to a set of reachability extension rules that are presented in the first subsection. The potential back propagations is contained in this closure. After this, we show how the estimated potential back propagation can be used to obtain sound sub-tableau caching as well as potential subsumers, respectively.

2.1 Potential Back Propagation

Due to the interaction of universal restrictions with inverse roles, constraints can propagate up and down the tableau tree. We refer to this upward propagation of constraints as *back propagation*. Here we present a way to synthesize the unfolding rules [15] to determine *potential back propagation* (PBP). We view PBP a reflexive and transitive relation, and formulate it as *reachability* in a directed graph (containing multi-edges and cycles). In the discourse, we use the notation $G = (V, E)$ for a directed multi-graph, where V denotes a set of nodes and E denotes a set of edges (binary relations) for pairs of nodes in V.

We say C (also D) is *non-modally used* by $C \sqcap D$ and $C \sqcup D$; and say $\forall R.(A \sqcup B)$ (resp. $\exists R.(A \sqcup B)$) *modally uses* A (also B) through $\forall R$ (resp. $\exists R$). Note we are not using the transitive *use relation*, for details please see [13].

Definition 1. *Given a $\mathcal{SHI}$ expression in* `NNF` *(negation normal form), its reachability graph is a multi-graph $G = (V, E)$ formed according to a function f such that: (1) for every concept literal C, there is a node $f(C) \in V$; (2) for every $\mathcal{SHI}$ (sub-)expression s, there is a unique node $f(s) \in V$; (3) if expression s_1 non-modally uses s_2, then $\epsilon \in E(f(s_1), f(s_2))$; if s_1 modally uses s_2 through $\otimes R$ for some role name R, then $\otimes R \in E(f(s_1), f(s_2))$, where $\otimes$ stands for the connective $\exists$ or $\forall$.*

Recursively an expression could be decomposed into concept literals and correspondingly G is constructed.

Definition 2. *Given $\mathcal{T}$, a $\mathcal{SHI}$ TBox of unfolding rules each of which is of the form $C_L \sqsubseteq D$, where C_L is a concept literal and D is in* `NNF`*, a reachability graph $G = (V, E)$ for $\mathcal{T}$ is formed such that for each rule $C_L \sqsubseteq D$: (1) there is a node $f(C_L) \in V$; (2) there is a node $f(D) \in V$; (3) $\epsilon \in E(f(C_L), f(D))$.*

The initial reachability graph G is built upon the syntactic structure only. A *non-modal use* relationship is represented by an ϵ-edge, a *modal use* relationship is represented by an edge tagged with the modality connective concatenated with the corresponding role name. The unfolding operation is represented by connecting the concept literal with an ϵ-edge to the root of the graph representation of its right-hand side.

In Table 1 three rules are given for extending the reachability relation from the initial $G = (V, E)$. We use E^* to denote the transitive closure of E over ϵ-edges.

$\exists\forall$-rule: if	there exist nodes x, y, z such that $\exists R_1 \in E^*(x, y)$ and $\forall R_2 \in E^*(y, z)$ and $R_1 \sqsubseteq^* \texttt{Inv}(R_2)$
then	$E(x, z) := E(x, z) \cup \{\epsilon\}$
$\forall\forall$-rule: if	there exist nodes x, y, z such that $\forall R_1 \in E^*(x, y)$ and $\forall S \in E^*(y, z)$ (for short let $\texttt{Inv}(S)$ be R_2) and there exists $R_3 \in \mathbf{R}$ s.t. $R_3 \sqsubseteq^* R_i$ for $i \in \{1, 2\}$
then	$E(x, z) := E(x, z) \cup \{\epsilon\}$
$\forall\exists$-rule: if	there exist nodes x, y, z such that $\forall R_1 \in E^*(x, y)$ and $\exists R_2 \in E^*(y, z)$ and there exists $\texttt{Trans}(S)$ s.t. $R_2 \sqsubseteq^* S \sqsubseteq^* R_1$
then	$E(y, y) := E(y, y) \cup \{\forall S\}$

Table 1. Reachability Extension Rules over $\mathcal{R}$ for $\mathcal{SHI}$.

Initially, the reachability graph $G = (V, E)$ is constructed according to all unfolding rules in $\mathcal{T}$. Then, an extension of G is based on a simulation of the $\forall$-rule in tableau algorithms w.r.t. $\mathcal{R}$. For a given TBox $\mathcal{T}$ and a role hierarchy $\mathcal{R}$ in DL $\mathcal{SHI}$, the PBP is contained in the final reachability relation in E^*.

To see how PBP is obtained, we compare the tableau expansion with the reachability extension. In tableau algorithms, both decomposition and selection are done by the $\sqcap$-rule and the $\sqcup$-rule; on the other hand, in the reachability graph, only decomposition is simulated. Also, in tableau algorithms, the firing of $\exists$-rule is subject to conditions, whereas in reachability graph, we make a simplification and assume it can be always fired. Furthermore, in tableau algorithms, the firing of $\forall$-rule depends on the existence of an edge (e.g., might be provided by a sibling expression), but in a reachability graph, we simplify that condition to only consider its predecessors and successors (as if that sibling always exists). In summary, by making these simplification and discarding inessential factors, the reachability graph is constructed in a very conservative way to grasp every potential back propagation. The essence for the reachability graph is that if PBP (ϵ-edge in the final $G = (V, E^*)$) is impossible, then it is guaranteed that no back propagation will take place in the corresponding application of the tableau algorithm.

For a given TBox $\mathcal{T}$ (of a set of unfolding rules) and a role hierarchy $\mathcal{R}$, there is a corresponding reachability graph $G = (V, E^*)$. For a concept literal C, we define a function $\texttt{Reach}(C) = \{y \in V \mid \epsilon \in E^*(f(C), y)\}$. In the following, we use the function $\texttt{Reach}$ without mentioning its $\mathcal{T}$ and $\mathcal{R}$.

2.2 Heuristic for Sound Sub-tableau Caching

Based on $\texttt{Reach}(C)$, we define the function $\texttt{Watch}$ applicable to a concept literal C: $\texttt{Watch}(C) = \bigcup_{x \in \texttt{Reach}(C)} \{\texttt{Inv}(R) \mid \exists y \in V \ s.t. \ \forall R \in E(x, y)\}$, where R is a (inverse) role. For a set of concept literals C_s, we define $\texttt{CWatch}(C_s) = \bigcup_{C \in C_s} \texttt{Watch}(C)$. Let R_s be a set of (inverse) roles, we use $R_s{\downarrow}$ to denote the union of subroles for each $r \in R_s$ w.r.t. role hierarchy $\mathcal{R}$. At this point, we are able to define a boolean function $\texttt{Safe}(C_s, R)$ that can be used to guarantee caching soundness by excluding potential unsound situations: $\texttt{Safe}(C_s, R) = \texttt{true}$ iff $R \notin \texttt{CWatch}(C_s){\downarrow}$. The following lemma follows naturally from the previous analysis.

Lemma 1 (Sound Caching). *Given the TBox $\mathcal{T}$ and the role hierarchy $\mathcal{R}$, let C_s and C_w be two sets of concept literals, R be a (inverse) role, The set C_s can be cached at the position of R-successor if: (1) $C_s \subseteq C_w$; and (2) $\texttt{Safe}(C_s, R)$; and (3) C_w has a model.*

Proof (sketch). (1) Assume C_w has the model $\mathcal{I}_w$. The construction process for $\mathcal{I}_w$ can be used to guide that of C_s, and C_s has a model $\mathcal{I}_s$ inside which each node's label is a subset of that of $\mathcal{I}_w$; (2) The condition $\texttt{Safe}(C_s, R)$ guarantees the root of $\mathcal{I}_s$ has no $\texttt{Inv}(R)$ predecessor, even if $\mathcal{I}_w$ needs a $\texttt{Inv}(R)$ predecessor.

2.3 Heuristic for Potential Subsumer

Classification is the process of computing the most-specific subsumption relationships (i.e., parents and children) of each concept name to other concept names mentioned in a TBox. The parents and children of a certain concept name are computed in the so-called top-down and bottom-up traversal phases, respectively. For large knowledge base, it is particularly important to avoid as many traversal as possible. A typical optimization is to use a heuristics-guided traversal which exploits information about *told subsumers* [3] or *potential subsumer* [12] to restrict the search space.

However, even the better *potential subsumer* technique considers only the transitive *non-modal use* relation [12] among concept names. For DLs with inverse roles, *potential subsumers* could appear in *modal-use* relations. For example, given a simple TBox $\mathcal{T} = \{A \sqsubseteq \exists R.B, B \sqsubseteq \forall R^{-}.C\}$ (note that $\mathcal{T}$ has only primitive definitions and no cyclic rules). The subsumption of $A \sqsubseteq C$ holds, and the subsumer C is neither explicitly told nor non-modal discernible.

For DLs with inverse roles, the *potential subsumer* technique remains useful if we take those concept names appearing in $\texttt{Reach}(A)$ into consideration. Similarly, this estimation is also applicable to related ABox reasoning tasks [15].

3 Discussion and Related Work

As a first step to evaluate the proposed techniques, i.e., caching and back-propagation estimation for DLs with inverse roles, a experimental tableau-based reasoner has been developed in Java. It employs several optimization techniques such as lexical normalization, semantic branching, backjumping (in a weak form), and a optimized version of dynamic blocking (tailored to DLs free of number restrictions). Besides the basic platform, two components were implemented, one for back-propagation estimation, and one for caching (the latter relies on the former). The proposed techniques were then evaluated with a set of synthetically generated satisfiability tests. These preliminary benchmarks indicated a performance gain of a factor of 2 to 10. A more realistic benchmark using RacerPro is in preparation.

The following papers in the literature are relevant in our context to caching and blocking. DeGiacomo et. al. [6] demonstrated for the DL $\mathcal{ALC}$ that the tableau caching technique (especially unsatisfiability caching) is necessary for obtaining a worst case optimal tableau algorithm. For super-set and sub-set caching see [12]. For various blocking techniques for DLs with inverse roles see [11, 8]. Papers which systematically addressed the classification problem include [3, 12]. However, they commonly focused on DLs without inverse roles. Recent papers focusing on the optimization of classification did not address the anomaly when inverse roles are present in the TBox.

Satisfiability tests for DLs with inverse roles are empirically hard for tableau-based reasoning systems. For instance, it is required that the blocking technique (i.e., the cycle detection mechanism) dynamically guarantees soundness. Furthermore, witness nodes are restricted to be ancestor nodes only. This dynamic behavior together with a limited choice of witness nodes generally makes the tableau-based procedures less efficient than necessary. Our preliminary experiments show that the proposed approach can be feasibly implemented as part of a one-pass parsing phase for tableau-based DL reasoners. The availability of the caching technique dramatically speeds up the reasoning process. First, the witness space is no longer restricted to ancestor nodes. Second, the caching technique requires

no re-checking and thus it is safe to discard nodes once they are successfully cached. By doing this, the tableau algorithm is more space-economic. Our approach can successfully deal with a set of (possibly cyclic) unfolding rules and a role hierarchy, thus it well meets the typical requirement from real applications.

4 Conclusion

The common caching techniques are no longer applicable in the presence of inverse roles, since information can be pushed backwards, which might invalidate the cache. Closely related to known techniques of knowledge compilation, we presented in this paper a technique for synthesizing the unfolding rules (of the terminological box) to estimate potential back propagations. This technique is currently applied to terminological knowledge bases with cyclic axioms and expressive roles in the expressive DL $\mathcal{SHI}$. The presented solution computes a reachability graph before testing the satisfiability of a concept in order to determine, when backpropagation is impossible and hence caching is safe. Based on this, a sound tableau caching technique was obtained in a relatively straightforward way. As was expected, better run-time performance was observed in our preliminary experiments due to the use of cache. We also pointed out that the estimation could be used for other purposes such as in locating potential subsumers during concept classification. We are aware that further refinement is possible for the technique presented here.

References

1. Brachman, R., McGuinness, D., Patel-Schneider, P., Resnick, L., Borgida, A.: Living with CLASSIC: When and how to use a KL-ONE-like language. Principles of Semantic Networks, edited by John Sowa, Morgan Kaufmann, (1991) 401-456
2. Schmidt-Schauss M., Smolka G.: Attributive concept descriptions with complements. Artificial Intelligence, Vol 48, (1991) 1-26
3. Baader, F., Hollunder, B., Nebel, B., Profitlich, H.-J., Franconi, E.: An Empirical Analysis of Optimization Techniques for Terminological Representation Systems - or - Making KRIS get a move on: KR-92, (1992) 270-281

4. Buchheit M., Donini F., Nutt W., Schaerf A.: Decidable Reasoning in Terminological Knowledge Representation Systems. Journal of Artificial Intelligence Research, Vol 1 (1993) 109-138.
5. Paolo Bresciani, Enrico Franconi, and Sergio Tessaris: Implementing and testing expressive description logics: a preliminary report. DL-95 (1995).
6. De Giacoma, G., Donini, F., Massacci, F.: EXPTIME Tableaux for ALC: DL-96, (1996)
7. Diego Calvanese, Giuseppe De Giacomo, Riccardo Rosati: A Note on Encoding Inverse Roles and Functional Restrictions in ALC Knowledge Bases. DL-98 (1998).
8. I. Horrocks and U. Sattler. A Description Logic with Transitive and Inverse Roles and Role Hierarchies. Journal of Logic and Computation, 9(3), (1999) 385-410
9. Ian Horrocks: Using an expressive description logic: FaCT or fiction?: KR-98 (1998)
10. Volker Haarslev, Ralf Möller: Consistency Testing: The RACE Experience: Proc. of TABLEAUX'2000, (2000) 57-61
11. F. Baader, U. Sattler: An Overview of Tableau Algorithms for Description Logics. Studia Logica, 69, (2001) 5-40
12. Volker Haarslev, Ralf Möller: High Performance Reasoning with Very Large Knowledge Bases: A Practical Case Study. Proc. of IJCAI'2001 (2001) 161-166
13. Volker Haarslev: Theory and Practice of Visual Languages and Description Logics: Habilitation Thesis, Computer Science Department, University of Hamburg, September (2001)
14. Volker Haarslev, Ralf Möller: RACER System Description: Proc. of IJCAR'2001 (2001)
15. Franz Baader, Diego Calvanese, Deborah McGuinness, Daniele Nardi, and Peter F. Patel-Schneider: The Description Logic Handbook: Theory, Implementation, and Applications. Cambridge University Press (2003)
16. Yu Ding, Volker Haarslev: Towards Efficient Reasoning for Description Logics with Inverse Roles. DL-05 (2005) 208-215

On the Scalability of Description Logic Instance Retrieval

Ralf Möller, Volker Haarslev, Michael Wessel

1 Introduction

Although description logics (DLs) are becoming more and more expressive, our experience has been that it is only for some tasks that the expressivity of description logics really comes into play; for many applications, it is necessary to be able to deal with largely deterministic knowledge very effectively (scalability problem). In the literature, the scalability problem has been tackled from different perspectives. We see two main approaches, the layered approach and the integrated approach. In the former approach the goal is to use databases for storing and accessing data, and exploit description logic ontologies for convenient query formulation. The main idea is to support ontology-based query expansion and optimization. See, e.g., [9, 4] (DLDB), [1] (Instance Store), or [2] (DL-Lite). We acknowlede the interesting progress of these approaches, but we note that the corresponding techniques only applicable if reduced expressivity does not matter. Despite the most appealing argument of reusing database technology (in particular services for persistent data), at the current state of the art it is not clear how expressivity can be increased to, e.g., $\mathcal{SHIQ}$ without losing the advantage of fast runtimes. Therefore, in this paper, we pursue the integrated approach that considers query answering with a description logic system. For the time being we ignore the problems associated with persistency and investigate specific knowledge bases (see below).

Continuing our work in [5], in this paper we investigate the constraints imposed on the architecture of description logic reasoning systems, and we investigate the assumptions underlying the heuristics used in instance retrieval algorithms. The contribution presents and analyzes the main results we have found about how to solve the scalability problem with tableau-based prover systems given large sets of data descriptions for a large number of individuals. Note that we do not discuss query answering speed of a particular system but investigate the effect of optimization techniques that could be exploited by any (tableau-based) DL inference system that already exists or might be built. We assume the reader is familiar with DLs in general and tableau-based decision procedures in particular.

2 Lehigh University Benchmark

In order to investigate the scalability problem, we use the Lehigh University BenchMark (LUBM, [3, 4]). LUBM data descriptions are automatically generated, but we do not exploit this in this paper (e.g., by tailoring optimization techniques to regular patterns stemming from generator peculiarities). The LUBM queries are formalized as conjunctive queries referencing concept, role, and individual names from the Tbox. A query language tailored to description logic applications that can express these queries is described in [8] (the language is called nRQL). Variables are only bound to individuals mentioned in the Abox. In the notation of nRQL queries used in this paper we assume that different variables may have the same bindings. In addition, variables that do not occur in the head of a query are assumed to be existentially quantified and are are also only bound to individual names mentioned in the

Abox.

Below, LUBM queries 9 and 12 are shown in order to present a flavor of the kind of query answering problems – note that $'www.University0.edu'$ is an individual and $subOrganizationOf$ is a transitive role. Please refer to [3, 4] for more information about the LUBM queries.

$Q9 : ans(x, y, z) \leftarrow Student(x), Faculty(y), Course(z),$
$advisor(x, y), takesCourse(x, z), teacherOf(y, z)$

$Q12 : ans(x, y) \leftarrow Chair(x), Department(y), memberOf(x, y),$
$subOrganizationOf(y,' www.University0.edu')$

In order to investigate the scalability problem, we used a TBox for LUBM with inverse and transitive roles as well as domain and range restrictions but no number restrictions, value restrictions or disjunctions. Among other axioms, the LUBM TBox contains axioms that express necessary and sufficient conditions on $Chair$. For instance, there is an axiom $Chair \doteq Person \sqcap \exists headOf.Department$. For evaluating optimization techinques for query answering we consider runtimes for a whole query set (queries 1 to 14 in the LUBM case).

3 Optimization Techniques

If the queries mentioned in the previous section are answered in a naive way by evaluating subqueries in the sequence of syntactic notation, acceptable answering times cannot be achieved. Determining all bindings for a variable (with a so-called generator) is much more costly than verifying a particular binding (with a tester). Treating the one-place predicates $Student$, $Faculty$, and $Course$ as generators for bindings for corresponding variables results in combinatorial explosion (cross product computation). Optimization techniques are required that provide for efficient query answering in the average case. For the discussion of optimization techniques, we assume that for transitive and inverse roles, the Abox is extended in the obvious way such that implicit role assertions are made explicit. In addition, the conclusions from domain and range restrictions are made explicit in the Abox as well.

3.1 Query Optimization

The optimization techniques that we investigated are inspired by database join optimizations, and exploit the fact that there are few $Faculties$ but many $Students$ in the data descriptions. For instance, in case of query Q9 from LUBM, the idea is to use $Faculty$ as a generator for bindings for y and then generate the bindings for z following the role $teacherOf$. The heuristic applied here is that the average cardinality of a set of role fillers is rather small. For the given z bindings we apply the predicate $Course$ as a tester (rather than as a generator as in the naive approach). Given the remaining bindings for z, bindings for x can be established via the inverse of $takesCourse$. These x bindings are then filtered with the tester $Student$.

If z was not mentioned in the set of variables for which bindings are to be computed, and the tester $Course$ was not used, there would be no need to generate bindings for z at all. One could just check for the existence of a $takesCourse$ role filler for bindings w.r.t. x.

In the second example, query Q12, the constant $'www.University0.edu'$ is mentioned. Starting from this individual the inverse of $subOrganizationOf$ is applied as a generator for bindings for y which are filtered with the tester $Department$. With the inverse of $memberOf$, bindings for x are computed which are then filtered with $Chair$. Since for the concept $Chair$ sufficient conditions are declared in the TBox, instance retrieval reasoning is required if $Chair$ is a generator. Thus, it is advantageous that $Chair$ is applied as a tester (and only instance tests are performed).

For efficiently answering queries, an ordered query execution plan must be determined. For computing a total order relation on query atoms with respect to a given set of data descriptions (assertions in an ABox), we need information about the number of instances of concept and role names. An estimate for this information can be computed in a preprocessing

step by considering given data descriptions, or could be obtained by examining the result set of previously answered queries (we assume that ABox realization is too costly, so this alternative is ruled out).

3.2 Indexing by Exploiting Told and Taxonomical Information

In many practical applications that we encountered, data descriptions often directly indicate of which concept an individual is an instance. Therefore, in a preprocessing step it is useful to compute an index that maps concept names to sets of individuals which are their instances. In a practical implementation this index might be realized with some form of hashtable.

Classifying the TBox yields the set of ancestors for each concept name, and if an individual i is an instance of a concept name A due to explicit data descriptions, it is also an instance of the ancestors of A. The index is organized in such a way that retrieving the instances of a concept A or one of its ancestors requires (almost) constant time. The index is particularly useful to provide bindings for variables if, despite all optimization attemps for deriving query execution plans, concept names must be used as generators. In addition, the index is used to estimate the cardinality of concept extensions. The estimates are used to compute an order relation for query atoms. The smaller the cardinality of a concept or a set of role fillers is assumed to be, the more priority is given to the query atom. Optimizing query $Q9$ using told information yields the following query execution plan.

$$Q9' : ans(x,y,z) \leftarrow Faculty(y), teacherOf(y,z), Course(z), \\ advisor^{-1}(y,x), Student(x), takesCourse(x,z)$$

Using this kind of rewriting, queries can be answered much more efficiently.

If the TBox contains only role inclusion axioms and GCIs of the form $A \sqsubseteq A_1 \sqcap \ldots \sqcap A_n$, i.e., if the TBox forms a hierachy, the index-based retrieval discussed in this section is complete (see [1]). However, this is not the case for LUBM. In LUBM, besides domain and range restrictions, axioms are also of the form $A \doteq A_1 \sqcap A_2 \sqcap \ldots \sqcap A_k \sqcap \exists R_1.B_1 \sqcap \ldots \sqcap \exists R_m.B_m$ (actually, $m = 1$). If sufficient conditions with exists restrictions are specified as in the case of *Chair*, optimization is much more complex. In LUBM data descriptions, no individual is explicitly declared as a *Chair* and, therefore, reasoning is required, which is known to be rather costly. If *Chair* is used as a generator and not as a tester such as in the simple query $ans(x) \leftarrow Chair(x)$, optimization is even more important.

3.3 Obvious Non-Instances: Exploiting Information from One Completion

The detection of "obvious" non-instances of a given concept C can be optimized by using a model merging operator defined for so-called individual pseudo models (aka pmodels) as specified in [5]. The central idea is to compute a pmodel from a completion that is derived by the tableau prover. Note that it is important that all restrictions for a certain individual are "reflected" in the pmodel. The idea of model merging is that there is a simple sound but incomplete test for showing that adding the assertion $i : \neg C$ to the ABox will not lead to a clash (see [5] for details) and, hence, i is not an instance of C. Note that, usually, the complete set of data structures for a particular completion is not maintained by a reasoner. The pmodels provide for an excerpt of a completion needed to determine non-instances.

3.4 Obvious Instances: Exploiting Information from the Precompletion

A central optimization technique to ensure scalability as it is required for LUBM is to also find "obvious" instances with minimum effort. Given an initial ABox consistency test one can consider all deterministic restrictions, i.e., consider only those tableau structures (from now on called constraints) for which there are no *previous* choice points in the tableau proof (in other words, consider only those constraints that do not have dependency information

attached). These constraints, which are computed during the tableau proof, describe a so-called precompletion.[1] Note that in a precompletion, no restrictions are violated.

Given the precompletion constraints, for each individual an approximation of the most-specific concept (MSC') of an individual i is computed as follows. For all constraints representing role assertions of the form $(i, j) : R$ (or $(j, i) : R$) add constraints of the form $i : \exists R.\top$ (or $i : \exists R^{-1}.\top$). Afterwards, constraints for a certain individual i are collected into a set $\{i : C_1, \ldots, i : C_n\}$. Then, $MSC'(i) := C_1 \sqcap \ldots \sqcap C_n$. Now, if $MSC'(i)$ is subsumed by the query concept C, then i must be an instance of C. In the case of LUBM many of the assertions lead to deterministic constraints in the tableau proof which, in turn, results in the fact that for many instances of a query concept C (e.g., *Faculty* as in query $Q9$) the instance problem is decided with a subsumption test based on the MSC' of each individual. Subsumption tests are known to be fast due to caching and model merging. The more precisely MSC'(i) approximates MSC(i), the more often an individual can be determined to be an obvious instance of a query concept. Obviously, it might be possible to determine obvious instances by directly considering the precompletion data structures. However, these are technical details. The main point is that, due to our findings, the crude approximation described above suffices to detect all "obvious" instances in LUBM.

If query atoms are used as testers, in LUBM it is the case that in a large number of cases the test for obvious non-instances or the test for obvious instances determines the result. However, for some individuals i and query concepts C both tests do not determine whether i is an instance of C (e.g., this is the case for *Chair*). Since both tests are incomplete, for some individuals i a refutational ABox consistency test resulting from adding the claim $i : \neg C$ must be decided with a sound and complete tableau prover. For some concepts C, the set of remaining instances i for which the "expensive" ABox instance test must be used is quite large, or, considering C as a generator, the set of candidates is quite large. In any case, considering the volume of assertions in LUBM (see below for details), it is easy to see that the refutational ABox consistency test must not start from the initial, unprocessed ABox in order to ensure scalability.

For large ABoxes and many repetitive instance tests it is mandatory not to "expand" the very same initial constraints over and over again. Therefore, the precompletion resulting from the initial ABox consistency test is used as a starting point for refutational instance tests. The tableau prover keeps the precompletion in memory. All deterministic constraints are expanded, so if some constraint is added, only a limited amount of work is to be done. Tableau provers are fast w.r.t. backtracking, blocking, caching and the like. But not fast enough if applied in a naive way. If a constraint $i : \neg C$ is added to a precompletion, the tableau prover must be able to very effectively determine related constraints for i that already have been processed. Rather than using linear search through lists of constraints, index structures are required.

3.5 Index Structures for Optimizing Tableau Provers

First of all, it is relatively easy to classify various types of constraints (for exists restrictions, value restrictions, atomic restrictions, negated atomic restrictions, etc.) and access them effectively according to their type. We call the corresponding data structure an active record of constraint sets (one set for each kind of constraint). For implementing a tableau prover, the question for an appropriate set data structure arises. Since ABoxes are not models, (dependency-directed) backtracking cannot be avoided in general. In this case, indexing the set of "relevant" constraints in order to provide algorithms for checking if an item is an element of a set or list (element problem) is all but easy. Indexing requires hashtables (or trees), but

[1] Cardinality measures for concept names, required for determining optimized query execution plans, could be made more precise if cardinality information is computed by considering a precompletion. However, in the case of LUBM this did not result in better query execution plans.

backtracking requires either frequent copying of index structures (i.e., hashtables) or frequent inserting and deleting items from hashtables. Both operations are known to be costly.

Practical experiments with LUBM and many other knowledge bases indicate that the following approach is advantageous in the average case. For frequent updates of the search space structures during a tableau proof, we found that simple lists for different kinds of constraints are most efficient, thus we have an active record of lists of constraints. New constraints are added to the head of the corresponding list, a very fast operation. During backtracking, the head is chopped off with minimum effort. The list representation is used if there are few constraints, and the element problem can be decided efficiently. However, if these lists of constraints get large, performance decreases due to linear search. Therefore, if some list from the active record of constraints gets longer than a certain threshold, the record is restructured and the list elements are entered into an appropriate index structure (hashtables with individuals as keys). Afterwards the tableau prover continues with a new record of empty lists as the active record. The pair of previous record of lists and associated hashtable is called a generation. From now on, new constraints are added to the new active record of constraints and the list(s) of the first generation are no longer used. For the element problem the lists from the active record are examined first (linear search over small lists) and then, in addition, the hashtable from the first generation is searched (almost linear search). If a list from the active record gets too large again, a new generation is created. Thus, in general we have a sequence of such generations, which are then considered for the element test in the obvious way. If backtracking occurs, the lists of the appropriate generation are installed again as the active record of lists. This way of dealing with the current search state allows for a functional implementation style of the tableau prover which we prefer for debugging purposes. However, one might also use a destructive way to manage constraints during backtracking. Obviously, all (deterministic) constraints from the initial Abox can be stored in a hashtable. In any case, the main point here is that tableau provers need an individual-based index to efficiently find all constraints an individual is involved in. In the evaluation of other optimization techniques (see below) we presuppose that a tableau prover is equipped with this technology.

3.6 Transforming Sufficient Conditions into Conjunctive Queries

Up to now we have discussed the optimization of concept atoms used as testers in the query execution plan. If concepts are used as generators, indeed a retrieval problem rather than only an instance test problem has to be solved. Using the results in [5] it is possible to linearly iterate over all individuals, check for obvious non-instances and obvious instances, and then, for the set of remaining candidates dependency-directed instance retrieval and binary partitioning can be used (see [5]). Our findings suggest that in the case of LUBM, for example for the concept $Chair$, the remaining tableau proofs are very fast. Nevertheless it is the case that the whole procedure is based on a linear iteration over all individuals. In application scenarios such as those we investigate with LUBM we have 200,000 individuals and more, and even if each single test lasts only a few dozen microseconds, query answering will be too slow, and hence additional techniques must be applied to solve the scalability problem.

The central insight for another optimization technique is that conjunctive queries can be optimized according to the above-mentioned arguments whereas for concept-based retrieval queries, optimization is much harder to achieve. Let us consider the query $ans(x) \leftarrow Chair(x)$. For $Chair$, sufficient conditions are given as part of the TBox (see above). Thus, in principle, we are looking for instances of the concept $Person \sqcap \exists headOf.Department$. The key to optimizing query answering becomes apparent if we transform the definition of $Chair$ into a conjunctive query and derive the optimized version $Q15'$

$Q15 : ans(x) \leftarrow Person(x), headOf(x, y), Department(y)$

$Q15' : ans(x) \leftarrow Department(y), headOf^{-1}(y, x), Person(x)$

Univs	Inds	Concept Assertions	Role Assertions
1	17174	53738	49336
3	55664	181324	166682
5	102368	336256	309393
10	207426	685569	630753

Table 1: Linearly increasing number of individuals, concept assertions and role assertions for different numbers of universities.

Because there exist fewer *Departments* than *Persons* in LUBM, search for bindings for x is substantially more focused in $Q15'$ (which is the result of automatic query optimization, see above). In addition, in LUBM, the extension of *Department* can be determined with simple index-based tests only (only hierarchies are involved). With the *Chair* example one can easily see that the standard approach for instance retrieval can be optimized dramatically if query atoms are rewritten as indicated in the above example.

If there is no specific definition or there are meta constraints, rewriting is not applied. It is easy to see that the rewriting approach is sound. However, it is complete only under specific conditions, which can be automatically detected. If we consider the Tbox $T = \{D \doteq \exists R.C\}$, the Abox $A = \{i : \exists R.C\}$ and the query $ans(x) \leftarrow D(x)$, then, due to the algorithm presented above, the query will be rewritten as $ans(x) \leftarrow R(x, y), C(y)$. For variable bindings, the query language nRQL (see above) considers only those individuals that are explicitly mentioned in the Abox. Hence, i will not be part of the result set because there is no binding for y in the Abox A. Examining the LUBM Tbox and Abox it becomes clear that in this case for every $i : \exists R.C$ appearing in a tableau proof there already exist constraints $(i, j) : R$ and $j : C$ in the original Abox. However, even if this is not the case, the technique can be employed under some circumstances.

Usually, in order to construct a model (or a completion to be more precise), tableau provers create a new individual for each constraint of the form $i : \exists R.C$ and add corresponding concept and role assertions. These newly created individuals are called anonymous individuals. Let us assume, during the initial Abox consistency test a completion is found. As we have discussed above, a precompletion is computed by removing all constraints that depend on a choice point. If there is no such constraint, the precompletion is identical to the completion that the tableau prover computed. Then, the set of bindings for variables is extended to the anonymous individuals found in the precompletion. The rewriting technique for concept query atoms is applicable (i.e., is complete) under these conditions. Even if the rewriting technique is not complete (i.e., s.th. is removed from the completion to derive the precompletion), it can be employed to reduce the set of candidates for binary partitioning techniques (c.f., [5]) that can speed of this process considerably in the average case.

The transformation approach discussed in this section is reminiscent to an early transformation approach discussed in [7]. In fact, ideas from translational approaches from DLs to disjunctive datalog [6] can also be integrated in tableau-based approaches. In the following section, we will evaluate how the optimization techniques introduced up to now provide a contribution to the data description scalability problem.

4 Evaluation and Conclusion

The significance of the optimization techniques introduced in this contribution is analyzed with the system RacerPro 1.9. RacerPro is freely available for research and educational purposes (`http://www.racer-systems.com`). The runtimes we present in this section are used to demonstrate the order of magnitude of time resources that are required for solving inference problems. They allow us to analyze the impact of proposed optimization techniques.

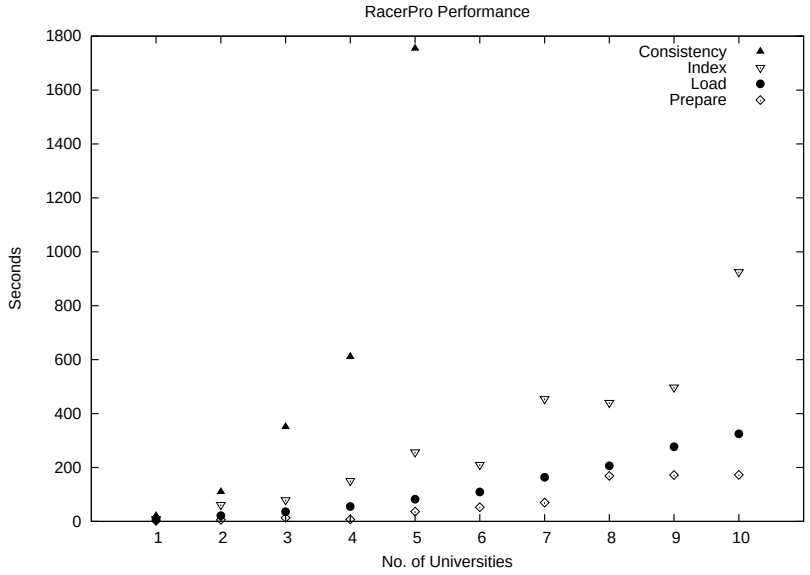


Figure 1: Runtimes for loading, preparation, abox consistency checking and indexing.

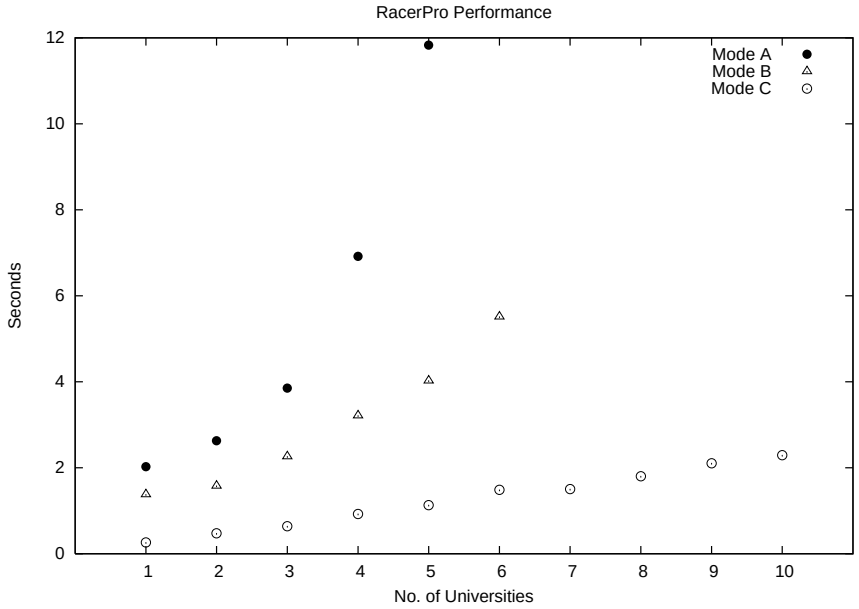


Figure 2: Runtimes of 14 LUBM queries with different optimization settings (see text).

An overview about the size of the LUBM benchmarks is given in Table 1. The runtimes for loading the data descriptions, transforming them into abstract syntax trees (preparation), and indexing are shown in Figure 1 (AMD 64bit processor, 4GB, Linux OS). It is important to note that the curves are roughly linear, thus, no reasoning is included in these phases. In

Figure 1, the runtimes for checking ABox consistency and precompletion data structures (see above) are indicated (the shape, which appears to be quadratic, reveals that this phase are subject to further optimizations).

In Figure 2, average query-answering times for running all 14 LUBM queries on data descriptions for an increasing number of universities are presented (see Table 1 for an overview on the number of individuals, concept assertions, and role assertions). In mode A and B, concept definitions are not rewritten into conjunctive queries (see Section 3.6). In mode A, full constraint reasoning on datatypes is provided. However, this is not required for LUBM. Therefore, in mode B, told value retrieval is performed only. As Figure 2 shows, this is much more efficient. Mode C in Figure 2 presents the runtimes achieved when definitions of concept names are rewritten to conjunctive queries (and told value reasoning on datatypes is enabled). Mode C indicates that scalability for instance retrieval can be achieved with tableau-based retrieval engines. Note that we argue that the concept rewriting technique is advantageous not only for RacerPro but also for other tableau-based systems. Future work will investigate optimizations of large Aboxes and more expressive Tboxes. Our work is based on the thesis that for investigating optimization techniques for Abox retrieval w.r.t. more expressive Tboxes, we first have to ensure scalability for Aboxes and Tboxes such as those that we discussed in this paper. We have shown that the results are encouraging.

References

[1] S. Bechhofer, I. Horrocks, and D. Turi. The OWL instance store: System description. In *Proceedings CADE-20*, LNCS. Springer Verlag, 2005.

[2] D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, and R. Rosati. Data complexity of query answering in description logics. In *Proc. of the 2005 Description Logic Workshop (DL 2005)*. CEUR Electronic Workshop Proceedings, http://ceur-ws.org/, 2005.

[3] Y. Guo, J. Heflin, and Z. Pan. Benchmarking DAML+OIL repositories. In *Proc. of the Second Int. Semantic Web Conf. (ISWC 2003)*, number 2870 in LNCS, pages 613–627. Springer Verlag, 2003.

[4] Y. Guo, Z. Pan, and J. Heflin. An evaluation of knowledge base systems for large OWL datasets. In *Proc. of the Third Int. Semantic Web Conf. (ISWC 2004)*, LNCS. Springer Verlag, 2004.

[5] V. Haarslev and R. Möller. Optimization techniques for retrieving resources described in OWL/RDF documents: First results. In *Ninth International Conference on the Principles of Knowledge Representation and Reasoning, KR 2004, Whistler, BC, Canada, June 2-5*, pages 163–173, 2004.

[6] B. Motik. *Reasoning in Description Logics using Resolution and Deductive Databases*. PhD thesis, Univ. Karlsruhe, 2006.

[7] B. Motik, R. Volz, and A. Maedche. Optimizing query answering in description logics using disjunctive deductive databases. In *Proceedings of the 10th International Workshop on Knowledge Representation Meets Databases (KRDB-2003)*, pages 39–50, 2003.

[8] M. Wessel and R. Möller. A high performance semantic web query answering engine. In *Proc. of the 2005 Description Logic Workshop (DL 2005)*. CEUR Electronic Workshop Proceedings, http://ceur-ws.org/, 2005.

[9] Z. Zhang. Ontology query languages for the semantic web: A performance evaluation. Master's thesis, University of Georgia, 2005.

High Performance Absorption Algorithms for Terminological Reasoning

Ming Zuo and Volker Haarslev
Concordia University, Montreal, Quebec, Canada
{ming_zuo|haarslev}@cse.concordia.ca

Abstract

When reasoning with description logic (DL) knowledge bases (KBs), performance is of critical concern in real applications, especially when these KBs contain a large number of axioms. To improve the performance, axiom absorption has been proven to be one of the most effective optimization techniques. The well-known algorithms for axiom absorption, however, still heavily depend on the order and the format of the axioms occurring in KBs. In addition, in many cases, there exist some restrictions in these algorithms which prevent axioms from being absorbed. The design of absorption algorithms for optimal reasoning is still an open problem. In this paper, we propose some new algorithms to absorb axioms in a KB to improve the reasoning performance. The experimental tests we conducted are mostly based on synthetic benchmarks derived from common cases found in real KBs. The experimental evaluation demonstrates a significant runtime improvement.

1 Motivation

When reasoning with description logic (DL) knowledge bases (KBs) which contain a large number of axioms, performance is the key concern in real application. To improve the reasoning performance, many optimization algorithms and techniques are employed in most of the modern reasoners such as RACER and FaCT++. Among the optimization algorithms, lazy unfolding is proven to be one of the most effective techniques [7]. Unfortunately, lazy unfolding does not work well for KBs containing a significant number of nonabsorbable GCIs (General Concept Inclusions). A GCI is called *nonabsorbable* if it cannot be rewritten into a rule axiom. We use the term *rule axiom* to represent axioms of the form of $A \Rightarrow C$ where $A \in NC$ and C is an arbitrary concept, while the form $C \sqsubseteq D$ represents a GCI where C and D are arbitrary concepts. The difference between $A \Rightarrow C$ and $A \sqsubseteq C$ is that $A \Rightarrow C$ represents only "if A then

C", while $A \sqsubseteq C$ represents both "if A then C" and "if $\neg C$ then $\neg A$" during reasoning. To convert a GCI into rule axiom(s), a technique called *absorption* is employed. Although preliminary absorption algorithms have been discussed in [6], there is little concern about the "best" absorption for optimal reasoning. The known algorithms also still heavily depend on the order and the format of axioms found in the KB of interest. In addition, in many cases, some restrictions in these algorithms prevent axioms from being absorbed [3].

2 Tableau Algorithm and Lazy Unfolding

Reasoning about a DL based KB is usually reduced to concept reasoning w.r.t. a *TBox*. For such kind of reasoning, a sound and complete tableau reasoning algorithm is usually employed. The basic idea behind a tableau algorithm is to take an input concept C w.r.t. a *TBox* $\mathcal{T}$, and try to prove the satisfiability of C w.r.t. $\mathcal{T}$ by constructing a model $\mathcal{I}$ of C w.r.t. $\mathcal{T}$. This is done by syntactically decomposing C so as to derive constraints on the structure of such a model. For example, any model of C must, by definition, contain some individual x such that x is an element of $C^{\mathcal{I}}$, and if C is of the form $\exists R.D$, then the model must also contain an individual y such that $\langle x, y\rangle \in R^I$ and y is an element of $D^{\mathcal{I}}$; if D is non-atomic, then continuing with the decomposition of D would lead to additional constraints. The construction fails if the constraints include an obvious contradiction, e.g., if some individual z must be an element of both C and $\neg C$ for some concept C [2, 5].

The decomposition and construction are usually carried out by applying so-called tableau expansion rules as described in [2]. During the tableau expansion, disjunctions are added to the label of each node of the tableaux for each GCI (one disjunction is added for axioms of the form $\mathcal{C}_1 \sqsubseteq \mathcal{C}_2$; two disjunctions are added for axioms of the form $\mathcal{C}_1 \equiv \mathcal{C}_2$ [5]). This leads to an exponential increase in the search space as the number of nodes and axioms increases [4].

An intuitive optimization technique is *lazy unfolding* — it only unfolds concepts if required during the expansion process [1]. It has been described by the additional tableau rules in [5].

Lazy unfolding cannot be applied to an arbitrary axiom in a *TBox* due to the atomic concept restriction on the left-hand side of the axiom. However, we can still divide an arbitrary *TBox* $\mathcal{T}$ into two parts: the unfoldable part $\mathcal{T}_u$, to which we can apply *lazy unfolding* directly, and the general part $\mathcal{T}_g$, in which we have to perform reasoning by general tableau expansion [5]. Therefore, there is an intuitive optimization technique to be considered: converting general axioms from $\mathcal{T}_g$ to $\mathcal{T}_u$ while keeping the semantics of *TBox* unchanged. This is the original idea of an *"absorption"*.

3 Standard Absorption

Let us first consider an absorption example. Suppose we have two *TBoxes* $\mathcal{T}$ and $\mathcal{T}'$.

$\mathcal{T} = \mathcal{T}_u \cup \mathcal{T}_g$ and $\mathcal{T}_u = \emptyset$; $\mathcal{T}_g = \{A \sqsubseteq C; \neg A \sqsubseteq D\}$;
$\mathcal{T}' = \mathcal{T}'_u \cup \mathcal{T}'_g$ and $\mathcal{T}'_u = \{A \Rightarrow C; \neg A \Rightarrow D\}$; $\mathcal{T}'_g = \emptyset$.

An obvious question is whether $\mathcal{T}' \equiv \mathcal{T}$?

According to the definition of a *correct absorption* proposed in [6], $\mathcal{T}'$ is a correct absorption of $\mathcal{T}$. Unfortunately, $\mathcal{T}' \neq \mathcal{T}$ since $\mathcal{T}' \models \mathcal{T}$ does not hold. To evaluate the correctness of an absorption, we introduce the notion of a *valid absorption*.

Definition 3.1 (valid absorption) *Let $\mathcal{T}$ be a TBox, and $\mathcal{T}'$ be an absorption of $\mathcal{T}$. If $\mathcal{T}' \models \mathcal{T}$ and $\mathcal{T} \models \mathcal{T}'$, then $\mathcal{T}'$ is called a valid absorption of $\mathcal{T}$.*

Based on the above definitions, the following absorptions are all *valid absorptions*, provided that A is an atomic concept and C, D and E are arbitrary concepts, and $\mathcal{T}$ is an *acyclic TBox* [2].

Proposition 3.1 *Let $\mathcal{T} = \mathcal{T}_u \cup \mathcal{T}_g$, $\mathcal{T}_u = \emptyset$ and $\mathcal{T}_g = \{A \sqsubseteq D\}$, $A \in NC$, and $\mathcal{T}' = \mathcal{T}'_u \cup \mathcal{T}'_g$; $\mathcal{T}'_u = \{A \Rightarrow D\}$ and $\mathcal{T}'_g = \emptyset$. Then $\mathcal{T}'$ is a valid absorption of $\mathcal{T}$.*

Proposition 3.2 *Let $\mathcal{T} = \mathcal{T}_u \cup \mathcal{T}_g$, $\mathcal{T}_u = \emptyset$ and $\mathcal{T}_g = \{A \equiv D\}$, $A \in NC$, and $\mathcal{T}' = \mathcal{T}'_u \cup \mathcal{T}'_g$, $\mathcal{T}'_u = \{A \Rightarrow D; \neg A \Rightarrow \neg D\}$ and $\mathcal{T}'_g = \emptyset$. Then $\mathcal{T}'$ is a valid absorption of $\mathcal{T}$.*

Proposition 3.3 *Let $\mathcal{T} = \mathcal{T}_u \cup \mathcal{T}_g$.*

(1) If $\mathcal{T}'$ is an arbitrary TBox, then $(\mathcal{T}_u, \mathcal{T}_g \cup \mathcal{T}')$ is a valid absorption of $\mathcal{T} \cup \mathcal{T}'$.

(2) If $\mathcal{T}'$ is a TBox that consists entirely of axioms in the form of $A \sqsubseteq D$, where $A \in NC$ and neither A nor $\neg A$ occur on the left-hand side in $\mathcal{T}_u$, then $(\mathcal{T}_u \cup \{A \Rightarrow D\}, \mathcal{T}_g)$ is a valid absorption of $\mathcal{T} \cup \mathcal{T}'$.

A question rises from Proposition 3.3 whether it is possible to absorb an axiom into $\mathcal{T}_u$ if either A or $\neg A$ occur on the left-hand side of $\mathcal{T}_u$.

Lemma 3.1 *Let $(\mathcal{T}_u, \mathcal{T}_g)$ be a valid absorption of a TBox $\mathcal{T}$. If $\mathcal{T}'$ is a TBox that consists entirely of axioms in the form of $A \sqsubseteq D$, where $A \in NC$ and A already has a rule definition in $\mathcal{T}_u$, say $A \Rightarrow C$, if $\neg A$ does not appear on the left-hand side of $\mathcal{T}_u$, then $(\mathcal{T}_u \cup \{A \Rightarrow (D \sqcap C)\}, \mathcal{T}_g)$ is a valid absorption of $\mathcal{T} \cup \mathcal{T}'$.*

Lemma 3.2 *Let $(\mathcal{T}_u, \mathcal{T}_g)$ be a valid absorption of a TBox $\mathcal{T}$. If $\mathcal{T}'$ is a TBox that consists entirely of axioms of the form $A \sqsubseteq D$, where $A \in NC$ and $\neg A$ already has a rule definition in $\mathcal{T}_u$, say $\neg A \Rightarrow C$, if A does not appear on the left-hand side of $\mathcal{T}_u$, then $(\mathcal{T}_u \cup \{A \Rightarrow D\}, \mathcal{T}_g \cup \{\top \sqsubseteq C \sqcup D\})$ is a valid absorption of $\mathcal{T} \cup \mathcal{T}'$.*

Lemma 3.3 *Let $(\mathcal{T}_u, \mathcal{T}_g)$ be a valid absorption of a TBox $\mathcal{T}$, if $\mathcal{T}'$ is a TBox that consists entirely of axioms of the form $A \sqsubseteq E$, where $A \in NC$. If both A*

and $\neg A$ *have a rule definition in* $\mathcal{T}_u$*, say* $A \Rightarrow C$ *and* $\neg A \Rightarrow D$*, then* $(\mathcal{T}_u \cup \{A \Rightarrow (C \sqcap E)\}, \mathcal{T}_g \cup \{\top \sqsubseteq D \sqcup E\})$ *is a valid absorption of* $\mathcal{T} \cup \mathcal{T}'$*.*

The above mentioned propositions and lemmas also hold for a cyclic *TBox* as long as no right-hand side concept in $\mathcal{T}_u$ *directly uses* [2] an atomic concept (regardless of a negation sign) occurring in the left-hand side of the same axiom.

4 Heuristic Absorption

Experimental experience suggests that reasoning efficiency is improved by either reducing the number of axioms in $\mathcal{T}_g$ or reducing the number of axioms in $\mathcal{T}_u$. Thus, we propose the "best" absorption is the one which can absorb a maximal number of axioms from $\mathcal{T}_g$ and keep only a minimal number of axioms in $\mathcal{T}_u$.

A first possible way of reducing axioms in $\mathcal{T}_g$ and $\mathcal{T}_u$ is to convert a *primitive definition* into a *complete definition*, i.e., selecting a concept definition instead of a concept inclusion by checking all axioms in the specified *TBox*. This can be achieved as follows.

Given an arbitrary TBox $\mathcal{T}$. Suppose A is an atomic concept.

1. *Simplify [8] and normalize* $\mathcal{T}$*. After normalization,* $\mathcal{T}_u = \emptyset$ *and* $\mathcal{T}_g$ *contains a set of axioms in the form* $\top \sqsubseteq C \sqcup D$ *where* C *and* D *are either atomic concepts or role concepts. Then each axiom in* $\mathcal{T}_g$ *can be expressed as a set. For example, the set* $\boldsymbol{G} = \{C, D\}$ *represents the axiom in the form* $\top \sqsubseteq C \sqcup D$*. As a consequence,* $\neg \boldsymbol{G}$ *represents a set containing the concept* $\neg C \sqcap \neg D$*. Therefore, each axiom in* $\mathcal{T}_g$ *only contains* A *or* $\neg A$ *or none of them. We also need a function* con *which returns for a given set* $\boldsymbol{G}$ *its represented concept, e.g., if* $\boldsymbol{G} = \{C, D\}$*, then* $\mathrm{con}(\boldsymbol{G})$ *returns* $C \sqcup D$*.*

2. *Initialize two sets* $\mathcal{T}_{g_1}$*,* $\mathcal{T}_{g_2}$ *to be empty and consider* A *the chosen (fixed) atomic concept.*

3. *For any set* $\boldsymbol{G}$*, if* $A \in \boldsymbol{G}$*, then remove* $\boldsymbol{G}$ *from* $\mathcal{T}_g$ *and add an item* $\neg(\boldsymbol{G} \backslash \{A\})$ *to* $\mathcal{T}_{g_1}$*; if* $\neg A \in \boldsymbol{G}$*, then remove* $\boldsymbol{G}$ *from* $\mathcal{T}_g$ *and add an item* $\boldsymbol{G} \backslash \{\neg A\}$ *to* $\mathcal{T}_{g_2}$*; otherwise, keep* $\boldsymbol{G}$ *in* $\mathcal{T}_g$*.*

4. *For each item* $\boldsymbol{G}_2$ *in* $\mathcal{T}_{g_2}$*, if* $\boldsymbol{G}_2$ *also appears in* $\mathcal{T}_{g_1}$*,*

   (a) *remove* $\boldsymbol{G}_2$ *from both* $\mathcal{T}_{g_1}$ *and* $\mathcal{T}_{g_2}$*;*

   (b) *add the axiom* $\{A \Rightarrow \mathrm{con}(\mathbf{G_2})\}$ *and* $\{\neg A \Rightarrow \mathrm{con}(\neg \mathbf{G_2})\}$ *to* $\mathcal{T}_u$*.*

5. *For each item set* $\boldsymbol{G}'_1$ *left in* $\mathcal{T}_{g_1}$*, create a new set* $\neg \boldsymbol{G}'_1 \cup \{A\}$ *and put it back into* $\mathcal{T}_g$*; for each item set* $\boldsymbol{G}'_2$ *in* $\mathcal{T}_{g_2}$*, create a new set* $\boldsymbol{G}'_2 \cup \{\neg A\}$ *and put it back into* $\mathcal{T}_g$*.*

The above algorithm can be further improved in step (4) by checking more than one item set in G_2. For more general KBs, the following heuristic absorption procedure could be applied.

Given an arbitrary TBox $\mathcal{T} = \mathcal{T}_u \cup \mathcal{T}_g$. $\mathcal{T}_u$ contains a set of axioms of the form $C \equiv D$ or $C \Rightarrow D$. $\mathcal{T}_g$ contains a set of axioms of the form $\top \sqsubseteq C$.

1. *Simplify and normalize $\mathcal{T}$ as described above.*
2. *Suppose c_p is the total number of appearance of A in $\mathcal{T}_g$; and c_n is the total number of appearance of $\neg A$ in $\mathcal{T}_g$. Among all c_ps and c_ns, select the greatest value and absorb the accordingly atomic concept to $\mathcal{T}_u$.*
3. *Repeat step (2) until no concept can be absorbed anymore.*

In fact, the above mentioned two algorithms can be combined to achieve a better performance.

5 Experimental Results

To check the effectiveness of the newly developed algorithms, we compare them with the ones currently employed by RACER. Firstly, we developed a customized RACER version by disabling its absorption module. After that, we developed a new external absorption module (as a stand-alone program) by implementing the algorithms described above. Each of our test KBs is processed by the external absorption module. Its output is used as input to the customized RACER version. We also process each original test KB with the standard RACER version. In our graphs we compare for each test KB the TBox classification time of the customized RACER version (denoted as "enhanced") with the standard version of RACER (denoted as "normal").

5.1 Primitive Definition to Complete Definition

Suppose A, B, C, D are all atomic concepts in the following example:

$TBox\ \mathcal{T} = \mathcal{T}_u \cup \mathcal{T}_g$;
$\mathcal{T}_u = \{A \Rightarrow B;\ C \Rightarrow B;\ B \Rightarrow A \sqcup C\}$ and $\mathcal{T}_g = \{\top \sqsubseteq A \sqcup C \sqcup \exists R.D\}$

After normalization and simplification, the above $TBox$ can be easily absorbed into the following $TBox\ \mathcal{T}'$ by heuristically creating these concept definitions:

$\mathcal{T}'_u = \{B \equiv (A \sqcup C); \neg A \Rightarrow C \sqcup \exists R.D\}$
$\mathcal{T}'_g = \emptyset$

The classification times for a KB where we replicated this pattern of axiom pairs are shown in the left graph in Figure 1.

From the graph one can see that the CPU time for reasoning with the algorithms currently employed by RACER is exponential. However, the performance based on the newly developed absorption algorithm is roughly linear.

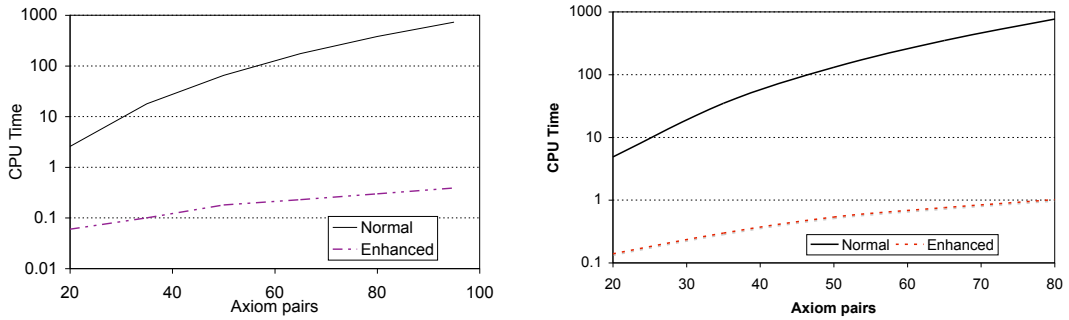


Figure 1: Classification times (seconds) for pattern "primitive to complete definition" (left graph) and for pattern "enhanced absorption" (right graph).

5.2 Enhanced Absorption Algorithm

Based on Lemma 3.2 and 3.3, we were able to develop an absorption algorithm by allowing both positive or negative atomic concepts to occur on the left-hand side of $\mathcal{T}_u$. Consider the following example ($A, B \in NC$):

$\mathcal{T} = \mathcal{T}_u \cup \mathcal{T}_g$;
$\mathcal{T}_u = \{A \sqsubseteq B\}$; $\mathcal{T}_g = \{\top \sqsubseteq A \sqcup \exists R.K\}$

By applying Lemma 3.3, we are able to completely absorb it as the following:

$\mathcal{T}_u = \{A \Rightarrow B; \neg A \Rightarrow \exists R.K; \neg B \Rightarrow \exists R.K\}$; $\mathcal{T}_g = \emptyset$

The test result is shown in the right graph in Figure 1. They show that the new absorption algorithm is much more effective than the classical absorption algorithm due to the elimination of absorption restrictions.

5.3 Heuristic Absorption

Suppose we have the following *TBox* $\mathcal{T}$ which consists of the following general axioms:

$\{\neg A \sqcup B; \neg C \sqcup \neg D \sqcup \exists R_1.C_1 \sqcup \exists R_2.C_2; \neg D \sqcup K; \neg M \sqcup N; C \sqcup D \sqcup \exists R_1.C_1 \sqcup M \sqcup A\}$

To absorb $\mathcal{T}$, we follow the procedure described in Section 4:
Step 1: List the statistics of T_g for each atomic concept. The result is as follows. (The format is $A(c_p, c_n)$, where c_p gives the number of positive (unnegated) occurrences and c_n the number of negative (negated) occurrences, and we ignore the atomic concepts occurring in the qualifications of existential and universal restrictions.)

A(0,1); B(1,0);C(1,1);D(1,2);M(1,1); N(1,0); K(1,0);$C_2(0,0)$; $C_1(0,0)$

We divide the atomic concepts into two groups by selecting the concepts where one of c_p or c_n values is zero and discarding the concepts that have both c_p and c_n values equal to zero. We obtain the following:

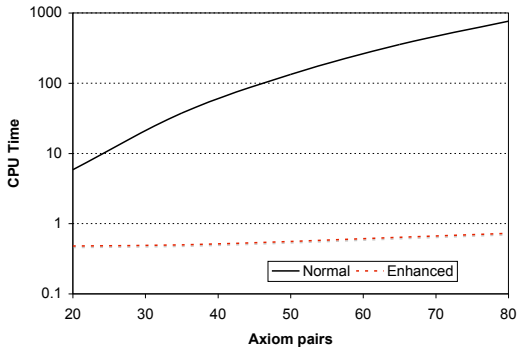


Figure 2: Classification times (seconds) for pattern "heuristic absorption".

Group 1: B(1,0); N(1,0); K(1,0)
Group 2: A(1,1); C(1,1);D(1,2);M(1,1)

Step 2: We give the concepts in the first group a higher priority. Then, the concepts B, N, K have the same priority. Suppose we absorb B first. We have T_u and T_g as the following:

T_u: $\{\neg B \Rightarrow \neg A\}$
T_g: $\{\neg C \sqcup \neg D \sqcup \exists R_1.C_1 \sqcup R_2.C_2;\ \neg D \sqcup K;\ \neg M \sqcup N;\ C \sqcup D \sqcup \exists R_1.C_1 \sqcup M \sqcup A\ \}$

Statistics group:

Group 1: A(1,0); N(1,0); K(1,0)
Group 2: C(1,1);D(1,2);M(1,1)

We repeat step 1 and step 2 until T_g is empty. At the end, T_u contains the following axioms:

$\neg B \Rightarrow \neg A$
$\neg A \Rightarrow C \sqcup D \sqcup \exists R_1.C_1 \sqcup M$
$D \Rightarrow K \sqcap (\neg C \sqcup \exists R_1.C_1 \sqcup R_2.C_2)$
$M \Rightarrow N$

and $T_g = \emptyset$

The test results using the above absorption scheme are shown in Figure 2. The improvement of the heuristic absorption algorithm is significant compared with the one currently employed by RACER.

6 Conclusion

We proposed criteria for the "best" absorption based on experimental experience. Then, we introduced novel heuristic absorption algorithms. We have demonstrated how these algorithms are working, and how they affect the reasoning performance. In addition, we have shown that some restrictions applied in the absorption algorithms of RACER could be eliminated. Therefore, the

absorption algorithms can be effectively applied to more general axioms.

We have also implemented the heuristic algorithm by incorporating the optimizations known from the RACER reasoner. We have illustrated their effectiveness by analyzing the reasoning performance of RACER when classifying benchmark KBs. The analysis shows that, not only are the new techniques highly effective, but also the reasoning performance is not significantly affected by the order and format of the axioms occurring in a KB.

References

[1] F. Baader, E. Franconi, B. Hollunder, B. Nebel, and H. Profitlich. An empirical analysis of optimization techniques for terminological representation systems. In *Proc. of the 3rd Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR92)*, pages 270–281, 1992.

[2] F. Baader and W. Nutt. Basic description logics. In F. Baader, D. Calvanese, D.L. McGuinness, D. Nardi, and P.F. Patel-Schneider, editors, *The Description Logic Handbook: Theory, implementation, and applications*, pages 47–100. 2003.

[3] V. Haarslev and R. Möller. High performance reasoning with very large knowledge bases. In *Proceedings of Seventeenth International Joint Conference on Artificial Intelligence*, pages 161–166, 2001.

[4] I. Horrocks. *Optimising Tableaux Decision Procedures for Description Logics.* PhD thesis, University of Manchester, 1997.

[5] I. Horrocks. Implementation and optimisation techniques. In F. Baader, D. Calvanese, D.L. McGuinness, D. Nardi, and P.F. Patel-Schneider, editors, *The Description Logic Handbook: Theory, implementation, and applications*, pages 313–355. 2003.

[6] I. Horrocks and S. Tobies. Optimisation of terminological reasoning. In *Proc. of the 2000 Description Logic Workshop*, pages 183–192, 2000.

[7] I. Horrocks and S. Tobies. Reasoning with axioms: Theory and practice. In *Proceedings of the Seventh International Conference on Principles of Knowledge Representation and Reasoning*, pages 285–296, 2000.

[8] F. Massacci. Simplification: A general constraint propagation technique for propositional and modal tableaux. In *Proceedings of the International Conference on Automated Reasoning with Analytic Tableaux and Related Methods*, pages 217–231, 1998.

Automated Benchmarking of Description Logic Reasoners

Tom Gardiner, Ian Horrocks, Dmitry Tsarkov
University of Manchester
Manchester, UK
{gardiner|horrocks|tsarkov}@cs.man.ac.uk

May 12, 2006

1 Introduction

Reasoning for expressive DLs implemented in state-of-the-art systems has high worst case complexity. The hope/claim is, however, that these systems perform well in "realistic" applications. In practice, this means in ontology applications. To check the validity of this claim it is necessary to test the performance of these systems with (the widest possible range of) ontologies derived from applications.

In addition, testing is useful in order to check the correctness of implementations. In small examples, it may be easy to check the correctness of a system's reasoning. However, for typical real-world examples, manual checking is not feasible. In these instances, the best (perhaps the only) way to check correctness is often by checking for consistency with the reasoning of other existing systems.

Real-world ontologies vary considerably in their size and expressivity. While they are all valuable test cases, it is still important to understand each ontology's properties in order to provide efficient and relevant testing.

System developers find this particularly useful, as it helps them to identify weaknesses in their systems and to devise and test new optimisations. Finally, testing is also useful for the developers and users of applications as they can use benchmarking results to determine if (the performance of) a DL reasoner is likely to satisfy their requirements, and if so which reasoner is likely to perform best in their application.

2 Background and Related Work

For the above mentioned reasons, there is extensive existing work on benchmarking DL (as well as modal logic) reasoners. E.g., TANCS comparisons

and benchmark suites [10], DL comparisons and benchmark suite [1], work on M-SPASS [8], work on FaCT and DLP [7, 6], the OWL benchmark suite and test results, and various test results from papers describing systems such as FaCT++ [13], Pellet (`http://www.mindswap.org/2003/pellet/`), Racer [5], KAON2 (`http://kaon2.semanticweb.org/`), Vampire [12], etc.

Due to the fact that relatively few (large and/or interesting) ontologies were available, earlier tests often used artificially generated test data. The Lehigh University Benchmark [4], for example, used a synthetic ontology and randomly generated data to test the capabilities of knowledge base systems using specific weightings to compare systems on characteristics of interest. Results from such tests are, however, of doubtful relevance when gauging performance on ontologies. The popularity of OWL has meant that many more ontologies are now available, and recent benchmarking work has focused on testing performance with such ontologies.

One such example [11] involved benchmarking a number of reasoners against a broad range of realistic ontologies. However, not all reasoners used in that comparison supports OWL as an input language, so quantitative comparison of performance would have been difficult/un-justified. From the other hand, the DIG interface [2] is recognised as a preferred choice by application developers and thus is implemented into a wide range of DL Reasoners.

Our work builds on these earlier efforts, taking advantage of the DIG standard to provide a generic benchmarking suite that allows the automatic quantitative testing and comparison of DL Reasoners on real-world examples with relevant properties. We aim to make the testing process as autonomous as possible, taking care, for example, of (re)starting and stopping reasoners as necessary, and the analysis of results be as flexible as possible, by allowing for arbitrary SQL queries against the collected data. We also aim to provide, as a publicly available resource, a library of test ontologies where each ontology has been checked for expressivity and syntactic conformance, translated into DIG syntax (which is much easier to work with that OWL's RDF/XML syntax), and includes (where possible) results (such as the concept hierarchy) that can be used for testing the correctness of reasoning systems.

3 Methodology

The system has two main functions. The first is to process ontologies and add them to the library, and the second is to benchmark one or more reasoners using the ontology library.

When processing ontologies, the system takes as input a list of OWL-ontology URI's. Before they can be used in testing, some preprocessing of these ontologies is required. The process involves generating valuable meta-data about each ontology, as well as converting each of the OWL-ontologies to DIG.

The meta-data is generated by code written for SWOOP [9], and provides the details of the expressivity (i.e. the constructs present in the ontology) together with the number of classes, object properties, data properties, individuals, class axioms, property axioms and individual axioms present. This is invaluable information in helping to understand the meaning of any results obtained through testing, in finding, for example, strengths and weaknesses of particular systems. The OWL-to-DIG conversion uses the OWL-API (`http://sourceforge.net/projects/owlapi`). This process is far from trivial as OWL's RDF syntax is extremely complex, and it is easy to (inadvertently) cause ontologies to be outside of OWL DL, e.g., by simply forgetting to explicitly type every object. Moreover, the DIG interface supports only the most basic of data types, such as Strings and Integers. The result is that many of the available OWL Ontologies we found could not be successfully converted to DIG.

Local copies are stored of both the OWL Ontology and the DIG version. This is not only for efficiency during the testing, but also to ensure consistency (as online ontologies rarely remain static). Moreover, this allows us to fix trivial errors in the OWL ontologies so that they can be used for testing purposes. The locations of these files, together with their properties/meta-data, are stored as database entries for easy access and manipulation.

The main function of the benchmark suite itself is timing the classification of each ontology by each Reasoner. To promote fairness, each Reasoner is terminated and then restarted for every ontology.

A problem with trying to compare different Reasoners is that they may perform tasks in different ways. For example, they may vary in the way in which they perform each part of the reasoning: some may take an "eager" approach, fully classifying the whole ontology and caching the results as soon as it is received; others may take a "lazy" approach, only performing reasoning tasks as required in order to answer queries. To try to get around this problem, we use a five step test, for each ontology, that forces every reasoners to fully classify that ontology. The steps are as follows:

1. TELL the reasoner the full ontology
2. ASK for all the concepts in the ontology
3. ASK for the satisfiability of the TOP concept
4. ASK for the satisfiability of all the concepts in the ontology
5. ASK for the ontology taxonomy (parents and children of all concepts)

Each of these individual steps are timed, providing interesting information about when different reasoners do most their work. It is, however, the total time for this complete (classification) test that we are most interested in.

Each test will end in one of three ways. It will either complete successfully, fail due to lack of time or fail for some other reasons. The latter may include failure due to lack of run-time memory, failure because the reasoner could not parse the ontology successfully, etc.

The benchmark suite is fully automatic, dealing with most errors autonomously,

meaning that the testing can be left to run over-night or over a week-end (which may be necessary when using a large time-out). All data is recorded in a MySQL database, making it easy for the user to view and analyse the data in a variety of ways.

As discussed in Section 1, in order to get a clearer indication of how DL Reasoners perform in the real world, we aim to build a large library of OWL ontologies from those that are publicly available. Currently, our library contains a little over 300 OWL-RDF Ontologies, but only 172 of these could successfully be converted to DIG. This has, however, provided us with a total of just under 72,000 classes and over 30,000 individuals in a DIG format. Only 18% of the ontologies were at least $\mathcal{ALC}$, which suggests that the majority of real-world ontologies aren't in fact very complex, but it also means we have a comfortable number of "interesting" examples too.

4 Testing

Our system is currently fully automatic and runs the classification tests successfully through our whole library. It does not, however, at this stage verify the correctness of each Reasoner's answers to the queries (from steps 2-5) and how they compare to the answers given by other Reasoners. This means that our measure of success is, for now, merely an indication that the Reasoner received and parsed the DIG successfully and returned a valid DIG response. This is generally a good indication, but should only be considered a preliminary result.

We have performed some tests on our system, as it stands, and we provide here some examples of the kinds of information that our system can produce.

FaCT++ v1.1.3, KAON2, Pellet v1.3 and RacerPro v1.8.1 are four of the most widely used OWL/DIG reasoners, and we therefore decided to use these to test the current capabilities of our system. The tests were performed using an Intel Pentium-M Processor 1.60 GHz and 1Gb of main memory on Windows XP. The time-out period was set to 10 minutes (in real time). Pellet and KAON2 are java applications, and for these tests were run with a maximum heap space of 200Mb. RacerPro and FaCT++ were left to run on their default settings. Our system does not try to optimise the performance of the Reasoners for particular ontologies, as we believe this is the job of the Reasoners themselves, not the application user.

Table 1 shows how the individual Reasoners performed firstly on all our ontologies and then on Ontologies which have particular characteristics. Finally, it shows their performance on OWL-Lite ontologies, which includes all those with expressivity up to SHIF.

In order to determine which were the most "challenging" ontologies (w.r.t. reasoning), we tried to order ontologies according to the difficulty of reasoning with them. To do this, we used all the ontologies that were successfully classified by at least two Reasoners and then ordered these by their average classification

Type	Status	FaCT++	KAON2	Pellet	RacerPro
All	Success	138	45	152	110
All	Failed	29	124	18	62
All	TimedOut	5	3	2	0
Nominals	Success	7	3	9	7
Nominals	Failed	4	9	3	5
Nominals	TimedOut	1	0	0	0
TransRoles	Success	15	9	18	13
TransRoles	Failed	4	11	3	9
TransRoles	TimedOut	3	2	1	0
Datatypes	Success	102	12	114	75
Datatypes	Failed	23	15	13	52
Datatypes	TimedOut	2	0	0	0
OWL-Lite	Success	33	31	33	34
OWL-Lite	Failed	5	6	5	6
OWL-Lite	TimedOut	2	3	2	0

Table 1: Sample of Overall Performance

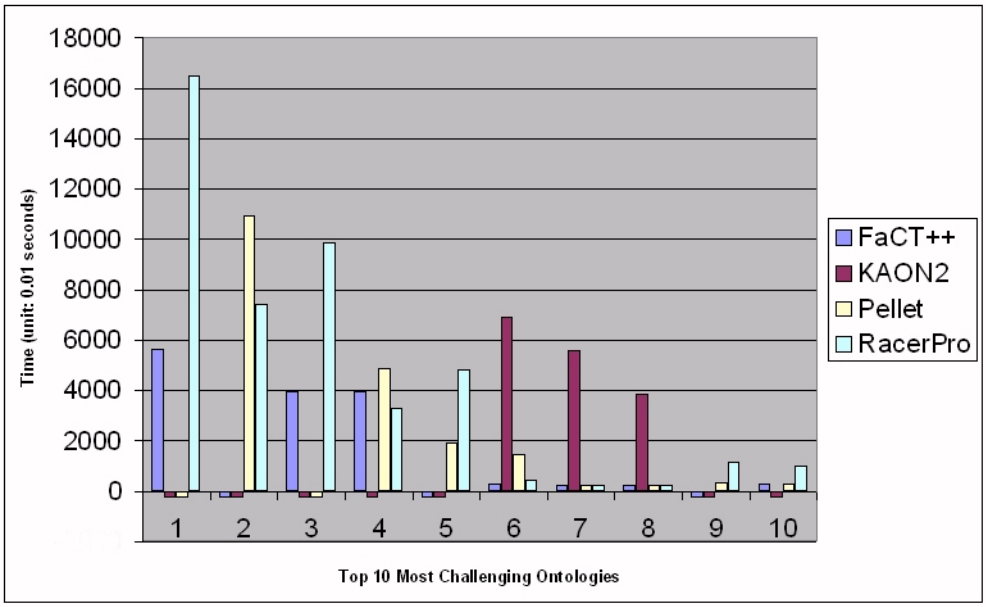


Figure 1: Comparison of Reasoners on the Top 10 Most Challenging Ontologies

Ontology	Expressivity	nClass	nIndiv	URL
1	DL-Lite	27652	0	http://...logy/nciOncology.owl
2	$\mathcal{SHF}$	3097	0	http://...ibrary/not-galen.owl
3	$\mathcal{ALR}+$	20526	0	http://archive.godatabase.org/
4	$\mathcal{SHF}$	2749	0	http://...Ontologies/galen.owl
5	RDFS(DL)	1108	3635	http://...world-fact-book.daml
6	RDFS(DL)	1514	0	http://...logy/data/center.owl
7	$\mathcal{ALCF}$(D)	87	0	http://...a/pizza/20041007.owl
8	$\mathcal{SHIF}$	37	0	http://...s/DOLCE-Lite/397.owl
9	RDFS(DL)	4	1899	http://...nt/AirportCodes.daml
10	$\mathcal{ALR}+\mathcal{HI}$(D)	5	2744	http://...ogicUnits/2003/09/hu

Table 2: Properties of Top 10 Most Time-consuming Ontologies

Reasoner	Tells	ConceptList	SatOfTop	SatOfClasses	Hierarchy
FaCT++	23%	35%	11%	9%	21%
KAON2	47%	45%	0%	3%	5%
Pellet	70%	20%	1%	2%	6%
RacerPro	58%	11%	4%	9%	19%

Table 3: Average Division of Task Time

time. Figure 1 shows the amount of time each Reasoner took to classify the 10 most challenging ontologies according to this measure (where negative time represents a failure to classify). Table 2 then shows some of the interesting information that is available on these "Top 10" Ontologies.

This table is useful in helping us understand what makes these particular Ontologies so time-consuming to reason over. In the case of the NCI and Gene Ontology's (1st and 3rd), it can be clearly seen that it is their shear size that provides the challenge. The 5th, 9th and 10th (world-fact-book, AirportCodes and Hydrolic Units) make up for their number of classes with an extensive array of individuals. Whereas Galen (2nd and 4th) simply uses some very complicated constructs and deep role hierarchy.

Our final table, Table 3, shows the average proportion of each classification test that each Reasoner spent on the separate tasks. This shows, for example, that Pellet performs a lot of caching on receiving the Ontology (TELLS), while FaCT++ does relatively little until the first ASK query.

5 Discussion

As we mentioned in the introduction, testing is useful for reasoner and tool developers as well as for users. Building on existing work, we have developed

a system for testing reasoners with available ontologies. The benefits of our approach include autonomous testing, flexible analysis of results and the development of a test library that should be a valuable resource for both the DL and ontology community. We will continue to extend the library, and will add classification results from tested reasoners so that correctness testing can also be performed.

While there are an increasingly large array of OWL-Ontologies available for public use, other Ontology formats (e.g. OBO: the Open Biomedical Ontologies, `http://obo.sourceforge.net`) are still widely in use and would make for valuable test examples. It is also the case, as describe in [3], that a large proportion of the available OWL-Full Ontologies, could in fact be validated as OWL-DL, just by adding a few extra clarifying statements. This means that of the 162 Ontologies that we had to throw away, many could be useful examples with a little work. In the future we hope to use these observations, together with any external contributions, to considerably increase the size of our ontology library.

The results produced by our tests provide an interesting insight into the variety and depth of information that can be extracted from such testing/benchmarking. However, for the system and its results to become a valuable resource, we need to test their correctness. We are currently assuming that both the OWL-to-DIG conversions and the Reasoner's responses are all valid and correct.

With regard to the OWL-API's conversions, this was the utility built alongside the original DIG specification. We therefore argue that this is the best conversion available and that our assumption is justified.

Regarding the responses, as discussed earlier, they can be almost impossible to check for correctness. Our best option is therefore to analyse the difference in responses received from different reasoners, and this route is thus one we aim to explore further. It will be interesting to see if reasoners (that should, in theory, all produce the same inferences to the same problems) will actually agree on the test ontologies.

So far we have focused on testing Tbox reasoning (classification). Although the use of nominals in $\mathcal{SHOIQ}$ blurs the separation between Tbox and Abox, it would still be useful to explicitly test Abox reasoning, e.g., by asking for the instances of some query class. This functionality will be added in a future version of the system.

Apart from the future work described above, there are a number of extensions to our benchmarking system that would enhance its utility. Allowing users to define their own customised test, rather than the 5 step classification we are using, is one example that would allow Reasoner developers to test specific optimisations and implementations as they are developed. Other relevant tests would include testing how multiple concurrent tests on a Reasoner affects performance, as well as simply not restarting a Reasoner between tests.

We intend for the whole system, including the ontology library, to be available for open-source use in the near future.

References

[1] P. Balsiger and A. Heuerding. Comparison of theorem provers for modal logics. In *Proceedings of Tableaux'98*, May 1998.

[2] Sean Bechhofer, Ralf Möller, and Peter Crowther. The DIG description logic interface. In *Proceedings of DL2003 International Workshop on Description Logics*, September 2003.

[3] Sean Bechhofer and Raphael Volz. Patching syntax in OWL ontologies. In *Proceedings of 3rd International Semantic Web Conference, ISWC*, 2004.

[4] Y. Guo, Z. Pan, and J. Heflin. LUBM: A benchmark for OWL knowledge base systems. *Journal of Web Semantics*, 3(2):158–182, 2005.

[5] V. Haarslev and R. Möller. RACER system description. In R. Goré, A. Leitsch, and T. Nipkow, editors, *International Joint Conference on Automated Reasoning, IJCAR'2001, June 18-23, Siena, Italy*, pages 701–705. Springer-Verlag, 2001.

[6] I. Horrocks. Benchmark analysis with FaCT. In *Proc. of TABLEAUX 2000*, number 1847, pages 62–66. Springer-Verlag, 2000.

[7] I. Horrocks and P. F. Patel-Schneider. FaCT and DLP. In *Proc. of TABLEAUX 98*, pages 27–30, 1998.

[8] U. Hustadt and R. A. Schmidt. MSPASS: Modal reasoning by translation and first-order resolution. 2000.

[9] Aditya Kalyanpur, Bijan Parsia, Evren Sirin, Bernardo Cuenca-Grau, and James Hendler. Swoop: A 'web' ontology editing browser. *Journal of Web Semantics*, 4(2), 2005.

[10] Fabio Massacci and Francesco M. Donini. Design and results of TANCS-00. volume 1847, 2000.

[11] Zhengxiang Pan. Benchmarking DL reasoners using realistic ontologies. In *Proc. of the OWL: Experiences and Directions Workshop*, 2005.

[12] A. Riazanov and A. Voronkov. The Design and Implementation of Vampire. *AI Communications*, 15(2-3):91–110, 2002.

[13] Dmitry Tsarkov and Ian Horrocks. FaCT++ Description Logic Reasoner: System Description. In *Proc. of the Int. Joint Conf. on Automated Reasoning (IJCAR 2006)*, 2006. To Appear.

Will My Ontologies Fit Together?

A preliminary investigation.

Bernardo Cuenca Grau, Ian Horrocks,
Oliver Kutz, and Ulrike Sattler
School of Computer Science, The University of Manchester, UK

1 Motivation

In realistic applications, it is often desirable to integrate different ontologies[1] into a single, reconciled ontology. Ideally, one would expect the individual ontologies to be developed as independently as possible, and the final reconciliation to be seamless and free from unexpected results. This allows for the modular design of large ontologies and facilitates knowledge reuse tasks. Few ontology development tools, however, provide any support for integration, and there has been relatively little study of these issues at a fundamental level. Understanding at this fundamental level would help us predict, for example, what logical consequences to expect from the integrated ontology, and whether the integration of ontologies preserves some desirable properties of its parts.

To the best of our knowledge, the problem of predicting and controlling the consequences of ontology integration has has been tackled only in [4]. The authors propose a set of reasoning services (with decidability and complexity results) to check whether, through integration with other ontologies, desirable properties of an ontology have been destroyed.

In this paper, we propose first steps towards a different approach, the so-called *normative* approach.[2] We specify certain properties that one would like to preserve in the integration and devise a set of restrictions that, when adhered to, guarantee to preserve these properties. Thus, while the approach of [4] determines the preservation of desirable properties ex post, our methodology prescribes some restrictions that guarantee the preservation of desirable properties. We introduce two 'integration scenarios' that, we believe, capture some of the common practices in ontology engineering, and postulate desirable properties that should be satisfied by the integrated ontology. We provide syntactic restrictions on the use of shared vocabulary that guarantee the preservation of

[1] Throughout this paper, we do not distinguish between ontologies and TBoxes.

[2] Thanks to Frank Wolter for coining this expression.

these properties. The two basic ontology integration scenarios we analyze here are the following:

1. *Foundational integration*: an ontology is integrated with a foundational (or "upper") ontology. The foundational ontology describes more general terms, and may be domain independent.
2. *Broadening integration*: two ontologies describing different (and largely independent) domains are integrated to cover a broader subject area.

We define, for each scenario, *semantic* properties that one might want to be satisfied by the integrated ontology, that is, we specify how the consequences of the integrated ontology relate to those from its parts. Next, we specify *syntactic* constraints on the ontologies to be integrated and show that they guarantee these properties: these syntactic constraints are referred to as *compliance* conditions. Furthermore, we sometimes need global semantic *safety* constraints on the ontologies used in the integration. In this paper, we use a condition called localness, which is identical to a condition found in [3].

Clearly, the syntactic constraints depend on the DL used in the ontologies, and mainly concern the way the symbols occurring in the different ontologies (their *signatures*) are used. Finally, we discuss whether these constraints are realistic for the scenario, i.e., whether users could be expected to stick happily to these constraints in order to ensure that the integrated ontology will satisfy the desired semantic properties. Since all constraints are purely syntactic, they are decidable in polynomial time, and preliminary tests indicate that many ontologies satisfy the constraints already.

We assume the reader to be familiar with the basics of description logics and use, throughout this paper, *axiom* for any kind of TBox, RBox, or ABox assertion, and $\mathsf{Sig}(\mathcal{T})$ for the set of concept and roles names in $\mathcal{T}$. This paper is accompanied by a technical report [2].

2 Integration Scenarios

Suppose that two ontologies $\mathcal{T}_1, \mathcal{T}_2$ are to be integrated in some application. The ontologies may be the result of a collaborative ontology development process and may have been designed in a coordinated way by different groups of experts, or they may have been simply "borrowed" from the Web. In any case, we assume that they have both been tested and debugged individually prior to the integration and, hence, are consistent and do not contain unsatisfiable concept names. To capture this notion, we call an ontology $\mathcal{T}$ *instantiable* if there exists a model $\mathcal{I}$ of $\mathcal{T}$ s.t. $A^I \neq \varnothing \neq R^{\mathcal{I}}$ for all concept and role names A, R in the

signature of $\mathcal{T}$.[3]

In the simplest case, one would construct an integrated ontology $\mathcal{T}$ by simply taking the *union* of the two ontologies $\mathcal{T}_1, \mathcal{T}_2$. In general, the ontologies $\mathcal{T}_1$ and $\mathcal{T}_2$ may be related and share symbols in their signatures $\mathsf{Sig}(\mathcal{T}_1)$ and $\mathsf{Sig}(\mathcal{T}_2)$.[4] We will first postulate the semantic properties that $\mathcal{T}$ should satisfy in order to capture the modeling intuitions of each scenario, and then to investigate "acceptable" syntactic restrictions on the $\mathcal{T}_i$ that make sure that $\mathcal{T}$ will behave as expected. The intuition is simple: the more liberal the syntactic constraints including the use of shared symbols, the more freedom is given to the modeler, but the less likely it is that the integrated ontology will behave as desired.

2.1 Foundational Integration

Often, interoperability between different *domain* ontologies $\mathcal{T}_{dom}$ and their data is achieved through the use of a *foundational* (or "upper") ontology $\mathcal{T}_{up}$. A well designed foundational ontology should provide a carefully conceived high level axiomatization of general purpose concepts. Foundational ontologies, thus, provide a structure upon which ontologies for specific subject matters can be based.

A prominent example of an ontology conceived as the integration of a foundational ontology and a set of domain ontologies is GALEN [8], a large medical ontology designed for supporting clinical information systems. The foundational ontology contains generic concepts, such as *Process* or *Substance*. The domain ontologies contain concepts such as *Gene* or *Research Institution*, which are specific to a certain subject matter. The domain ontologies in GALEN are connected to the foundational ontology through subsumption relations between concept and role names. For example, Microorganism in one of the domain ontologies is a subconcept of Organism in the foundational ontology: $\mathsf{Microorganism} \sqsubseteq \mathsf{Organism}$. Some prominent ontologies, such as CYC, SUMO and DOLCE have been designed specifically to be used in applications as foundational ontologies. For example, given a large dataset about chemicals annotated with concepts in a certain biomedical ontology, e.g., the National Cancer Institute Thesaurus (NCI) [5], one may want to annotate it semi-automatically with concepts of a different ontology. For such a purpose, one may align organic chemicals in NCI to substances in SUMO using the axiom: $\mathsf{Organic\_Chemical} \sqsubseteq \mathsf{Substance}$. Similarly, one may want to use a foundational ontology to generalize the roles of a given domain ontology. For example, a University ontology may use SUMO to

[3] For $\mathcal{T}$ a TBox in a logic whose models are closed under disjoint unions, such as $\mathcal{SHIQ}$, $\mathcal{T}$ is instantiable if all concept and role names in $\mathcal{T}$ are satisfiable.

[4] There may be some previous reconciliation w.r.t. symbols, e.g., to identify different symbols in the two ontologies that have the same intended meaning [7]. This is a separate problem, often referred to as *ontology alignment*, which we do not address here.

generalize the role writes as follows: writes $\sqsubseteq$ authors , where authors is defined in SUMO and does not occur in the University ontology.

Foundational ontologies are well-established ontologies that one does not control and, typically, does not fully understand. When one of these ontologies is borrowed from the Web and integrated in an application, it is especially important to make sure that the merge preserves their semantics. In particular, we do not want the classification tree in $\mathcal{T}_{up}$ to change as a consequence of the merge. This property can be conveniently formalized by the notion of a *conservative extension* [4].

Definition 1 *The TBox $\mathcal{T} = \mathcal{T}_1 \cup \mathcal{T}_2$ is a* conservative extension *of $\mathcal{T}_1$ if, for every axiom α in the signature of $\mathcal{T}_1$: $\mathcal{T} \models \alpha$ implies $\mathcal{T}_1 \models \alpha$.*

Clearly, if $\mathcal{T}$ is a conservative extension of $\mathcal{T}_1$ and $\mathcal{T}_1, \mathcal{T}_2$ are consistent, then so is $\mathcal{T}$. However, conservativeness is a much stronger condition than instantiability: even if $\mathcal{T}$ is instantiable, new (and probably unintended) subsumptions between (possibly complex) concepts in $\mathcal{T}_1$ may still occur as a consequence of the merge.

In general, it may still be tolerable, and even desirable, to allow new subsumptions to occur in the domain ontology as a consequence of the integration and, in such a case, $\mathcal{T}$ will not be a conservative extension of $\mathcal{T}_{dom}$.

Also, the notion of a conservative extension is not sufficient to capture all the intended and unintended consequences. In particular, one would not expect concept names originally in $\mathcal{T}_{up}$ to be subsumed by concepts originally in $\mathcal{T}_{dom}$. In other words, the rôles of the foundational and domain ontologies should not be inverted after the merge. In contrast, new subsumptions may and should be entailed between concepts (respectively roles) in $\mathcal{T}_{dom}$ and concepts (roles) in $\mathcal{T}_{up}$. For example, since the shared concept Substance is subsumed by SelfConnectedObject in SUMO, it is expected that $\mathcal{T} = \mathcal{T}_{NCI} \cup \mathcal{T}_{SUMO}$ will entail the subsumption Organic_Chemical $\sqsubseteq$ SelfConnectedObject, where Organic_Chemical occurs in NCI, but not in SUMO, whereas SelfConnectedObject occurs in SUMO, yet not in NCI.

Next, we specify syntactic restrictions that will ensure these "nice" properties of $\mathcal{T} = \mathcal{T}_{up} \cup \mathcal{T}_{dom}$. Given the examples, it seems reasonable to limit the coupling between $\mathcal{T}_{up}$ and $\mathcal{T}_{dom}$ to subsumptions relating concept (role) names in $\mathcal{T}_{dom}$ and concept (role) names occurring in $\mathcal{T}_{up}$.

Definition 2 *The pair $\Im = \langle \mathcal{T}_{up}, \mathcal{T}_{dom} \rangle$ is* f-compliant[5] *if, for $\mathbf{S} = \mathsf{Sig}(\mathcal{T}_{up}) \cap \mathsf{Sig}(\mathcal{T}_{dom})$ the shared signature, concept and role names $A, R \in \mathbf{S}$ occur in $\mathcal{T}_{dom}$ only in axioms of the form $B \sqsubseteq A$ and $S \sqsubseteq R$ respectively, where $B, S \in \mathsf{Sig}(\mathcal{T}_{dom}) \setminus \mathbf{S}$.*

[5] "f" stands for "foundational".

f-compliance suffices for capturing the coupling between the foundational and the domain ontologies in GALEN. However, is f-compliance enough to guarantee our "nice" properties for $\mathcal{T} = \mathcal{T}_{up} \cup \mathcal{T}_{dom}$? A simple example will provide a negative answer: just assume that $\mathcal{T}_{dom}$ contains a GCI of the form $\top \sqsubseteq A$; after the merge, every concept in $\mathcal{T}_{up}$ will be subsumed by $A \in \mathsf{Sig}(\mathcal{T}_{dom})$ and, thus, the foundational ontology does not act as such anymore.

As mentioned above, we use an additional safety condition—called localness—which is defined as follows [3]: if $\mathcal{I} = (\Delta^{\mathcal{I}}, .^{\mathcal{I}})$ and $\mathcal{J} = (\Delta^{\mathcal{J}}, .^{\mathcal{J}})$ are interpretations such that $\Delta^{\mathcal{J}} = \Delta^{\mathcal{I}} \cup \nabla$, where ∇ is a non-empty set disjoint with $\Delta^{\mathcal{I}}$, $A^{\mathcal{J}} = A^{\mathcal{I}}$ for each concept name, and $R^{\mathcal{J}} = R^{\mathcal{I}}$ for each role name, then $\mathcal{J}$ is called the *expansion* of $\mathcal{I}$ with ∇. Intuitively, the interpretation $\mathcal{J}$ is identical to $\mathcal{I}$ except for the fact that it contains some additional elements in the interpretation domain. These elements do not participate in the interpretation of concepts or roles. Now, *local* ontologies are precisely those whose models are closed under domain expansions, i.e., $\mathcal{T}$ is *local* if, for every $\mathcal{I} \models \mathcal{T}$ and every set ∇ disjoint with $\Delta^{\mathcal{I}}$, the expansion $\mathcal{J}$ of $\mathcal{I}$ with ∇ is a model of $\mathcal{T}$.

Intuitively, local ontologies contain only GCIs with a limited "global" effect. Examples of non-local axioms are GCIs that fix the size of the domain in every model of the ontology (e.g. $\top \sqsubseteq \mathit{bob}$ for a nominal *bob*), or GCIs that establish the existence of a "universal" named concept (e.g. $\top \sqsubseteq \mathit{Car}$). In contrast, role domain and range and concept disjointness are local. In order to show that localness is a reasonable condition to impose, we have implemented a localness checker, tested it on about 800 ontologies available on the Semantic Web, and found that less than 1% of them contain non-local axioms. In [2], we provide the proofs of our initial results, a precise syntactic characterisation of localness for $\mathcal{SHIQ}$ and further details on our experimental results.

Theorem 1 *Let $\Im = \langle \mathcal{T}_{up}, \mathcal{T}_{dom} \rangle$ be* f*-compliant. If $\mathcal{T}_{dom}$ is a local $\mathcal{SHOIQ}$ TBox, $\mathcal{T}_{up}$ is a $\mathcal{SHIQ}$ TBox (not necessarily local), and $\mathcal{T} = \mathcal{T}_{up} \cup \mathcal{T}_{dom}$ is instantiable, then*

1. *$\mathcal{T} = \mathcal{T}_{up} \cup \mathcal{T}_{dom}$ is a conservative extension of $\mathcal{T}_{up}$,*
2. *$\mathcal{T} \not\models A \sqsubseteq B$, for all concept names $A \in \mathsf{Sig}(\mathcal{T}_{up})$ and $B \in \mathsf{Sig}(\mathcal{T}_{dom}) \setminus \mathbf{S}$, and*
3. *$\mathcal{T} \not\models R \sqsubseteq S$, for all role names $R \in \mathsf{Sig}(\mathcal{T}_{up})$ and $S \in \mathsf{Sig}(\mathcal{T}_{dom}) \setminus \mathbf{S}$.*

This theorem states that our desirable properties are indeed preserved after the merge and, most importantly, the rôles of the foundational and domain ontologies are preserved (Items 2 and 3). Note, however, that f-compliance does not suffice for ensuring the instantiability of the merge: only if $\mathcal{T}$ is consistent and free from unsatisfiable names do the guarantees provided by the theorem apply. Although instantiability, as opposed to conservative extensions, can be easily checked using a reasoner, one might want to strengthen the theorem (and

the corresponding syntactic restrictions) to ensure the instantiability of $\mathcal{T}$ as well. Also, note that localness certainly is a too restrictive safety condition since it rules out "harmless" GCIs as well. The investigation of new f-compliance conditions that ensure the instantiability of the integrated ontology and of less strict safety conditions is the focus of our ongoing work.

2.2 Broadening Integration

In this scenario, an ontology $\mathcal{T}_1$ is to be integrated with another $\mathcal{T}_2$ that describes in more detail one or more of the domains that are only touched on in $\mathcal{T}_1$. For example, we may wish to integrate the Wine Ontology [9] with an ontology describing, in more detail, the regions in which wines are produced or the kinds of grapes they contain.

The Wine Ontology illustrates a common pattern: although ontologies usually refer to a *core* application domain, they also refer to other *secondary* domains that deal with different objects. This modeling paradigm is not only characteristic of small and medium sized ontologies, but also occurs in large, high-quality knowledge bases, written by groups of experts. A prominent example is the NCI Thesaurus [5], a huge ontology covering areas of basic and clinical science. The core of NCI is focused on genes; other subject matters described in the ontology include diseases, drugs, chemicals, diagnoses treatments, professional organizations, anatomy, organisms, and proteins.

In this scenario, concepts in the core application domain can be defined in terms of concepts in the secondary domains. For example, in the Wine Ontology, a Bordeaux is described as a Wine produced in France, where France is defined in the Regions ontology: $\mathsf{Bordeaux} \sqsubseteq \mathsf{Wine} \sqcap \exists \mathsf{producedIn}.\mathsf{France}$ In NCI, the gene ErbB2 is an Oncogene that is found in humans and is associated with a disease called Adrenocarcinoma.

$$\mathsf{ErbB2} \sqsubseteq \mathsf{Oncogene} \sqcap \exists \mathsf{foundIn}.\mathsf{Human} \sqcap \exists \mathsf{associatedWith}.\mathsf{Adrenocarcinoma}$$

Concepts in secondary ontologies, however, do not use the core concepts in their definitions, i.e., regions are not defined in terms of wines or diseases in terms of genes. Note, in this connection, that a 'broadening scenario' in this interpretation is closely related to the way ontologies would be integrated using the framework of $\mathcal{E}$-connections, but is rather mimicking than directly adopting the syntax and semantics of $\mathcal{E}$-connections [6].

Ontologies following this pattern can evolve by expanding their domain of discourse with knowledge about new subject matters. For example, we may extend the Wine ontology by representing the kinds of dishes each wine is most appropriate for, or NCI by adding information about useful publications on cancer research. This evolution process will typically consist of adding a new

"secondary" ontology, either developed by a group of experts, or borrowed directly from the Web. As a consequence, this ontology should be "good" as it is, and thus we want to make sure that it will not be affected by the integration, i.e., we should require $\mathcal{T} = \mathcal{T}_{core} \cup \mathcal{T}_{side}$ to be a conservative extension of $\mathcal{T}_{side}$.

Furthermore, since we assume $\mathcal{T}_{core}$ and $\mathcal{T}_{side}$ to cover different aspects of the world, we require that the merged ontology $\mathcal{T}$ does not entail subsumptions in any directions between non-shared concept names $A \in \mathsf{Sig}(\mathcal{T}_{core})$ and $B \in \mathsf{Sig}(\mathcal{T}_{side})$. This condition ensures that the ontologies actually describe different objects.

Let $\mathcal{T}_{core}$ and $\mathcal{T}_{side}$ be ontologies with signatures $\mathbf{S}_{core} = \mathbf{C}_{core} \cup \mathbf{R}_{core}$ and $\mathbf{S}_{side} = \mathbf{C}_{side} \cup \mathbf{R}_{side}$, let the shared signature $\mathbf{S} = \mathbf{S}_{core} \cap \mathbf{S}_{side}$ contain only concept names, and let $\mathbf{R}_{out} \subseteq \mathbf{R}_{core}$ be a distinguished subset of roles. Intuitively, the roles in $\mathbf{R}_{out}$ connect objects in different ontologies. Some concepts in $\mathcal{T}_{core}$ are defined in terms of restrictions on these roles; for example, the Bordeaux wines are related to France via the role producedIn and the ErbB2 oncogenes with organisms and diseases through the roles foundIn and associatedWith, respectively.

Definition 3 *The pair $\Im = \langle \mathcal{T}_{core}, \mathcal{T}_{side} \rangle$ is* b-compliant *if:* ***1)*** *$\mathbf{S} = \mathbf{S}_{core} \cap \mathbf{S}_{side} = \mathbf{C}_{core} \cap \mathbf{C}_{side}$,* $\varnothing \neq \mathbf{R}_{out} \subseteq \mathbf{R}_{core}$*;* ***2)*** *for every role inclusion axiom $R \sqsubseteq S \in \mathcal{T}_{core}$, either both $R, S \in \mathbf{R}_{out}$ or both $R, S \notin \mathbf{R}_{out}$;* ***3)*** *for every GCI $C_1 \sqsubseteq C_2 \in \mathcal{T}_{core}$, C_1, C_2 can be generated using the following grammar:*

$$C_i \leftarrow A | C \sqcap D | \neg C_i | \exists R.C_i | \exists P.A' | \geq nR.C_i | \geq nP.A'$$

where $A \in \mathbf{C}_{core} \setminus \mathbf{C}_{side}$, C, D and C_i are concepts generated using the grammar, $A' \in \mathbf{C}_{side}$, $R \notin \mathsf{RoI}(\mathbf{R}_{out})$, and $P \in \mathbf{R}_{out}$.

As a consequence, concept names in $\mathcal{T}_{side}$ can *only* be used in $\mathcal{T}_{core}$ through restrictions on the "outgoing" relations. Condition **2)** makes sure that the hierarchies for the two kinds of roles are disconnected from each other. It turns put that the Wine Ontology and the "modules" that can be extracted from NCI [3] are local and b-compliant. As in the foundational scenario, the theorem requires the instantiability (and thus the consistency) of the merged ontology $\mathcal{T}$.

Theorem 2 *Let $\Im = \langle \mathcal{T}_{core}, \mathcal{T}_{side} \rangle$ be* b*-compliant. If $\mathcal{T}_{core}$ is a local $\mathcal{SHOIQ}$ TBox, $\mathcal{T}_{side}$ is a local $\mathcal{SHIQ}$ TBox, and $\mathcal{T} = \mathcal{T}_{core} \cup \mathcal{T}_{side}$ is instantiable, then*

1. *$\mathcal{T} = \mathcal{T}_{core} \cup \mathcal{T}_{side}$ is a conservative extension of $\mathcal{T}_{side}$,*
2. *For all $A \in \mathsf{Sig}(\mathcal{T}_{core}) \setminus \mathbf{S}$ and $B \in \mathsf{Sig}(\mathcal{T}_{side})$: $\mathcal{T} \not\models A \sqsubseteq B$ and $\mathcal{T} \not\models B \sqsubseteq A$,*
3. *For all $R \in \mathsf{Sig}(\mathcal{T}_{side})$ and $S \in \mathsf{Sig}(\mathcal{T}_{core})$: $\mathcal{T} \not\models R \sqsubseteq S$ and $\mathcal{T} \not\models S \sqsubseteq R$,*
4. *For all $R \in \mathbf{R}_{out}$ and $S \in \mathbf{R}_{core} \setminus \mathbf{R}_{out}$: $\mathcal{T} \not\models R \sqsubseteq S$ and $\mathcal{T} \not\models S \sqsubseteq R$.*

3 Outlook

So far, the problem of predicting and controlling the consequences of ontology integration has been largely overlooked by the Ontology Engineering and Semantic Web communities.

In this paper, we have formalized two basic scenarios for ontology integration. In each case, we have identified a set of *semantic* properties that the integrated ontology should satisfy and, under certain simplifying assumptions, we have shown how these properties can be guaranteed by imposing certain *syntactic* constraints on the ontologies to be integrated.

So far we have been very conservative in both the (syntactic) compliance and safety conditions (localness) in the scenarios. In the future, we aim at investigating how these can be relaxed in each case without losing the nice properties of the integrated ontology. We expect that our results will constitute the basis for a normative methodology for ontology integration that is both well-founded and understandable to modelers, and that can be supported by ontology development tools.

References

[1] F. Baader, C. Lutz, H. Sturm, and F. Wolter. Fusions of Description Logics and Abstract Description Systems. *JAIR*, 16:1–58, 2002.

[2] B. Cuenca-Grau, I. Horrocks, O. Kutz, and U. Sattler. Will my Ontologies Fit Together? Technical report, University of Manchester, 2006. Available at `http://www.cs.man.ac.uk/~bcg/Integration-TR.pdf`.

[3] B. Cuenca-Grau, B. Parsia, E. Sirin, and A. Kalyanpur. Modularity and Web Ontologies. In *Proc. of KR-2006*, 2006.

[4] S. Ghilardi, C. Lutz, and F. Wolter. Did I Damage My Ontology? A Case for Conservative Extensions in Description Logics. In *Proc. of KR-2006*, 2006.

[5] J. Golbeck, G. Fragoso, F. Hartel, J. Hendler, B. Parsia, and J. Oberthaler. The National Cancer Institute's Thesaurus and Ontology. *J. of Web Semantics*, 1(1), 2003.

[6] O. Kutz, C. Lutz, F. Wolter, and M. Zakharyaschev. $\mathcal{E}$-connections of Abstract Description Systems. *Artificial Intelligence*, 1(156):1–73, 2004.

[7] N. Noy. Semantic Integration: A Survey on Ontology-based Approaches. *SIGMOD Record*, 2004.

[8] A. Rector. Modularisation of Domain Ontologies Implemented in Description Logics and Related Formalisms, including OWL. In *Proc. of FLAIRS*, 2003.

[9] M.K. Smith, C. Welty, and D.L. McGuinness. OWL Web Ontology Language Guide. *W3C Recommendation*, 2004.

Description logic reasoning using the PTTP approach

Zsolt Nagy, Gergely Lukácsy, Péter Szeredi
Budapest University of Technology and Economics
Department of Computer Science and Information Theory
{zsnagy, lukacsy, szeredi}@cs.bme.hu

Abstract

The goal of this paper is to present how the Prolog Technology Theorem Proving (PTTP) approach can be used for ABox-reasoning. This work presents an inference algorithm over the language $\mathcal{ALC}$, and evaluates its performance highlighting the advantages and drawbacks of this method.

1 Introduction and motivation

Work reported in this paper is being carried out in the SINTAGMA[1] project, which aims at the development of a knowledge management tool-set for the integration of heterogenous information sources. This is an extension of the SILK[2] [2] technology, for retrieving information spanning over several data sources stored in the model warehouse of the system. The model warehouse contains UML models as well as models given using description logics (DL) [1].

Currently we are working on extending the capabilities of the SINTAGMA tool-set by designing and implementing description logic inference methods used for querying information sources containing large amounts of data. The first step of this research process resulted in a resolution-based transformation of ABox-reasoning problems to Prolog [10]. This algorithm is able to answer *instance-check* and *instance-retrieval* queries over the DL language $\mathcal{ALC}$ and an empty TBox. In this paper, we examine how ABox-reasoning services can be provided with respect to a *non-empty* TBox using Prolog technology.

This paper is structured as follows: Section 2 presents related work on ABox-inference in description logics. Section 3 details the method how ABox-reasoning

[1] Semantic INtegration Technology Applied in Grid-based Model-driven Architectures
[2] Semantic Integration via Logic and Knowledge

is performed in our framework. Section 4 evaluates the performance of our technique, highlighting its strong points and weaknesses. Finally, Section 5 concludes this work and addresses future research challenges.

2 Related Work

Traditional tableau-based Description Logic reasoners such RACER are slow when performing ABox-reasoning on large amounts of instances [4]. The work [7] describes a resolution-based inference algorithm, which is not as sensitive to the increase of the ABox size as a tableau-based method. The system KAON2 [8] implements this method and provides reasoning services over the description logic language $\mathcal{SHIQ}$.

The work [5] discusses how a first order theorem prover such as VAMPIRE can be modified and optimized for reasoning over description logic knowledge bases.

Paper [3] describes a direct transformation of $\mathcal{ALC}$ description logic axioms to Horn-clauses. Although [3] restricts the expressive power of $\mathcal{ALC}$ by disallowing constructs that result in non-Horn clauses, the main advantage of the approach is that the transformed clauses can be supplemented with other non-DL Horn-clauses.

Both the modified VAMPIRE and KAON2 aim to provide inference services over knowledge bases defined using the expressive power of the DL language in question plus the expressive power of a restricted fragment of first order logic. In case of KAON2, the restrictions involving FOL components are such that the inference algorithm remains to be a decision procedure. It has been proved in [9] that query-answering over a knowledge base containing $\mathcal{SHOIN}$ axioms and so-called DL-safe rules is decidable. However in practice, this decision procedure has been shown to be highly inefficient due to don't know nondeterminism (backtrack search). For efficiency reasons, only a subset of the $\mathcal{SHOIN}$ description logic language is used in KAON2.

In our previous work [10], we have provided a possible resolution-based alternative to ABox reasoning over the language $\mathcal{ALC}$ with respect to an empty TBox. In that approach, a query-plan was derived before the first ABox-access and the query-plan was executed using Prolog. This solution could be viewed as a two-phase proof of an ABox-query: first, the ABox-independent part of the proof is constructed, resulting in the query-plan as a Prolog program, and second, the query-plan is executed on the ABox.

Our current work deals with $\mathcal{ALC}$ ABox-reasoning in the presence of non-empty TBoxes, with some restrictions on the form of the TBox-axioms. However, we do allow full negation and disjunction on each side of the DL axioms, generalizing the transformation of [3]. In contrast with the earlier approach, *we delegate the whole reasoning process to Prolog*, building on the *Prolog Technology*

Theorem Proving (PTTP) approach [13].

Paper [6] introduces a fragment of the $\mathcal{SHIQ}$ language that can be transformed into Horn-clauses. The Horn-$\mathcal{SHIQ}$ language presented there allows more concept constructors than our restricted $\mathcal{ALC}$ framework. On the other hand, our approach poses less restrictions on use of disjunctions.

3 Transforming DL axioms to Prolog clauses

In this section, we describe the transformation of $\mathcal{ALC}$ TBox-axioms into executable Horn-clauses. The aim of this transformation is to provide *concept-instance check* and *concept-instance retrieval* services over an $\mathcal{ALC}$ knowledge-base containing both ABox- and TBox-axioms.

In order to avoid the appearance of Skolem-functions in the transformed Horn-clauses, some restrictions are posed on the DL-axioms. We exclude subsumption axioms $C \sqsubseteq D$ where $\forall R.E$ is a subconcept of the negational normal form of C or $\exists R.E$ is a subconcept of the negational normal form of D. Although the method described below can cope with ABox-inference on some TBoxes containing such axioms, allowing them in general may lead to non-termination when transformed to Prolog. The problem of termination for transformed clauses containing Skolem-functions may be addressed by reverting to a two-phase transformation process (as in [10]) and using ordered resolution [7] in the first phase, or by applying a proper meta-level cycle-detection technique.

The goal of this section is to show that an arbitrary DL knowledge-base obeying the above restriction can be transformed into a set of executable Prolog-clauses. There are three types of clauses: the TBox-clauses, ABox-facts and the clauses belonging to the instance-check and instance-retrieval queries. Currently, we use an interpreter written in Prolog for executing these Prolog clauses, in order to ease the development process and experimentation. However, it is fairly easy to transform these clauses further to code directly executable on a Prolog system.

Description of the transformation. Let an ABox $\mathcal{A}$ and a TBox $\mathcal{T}$ be given, where $\mathcal{T}$ consists of axioms $C \sqsubseteq D$, where C and D are in negational normal form, and C does not contain subconcepts of form $\forall R.E$, while D does not have a subconcept $\exists R.E$. The axioms of the TBox are transformed into Prolog clauses using the transformation steps below.

1. Based on the well-known mapping described e.g. in [1], we transform the description logic axioms into first order logic formulas.

2. The formulas corresponding to the TBox-axioms are then transformed into clausal form [12]. According to the properties of clause transformation, the generated clauses have the following properties:

- clauses are disjunctions of possibly negated literals;
- all variables in the clauses are universally quantified.

Due to our restrictions posed, no Skolem functions appear in the clausal form of the concepts. The general form of a transformed TBox-clause is thus the following:

$$\bigvee_m C_m(x_{i_m}) \vee \bigvee_n \neg D_n(x_{j_n}) \vee \bigvee_p \neg R_p(x_{k_p}, x_{l_p}), \tag{1}$$

where the literals C_m and D_n correspond to atomic concepts, and literals R_p correspond to role names, while x-es denote variables. Note that while both positive and negative unary literals can appear in the clauses, binary literals are only negative. Positive binary literals do not appear in any clause, since this would correspond to a role negation in the corresponding DL axiom.

3. Each TBox-clause

$$L_1 \vee L_2 \vee \ldots \vee L_n \tag{2}$$

is transformed into n clauses of the following form $(i = 1, \ldots, n)$

$$L_i \leftarrow L_1^{neg} \wedge L_2^{neg} \wedge \ldots \wedge L_{i-1}^{neg} \wedge L_{i+1}^{neg} \wedge \ldots \wedge L_n^{neg}, \tag{3}$$

where L_i is the head of the clause, and all other literals are body literals. L^{neg} is equal to $\neg L$ if L is a positive literal, and $L^{neg} = T$ if $L = \neg T$. A clause of form (3) is called a *contrapositive* of the clause (2).

Execution in Prolog. The resulting contrapositives are then transformed to Prolog syntax and are executed by our interpreter. The transformation and interpretation techniques as well as the usage of contrapositives have been borrowed from PTTP (Prolog Technology Theorem Prover) [13]. We briefly describe how we handle these issues:

- *Occurence of positive and negative literals*: To transform an arbitrary clause to Prolog format we introduce new predicate names. For each concept-name C (i.e. one of C_m or D_n of Formula (1)) we add a new predicate name $nonC$, and replace all occurrences of $\neg C(X)$ by $nonC(X)$ both in the head and in the body. The link between the separate predicates C and $nonC$ is created by ancestor resolution, see below.

- *Ancestor resolution*: *open* predicate calls[3] are collected in an *ancestor list*. If the ancestor list contains a literal which can be unified with the first

[3]I.e. calls which were entered or re-entered, but have not been exited yet, according to the Procedure-Box model of Prolog execution. [11]

literal of the goal, then the call corresponding to the goal literal succeeds. Program execution is divided into two branches: one reflecting the modifications caused by the ancestor resolution step, and one without using ancestor resolution.

- *Loop elimination*: if the first literal of the goal can be found in the ancestor list, we stop the given branch with a failure. The term 'can be found' is interpreted by the `==` Prolog built-in predicate, which succeeds if its operands are identical.

Prolog execution uses SLD-resolution [11], which is a linear resolution strategy always resolving the first literal of the *goal* with the head of a corresponding definite-clause in the Prolog-program. When all n contrapositives of a clause (2) are available, the goal can be resolved with *any* literal L_i of the clause. Thus, any linear refutation can be simulated in Prolog at the expense of introducing multiple variants of the clauses.

Soundness, completeness, termination. Soundness and completeness of this approach is based on the properties of the PTTP technique [13], since PTTP is a sound and complete first order theorem prover. A DL ABox-inference problem is handled with these theorem-proving techniques by resolving the set of produced TBox-, ABox- and query-clauses in a PTTP framework using Prolog.

Regarding termination, consider the following example: if we transform the axiom $C \sqsubseteq \forall R.C$[4] into clausal form, we get the clause $\neg C(x) \vee \neg R(x, y) \vee C(y)$. Executing an instance-retrieval query on concept C may lead to an infinite loop if the literals in the contrapositives are not ordered properly. Whenever we call the contrapositive

```
C(Y) :- C(X), R(X, Y).
```

we introduce a new variable inside the literal `C`, which is not detected by the loop elimination technique. This example shows that without further provisions, the termination of the Prolog program is not guaranteed.

Conjecture: *The execution of the resulting clauses of the above described transformation always terminates if all role literals in the body of the clauses are moved before concept literals.* This rearrangement ensures that at the time of a concept-predicate call the variable in the call either (a) occurs in a role literal previously called or (b) is equal to the head-argument of the clause. Role literals can only be resolved with ABox-facts, so all variables occurring in a role literal are unified with ABox-instances. In case (a), the variable in the concept-predicate call is instantiated due to the previous role-predicate call. In case (b), the head-argument of the clause is either instantiated, or is identical to an

[4]Suppose that C is an atomic concept.

older head argument. By induction, this means that any uninstantiated concept argument must be identical to the parameter of the instance-retrieval query. The number of clauses and instances is finite and only one variable, namely the parameter of the query may occur in a predicate call. Therefore, for a set of clauses, only a finite number of different predicate calls is possible. Since loop elimination ensures that the same predicate call never executes twice, execution always terminates.

Optimizations. Note that not all contrapositives of the clauses are needed for acquiring a complete decision procedure. Contrapositives containing a role literal in the head can be omitted, since they cannot be called within the execution of an instance-check or an instance-retrieval query. Although these clauses are never called by the program, the size of the program gets smaller, reducing administration overhead.

It is often the case that a concept predicate is called with a ground argument. In the presence of disjunctions, such a Prolog goal could be executed in multiple ways. Obviously, once the goal exited successfully, there is no point in exploring alternative branches within this goal. Therefore we modified the interpreter to perform a Prolog cut operation (`!`) after a successful exit from a ground goal. This optimization resulted in a measurable performance boost.

Composite queries. The outcome of the transformation is a set of Prolog-clauses, which are usable for instance-check or instance-retrieval queries on possibly negated atomic concepts. For allowing ABox-inference queries containing a composite query-concept Q, an atomic concept-name A has to be introduced for the query-concept with the axiom $A \equiv Q$. The equivalence-axiom $A \equiv Q$ can be written in form $A \sqsubseteq Q$ and $Q \sqsubseteq A$. From these two subsumption axioms, only the second one has to be added to the TBox for answering the ABox query. This axiom is transformed to Prolog clauses in the same way as the other axioms in the TBox.

Note that directly transforming the composite query-concept Q to a Prolog query is not sufficient, because all the contrapositives of the query-clause $Q \sqsubseteq A$ may be needed for handling case-analysis. This technique is an alternative for passing the parameter of the instance-check and the instance-retrieval queries when handling case-analysis, as detailed in [10].

4 Performance evaluation

We have carried out the preliminary performance-evaluation of our approach and compared it with the available state-of-the-art description logic reasoners, such as RACER and KAON2. Since the capabilities and usage of these systems

are different, the goal of this comparison is to analyze the behavior of these approaches for different reasoning tasks and not to declare a winner among these systems. It is important to note that these inference engines provide reasoning services over different description logic languages, so the comparison has to be evaluated with care.

The tests were run on an AMD Athlon CPU running at 1.81GHz with 1GB RAM and Windows XP operating system with Service Pack 2. After a test had been performed, RACER[5], KAON2[6] and our Prolog interpreter[7] were always reinitialized.

We show the results for two instance-retrieval test cases. The first test (the first five-line block of Table 1) summarizes the results of a test-case over a TBox containing TBox-axioms with no disjunctions. The second test was run on a TBox containing complex TBox-axioms with disjunctions and composite concepts. The last line belongs to the second test case indicating a special case explained below. The ABox of the tests was randomly generated using a Prolog program.

KB characteristics	# instances	#role assertions	#concept assertions	RACER	KAON2	SICSTUS
simple	500	1269	337	1.1s	0.203s	0.000s
TBox	1000	2499	688	3.1s	0.531s	0.000s
axioms	2000	2133	1333	13.2s	0.515s	0.016s
	5000	12599	3300	91.4s	0.468s	0.016s
	10000	25036	6627	>300s	0.891s	0.031s
complex	500	261	323	0.0s	0.078s	0.890s
TBox	1000	503	635	0.7s	0.375s	1.953s
axioms	2000	972	1392	1.5s	1.062s	7.875s
	5000	2493	3367	7.1s	5.156s	44.265s
Special complex	5000	2493	3367	7.0s	3.579s	0.813s

Table 1: Test results

The first block shows that the resolution-based approaches are not as sensitive to the increase of the size of the ABox as the tableau-based RACER, when not many disjunctions are present[8]. When comparing execution times of KAON2 and our interpreter, one should note that KAON2 provides inference services over a description logic language with higher expressive power.

[5]version 1.7.6

[6]version released on 8th December 2005

[7]running under SICStus Prolog version 3.12.2

[8]Note that in special cases as shown in [10], KAON2 fails to scale.

The second test case highlights that our interpreter does not handle disjunctions efficiently enough, due to the many introduced contrapositives that often make the program explore irrelevant parts of the search space. For instance, if we know that some TBox-axioms are not needed for answering a simple query, the clauses belonging to the axioms not needed can be left out, reducing execution time below one second for even the ABox with size 5000. This phenomenon is described as a special test case in the last row of Table 1. Here the axioms not needed for answering the instance-retrieval query are omitted from the TBox. Note that for TBoxes containing less disjunctions, the execution time was only reduced by a small constant factor when omitting irrelevant axioms. KAON2 seems to handle disjunctions fairly well. Although RACER was faster than KAON2 for the first test case, execution times increased with the overhead of examining all the instances of larger ABoxes.

A possible reason why our approach does not handle disjunctions well enough is that – as opposed to our previous work [10] – no preprocessing is made on the transformed PTTP-clauses. We believe that the current approach can be optimized by cutting the search space at compile time.

5 Conclusion and future work

This paper has described an ABox-reasoning technique for the DL language $\mathcal{ALC}$ which is able to handle TBoxes obeying certain restrictions. We have written an interpreter which executes the resolution-based proof belonging to instance-check and instance-retrieval queries and have also evaluated the performance of this approach, pointing out its advantages and disadvantages.

In the future, our plan is to examine how the services provided by the interpreter can be included in the Prolog-program serving as the query-plan. Our other aim is to apply optimizations guiding the Prolog execution of the refutation. By combining this approach with the approach seen in our previous work, we believe that the performance of this reasoning technique can be largely enhanced. A method handling all forms of refutation involving Skolem-functions and DL-constructs not addressed in this paper is also subject of future research.

References

[1] F. Baader, D. Calvanese, D. McGuinness, D. Nardi, and P. F. Patel-Schneider, editors. *The Description Logic Handbook: Theory, Implementation and Applications*. Cambridge University Press, 2003.

[2] Tamás Benkő, Gergely Lukácsy, Attila Fokt, Péter Szeredi, Imre Kilián, and Péter Krauth. Information integration through reasoning on meta-data. In *Proceedings of the Workshop "AI Moves*

to IA", IJCAI'03, Acapulco, Mexico, pages 65–77, August 2003. `http://www.dimi.uniud.it/workshop/ai2ia/cameraready/benko.pdf`.

[3] Benjamin N. Grosof, Ian Horrocks, Raphael Volz, and Stefan Decker. Description Logic Programs: Combining Logic Programs with Description Logics. In *Proc. of the Twelfth International World Wide Web Conference (WWW 2003)*, pages 48–57. ACM, 2003.

[4] V. Haarslev and R. Möller. Optimization techniques for retrieving resources described in OWL/RDF documents: First results. In *Ninth International Conference on the Principles of Knowledge Representation and Reasoning, KR 2004, Whistler, BC, Canada, June 2-5*, pages 163–173, 2004.

[5] Ian Horrocks and Andrei Voronkov. Reasoning support for expressive ontology languages using a theorem prover. In *FoIKS*, volume 3861 of *Lecture Notes in Computer Science*, pages 201–218. Springer, 2006.

[6] Ullrich Hustadt, Boris Motik, and Ulrike Sattler. Data complexity of reasoning in very expressive description logics. In *Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence (IJCAI 2005)*, pages 466–471. International Joint Conferences on Artificial Intelligence, 2005.

[7] U. Hustadt, B. Motik, and U. Sattler. Reasoning for description logics around $\mathcal{SHIQ}$ in a resolution framework. Technical Report 3-8-04/04, FZI, Karlsruhe, Germany, June 2004.

[8] KAON2: Ontology management tool for the semantic web. `http://kaon2.semanticweb.org/`.

[9] Boris Motik, Ulrike Sattler, and Rudi Studer. Query answering for OWL-DL with rules. *Journal of Web Semantics: Science, Services and Agents on the World Wide Web*, 3(1):41–60, July 2005.

[10] Zsolt Nagy, Gergely Lukácsy, and Péter Szeredi. Translating description logic queries to Prolog. In *PADL*, volume 3819 of *Lecture Notes in Computer Science*, pages 168–182, 2006.

[11] U. Nilsson and J. Maluszynski, editors. *Logic, Programming and Prolog*. John Wiley and Sons Ltd., 1990.

[12] S.J. Russel and P. Norvig, editors. *Artificial intelligence: a modern approach*. Prentice Hall, 1994.

[13] M. Stickel. A Prolog technology theorem prover: A new exposition and implementation in Prolog, Technical Note 464, SRI international, 1989.

Extending the $\mathcal{SHOIQ}(D)$ Tableaux with DL-safe Rules: First Results

Vladimir Kolovski and Bijan Parsia and Evren Sirin
University of Maryland
College Park, MD 20740
kolovski@cs.umd.edu
bparsia@isr.umd.edu
evren@cs.umd.edu

Abstract

On the Semantic Web, there has been increasing demand for a rules-like expressivity that goes beyond OWL-DL. Efforts of combining rules languages and description logics usually produce undecidable formalisms, unless constrained in a specific way. We look at one of the most expressive - but decidable - such formalisms proposed: DL-safe rules, and present a tableaux-based algorithm for query answering in OWL-DL augmented with a DL-safe rules component. In this paper we present our algorithm which is an extension of the $\mathcal{SHOIQ}$ tableaux algorithm and show the preliminary empirical results. Knowing that we pay a high price in performance for this expressivity, we also explore possible optimizations.

1 Introduction

Even though OWL-DL [3] has been a great success as a language for ontology representation on the Semantic Web, there is already demand for even greater expressivity in the form of logic programming (LP) rules. Community interest in this area is growing, as evidenced by the recent (November 2005) creation of the Rule Interchange Format Working Group (RIF [2]). One approach to provide the needed additional expressivity is to couple in some way Description Logic with LP rule systems (the hybrid approach).

However, combining expressive Description Logics with rules is non-trivial since it can easily lead to undecidability if the integration is not restricted in some manner. There have been a number of proposals that follow the hybrid approach

while still retaining decidability [12, 4, 9]. The logic of DL-safe rules [12] is one of the most expressive of these, combining $\mathcal{SHIQ}(\mathcal{D})$ and positive datalog rules. DL-safe rules allow classes and properties from the Description Logic component to occur freely in the head or the body of the rule, with the only restriction being that they can be applied only to explicitly named individuals. The same authors who coined the term also proposed the only practical reasoning algorithm for this formalism [11]. The algorithm is based on reducing the description logic knowledge base to a positive disjunctive datalog program, appending the DL-safe rules to this program and using optimized deductive database techniques for query answering.

As an alternative to a reduction to disjunctive datalog, in this paper we present a direct tableaux algorithm for DL-safe rules, as an extension to the $\mathcal{SHOIQ}$ tableaux algorithm. Instead of converting the Description Logic knowledge base to a datalog program, we extend a highly optimized OWL-DL tableaux reasoner to handle these rules. We believe that we will benefit from layering our algorithm on top of an efficient tableaux reasoner and our initial empirical results support this claim. Also, this approach allows our algorithm to cover the full expressivity of OWL-DL ($\mathcal{SHOIN}(\mathcal{D})$) and DL-safe rules.

In the following section, we provide preliminaries and definitions of terms used later in this paper. Then, after presenting our algorithm, we discuss its performance in the preliminary tests, and possible optimizations and future work.

2 Background

In this section we provide a brief overview of the terms and notation used later in the paper. We assume that the reader is familiar with the syntax and semantics of $\mathcal{SHOIN}(D)$. Before introducing DL-safe rules, we need to describe the standard notation we will be using for rules. A *term* is either a constant (denoted by *a, b, c*) or a variable (denoted by *x, y, z*). An *atom* has the form $P(s_1, ..., s_n)$, where P is a predicate symbol and s_i are terms. A literal is an atom or a negated atom. A *rule* has the form

$$H \leftarrow B_1, \ldots, B_n$$

where H and B_i are atoms; H is called the rule *head*, and the set of all B_i is called the rule *body*. A *program* P is a finite set of rules. As for the semantics, we treat rules as logical implications, thus they can also be represented as:

$$H \vee \neg B_1 \vee \neg B_2 \vee \ldots \neg B_n$$

Definition 1 *(Semantics of a DL-safe Knowledge Base)*

We start with a $\mathcal{SHOIN}(\mathbf{D})$ knowledge base, $\mathcal{K}$. Let V_C, V_{IP}, V_{DP} be countable and pair-wise disjoint sets of class, object property and datatype property names respectively. Let V_P be a set of predicate symbols that can be used as atoms in the rules, such that $V_C \cup V_{IP} \cup V_{DP} \subseteq V_P$. A DL-atom is an atom of the form $A(s)$ were A belongs to V_C, or of the form $R(s,t)$ where R belongs to $V_{IP} \cup V_{DP}$. A rule r is called DL-safe if each variable in r occurs in a non-DL-atom in the rule body. A program is DL-safe iff all of its rules are DL-safe.

Let $\mathcal{K}$ be a $\mathcal{SHOIN}(\mathbf{D})$ knowledge base and P a set of DL-safe rules. A ***DL-safe Knowledge Base*** *is a pair $(\mathcal{K}, P)$. Since $\mathcal{SHOIN}(\mathbf{D})$ is a subset of first order logic, we can give the semantics of $(\mathcal{K}, P)$ by $\pi(\mathcal{K}) \cup P$ where $\pi(\mathcal{K})$ is a translation mapping $\mathcal{SHOIN}(\mathbf{D})$ to first order logic.*

3 Tableaux Algorithm

In this section we present a tableaux-based algorithm for deciding the satisfiability of DL-safe knowledge bases. Our algorithm is inspired by the decidability proof for query answering in combined $\mathcal{SHOIN}(D)$ and DL-safe rules [12]. The main difference is that we push the rule reasoning inside the tableaux algorithm in order to get some goal directed behavior. Our approach extends the $\mathcal{SHOIQ}$ algorithm [7] thus we are able to handle the full expressivity of OWL-DL and DL-safe rules. We assume that the reader is familiar with the basics of the $\mathcal{SHOIQ}$ algorithm (detailed description is available in [7]).

The main reasoning service that our algorithm provides is query answering. More specifically, answering for a given query α and a DL-safe knowledge base $KB = (\mathcal{K}, P)$ whether $KB \models \alpha$. This can be reduced to a consistency check of $KB \cup \{\neg\alpha\}$, so $KB \models \alpha$ iff $KB \cup \{\neg\alpha\}$ is not consistent.

To check whether a DL-safe KB $(\mathcal{K}, P)$ is consistent, we need to perform a consistency check of its ABox with respect to the TBox, the RBox and the rules. Since it was shown in [7] that we can reduce reasoning w.r.t. general TBoxes and role hierarchies to reasoning w.r.t. role hierarchies only, from now on without loss of generality we will assume that we have internalized the TBox and we do reasoning w.r.t. to the RBox and the rules program.

Since ABoxes in general involve multiple individuals with arbitrary role relationships between them, the completion algorithm will work on a forest instead of a tree. Such a forest is a collection of trees whose root nodes correspond to the individuals present in the input ABox. The individuals in the input ABox are the explicitly asserted individuals, and are the *only* individuals to which the rules in P are applicable. A formal description of a completion forest is given in the following definition.

Definition 2 *($\mathcal{SHOIQ}$ + DL-safe Completion Forest) (extended version of [8]) A completion forest $\mathcal{F}$ for a $\mathcal{SHOIQ}$ Abox $\mathcal{A}$ w.r.t a role hierarchy $\mathcal{R}$ and a*

DL-safe KB $(\mathcal{K}, P)$ *is a collection of trees whose distinguished root nodes may be connected by edges in an arbitrary way. The completion forest can be described as a directed graph* $\mathcal{G} = (\mathcal{V}, \mathcal{L}, \mathcal{E}, P, \neq)$ *where each individual* $x \in \mathcal{V}$ *can be labeled with a set of concept labels* $\mathcal{L}(x)$ *and each edge* (x, y) *can be labeled with a set of role names* $\mathcal{E}(x, y)$*. P is a program consisting of DL-safe rules* R_i *,each one of the form*

$$H_i \vee \neg B_{i1} \vee \neg B_{i2} \vee \ldots \neg B_{in}$$

We also keep track of inequalities between nodes of the graph with a symmetric binary relation $\neq$ *between the nodes of* $\mathcal{G}$*. Given a* $\mathcal{SHOIQ}$ *Abox* $\mathcal{A}$*, a role hierarchy* $\mathcal{R}$ *and a DL-safe program P, the algorithm initializes a completion forest* $\mathcal{F}$ *that contains a root node* x_0^i *for each individual* $a_i \in \mathcal{A}$*, and an edge* (x_0^i, x_0^j) *if* $\mathcal{A}$ *contains an assertion* $(a_i, a_j) : R$ *for a role R. The labels of these nodes and edges are initialized as follows:*

$$\mathcal{L}(x_0^i) = \{C | a_i : C \in \mathcal{A}\}$$
$$\mathcal{L}((x_0^i, x_0^j)) = \{R | (a_i, a_j) : R \in \mathcal{A}\}$$
$$x_0^i \neq x_0^j \text{ iff } a_i \neq a_j \in \mathcal{A}$$

For the purpose of our paper, we assume standard blocking and clash definitions [7].

The tableaux algorithm starts with the completion forest $\mathcal{F}$*. It applies the completion rules and stops when a clash occurs. The ABox* $\mathcal{A}$ *is consistent w.r.t. to* $\mathcal{R}$ *and P iff the completion rules can be applied in a way that they yield a complete and clash-free completion forest, and inconsistent otherwise.*

Our extension of the algorithm is in the form of a new completion rule, R-rule [1]. We reuse all of the original $\mathcal{SHOIQ}$ completion rules, and add the R-rule. The R-rule starts with generating all of the possible ground instantiations for a DL-safe rule (we will refer to them as *bindings* henceforth). We need to be careful with the bindings because only named individuals **not** introduced by the tableaux expansion rules are considered. For every such binding, we check whether at least one of the ground literals holds true. If that is not the case, we have found a clash and the DL-safe knowledge base is inconsistent. Thus, for every literal in the rule we create a disjunction branch, in which we assert that the literal holds (in a manner explained in Figure 1) - this branch is then explored further. This is the most naive version of the algorithm, and optimizations are discussed in the next sections.

Please note here that for brevity, we do not include all of the $\mathcal{SHOIQ}$ tableaux completion rules (they are defined in [7]). We only present our new expansion rule which is to be applied with lowest priority.

[1] Thanks to the anonymous reviewer for the suggestion

$\rightarrow R$ **rule**:
foreach DNF clause $R_i \in P$:
foreach ground substitution to R_i *that produces a ground clause* G :

foreach monadic ground literal $B(a) \in G$:
if $B \notin \mathcal{L}(a)$, *then create a new branch where* $\mathcal{L}(a) = \mathcal{L}(a) \cup \{B\}$

foreach dyadic ground literal $B(a, b) \in G$:
if B *is of type* $\neg A(a, b)$ *and* $\forall A.\neg\{b\} \notin \mathcal{L}(a)$, *then*
create a new branch where $\mathcal{L}(a) = \mathcal{L}(a) \cup \{\forall A.\neg\{b\}\}$

if B *is of type* $A(a, b)$ *and* $(a, b) \notin \mathcal{G}$ *then*
create a new branch with $\mathcal{G} = \mathcal{G} \cup \{(a, b)\}$ *and* $\mathcal{L}(a, b) = \{A\}$

if B *is of type* $A(a, b)$, $(a, b) \in \mathcal{G}$ *and* $A \notin \mathcal{L}(a, b)$ *then*
create a new branch where $\mathcal{L}(a, b) = \mathcal{L}(a, b) \cup \{A\}$

Figure 1: New Expansion Rule

4 Implementation and Optimizations

We implemented a proof of concept by extending the open source reasoner Pellet [1]. Obviously, the bottleneck of the algorithm is the non-determinism introduced by creating and exploring tableaux branches for every binding. Thus, as an easy optimization, we check whether, for a particular binding, the rule ground clause is trivially satisfied (one of the disjuncts occurs in the tableaux). If this is the case, we do not generate any new branches for that clause in the tableaux.

More advanced, yet practical, optimizations are possible. For example, we do not need to generate all possible groundings of the rules. Thus, we use a fix-point evaluation procedure as an optimization for our algorithm. By running that procedure first, we make sure that all of the obvious rules are fired - by obvious we mean those rules whose patterns we can match syntactically against the individuals in the completion forest. We use the efficent pattern matching RETE [5] algorithm for this purpose. After the fixpoint is reached, we continue with our tableaux-based algorithm. The speed up comes from the fact that for a given grounding for a rule, if the rule was already fired in the pattern-matching phase, now we do not have to try each disjunct of its horn clause separately, thus decreasing non-determinism.

Another optimization would be to sort the rules after the fix point optimization and before running our algorithm. For a given rule r:

$$H \leftarrow B_1, \ldots, B_n$$

we would like to try those groundings that satisfy the maximum amount of

conditions B_i first. By doing this we will succeed in firing the 'almost obvious' applications of the rules first, and add more information in the concept labels of the tableaux. In general, the more information we have in the concept labels, the higher the chances to reach a clash when grounding the horn clauses of the rules, yielding less non-determinism in the algorithm.

5 Evaluation

We conducted an experimental study to compare the performance of our approach with KAON2 [10]. Our test case of choice was a modified version of the LUBM benchmark [6], with one university and increasing ABox sizes. There are 46 defined classes and 30 object properties in the ontologies. We extended this ontology by adding a rules component consisting of 2 rules:

$GraduateStudent(X)$:-
$Person(X), takesCourse(X, Y), GraduateCourse(Y)$.

$SpecialCourse(Z)$:-
$FullProfessor(X), headOf(X, Y), teacherOf(X, Z)$.

We ran the experiments on an IBM ThinkPad T42 with Pentium Centrino 1.6GHz processor and 1.5GB of memory. The performance results (time in milliseconds) for the consistency check of the ontologies are shown in table 1. *Rule Branch* stands for the number of additional branches introduced by the R-rule.

No	Size of ontology	Pellet DL-Safe		KAON2
		Rule Branches	Consistency Check Time	Consistency Check Time
1	LUBM, 1 rule, consistent, 17 individuals	34	100	661
2	LUBM, 1 rule, consistent, 94 individuals	107	250	621
3	LUBM, 1 rule, inconsistent, 94 individuals	120	381	651
4	LUBM, 2 rules, consistent, 17 individuals	109	421	701
5	LUBM, 2 rules, inconsistent, 94 individuals	422	951	691
6	LUBM, 2 rules, inconsistent, 94 individuals	0	90	691
7	LUBM, 2 rules, consistent, 94 individuals	422	1232	5748

Table 1: Evaluation Results for Consistency Checking. Rule Branches stands for the total number of branches introduced by the R-rule

Please note that the first rule has 2 variables, and the second has 3. The additional variable in the second rule impacts the performance, since there are considerably more bindings possible for that rule. As an example of the exponential blow up of the bindings, consider that in the case of LUBM with two rules and 94 individuals, in the naive implementation there were 804264 bindings to try. This observation lead us to another optimization strategy - prune

the bindings search space. To accomplish this, we embed the trivial satisfiability check in the generation of the bindings. This strategy, along with the implementation of the RETE algorithm as a preprocessor, improved the performance considerably.

We present our results compared to KAON2 in Table 1. As it can be seen, for smaller cases we perform favorably, however the faster rate of increase of consistency check time on our part suggests that for larger ontologies we will be slower than KAON2. The last two test cases merit more discussion. In case of 6), after the RETE algorithm fired the obvious rules we got a clash early and avoided generation of groundings. In the lasr test case, we added another class , `OverachievingProfessor` which is equivalent to a `FullProfessor` who teaches at least 9 courses. This was to demonstrate how we perform better in presence of cardinality constraints.

6 Related Work

The most relevant related work to ours is the only other practical reasoning algorithm that processes DL-Safe KBs . The algorithm is described in [11] (implemented in [10]) and is based on the $\mathcal{SHIQ}(D)$ logic and having restricted the DL-atoms in rules to concepts and simple roles. The algorithm is based on reducing the description logic knowledge base to a positive disjunctive datalog program which entails the same set of ground facts as the original knowledge base. As shown in Section 5, our algorithm compares reasonably well for smaller ontologies and for cases when cardinality constraints are used.

7 Conclusions and Future Work

This paper makes the following contributions:

- Provides a direct tableaux procedure for the combination of OWL-DL and DL-safe rules.

- Provides preliminary empirical results of an implementation and discussion of possible optimizations.

As for our next steps, we noticed that a the reordering of the literals in the rule makes a noticeable difference in the running time, so we will try to work out the optimum orderings. Also, in longer term, we plan to work on evolving our algorithm toward SWRL by integrating a first order Free Variable tableaux procedure with the $\mathcal{SHOIQ}$ algorithm.

References

[1] Pellet - OWL DL Reasoner. http://www.mindswap.org/2003/pellet.

[2] Rule interchange format working group, 2005. http://www.w3.org/2005/rules/wg.

[3] M. Dean, D. Connolly, F. van Harmelen, J. Hendler, I. Horrocks, D. L. McGuinness, P. F. Patel-Schneider, and L. A. Stein. Web Ontology Language (OWL) Reference Version 1.0. W3C Working Draft 12 November 2002 http://www.w3.org/TR/2002/WD-owl-ref-20021112/.

[4] F. M. Donini, M. Lenzerini, D. Nardi, and A. Schaerf. Al-log: integrating datalog and description logics. *Journal of Intelligent Information Systems*, 10:227–252, 1998.

[5] C. Forgy. Rete: A fast algorithm for the many pattern/many object pattern match problem. *Artificial Intelligence*, 19:17–37, 1982.

[6] J. Heflin, Z. Pan, and Y. Guo. The lehigh university benchmark LUBM. http://swat.cse.lehigh.edu/projects/lubm/, 2003.

[7] I. Horrocks and U. Sattler. A tableaux decision procedure for shoiq. In *Proc. of IJCAI 2005*, pages 448 – 453.

[8] I. Horrocks, U. Sattler, and S. Tobies. Reasoning with individuals for the description logic shiq. In D. MacAllester, editor, *Proc. of the CADE 2000*, number 1831, pages 482–496. Springer-Verlag, 2000.

[9] A. Levy and M.-C. Rousset. CARIN: A representation language combining horn rules and description logics. *Artificial Intelligence*, 104(1-2):165–209, 1998.

[10] B. Motik. KAON2 - ontology management for the semantic web. http://kaon2.semanticweb.org/, 2005.

[11] B. Motik. *Reasoning in Description Logics using Resolution and Deductive Databases*. PhD thesis, Univesitt Karlsruhe, Germany, January 2006.

[12] B. Motik, U. Sattler, and R. Studer. Query answering for owl-dl with rules. In *Proc. of ISWC 2004*, pages 549–563.

Description Logic Reasoning for Dynamic ABoxes

Christian Halaschek-Wiener, Bijan Parsia, Evren Sirin,
and Adtiya Kalyanpur
Maryland Information and Network Dynamics Laboratory
8400 Baltimore Ave., College Park, MD, 20740 USA
{halasche,evren,aditya}@cs.umd.edu,bparsia@isr.umd.edu

1 Introduction

Recently, there has been interest in providing formal representation of Web content, which can then be processed using automated reasoning techniques. Due to data sources that produce fluctuating data, there exists a variety of description logic reasoning use cases which require frequent updates at the assertional level. These include prominent web services frameworks (e.g., OWL-S) that use description logics for service discovery and matchmaking, where devices register or deregister their descriptions (and supporting ontologies) quite rapidly. Additionally, Semantic Web portals often allow content authors to modify or extend the ontologies which organize their site structure or page content. Lastly, one of the common uses of description logic reasoners is in ontology editors. Most editors do not do continuous reasoning while one is editing, relying on an analogue of the *edit-compile-test* loop of most programming environments.

While there exists such use cases for reasoning under update, today's description logic reasoning services have been developed considering relatively static knowledge bases. In this paper, we investigate the process of incrementally updating tableau completion graphs created during consistency checking in the expressive description logics $\mathcal{SHOQ}(\mathcal{D})$ and $\mathcal{SHIQ}(\mathcal{D})$, which correspond to a large subset of the W3C standard Web Ontology Language, OWL-DL. We present an algorithm for updating completion graphs under both the addition and removal of ABox assertions; this provides a critical step towards reasoning over fluctuating/streaming data, as it has been shown that in KBs with substantially sized ABoxes, consistency checking can dominate reasoning time. We also provide an empirical analysis of the optimizations through an experimental implementation in an open source OWL-DL reasoner, Pellet.

2 Background

DL tableau-based algorithms decide the consistency of an ABox A w.r.t a TBox T (by TBox, we are additionally referring to all axioms for properties) by trying to construct (an abstraction of) a common model for A and T, called a *completion graph* [5]. Formally, a completion graph for an ABox A with respect to T is a directed graph $G = (\mathcal{V}, \mathcal{E}, \mathcal{L}, \dot{\neq})$. Each node $x \in \mathcal{V}$ is labeled with a set of concepts $\mathcal{L}(x)$ and each edge $e = \langle x, y \rangle$ with a set $\mathcal{L}(e)$ of role names. The binary predicate $\dot{\neq}$ is used for recording inequalities between nodes. This graph is constructed by repeatedly applying a set of *expansion rules*.

For ABox updates, we consider addition and deletion of individual equality and inequality assertions $x = y$ and $x \neq y$, concept assertions $C(x)$ and role assertions $R(x, y)$. Updating ABox assertions in the presence of a TBox and an RBox brings up several issues with the semantics. For the purposes of this paper we use a straight-forward update semantics, which we call *Edit Semantics*. Intuitively, *Edit Semantics* can be described as an update in which all new ABox assertions are directly added (or removed) to the KB and no further changes/revisions are made. As the above example shows removing an assertion from the ABox under these semantics will not guarantee that the removed assertion will not be entailed anymore. Furthermore, an addition of a new axiom may cause inconsistencies in the KB as no additional changes can be made besides the addition or removal invoked by the update. Formally, we desribe this semantics as:

Definition 1 *(Edit Semantics) Let S be the set of assertions in an initial ABox A. Then under* Edit Semantics*, updating S with an an ABox addition (resp. deletion) α, written as $A + \alpha$ (resp. $A - \alpha$), results in an updated set of ABox axioms S' such that $S' = S \cup \{\alpha\}$ (resp. $S' = S \setminus \{\alpha\}$).*

This semantics might be a little unintuitive (especially from a belief revision point of view) but real world use of the update semantics is directly present in ontology editors and many document-oriented servers, e.g. Semantic Web Service repositories. For example, an OWL-S Web Service description can be seen as a set of ABox assertions and publishing/retracting this service description to/from a repository would be typically done under *Edit Semantics*. Edit semantics additionally aligns with the Formula based update semantics defined in [9].

3 Update Algorithm

The goal of the algorithm presented here is to update a previously constructed completion graph in such a way that it is still a model of the KB (if one exists).

The approach presented here is applicable to the Description Logics $\mathcal{SHOQ}$ [3] and $\mathcal{SHIQ}$ [4], as the difference between the two completion algorithms is independent of the update algorithm.

3.1 ABox Additions

Conceptually, tableau algorithms for $\mathcal{SHOQ}$ [3] and $\mathcal{SHIQ}$ [4] can be thought of as incremental in nature - *expansion rules* are applied in a non-deterministic manner to labels in the completion graph. Hence, new ABox assertions can be added even after previous completion rules have been fired for existing nodes and edges in the graph. After the addition, expansion rules can subsequently be fired for the newly added nodes, edges and their labels.

In order to update a completion graph upon the addition of a type assertion, $C(x)$, the algorithm first checks if the individual exists in the completion graph. If $x \notin \mathcal{V}$, then x is added to $\mathcal{V}$. Then C is added to $\mathcal{L}(x)$, if it does not already exist. If an individual inequality relation, $x \neq y$, is added to the completion graph, the algorithm checks if $x \in \mathcal{V}$ and $y \in \mathcal{V}$. If either do not exists, then they are added to the graph. Additionally, if the $x \neq y \notin x \dot{\neq} y$, then it is added. Alternatively, if an individual equality relation, $x = y$, is added to the completion graph, the algorithm checks if $x \in \mathcal{V}$ and $y \in \mathcal{V}$. If either do not exists, then they are added to the graph. Additionally, x and y are merged. Lastly, if a role, $R(x, y)$, is added to the ABox and $\langle x, y \rangle \notin \mathcal{E}$, then $\langle x, y \rangle$ is added to $\mathcal{E}$ and R is added to $\mathcal{L}(\langle x, y \rangle)$. If however, $\langle x, y \rangle \in \mathcal{E}$ but $R \notin \mathcal{L}(e)$, then R is added to $\mathcal{L}(x)$.

After the graph has been updated, the completion rules are reapplied to the updated completion graph, as the update may induce additional expansion rules to be fired. We note here that if there has previously been a deletion update, then previously explored branches which had a clash must be re-explored as the deletion could have removed the axiom which caused the clash.

3.2 ABox Deletions

When updating a completion graph under ABox axiom removals, *components* (nodes, edges, labels, etc.) in the graph that correspond to the removed axiom cannot be simply removed. This is because as completion rules are applied, newly added portions of the graph are dependent on the presence of the original axioms in the KB. By deleting an ABox assertion, components of the graph that are dependent on that assertion need to be updated as well.

In order to account for this, we propose using axiom pinpointing [6], which tracks the dependencies of graph components on original source axioms from the ontology through the tableau expansion process. More specifically, the application of the expansion rules triggers a set of *events* that change the state of

the completion graph, or the flow of the algorithm. In [6] a set of change events is defined, including adding labels to nodes and edges, etc. In order to record the changes on the completion graph, [6] introduces a *tracing function*, which keeps track of the asserted axioms responsible for changes to occur. The *tracing function*, τ, maps each event $e \in \mathcal{E}$ to a collection of sets, each set containing a fragment of K (axioms) that cause the event to occur. This tracing function is maintained throughout the application of tableau expansion rules defined in [5].

For purpose of this work, the original set of *change events* has been extended [2] to include all possible events that can occur during the application of expansion rules. The extension of events includes removing nodes, edges, labels (in case of merges), adding nodes and edge, etc. Additionally the tracing function maintenance through expansion rule application has been extended. Although the tracing extension has been provided for $\mathcal{SHOIQ}$, it is trivial to see how they are applied to $\mathcal{SHIQ}$ and $\mathcal{SHOQ}$ completion strategies. For brevity, further details are omitted, however they can be found in [2].

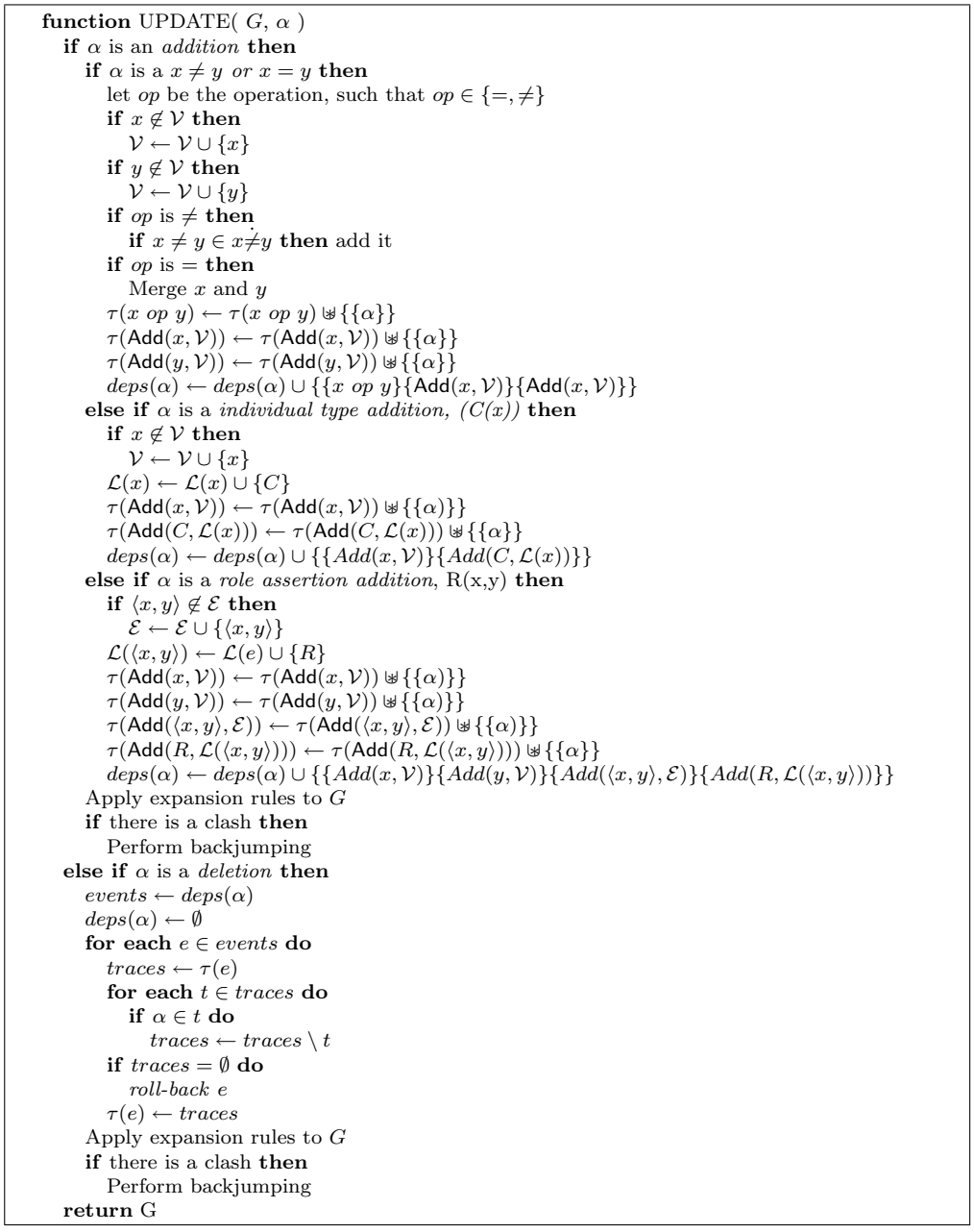
In general, the update algorithm for deletion works as follows. When an ABox axiom is removed, the algorithm performs a lookup in the graph for all *change events* whose axiom traces include the axiom number of the deleted axiom. These events are *rolled-back* if and only if their axiom trace sets only include sets which contain the deleted axiom, possibly among other axioms. By *roll-back* we refer to simply undoing (the inverse) the event (e.g., rolling back the event $Add(x, \mathcal{V})$ would be the process of removing x from $\mathcal{V}$). If there exists additional axiom traces for that particular event that do not include the removed axiom, then only the sets including the removed axiom are deleted from the axiom trace set; in this situation the actual event is not *rolled-back*. This is intuitive as there still exists support for that particular event.

As in the approach for additions, the completion rules must be reapplied to the updated completion graph as it is possible for the graph to be incomplete for the KB. Axiom tracing additionally requires a slight modification to the update approach for ABox additions. For example, in the case that a individual type assertion is added to the KB, the algorithm must add a new tracing set to the axiom trace for the affected components (this set will consist of the new axiom number). The update algorithm ($UPDATE(G, \alpha)$) is provided in Figure 1. Note that *deps* (dependents) is the inverted tracing function index. Additionally, the operator $\uplus$ is defined as follows: let $\mathbf{S_1} = \{S_1^1, ..., S_m^1\}$ and $\mathbf{S_2} = \{S_1^2, ..., S_n^2\}$ be two collections of sets, then:

$$S_i^1 \ (1 \leq i \leq m) \in S_1 \uplus S_2 \text{ iff } S_i^1 \not\subset S_j^2 \text{ for all } j,\ 1 \leq j \leq n$$
$$S_j^2 \ (1 \leq j \leq n) \in S_1 \uplus S_2 \text{ iff } S_j^2 \not\subset S_i^1 \text{ for all } i,\ 1 \leq i \leq m$$

Due to space limitation soundness proofs for the update algorithm are omitted, however they can be found in a more detailed technical report[1].

[1] Description Logic Reasoning for Dynamic ABoxes, UMIACS Technical Report. Available



function UPDATE(G, α)
  if α is an *addition* **then**
    if α is a $x \neq y$ *or* $x = y$ **then**
      let *op* be the operation, such that $op \in \{=, \neq\}$
      if $x \notin \mathcal{V}$ **then**
        $\mathcal{V} \leftarrow \mathcal{V} \cup \{x\}$
      if $y \notin \mathcal{V}$ **then**
        $\mathcal{V} \leftarrow \mathcal{V} \cup \{y\}$
      if *op* is $\neq$ **then**
        if $x \neq y \in x \dot{\neq} y$ **then** add it
      if *op* is $=$ **then**
        Merge x and y
      $\tau(x\ op\ y) \leftarrow \tau(x\ op\ y) \uplus \{\{\alpha\}\}$
      $\tau(\mathsf{Add}(x, \mathcal{V})) \leftarrow \tau(\mathsf{Add}(x, \mathcal{V})) \uplus \{\{\alpha\}\}$
      $\tau(\mathsf{Add}(y, \mathcal{V})) \leftarrow \tau(\mathsf{Add}(y, \mathcal{V})) \uplus \{\{\alpha\}\}$
      $deps(\alpha) \leftarrow deps(\alpha) \cup \{\{x\ op\ y\}\{\mathsf{Add}(x, \mathcal{V})\}\{\mathsf{Add}(x, \mathcal{V})\}\}$
    else if α is a *individual type addition, (C(x))* **then**
      if $x \notin \mathcal{V}$ **then**
        $\mathcal{V} \leftarrow \mathcal{V} \cup \{x\}$
      $\mathcal{L}(x) \leftarrow \mathcal{L}(x) \cup \{C\}$
      $\tau(\mathsf{Add}(x, \mathcal{V})) \leftarrow \tau(\mathsf{Add}(x, \mathcal{V})) \uplus \{\{\alpha)\}\}$
      $\tau(\mathsf{Add}(C, \mathcal{L}(x))) \leftarrow \tau(\mathsf{Add}(C, \mathcal{L}(x))) \uplus \{\{\alpha\}\}$
      $deps(\alpha) \leftarrow deps(\alpha) \cup \{\{Add(x, \mathcal{V})\}\{Add(C, \mathcal{L}(x))\}\}$
    else if α is a *role assertion addition*, R(x,y) **then**
      if $\langle x, y\rangle \notin \mathcal{E}$ **then**
        $\mathcal{E} \leftarrow \mathcal{E} \cup \{\langle x, y\rangle\}$
      $\mathcal{L}(\langle x, y\rangle) \leftarrow \mathcal{L}(e) \cup \{R\}$
      $\tau(\mathsf{Add}(x, \mathcal{V})) \leftarrow \tau(\mathsf{Add}(x, \mathcal{V})) \uplus \{\{\alpha)\}\}$
      $\tau(\mathsf{Add}(y, \mathcal{V})) \leftarrow \tau(\mathsf{Add}(y, \mathcal{V})) \uplus \{\{\alpha)\}\}$
      $\tau(\mathsf{Add}(\langle x, y\rangle, \mathcal{E})) \leftarrow \tau(\mathsf{Add}(\langle x, y\rangle, \mathcal{E})) \uplus \{\{\alpha)\}\}$
      $\tau(\mathsf{Add}(R, \mathcal{L}(\langle x, y\rangle))) \leftarrow \tau(\mathsf{Add}(R, \mathcal{L}(\langle x, y\rangle))) \uplus \{\{\alpha\}\}$
      $deps(\alpha) \leftarrow deps(\alpha) \cup \{\{Add(x, \mathcal{V})\}\{Add(y, \mathcal{V})\}\{Add(\langle x, y\rangle, \mathcal{E})\}\{Add(R, \mathcal{L}(\langle x, y\rangle))\}\}$
    Apply expansion rules to G
    if there is a clash **then**
      Perform backjumping
  else if α is a *deletion* **then**
    $events \leftarrow deps(\alpha)$
    $deps(\alpha) \leftarrow \emptyset$
    for each $e \in events$ **do**
      $traces \leftarrow \tau(e)$
      for each $t \in traces$ **do**
        if $\alpha \in t$ **do**
          $traces \leftarrow traces \setminus t$
      if $traces = \emptyset$ **do**
        roll-back e
      $\tau(e) \leftarrow traces$
    Apply expansion rules to G
    if there is a clash **then**
      Perform backjumping
  return G

Figure 1: Pseudo-code of update procedure for $\mathcal{SHOQ}(\mathcal{D})$ and $\mathcal{SHIQ}(\mathcal{D})$ ABoxes

4 Implementation and Evaluation

We have implemented the approach presented in this paper in an open source OWL-DL reasoner, Pellet [2]. In order to evaluate our update algorithm, we have performed an emperical evaluation using two different KBs with large ABoxes - the Lehigh University Benchmark (LUBM)[3] and AKT Reference Ontology [4].

Three tests were run over three KBs consisting of one, two, and lastly three universities from the LUBM dataset generator. First an initial consistency check was performed and then in each test, a random update was selected which was used to update the KB. In the regular version of the reasoner, the cached completion graph was discarded, while in the optimized reasoner the update algorithm was utilized. For each KB size, varying sized additions and deletions were randomly selected from the dataset. Update sizes include single axiom, twenty-five axioms, and fifty axioms (individual and/or role). Each test was performed twenty-five times and the results were averaged. Expressivity and KB statistics are provided in Table 1. Results for additions and removals in the LUBM

Name	Classes / Properties / Individuals / Assertions	Expressivity
LUBM-1 Univ	43 / 32 / 18,257 / 97,281	$\mathcal{SHI}$
LUBM-2 Univ	43 / 32 / 47,896 / 254,860	$\mathcal{SHI}$
LUBM-3 Univ	43 / 32 / 55,110 / 295,728	$\mathcal{SHI}$
AKT-1	160 / 152 / 16,505 / 70,948	$\mathcal{SHIF}$
AKT-2	160 / 152 / 32,926 / 143,334	$\mathcal{SHIF}$

Table 1: LUBM and AKT Portal Dataset Statistics

test are presented in Figure 2 (timing results are shown in milliseconds and the scale is logarithmic). We note that the '0' axiom value represents the initial consistency check. In both versions of the reasoner, initial consistency checks are comparable. However for both update types, performance improvements ranging from one to three orders of magnitude are achieved under updates in the reasoner with the optimized update algorithm. This is due to the avoidance of re-firing of completion rules by maintaining the previous completion graph; therefore very few (if any in some cases) completion rules must be refired. This provides direct empirical evidence for the effectiveness of the update algorithm. In a second evaluation, two datasets[5] adhering to the AKT Reference ontology were used (statistics shown in Table 1). The tests were structured in the same manner as the LUBM test. Again, each test was performed twenty-five times and the results are averaged over these iterations. All timings are in milliseconds and the scale is logarithmic. Similar to the LUBM test, update performance is

at http://www.mindswap.org/papers/2006/aboxUpdateTR2006.pdf

[2]Pellet OWL-DL Reasoner: http://www.mindswap.org/2003/pellet/

[3]LUBM Ontology: http://swat.cse.lehigh.edu/projects/lubm/

[4]AKT Ontology: http://www.aktors.org/publications/ontology/

[5]Hyphen-REA: http://www.hyphen.info/rdf/hero_complete.zip

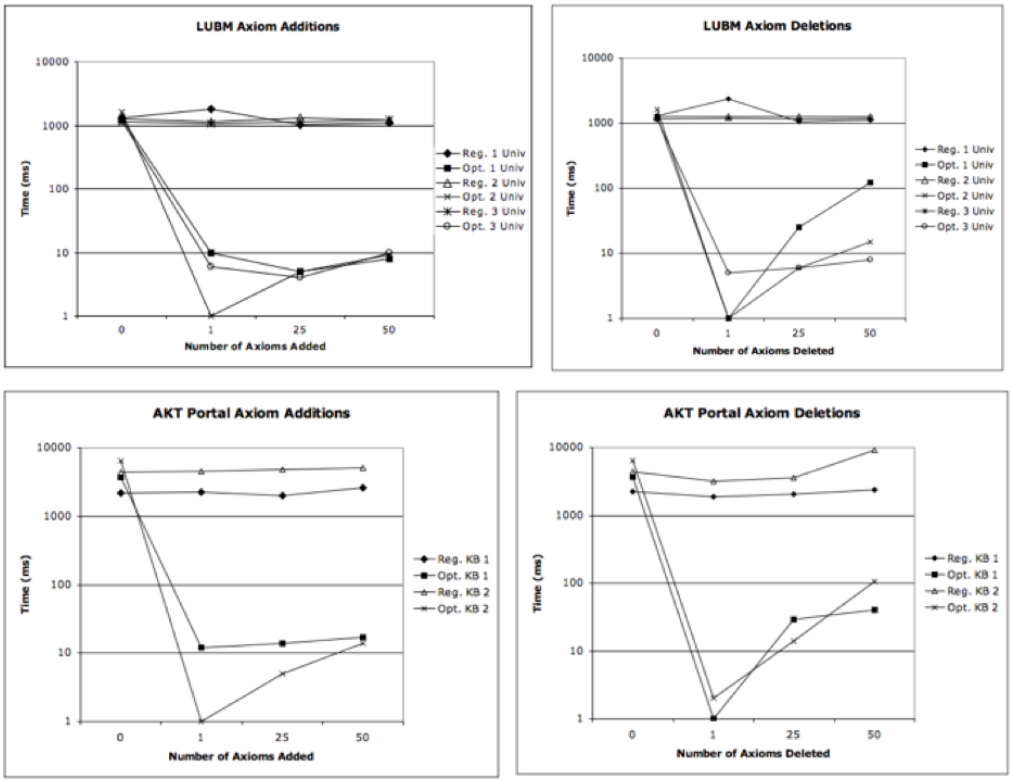


Figure 2: Addition and Removal Updates of LUBM and AKT datasets

improved between one to three orders of magnitude. Note that the performance of the update is better in the LUBM test cases; this is primarily due to the increased complexity of the AKT Reference Ontology. Therefore, a larger number of expansion rules are applied after the update. However the update algorithm greatly outperforms the regular reasoner, again demonstrating the effectiveness and overall impact of the update algorithm.

5 Related Work and Conclusion

To our knowledge there has been no previous work in DL reasoning algorithms for incremental maintenance of completion graphs. We do note that this work can be paralleled to view maintenance and truth maintenance; however we deal with a more expressive logic and a different proof theory. There has also been recent work in optimizing incremental instance retrieval in expressive description logics [1]; however previous work has not addressed the notion of updating the completion graphs for consistency checking. Additionally, in [7, 8] the authors specify update semantics for descriptions logics; however this work is less related to belief revision and is independent as we are concerned with maintaining the

internal state of the reasoner.

In this paper, we have presented an algorithm for updating completion graphs for the Description Logics $\mathcal{SHIQ}(\mathcal{D})$ and $\mathcal{SHOQ}(\mathcal{D})$ under both the addition and removal of ABox assertions, providing a critical step towards reasoning procedures for fluctuating or streaming data. We have provided an empirical analysis of the algorithm through an experimental implementation in the Pellet reasoner. Our initial results are very promising as they demonstrate orders of magnitude performance improvement.

References

[1] Volker Haarslev and Ralf Moller. Incremental query answering for implementing document retrieval services. In *International Workshop on Description Logics*, pages 85–94, 2003.

[2] Christian Halaschek-Wiener, Aditya Kalyanpur, and Bijan Parsia. Extending tableau tracing for abox updates. In *UMIACS Tech Report*, 2006. http://www.mindswap.org/papers/2006/aboxTracingTR2006.pdf.

[3] I. Horrocks and U. Sattler. Ontology reasoning in the SHOQ(D) description logic. In B. Nebel, editor, *Proc. of the 17th Int. Joint Conf. on Artificial Intelligence (IJCAI 2001)*, pages 199–204. Morgan Kaufmann, 2001.

[4] I. Horrocks, U. Sattler, and S. Tobies. Practical reasoning for expressive description logics. In *Proc. of the 6th Int. Conference on Logic for Programming and Automated Reasoning (LPAR'99)*, number 1705 in Lecture Notes in Artificial Intelligence, pages 161–180. Springer-Verlag, 1999.

[5] Ian Horrocks and Ulrike Sattler. A tableaux decision procedure for SHOIQ. In *Proc. of the 19th Int. Joint Conf. on Artificial Intelligence (IJCAI 2005)*. Morgan Kaufman, 2005.

[6] Aditya Kalyanpur, Bijan Parsia, Bernardo Cuenca-Grau, and Evren Sirin. Tableau tracing in shoin. UMIACS Tech Report.

[7] H. Liu, C. Lutz, M. Milicic, and F. Wolter. Updating description logic aboxes. In *International Conference of Principles of Knowledge Representation and Reasoning(KR)*, 2006.

[8] Mathieu Roger, Ana Simonet, and Michel Simonet. Toward updates in description logics. In *International Workshop on Knowledge Representation meets Databases*, 2002.

[9] M. Winslett. Updating logical databases. In *Updating Logical Databases*. Cambridge University Press, 1990.

Beyond Asserted Axioms: Fine-Grain Justifications for OWL-DL Entailments

Aditya Kalyanpur[1] and Bijan Parsia[1] and Bernardo Cuenca Grau[2]
University of Maryland, USA[1], University of Manchester, UK[2]
aditya@cs.umd.edu, bparsia@isr.umd.edu, bcg@cs.man.ac.uk

1 Motivation

The Ontology Engineering community widely agrees on the importance of helping the user understand the output of a DL reasoner. The most recent approaches to the problem [4] [3] are based on the notion of a *Minimal Unsatisfiability Preserving Sub-TBoxes* (MUPS). Roughly, a MUPS for an atomic concept A is a minimal fragment $\mathcal{T}' \subseteq \mathcal{T}$ of a TBox $\mathcal{T}$ in which A is unsatisfiable.

For example, given the TBox:

1: $A \sqcup C \sqsubseteq B \sqcap \exists R.D \sqcap E \sqcap \neg B$, 2: $C \sqsubseteq F \sqcup \geq 1.R$
3: $A \sqsubseteq E \sqcap \forall R.\neg D$, 4: $B \sqsubseteq \neg D \sqcup E$

where the concept A is unsatisfiable, the TBox $\mathcal{T}' = \{1\}$ is a MUPS for A in $\mathcal{T}$.

In our earlier work [2], we extended the notion of a MUPS to arbitrary entailments in $\mathcal{SHOIN}$ and devised a non-standard inference service, *axiom pinpointing*, which, given $\mathcal{T}$ any of its logical consequences α, determines the minimal fragments of $\mathcal{T}$ in which α holds.

However, these services, though useful for debugging and explanation purposes, suffer from a fundamental *granularity* problem: since they work at the asserted axiom level, they fail to determine which *parts* of the asserted axioms are *irrelevant* for the particular entailment under consideration to hold. For instance, in our example, the conjunct E in axiom 1 is irrelevant for the unsatisfiability of A. Moreover, additional parts of axioms that could contribute to the entailment are *masked*, e.g., the axioms 1, 3 can be broken down into $A \sqsubseteq \exists R.D$ and $A \sqsubseteq \forall R.\neg D$ which also entail the unsatisfiability of A, however, this condition cannot be captured.

In this paper, we aim at extending the axiom pinpointing service to capture *precise justifications*, which are at a finer granularity level than the original asserted axioms. In this context, we provide a formal notion of precise justification and propose a decision procedure for the problem. We discuss implementation details of the service

and show that it is feasible in practice. Our results are applicable to $\mathcal{SHOIN}$ and hence to OWL-DL.

2 Precise Justifications

Since we aim at identifying relevant parts of axioms, we define a function that splits the axioms in a TBox $\mathcal{T}$ into "smaller" axioms to obtain an equivalent TBox $\mathcal{T}_s$ that contains as many axioms as possible.

Definition 1 ***(split_KB)*** *Given a $\mathcal{SHOIN}$ concept C in negation normal form (NNF), the function $split(C)$ returns a set of concepts, inductively defined as follows:*

$split(C) \leftarrow C$, *if C is A, $\neg A$, $\geq n.R$, $\leq n.R$, $= n.R$, $\top, \bot$ or a nominal node $\{o\}$*
$split(C \sqcap D) \leftarrow split(C) \cup split(D)$
$split(C \sqcup D) \leftarrow (split(C) \otimes_{\sqcup} split(D))$
$split(\forall R.C) \leftarrow \forall R. \prod(split(C))$
$split(\exists R.C')$ – *we get two cases depending on C'*
if C' is of the form $C \sqcap D$,
$split(\exists R.C') \leftarrow \exists R.E \cup split(\neg E \sqcup C) \cup split(\neg E \sqcup D) \cup split((\neg C \sqcup \neg D) \sqcup E)$, *where E is a fresh concept*
else,
$split(\exists R.C') \leftarrow \exists R. \prod split(C')$

Given a KB $= \{\alpha_1, \alpha_2, ...\alpha_n\}$, where each axiom α_i is of the form $C_i \sqsubseteq D_i$, let C_α be the concept representing α, i.e., $\neg C_i \sqcup D_i$, converted to NNF. Then, $T_s = split\_KB() = \sum \top \sqsubseteq C_s$ for each concept $C_s \in split(\neg C_1 \sqcup D_1) \cup split(\neg C_2 \sqcup D_2)..split(\neg C_n \sqcup D_n)$.

The idea of the above function is to rewrite the axioms in in a convenient normal form and split across conjunctions in the normalized version, e.g., rewriting $A \sqsubseteq C \sqcap D$ as $A \sqsubseteq C, A \sqsubseteq D$.[1] In some cases, we are forced to introduce new concept names, only for the purpose of splitting axioms into smaller sizes (which prevents any arbitrary introduction of new concepts); for example, since the axiom $A \sqsubseteq \exists R.(C \sqcap D)$ is not equivalent to $A \sqsubseteq \exists R.C$, $A \sqsubseteq \exists R.D$, we introduce a new concept name, say E, and transform the original axiom into the following set of "smaller" axioms: $A \sqsubseteq \exists R.E$, $E \sqsubseteq C$, $E \sqsubseteq D$, $C \sqcap D \sqsubseteq E$.

Table 1 shows an algorithm to split a TBox based on the above definition. In this algorithm, we also keep track of the correspondence between the new axioms and the axioms in $\mathcal{T}$ by using a function σ.

[1] Note that the $\otimes_{\sqcup}$ operator generates a cross-product of disjunctions, e.g, $split((A \sqcap B) \sqcup (C \sqcap D)) = \{(A \sqcup C), (A \sqcup D), (B \sqcup C), (B \sqcup D)\}$; and the $\prod$ operator applies the set of concepts to the preceding universal or existential operator, e.g., $split(\forall R.(B \sqcap (C \sqcup D))) = \{\forall R.B, \forall R.(C \sqcup D)\}$.

Algorithm: Split KB
Input: TBox $\mathcal{T}$
Output: TBox $\mathcal{T}_s$, Axiom Correspondence Function σ

$\mathcal{T}_s \leftarrow \emptyset$
initialize axiom correspondence function σ
initialize substitution $cache$
for each subclass axiom $\alpha \in \mathcal{T}$
  from $\alpha := C \sqsubseteq D$ generate $C_\alpha := \neg C \sqcup D$
  normalize C_α to NNF (pushing negation inwards)
  $\mathcal{T}_s \leftarrow \mathcal{T}_s \cup \{\top \sqsubseteq C_\alpha\}$
  $\sigma(\{\top \sqsubseteq C_\alpha\}) \leftarrow \alpha$
while there exists an axiom $\{\top \sqsubseteq C_\alpha\} \in \mathcal{T}_s$ s.t. **can-split**$(C_\alpha) \neq \emptyset$,
  $\mathcal{T}_s \leftarrow \mathcal{T}_s - \{\top \sqsubseteq C_\alpha\}$
  $\langle T, v, \pi, A, B\rangle \leftarrow$ **can-split**(C_α)
  if $L_t(\langle \prec_t (v), v\rangle) \neq \exists R$, **then**
    $C_A \leftarrow C_\alpha[A]_\pi$; $\sigma(C_A) \leftarrow \sigma(C_A) \cup \sigma(C_\alpha)$; $\mathcal{T}_s \leftarrow \mathcal{T}_s \cup \{\top \sqsubseteq C_A\}$
    $C_B \leftarrow C_\alpha[B]_\pi$; $\sigma(C_B) \leftarrow \sigma(C_B) \cup \sigma(C_\alpha)$; $\mathcal{T}_s \leftarrow \mathcal{T}_s \cup \{\top \sqsubseteq C_B\}$
  else
    if $cache(A \sqcap B) = \emptyset$, **then**
      let E be a new concept not defined in $\mathcal{T}_s$
      $\mathcal{T}_s \leftarrow \mathcal{T}_s \cup \{E \sqsubseteq A, E \sqsubseteq B, A \sqcap B \sqsubseteq E\}$
      $cache(A \sqcap B) \leftarrow E$
    else $E \leftarrow cache(A \sqcap B)$
    $C_E \leftarrow C_\alpha[E]_\pi$; $\sigma(C_E) \leftarrow \sigma(C_\alpha)$; $\mathcal{T}_s \leftarrow \mathcal{T}_s \cup \{\top \sqsubseteq C_E\}$

$subroutine$: **can-split**(C_α)
let $T = (W_t, \prec_t, L_t)$ be the parse tree for the disjunction C_α
if there exists a node $v \in W_t$ in T s.t. $L_t(v) = \sqcap$ with children A, B
  let π be the position of node v
  return $\langle T, v, \pi, A, B\rangle$
else return $\emptyset$

Table 1: Splitting a TBox

A *justification* is a generalization of the notion of a MUPS for arbitrary entailments:

Definition 2 ***(Justification)*** *Let $\mathcal{T} \models \alpha$ where α is an axiom. A fragment $\mathcal{T}' \subseteq \mathcal{T}$ is a justification for α in $\mathcal{T}$ if $\mathcal{T}' \models \alpha$, and $\mathcal{T}'' \not\models \alpha$ for every $\mathcal{T}'' \subset \mathcal{T}'$.*

We denote by $\mathsf{JUST}(\alpha, \mathcal{T})$ the set of all justifications for α in $\mathcal{T}$.
Finer-grained justifications can be defined as follows:

Definition 3 ***(Precise Justification)***
*Let $\mathcal{T} \models \alpha$. A TBox $\mathcal{T}'$ is a **precise justification** for α in $\mathcal{T}$ if $\mathcal{T}' \in \mathsf{JUST}(\alpha, \mathsf{split\_TBox}(\mathcal{T}))$.*

We denote by $\mathsf{JUST}_\mathsf{p}(\alpha, \mathcal{T})$ the set of all precise justifications for α in $\mathcal{T}$.
In $\mathcal{SHOIN}$, entailment can be reduced to concept unsatisfiability. As a consequence, it is easy to show that the problem of finding all the justifications for an

unsatisfiable concept and the problem of finding all the justifications for a given entailment can be reduced to each other. In what follows, we shall restrict our attention, without loss of generality, to the problem of finding all the precise justifications for an unsatisfiable concept w.r.t to a $\mathcal{SHOIN}$ TBox.

3 Finding Precise Justifications

The problem of finding all precise justifications for an entailment α of a TBox $\mathcal{T}$ now reduces to the problem of finding all justifications for α in the $split\_TBox(\mathcal{T})$. Thus, we can use the algorithm listed in Table 1 to split a TBox, and then apply any decision procedure to find all justifications for the entailment in the split version of the TBox.

We briefly sketch the algorithm we have developed in our earlier work to find all justifications for an arbitrary entailment in a TBox. For details, we refer the reader to the technical report [2].

The first part of the algorithm is based on the decision procedure for concept satisfiability in $\mathcal{SHOIN}$ [1], and it finds any one arbitrary justification for the entailment. It keeps track of the axioms from the TBox responsible for each change in the completion graph, namely, the addition of a particular concept (or role) to the label of a specific node (or edge), the addition of an equality (merge) or inequality relation between nodes, or the detection of a contradiction (clash) in the label of a node.

In order to ensure correctness, our algorithm imposes an ordering among the deterministic rules: the $unfolding$ and CE rules are only applied when *no other* deterministic rule is applicable. The rationale for this strategy relies on the definition of justification that establishes minimality w.r.t. the number of axioms considered; the new rules cause new axioms to be considered in the tracing process and additional axioms should only be considered if strictly necessary.

The algorithm works on a tree $\mathbf{T}$ of completion graphs. that is incrementally built using the set of available *expansion rules*, and the traces for the events triggered by the rules are updated on the fly. For a full specification of the expansion rules, we refer the reader to [2]. The application of non-deterministic rules results in the addition of new completion graphs as leaves of $\mathbf{T}$, one for each different non-deterministic choice. However, since C is unsatisfiable w.r.t. $\mathcal{T}$, the algorithm detects a clash in each of the leaves. Upon termination, the trace of the detected clash (or clashes) in the leaves of $\mathbf{T}$ yields a justification.

Once a single justification has been found, we find the remaining justifications by employing a variation of the classical Hitting Set Tree (HST) algorithm.

Given a collection S of conflict sets, Reiter's algorithm constructs a labeled tree called *Hitting Set Tree* (HST). Nodes in an HST are labeled with a set $s \in S$ and: **1)** if $H(v)$ is the set of edge labels on the path from the root to the node v, then $\mathcal{L}(v) \cap H(v) = \emptyset$; **2)** for each $\sigma \in \mathcal{L}(v)$, v has a successor w and $\mathcal{L}(\langle v, w \rangle) = \sigma$; **3)** if $\mathcal{L}(v) = \emptyset$, then $H(v)$ is a hitting set for S.

In our approach, we form an HST where nodes correspond to single justification sets (computed on the fly), edges correspond to the removal of axioms from the KB, and in building the tree to compute all the minimal hitting sets, we in turn obtain all the justification sets as well. The correctness and completeness of this approach is provided in [2].

3.1 Optional Post-Processing

Having derived the justifications for the entailment $A \equiv \bot$ in the maximally split version of KB $\mathcal{T}$, we remove any term in a justification set that was introduced in the KB in step 1 using the procedure described in Table 2. The output of this stage gives us precise justifications for the entailment in terms of sub-axioms.

Algorithm: Remove New Terms **Input:** Collection of Axiom Sets $\mathcal{J}$, Original KB $\mathcal{T}$ **Output:** $\mathcal{J}$

for each axiom set $j \in J$ **do**,
  while there exists a term $C' \in \mathsf{vocab(j)}$ s.t. $C' \notin \mathsf{vocab}(\mathcal{T})$, **do**
  $S \leftarrow \emptyset$
  for each axiom $C' \sqsubseteq C_i \in j$, **do**
  $S \leftarrow S \cup C_i$
  if $S \neq \emptyset$, **then**
  $C \leftarrow C_1 \sqcap C_2 \sqcap ..C_n$ (for all $C_i \in S$)
  else
  $C \leftarrow \top$
  substitute C' with C in j

Table 2: Post-Processing

3.2 Example

We now present a detailed example to demonstrate how the algorithm finds precise justifications correctly.

Consider a KB $\mathcal{T}$ composed of the following axioms:

1. $A \sqcup B \sqsubseteq \exists R.(C \sqcap \neg C) \sqcap D \sqcap E$
2. $A \sqsubseteq \neg D \sqcap B \sqcap F \sqcap D \sqcap \forall R.\bot$
3. $E \sqsubseteq \forall R.(\neg C \sqcap G)$

Note, $\mathsf{vocab}(\mathcal{T}) = \{A, B, C, D, E, F, R\}$, and the concept A is unsatisfiable w.r.t $\mathcal{T}$.

Given $A, \mathcal{T}$ as input, the algorithm proceeds as follows:

Step 1: After pre-processing, we obtain a maximally split equivalent KB $\mathcal{T}_s$:

$T_s = \{A \sqsubseteq \exists R.H^1; B \sqsubseteq \exists R.H^1; A \sqsubseteq D^{1,2}; B \sqsubseteq D^1; A \sqsubseteq E^1; B \sqsubseteq E^1; H \sqsubseteq C^1; H \sqsubseteq \neg C^1; A \sqsubseteq \neg D^2; A \sqsubseteq B^2; A \sqsubseteq F^2; A \sqsubseteq \forall R.\bot^2; E \sqsubseteq \forall R.\neg C^3; E \sqsubseteq \forall R.G^3\}$.

The superscript of each axiom in $\mathcal{T}_s$ denotes the corresponding axiom in $\mathcal{T}$ that it is obtained from. This correspondence is captured by the function σ in the Split-KB algorithm (see Table 1). Notice that the superscript of the axiom $A \sqsubseteq D$ in $\mathcal{T}_s$ is the set $\{1, 2\}$ since it can be obtained from two separate axioms in $\mathcal{T}$. Also, we have introduced a new concept H in the split KB, which is used to split the concept $\exists R.(C \sqcap \neg C)$ in axiom 1.

Step 2: Now, we obtain the justifications for the unsatisfiability of A w.r.t $\mathcal{T}_s$. This gives us the following axiom sets J:

$J = \{\{A \sqsubseteq \exists R.H^1, H \sqsubseteq C^1, H \sqsubseteq \neg C^1\}; \{A \sqsubseteq D^{1,2}, A \sqsubseteq \neg D^2\}; \{A \sqsubseteq \exists R.H^1, H \sqsubseteq C^1, A \sqsubseteq E^1, E \sqsubseteq \forall R.\neg C^3\}; \{A \sqsubseteq \exists R.H^1, A \sqsubseteq \forall R.\bot^2\}\}$

Step 3: Finally, we remove the concept H introduced in $\mathcal{T}_s$ from the justification sets in J to get:

$\mathcal{T}' = \{\{A \sqsubseteq \exists R.(C \sqcap \neg C)^1\}; \{A \sqsubseteq D^1, A \sqsubseteq \neg D^2\}; \{A \sqsubseteq D^2, A \sqsubseteq \neg D^2\}; \{A \sqsubseteq \exists R.C^1, A \sqsubseteq E^1, E \sqsubseteq \forall R.\neg C^3\}; \{A \sqsubseteq \exists R.\top^1, A \sqsubseteq \forall R.\bot^2\}\}$

As can be seen, the final output is the complete set of precise justifications for $A \equiv \bot$ in $\mathcal{T}$.

4 Implementation Issues

We have shown in [3], [2] that the axiom pinpointing service used to find all justifications for an entailment of a $\mathcal{SHOIN}$ KB performs reasonably well on a lot of real-world OWL-DL ontologies.

The main additional overhead incurred for capturing precise justifications is due to the pre-processing phase where we split the KB. The concern here, from a reasoning point of view, is the introduction of GCIs during the splitting process, e.g. $A \equiv B \sqcap C$ is replaced by (among other things) $B \sqcap C \sqsubseteq A$. Even though these GCIs are absorbed, they still manifest as disjunctions and hence adversely affect the tableau reasoning process. [2]

Alternately, a more optimal implementation is the following: instead of splitting the entire KB beforehand, we can perform a *lazy splitting* of certain specific axioms (on the fly) in order to improve the performance of the algorithm. The modified algorithm with lazy splitting becomes:

- Given A unsatisfiable w.r.t $\mathcal{T}$, find a single justification set, $J \in \mathsf{JUST}(\mathsf{A} \equiv \bot, \mathcal{T})$
- Split axioms in J to give J_s. Prune J_s to arrive at a minimal precise justification set J_p

[2] In this case, it is possible to introduce a new absorption rule in the reasoner which would transform axioms of the form $B \sqsubseteq A, C \sqsubseteq A, B \sqcap C \sqsubseteq A$ to $A \equiv B \sqcap C$ internally. This obviously seems odd given that we split the KB initially and re-combine axioms back in the reasoner, but note that the goal here is finding precise justifications.

- Replace J by J_s in $\mathcal{T}$.
- Form Reiter's HST using J_p as a node, with each outgoing edge being an axiom $\alpha \in J_p$ that is removed from $\mathcal{T}$

The advantage of this approach is that it only splits axioms in the intermediate justification sets in order to arrive at precise justifications, and re-inserts split axioms back into the KB dynamically. For details of the procedure, see [2]

5 Conclusion and Future Work

We have implemented the axiom pinpointing service in the OWL-DL reasoner, Pellet, and exposed it in the OWL Ontology Editor, Swoop. We are in the process of extending it to capture precise justifications.

From a UI perspective, maintaining the correspondence between the subaxioms in the precise justification and the original asserted axioms is useful, as it allows us to strike out irrelevant parts of the axioms directly, which is helpful to understand the entailment better. (see Figure 5).

As future work, we plan to optimize the implementation and do a thorough performance and usability evaluation.

Axioms causing the inference
MaleStudentWith3Daughters ⊑ Parent:
1) (MaleStudentWith3Daughters ≡ (~~Student~~ ⊓ (∃hasGender . {~~male~~}) ⊓ (∀~~hasChildren~~ . ~~Female~~) ⊓ (= 3 hasChildren)))
2) |_(hasGender domain Animal)
3) (Parent ≡ (Animal ⊓ (≥ 1 hasChildren)))

Figure 1: Justification for MaleStudentWith3Daughters ⊑ Person in the Koala OWL Ontology

References

[1] Ian Horrocks and Ulrike Sattler. A tableaux decision procedure for SHOIQ. In *Proc. of IJCAI 2005*, 2005.

[2] Aditya Kalyanpur, Bijan Parsia, Bernardo Cuenca, and Evren Sirin. Axiom pinpointing: Finding (precise) justifications for arbitrary entailments in OWL-DL, 2006. Available at http://www.mindswap.org/papers/2006/AxiomPinpointing.pdf.

[3] Aditya Kalyanpur, Bijan Parsia, Evren Sirin, and James Hendler. Debugging unsatisfiable classes in OWL ontologies. *J. of Web Semantics, Vol 3, Issue 4*, 2005.

[4] S. Schlobach and R. Cornet. Non-standard reasoning services for the debugging of description logic terminologies. In *Proc. of IJCAI, 2003*, 2003.

Optimizations for Answering Conjunctive ABox Queries: First Results

Evren Sirin and Bijan Parsia
Maryland Information and Network Dynamics Lab,
8400 Baltimore Avenue, College Park, MD, 20740 USA
evren@cs.umd.edu, bparsia@isr.umd.edu

1 Introduction

Conjunctive query answering is an important task for many applications on Semantic Web. It is important to efficiently answer queries over knowledge bases with large ABoxes. Although answering conjunctive queries has been studied from a theoretical point of view, until very recently, there was no reasoner supporting such functionality. As a result, there is not enough implementation experience and optimization techniques developed far answering conjunctive queries.

In this paper, we address the problem of answering conjunctive ABox queries efficiently without compromising soundness and completeness. Our focus is on answering queries asked against a very large ABox. We consider queries with only distinguished variables. We focus on the case of distinguished variables because such queries can be answered more efficiently and such queries occur more frequently in realistic scenarios.

We start with the discussion of answering atomic ABox queries and then describe a sound and complete conjunctive query answering algorithm. We then present optimization methods to improve the query evaluation analogous to the optimization methods developed and studied in the context of relational databases. We discuss how the ordering of query evaluation can affect performance, describe some simple yet effective metrics of evaluating the cost of a given ordering and present some heuristics to find (near-)optimal orderings. In the end, we show that the cost model we described provides a sufficiently accurate approximation to the actual query answering time.

2 Answering Atomic ABox Queries

Retrieving Instances The ground query $C(a)$, so-called *instance check*, is answered by adding the negated statement $\neg C(a)$ to the ABox and checking for (in)consistency.

Pseudo model merging technique [2] can be used to test if the pseudo-model of the individual can be merged with pseudo-model of the negated concept $\neg C$ to detect obvious non-instances without doing a consistency test.

The naive way to answer the unground atomic query $C(x)$, so-called *instance retrieval*, is to iterate over all the individuals in the ABox and do a consistency check when the above methods fail to detect an obvious instance or non-instance. Generally, most of the remaining individuals are non-instances and would not cause an inconsistency when the negated statement is added to the KB. *Binary instance retrieval* technique presented in [1] exploits this characteristic and combines many instance checks in one ABox consistency test. If there is no inconsistency all candidates are proven to be non-instances, otherwise the method splits the set of candidates into two sets and continues.

The effectiveness of binary instance retrieval is maximized if obvious instances are found upfront and not put into the candidate list. Model merging technique can be used to detect *obvious instances* by caching dependency set information in the pseudo models. Model merging technique checks for possible interactions between models but finding a clash between models does not mean these models are not mergable because the clash might depend on a non-deterministic choice in tableau completion. There might be other completions of the ABox without the clash. Using the cached dependency set information in pseudo models distinguish a deterministic clash form others and identifies where the models *cannot* be combined in any possible completion meaning that the the individual is necessarily an instance of the concept.

One case where finding obvious instances fails is the case of *defined concepts*. Suppose a concept C is defined as $C \equiv D \sqcap \exists p.A$. Any instance of D that is related to an A instance with a p role is also an instance of C. Unfortunately, above methods would fail to detect such a case. If we are trying to find the instances of a defined concept then breaking up the concept description and applying the above method would help us to find an instance without doing a consistency check. This technique has proved to be very effective in our experiments.

Retrieving Role Fillers In OWL-DL, the role constructors are much less expressive compared to concept constructors. Therefore, verifying $p(\mathsf{a}, \mathsf{b})$ holds only if in the original ABox it is asserted that a and b is related by p or one of its subroles. The interactions between the role hierarchy and number restrictions invalidate this assumption, e.g. a super role assertion combined with cardinality restrictions may cause the relation to hold. Transitive roles complicate the situation even more, now a path between individuals is enough for the relation to hold. Having nominals in the KB completely makes things even more complicated as nominals might relate individuals from disconnected parts of the ABox.

Even if asserted facts in the ABox does not help to find identify all the role assertions between individuals, the completion graph generated for the ABox consistency test can be used to find obvious relations and non-relations. There might be a rela-

tion between disconnected individuals only if such a relation occurs in every model. This suggests by examining the completion graph we can still detect non-relations because if two individuals are disconnected in a completion graph then there is at least one model they are not related and the entailment does not hold. The details of this approach can be found in [4] where we have described this technique in detail for detecting non-subsumptions between concepts.

After finding all obvious relations and non-relations, one might still be left with some possible candidates that might or might not be related. At this point, we can reduce the query $p(x, \mathtt{a})$ (resp. $p(\mathtt{a}, x)$) to an instance retrieval query for concept $\exists p.\{a\}$ (resp. $\exists p^{-}.\{a\}$). If both arguments in the query are unground as in $p(x, y)$, then we first need to generate all candidates for x and then use the above techniques to find corresponding y values.

3 Answering Conjunctive Queries

In this section, we consider answering arbitrary conjunctive queries. W.l.o.g. we assume the query graph is connected. For queries with disconnected components, we can answer each component separately and then take the Cartesian product of generated solutions.

3.1 Query Answering Algorithm

The pseudo-code of the conjunctive query answering algorithm is given in Figure 1. The algorithm simply iterates through all the atoms in the query and either generates bindings for a variable or tests if the previous bindings satisfy the query atom. Generating bindings are done by invoking the instance retrieval function $retrieve$ which in turn might perform several consistency checks as described earlier. Theoretically, testing the satisfaction of a query atom might also require a consistency check. However, as explained in the previous section, most of these tests can be answered without doing a consistency test. Especially, the retrieval operations regarding the role assertions, e.g. $retrieve(\exists p.\{\mathsf{u}\})$, do not typically require any consistency check.

Initially the algorithm is invoked by $AnswerQuery(\mathcal{K}, A, \emptyset, \emptyset)$ where A is an ordering of the atoms in the query. The correctness of this algorithm is quite clear as the satisfaction of every binding is reduced to KB entailment. Thus, this is a sound and complete procedure for answering conjunctive queries.

The efficiency of this algorithm depends very much on the order query atoms are processed. For example, in the query $C(x) \wedge\ p(x, y) \wedge\ D(y)$, suppose C has 100 instances, each instance has one p value and D has 10.000 instances. The ordering $[C(x), p(x, y), D(y)]$ would be much more efficient compared to the ordering $[D(x), p(x, y), C(y)]$. We would do one instance retrieval operation to get 100 instances, find the corresponding p values and test whether these are D instances. The

```
function AnswerQuery(K, A, B, Sol)
  input  K is the input KB, A is a list of query atoms, B is the binding
         built so far, Sol is the set of all bindings that satisfy the query

if A = [ ] then return Sol ∪ {B}
Let a = first(A) and R = rest(A)
Substitute the variables in a based on the bindings in B
if a = C(v) and K ⊨ C(v) then return AnswerQuery(K, R, B, Sol)
if a = C(x) then
   for each v ∈ retrieve(C) do
      Let Sol = AnswerQuery(R, B ∪ {x ← v}, Sol)
if a = p(v, u) and K ⊨ p(v, u) then return AnswerQuery(K, R, B, Sol)
if a = p(x, v) (resp. p(v, x)) then
   for each u ∈ retrieve(∃p.{v}) (resp. u ∈ retrieve(∃p⁻.{v})) do
      Let Sol = AnswerQuery(K, R, B ∪ {x ← u}, Sol)
if a = p(x, y) then
   for each v ∈ retrieve(∃p.⊤) do
      for each u retrieve(∃p.{v}) do
         Let Sol = AnswerQuery(R, B ∪ {x ← u} ∪ {y ← u}, Sol)
return Sol
```

Figure 1: Pseudo-code for the conjunctive query answering algorithm

second ordering, on the other hand, requires us to iterate over 10.000 individuals and check for a p^- value that does not exists for most d instances.

3.2 Cost-based Query Reordering

There are several important challenges to finding an optimal query reordering. In query optimization for relational databases, the main objective is to find an optimal join order and generally the bottleneck is reading data from disk. In a DL reasoner, the most costly operation is consistency checking so we should try to minimize the number of consistency checks performed.

There are two parameters that will help us to estimate the cost of answering a query. First we need to estimate how costly an atomic query is, e.g. for an instance retrieval query, estimate how long it will take to find all the instances, and then estimate the size of results, e.g. how many instances a concept has. These two parameters are interdependent to some degree. For example, if C has 100.000 instances and D has only 10 instances, retrieving C instances can be more costly. However, this is not always true because all the 100.000 individuals might be considered as possible D instances (if the methods described in Section 2 fail) and force us to do expensive consistency tests. For this reason, there is no easy way of estimating these parameters that would work for different ontologies. In the implementation section, we will briefly describe some preliminary methods we devised to compute these parameters.

For now, we will assume that for each atomic query type there are cost functions $\mathcal{C}_{ir}(C)$, $\mathcal{C}_{ic}(C)$, $\mathcal{C}_{rr}(r)$ and $\mathcal{C}_{rc}(r)$ that returns the cost of instance retrieval for concept

C, the cost of a single instance checking for concept C, the cost of role filler retrieval for role r and the cost of verifying a role filler for role r, respectively. Note that, we are assuming the cost of instance checking for a concept is same for all the different individuals in the KB. This assumption may not be very accurate but considering that we will typically deal with large number of individuals, it is not practical to compute estimates for every individual.

In addition, we need to estimate the number of instances of a concept and how many role fillers exist for a given individual and a role. Assuming the size estimates are computed at a preprocessing step, we will use $|C|$ to denote the number of C instances and $|p|$ to denote the total number of tuples in p relation. The average number of p fillers for an individual is denoted by $avg(p)$ and computed as $|p|/|\exists p.\top|$.

Given the parameters for the cost computation and the size estimates, algorithm described in Figure 2 computes an estimate for the cost of query answering for a certain ordering.

Cost estimation is linear in the number of query atoms, provided that size estimates are already computed. However, there are exponentially many orderings to try so an exhaustive search would still be very expensive. Again, as in relational databases, it is possible to use some heuristics to prune the search space. The heuristics we use are: 1) For each atom at position $i > 1$ in the ordered list, there should be at least one atom at position $j < i$ s.t. two atoms share at least one variable. 2) Atoms of the form $p(x, \mathsf{v})$ and $p(\mathsf{v}, x)$ should appear before other atoms involving x. 3) An atom of the form $C(x)$ should come immediately after the first atom that contains x.

First rule is similar to the general query optimization rule that cross products should be avoided. Second rule makes use of the fact that generally an individual is related to limited number of other individuals. And the last rule is to discard the orderings such as $[C(x), p(x, y), q(y, z), D(y)]$. This ordering is not desirable because if $p(x, y)$ finds a binding for y such that $D(y)$ is not satisfied, we would unnecessarily retrieve the q fillers before realizing the failure.

function $EstimateCost(A, B)$
if $A = [\,]$ **then return** 1
Let a = first(A) and R = rest(A)
if $a = C(x)$ and $x \in B$ **then return** $\mathcal{C}_{ic}(C) + EstimateCost(R, B)$
if $a = C(x)$ and $x \notin B$ **then return** $\mathcal{C}_{ir}(C) + |C| * EstimateCost(R, B \cup \{x\})$
if $a = p(x, y)$ and $\{x, y\} \subseteq B$ **then return** $\mathcal{C}_{rc}(p) + EstimateCost(R, B)$
if $a = p(x, y)$ and $\{x, y\} \cap B = \{x\}$ **then return** $\mathcal{C}_{rr}(p) + avg(p) * EstimateCost(R, B \cup \{y\})$
if $a = p(x, y)$ and $\{x, y\} \cap B = \{y\}$ **then return** $\mathcal{C}_{rr}(p) + avg(p^-) * EstimateCost(R, B \cup \{x\})$
if $a = p(x, y)$ and $\{x, y\} \cap B = \emptyset$ **then return**
$\mathcal{C}_{ir}(\exists p.\top) + |p| * \mathcal{C}_{rr}(p) * EstimateCost(R, B \cup \{x, y\})$

Figure 2: Pseudo-code for estimating the cost of an ordering

3.3 Query Simplification

In some cases there might be redundant axioms in a query that can be safely removed from the query without affecting the results. For example, if $C \sqsubseteq D$ then the query $C(x) \wedge D(x)$ is logically equivalent to query $C(x)$. Such redundant atoms do not cause to make additional consistency tests (methods described in Section 2 are quite effective for these cases) but even repeating computationally cheap operations many times causes a noticeable overhead in the end.

The idea behind query simplification is to discover redundant atoms with cheap concept satisfiability tests. But performing too many concept satisfiability tests for simplifications that do not occur frequently in queries is wasteful. For example, simplification based on subsumption of named concepts and roles are nearly never applicable in real world queries or in the benchmarking problems for query answering. We have pinpointed the following two common query simplifications:

- Simplify $C(x) \wedge p(x,y) \wedge D(y)$ to
  - $p(x,y) \wedge D(y)$ if $\exists p.\top \sqsubseteq C$ (*Domain simplification*)
  - $C(x) \wedge p(x,y)$ if $C \sqsubseteq \forall p.D$ (*Range simplification*)

Note that range simplification can also be done even if one of the atoms $C(x)$ or $D(y)$ is missing since we can simply insert $\top(x)$ or $\top(y)$ as an additional atom. In such cases global domain/range restrictions of properties can be directly used and simplification can be done with no subsumption test.

4 Implementation and Experimental Evaluation

The optimization techniques described in this paper have been fully implemented in OWL-DL reasoner Pellet. Right now, size estimation for concepts is done by inspecting the completion graph generated by the initial ABox consistency test. Due to space constraints we will only outline the size estimation algorithm w.r.t concepts but the general idea is same for roles, too.

The size estimation algorithm iterates over a random sample of individuals in the ABox and for each concept tries to determine if this individual is an instance of the given concept without doing a consistency check. The techniques described in Section 2 might return `true`, `false`, or `unknown`. A result of `unknown` means that the individual may or may not be an instance but the reasoner cannot conclude without a consistency test. In such cases, we estimate that with probability κ such individuals would indeed be instances of that concept. Thus, the total estimate for a concept is $|C| = \sigma * (|C_{known}| + \kappa * |C_{unknown}|)$ where σ is the sampling ratio. We also use $|C_{unknown}|$ to have an estimate about $\mathcal{C}_{ir}(C)$ and $\mathcal{C}_{ic}(C)$. If $|C_{unknown}| = 0$ then it means that all the individuals can be retrieved without any consistency test. As $|C_{unknown}|$ increases $\mathcal{C}_{ir}(C)$ would typically increase. Of course there are other factors affecting $\mathcal{C}_{ir}(C)$ but these are not yet considered in our implementation.

In our experiments, we first tested the accuracy of the size estimation. For this purpose, have used the data from Lehigh University Benchmark (LUBM) [5] and ontologies Vicodi and Semintec from [3]. The following tables show the number of individuals in each dataset, the time spent for estimating the size of all the concepts in each dataset, and the mean normalized error over all concepts in the given ontology. For example, an error of 3.6 means that if the actual number of instances for a concept was 200, the algorithm returned 200 ± 7.2. We have changed the sampling percentage from $\%20$ to $\%100$.

		Sampling Percentage				
Dataset	Size	%20	%40	%60	%80	%100
LUBM	55664	3.6	6.7	9.7	11.8	15.0
Semintec	17941	0.9	1.6	2.4	3.2	3.9
Vicodi	16942	1.8	3.4	5.1	6.9	8.6

(a) Time spent in seconds

	Sampling Percentage				
Dataset	%20	%40	%60	%80	%100
LUBM	0.7	0.3	0.6	0.4	0.0
Semintec	6.4	4.4	4.9	3.7	0.0
Vicodi	18.5	11.3	7.6	4.7	0.9

(b) The mean normalized error

As expected, error in size estimation decreases as we inspect more and more individuals. More interestingly, for these ontologies, sizes can be computed with perfect accuracy if all the individuals are inspected. However, computation time also increases. Looking at these results we decided to use a sampling ratio of $\%20$ which yields fairly accurate results with reasonable computation time.

Next, we looked at the effectiveness of the cost model defined in this paper. Query ordering has a significant effect especially when there are many atoms in the query. Therefore, we used three conjunctive queries, Q2 (6 atoms), Q8 (5 atoms), Q9 (6 atoms), from LUBM and Q2 (5 atoms) from Semintec and Q2 (3 atoms) from Vicodi. We generated all the possible query orderings, pruned the orderings based on the aforementioned heuristics and computed the time to answer each query with that ordering. Figure 3 shows the scatter plot of different query orderings where X axis is the estimated cost and Y axis is the actual time it took to generate the answers. The correlation factor for each query is different ranging from low (~ 0.5) to perfect score ($= 1$). More importantly, in each case, the lowest-cost query ordering found by the estimation algorithm is very close to the optimal value.

Next table shows these results in more detail and displays query evaluation time (in milliseconds) for the minimum cost ordering, the minimum time ordering, the

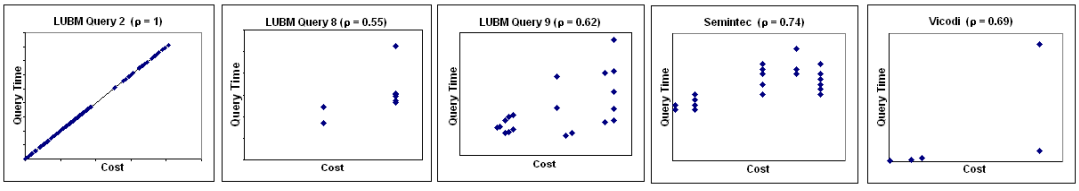


Figure 3: The correlation between the cost estimates and the actual query evaluation time. Each data point represents a different ordering of the corresponding query. Note that, due to simplification and heuristic pruning, number of data points is less than all possible orderings.

maximum time ordering and the median of all orderings (still excluding the heuristically pruned orderings). Although the minimum cost ordering does not always take minimum time, we can still see that the improvement in query evaluation time compared to an arbitrary ordering can be more than one order of magnitude.

	LUBM Q2	LUBM Q8	LUBM Q9	Semintec	Vicodi
Min Cost Ordering	20	320	227	100	81
Min Time Ordering	10	285	164	100	81
Median	1183	341	311	135	255
Max Time Ordering	1233	348	400	140	2123

5 Conclusion and Discussion

In this paper, we have described several different techniques to improve conjunctive query answering for DL reasoners. The query answering algorithm and the optimization techniques we describe do not compromise the soundness and completeness of the reasoner. Furthermore, the additional memory requirement is minimal (linear in the size of TBox but independent of the size of ABox). Our preliminary experimental evaluation shows that the preprocessing time spent for cost estimation is quite reasonable and Pellet can very efficiently answer conjunctive queries over large ABoxes.

There are many open issues left for future work. We are investigating how to estimate the atomic query costs more accurately. Secondly, it is not clear to what extent the effectiveness of our techniques depend on the knowledge base and the query. For our experiments, we picked benchmark datasets that were used in recent publications and considered to be realistic. We are now looking into the evaluation of our techniques on datasets with different characteristics, typically with more complicated TBox components.

References

[1] V. Haarslev and R. Möller. Optimization techniques for retrieving resources described in OWL/RDF documents: First results. In *Proc. KR 2004*, 2004.

[2] V. Haarslev, R. Möller, and A.Y. Turhan. Exploiting pseudo models for TBox and ABox reasoning in expressive description logics. In *IJCAR 2001, Italy*, 2001.

[3] B. Motik and U. Sattler. Practical DL reasoning over large ABoxes with KAON2. In *Proc. KR-2006*, 2006.

[4] Evren Sirin, Bernardo Cuenca Grau, and Bijan Parsia. From wine to water: Optimizing description logic reasoning for nominals. In *Proc. KR-2006*, 2006.

[5] Z. Pan Y. Guo and J. Heflin. LUBM: A benchmark for OWL knowledge base systems. *Journal of Web Semantics*, 3(2):158–182, 2005.

Model checking the basic modalities of CTL with Description Logic

Shoham Ben-David Richard Trefler Grant Weddell

David R. Cheriton School of Computer Science
University of Waterloo

Abstract. Model checking is a fully automated technique for determining whether the behaviour of a finite-state reactive system satisfies a temporal logic specification. Despite the fact that model checking may require analyzing the entire reachable state space of a protocol under analysis, model checkers are routinely used in the computer industry. To allow for the analysis of large systems, different approaches to model checking have been developed, each approach allowing for a different class of systems to be analyzed. For instance, some model checkers represent program state spaces and transitions explicitly, others express these concepts implicitly. The determination of which flavour best suits a particular model must often be left to experimentation. Description Logic (DL) reasoners are capable of performing subsumption checks on large terminologies. In this paper, we show how to perform explicit state model checking with a DL reasoner. We formulate the check that a reactive system satisfies a temporal specification as a consistency check on a terminology in the DL $\mathcal{ALC}$ and demonstrate our method on an example.

1 Introduction

Model checking [4] (cf. [5]) is a fully automated technique for verifying that the behaviour of a finite-state reactive system satisfies a temporal logic specification. As such, it is extremely useful in verifying important aspects of safety critical reactive systems.

The main challenge in this area, known as the *state explosion problem*, arises because systems may have short textual descriptions encoding exponentially larger state spaces that are analyzed by the model checker. While model checking techniques are widely used in industry [1, 6, 11], methods of attacking the state explosion problem and increasing the applicability of this technique are of substantial interest.

Given a finite state model M (a non-deterministic Kripke structure), a model checker verifies that the behaviours of M satisfy a temporal logic specification φ, typically given in Computation Tree Logic (CTL) [4], i.e. $M \models \varphi$.[1] The following is a typical specification: *it is always the case that at most one of several concurrent processes has access to a shared resource*. Letting c_i indicate that process i has access to the shared resource of interest, the specification is expressed succinctly in CTL as $\mathsf{AG}\neg(c_1 \wedge c_2)$. Here, the basic modality AG is composed of two temporal operators A, representing *all future paths*, and G, representing *all states on a path*. Formulas of the form $\mathsf{AG}(p)$, with p being a Boolean expression, are of special importance. In our experience, over 90% of the formulas written in practice can be converted into $\mathsf{AG}(p)$ form.

To cope with the state explosion problem several different techniques have been proposed. The different approaches work well on large classes of examples but no one technique best

[1] The symbol $\models$ is overloaded. We use it in this paper both in the context of model checking and in the context of DL reasoning.

suits all models. One method describes the model *symbolically* by representing the system under verification by boolean functions. Two main symbolic methods are used to perform model checking. The first, known as *SMV* [10], was introduced by McMillan in 1992. This method is based on Binary Decision Diagrams (BDDs) [3] for representing the state space as well as for performing the model checking procedure. The second is known as Bounded Model Checking [2]. Using this method, the model under verification is translated into a Boolean formula, and a satisfiability solver is applied to it to find a satisfying assignment. Such an assignment, if found, demonstrates a bug in the model.

Other important methods represent states and transitions explicitly. Explicit state methods appear to be more amenable to sophisticated state space reduction techniques. In this paper we show how to use Description Logic reasoning to perform explicit state model checking of the basic modalities of CTL on a synchronous model of computation, with the hope of benefiting from the powerful DL reasoners currently available [7–9]. Let MD be a model description, whose sematics is given by the Kripke structure M_{MD}, and let φ be a specification. We formulate a terminology $\mathcal{T}_{MD,\varphi}$ over the DL dialect $\mathcal{ALC}$, and define a concept C_{MD} such that by checking if $\mathcal{T}_{MD,\varphi} \models C_{MD}$ is consistent, we can determine whether $M_{MD} \models \varphi$.

In the next section we give the necessary background and definitions. Section 3 presents the translation of a model checking problem into a terminology over $\mathcal{ALC}$, and demonstrate it through an example. Section 4 concludes the paper.

2 Background and Definitions

Definition 1 (Description Logic $\mathcal{ALC}$) *Let* NC *and* NR *be sets of atomic concepts* $\{A_1, A_2, \ldots\}$ *and atomic roles* $\{R_1, R_2, \ldots\}$*, respectively. The set of* concepts C *of the description logic* $\mathcal{ALC}$ *is the smallest set including* NC *that satifies the following.*

- *If* $C_1, C_2 \in \mathsf{C}$*, then so are*
  - $\neg C_1$ • $C_1 \sqcap C_2$
- *If* $C \in \mathsf{C}$*, and* $R \in \mathsf{NR}$*, then so is*
  - $\exists R.C$

Additional concepts are defined as syntactic sugaring of those above:
• $\top = A \sqcup \neg A$*, for some* A • $\forall R.C = \neg\exists R.\neg C$
• $C_1 \sqcup C_2 = \neg(\neg C_1 \sqcap \neg C_2)$

An inclusion dependency *is an expression of the form* $C_1 \sqsubseteq C_2$*. A* terminology $\mathcal{T}$ *consists of a finite set of inclusion dependencies.*

The semantics *of expressions is defined with respect to a structure* $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$*, where* $\Delta^{\mathcal{I}}$ *is a non-empty set, and* $\cdot^{\mathcal{I}}$ *is a function mapping every concept to a subset of* $\Delta^{\mathcal{I}}$ *and every role to a subset of* $\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$ *such that the following conditions are satisfied.*
• $(\neg C)^{\mathcal{I}} = \Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}$ • $(C_1 \sqcap C_2)^{\mathcal{I}} = C_1^{\mathcal{I}} \sqcap C_2^{\mathcal{I}}$
• $\exists R.C = \{x \in \Delta^{\mathcal{I}} \mid \exists y \in \Delta^{\mathcal{I}} \text{ s.t. } (x, y) \in R^{\mathcal{I}} \wedge y \in C^{\mathcal{I}}\}$

A structure satisfies an inclusion dependency $C_1 \sqsubseteq C_2$ *if* $C_1^{\mathcal{I}} \subseteq C_2^{\mathcal{I}}$*. The* consistency problem for $\mathcal{ALC}$ *asks if* $\mathcal{T} \models C$ *holds; that is, if there exists* $\mathcal{I}$ *such that* $C^{\mathcal{I}}$ *is non-empty and such that* $C_1^{\mathcal{I}} \subseteq C_2^{\mathcal{I}}$ *holds for each* $C_1 \sqsubseteq C_2$ *in* $\mathcal{T}$*.*

Definition 2 (Kripke Structure) *Let* V *be a set of Boolean variables. A* Kripke structure M *over* V *is a four tuple* $M = (S, I, R, L)$ *where*

1. *S is a finite set of states.*
2. *$I \subseteq S$ is the set of initial states.*
3. *$R \subseteq S \times S$ is a transition relation that must be total, that is, for every state $s \in S$ there is a state $s' \in S$ such that $R(s, s')$.*
4. *$L : S \to 2^V$ is a function that labels each state with the set of variables true in that state.*

In practice, the full Kripke structure of a system is not explicitly given. Rather, a model is given as a set of Boolean variables $V = \{v_1, ..., v_n\}$, their initial values and their next-state assignments. The definition we give below is an abstraction of the input language of *SMV* [10].

Definition 3 (Model Description) *Let $V = \{v_1, ..., v_k\}$ be a set of Boolean variables. A* Model Description *over V is a tuple $MD = (I_{MD}, [\langle c_1, c'_1\rangle, ..., \langle c_k, c'_k\rangle])$, where $I_{MD} = \{v_1 = b_1, ..., v_k = b_k\}$, $b_i \in \{0, 1\}$, and c_i, c'_i are Boolean expressions over V.*

The semantics of a model description is a Kripke structure $M_{MD} = (S, I_M, R, L)$, where $S = 2^V$, $L(s) = s$ for $s \in S$, $I_M = \{I_{MD}\}$ and $R = \{(s, s') : \forall 1 \leq i \leq k,\ s \models c_i$ implies $s' \models (v_i = 0)$ and $s \models c'_i \wedge \neg c_i$ implies $s' \models (v_i = 1)\}$.

Intuitively, a pair $\langle c_i, c'_i\rangle$ defines the next-state assignment of variable v_i in terms of the current values of $\{v_1, ..., v_k\}$. That is,

$$\text{next}(v_i) = \begin{cases} 0 & \text{if } c_i \\ 1 & \text{if } c'_i \wedge \neg c_i \\ \{0, 1\} & \text{otherwise} \end{cases}$$

where the assignment $\{0, 1\}$ indicates that for every possible next-state value of variables $v_1, ...v_{i-1}, v_{i+1}, ..., v_n$ there must exist a next-state with $v_i = 1$, and a next-state with $v_i = 0$.

We give the definition of Basic CTL formulas below. Our definition differs from full CTL in that temporal operators cannot be nested. Note though, that most of the formulas written in practice can be converted into $AG(p)$ form, which is included in Basic CTL.

Definition 4 (Basic CTL formulas [4]) – *The formula ($v = 1$) is an atomic CTL formula*
- *If p and q are atomic CTL formulas, then so are*
  - $\neg p$ - $p \wedge q$
- *If p and q are atomic CTL formulas then the following are Basic CTL formulas:*
  - EXp - AXp - $E[pVq]$ - $A[pVq]$
- *If φ is a Basic CTL formula then so is* $\neg\varphi$.

Additional operators are defined as syntactic sugaring of those above:

- $E[pUq] = \neg A[\neg pV\neg q]$
- $A[pUq] = \neg E[\neg pV\neg q]$
- $AFp = A[true\ Up]$
- $EFp = E[true\ Up]$
- $AGp = \neg EF\neg p$
- $EGp = \neg AF\neg p$

The semantics of a CTL formula is defined with respect to a Kripke structure $M = (S, I, R, L)$ over a set of variables $V = \{v_1, ..., v_k\}$. A path in M is an infinite sequence of states $(s_0, s_1, ...)$ such that each successive pair of states (s_i, s_{i+1}) is an element of R. The notation $M, s \models \varphi$, means that the formula φ is true in state s of the model M.

- $M, s \models (v = 1)$ iff $v \in L(s)$
- $M, s \models p \wedge q$ iff $M, s \models p$ and $M, s \models q$
- $M, s \models \neg\varphi$ iff $M, s \not\models \varphi$

- $M, s_0 \models AXp$ iff for all paths $(s_0, s_1, ...)$, $M, s_1 \models p$
- $M, s_0 \models EXp$ iff for some path $(s_0, s_1, ...)$, $M, s_1 \models p$
- $M, s_0 \models A[pVq]$ iff for all paths $(s_0, s_1, ...)$, either for all $i \geq 0$, $M, s_i \models q$ or there exists $n \geq 0$ such that $M, s_n \models p$ and for all $0 \leq i \leq n, M, s_i \models q$
- $M, s_0 \models E[pVq]$ iff for some path $(s_0, s_1, ...)$, either for all $i \geq 0$, $M, s_i \models q$ or there exists $n \geq 0$ such that $M, s_n \models p$ and for all $0 \leq i \leq n, M, s_i \models q$

We say that a Kripke structure $M = (S, I, R, L)$ satisfies a Basic CTL formula φ ($M \models \varphi$) if for all $s_i \in I$, $M, s_i \models \varphi$.

Definition 5 (Formula type) *Let φ be a Basic CTL formula, expressed in terms of EX, AX, $E[pVq]$ or $A[pVq]$. We say that φ is of Type A if the outermost path quantifier is A, and Type E otherwise. We say that φ is a* negated *formula if its path quantifier is preceded by an odd number of negations.*

3 Model Checking Using Description Logic Reasoning

We give a linear reduction of a model checking problem into a consistency check over $\mathcal{ALC}$. Let $MD = (I, [\langle c_1, c'_1\rangle, ..., \langle c_k, c'_k\rangle])$ be a model description for the model $M_{MD} = (S, I, R, L)$, over $V = \{v_1, ...v_k\}$. Let φ be a Basic CTL formula. We generate a terminology $\mathcal{T}_{MD,\varphi}$, linear in the size of MD and constant in the size of φ, and a concept $Init$, such that by checking if $\mathcal{T}_{MD,\varphi} \models Init$ is consistent, we can determine whether $M_{MD} \models \varphi$.

We construct $\mathcal{T}_{MD,\varphi}$ as the union of three terminologies, $\mathcal{T}_{MD,\varphi} = \mathcal{T}_k \cup \mathcal{T}_{MD}^{type} \cup \mathcal{T}_\varphi$ where $\mathcal{T}_k$ depends only on the number k of variables in V, the terminology $\mathcal{T}_{MD}^{type}$ depends on the model description as well as on the *type* of the formula φ (with *type* being *A* or *E*), and $\mathcal{T}_\varphi$ depends only on the formula φ.

We start by describing the primitive concepts and roles which are used in all of the terminologies, and then provide the construction for each of the three terminologies. We conclude this section with a proposition that relates the consistency of the concept $Init$ with respect to $\mathcal{T}_{MD,\varphi}$ to the satisfaction of φ in the model M_{MD}.

Concepts and Roles We introduce one primitive role R corresponding to the transition relation of the model. For each variable $v_i \in V$ we introduce three primitive concepts: V_i, V_iN and V_iT. The concept V_i corresponds to the variable v_i, where V_i denotes $v_i = 1$ and $\neg V_i$ denotes $v_i = 0$. The concept V_iN corresponds in a similar way to the next-state value of V_i. The concept V_iT is needed to encode an execution step of the model as a sequence through R, as will be explained in the sequel. Finally, one primitive concept C_φ is introduced to assist the encoding of the specification φ. Depending on φ, this concept is not always needed. In total, for a set V with k variables, the terminology T_{MD}^{φ} will consist of $3k + 1$ primitive concepts and one role.

Constructing $\mathcal{T}_k$ For a transition $(s, s')\in R$ in the model M_{MD}, the state s may differ from s' in the values of some or all of the variables $v_1, ..., v_k$. That is, in one transition, many variables may simultaneously change their values. To achieve this with our single role R, we encode every transition of the model M_{MD} as a series of "micro-transitions" through R, each of which determines the next-state value of one variable only. To ensure that every variable

makes a move only on its turn, we use the concepts V_iT. We allow exactly one of $V_1T, ..., V_kT$ to hold on each state, and in the correct order, by introducing the following concept inclusions.

- $V_iT \sqsubseteq (\forall R.V_{i+1}T) \quad \neg V_iT \sqsubseteq (\forall R.\neg V_{i+1}T)$ for $1 \leq i < k$ and
  $V_kT \sqsubseteq (\forall R.V_1T) \quad \neg V_kT \sqsubseteq (\forall R.\neg V_1T)$

The actual model states correspond to where concept V_1T holds, after a cycle where each of the variables has taken a turn. Thus, we have to make sure to propagate V_i, V_iN values appropriately, by introducing the following inclusions for $1 \leq i \leq k$.

- When V_1T holds, V_i should assume its next value that has been stored in V_iN.
  $(V_1T \sqcap V_iN) \sqsubseteq V_i \quad (V_1T \sqcap (\neg V_iN)) \sqsubseteq (\neg V_i)$
- Propagate V_i at every step in the cycle, except the last one.
  $((\neg V_kT) \sqcap V_i) \sqsubseteq (\forall R.V_i) \quad ((\neg V_kT) \sqcap (\neg V_i)) \sqsubseteq (\forall R.(\neg V_i))$
- Propagate V_iN, unless V_iT holds, (in which case a new value is computed).
  $((\neg V_iT) \sqcap V_iN)) \sqsubseteq (\forall R.V_iN) \quad ((\neg V_iT) \sqcap (\neg V_iN)) \sqsubseteq (\forall R.(\neg V_iN))$

In total, for k variables, $\mathcal{T}_k$ will consist of $8k$ concept inclusions, each of them of constant size.

Constructing $\mathcal{T}_{MD}^{type}$ We now translate the model description $MD = (I, [\langle c_1, c_1' \rangle, ..., \langle c_k, c_k' \rangle])$. Let $I = \{v_1 = b_1, ..., v_k = b_k\}, b_i \in \{0, 1\}$. To encode the initial condition of the model, we introduce the concept inclusion

- $Init \sqsubseteq (D_1 \sqcap, ..., \sqcap D_k \sqcap V_1T \sqcap \neg V_2T \sqcap, ..., \sqcap \neg V_kT)$

Where $D_i = V_i$ if $(v_i = 1) \in I$, and $D_i = \neg V_i$ if $(v_i = 0) \in I$. The V_iTs are needed to ensure the initial condition corresponds to a real model state (V_1T), and that only V_1 is allowed to make a move ($\neg V_iT$).
Let the pair $\langle c_i, c_i' \rangle$ describe the next state behavior of the variable v_i. That is,

$$\text{next}(v_i) = \begin{cases} 0 & \text{if } c_i \\ 1 & \text{if } c_i' \wedge \neg c_i \\ \{0, 1\} & \text{otherwise} \end{cases}$$

where c_i, c_i' are Boolean expressions over $v_1, ..., v_k$, and $\{0, 1\}$ is a non-deterministic assignment, allowing v_i to assume both 0 and 1 in the next state. Let C_i be the concept generated by replacing every v_i in c_i with the concept V_i, and $\wedge$ with $\sqcap$. Let C_i' be the concept corresponding to c_i' in the same way. We introduce the following concept inclusions.
$(V_iT \sqcap C_i) \sqsubseteq \exists R.\neg V_iN$
$(V_iT \sqcap \neg C_i \sqcap C_i') \sqsubseteq \exists R.V_iN$
The encoding of the non-deterministic assignment as a concept inclusion, has two flavors, depending on the *type* of the formula φ.

- If φ is of type A, we call the terminology $\mathcal{T}_{MD}^{A}$, and introduce the inclusion:
  $(V_iT \sqcap \neg C_i \sqcap \neg C_i') \sqsubseteq (\exists R.V_iN \sqcap \exists R.\neg V_iN)$.
- If φ is of type E, we call the terminology $\mathcal{T}_{MD}^{E}$, and introduce the inclusion:
  $(V_IT \sqcap \neg C_i \sqcap \neg C_i') \sqsubseteq (\exists R.V_iN \sqcup \exists R.\neg V_iN)$.

In total, $\mathcal{T}_{MD}^{type}$ will consist of one concept, $Init$, of size $2k$, and 3 concept inclusions of constant size.

Constructing $\mathcal{T}_\varphi$ Let φ be the formula to be verified, written in terms of EXp, AXp,$E[pVq]$ or $A[pVq]$. Let ψ be the formula derived from φ by peeling off all negations preceding the outermost path quantifier, as well as the path quantifier itself. (e.g., $\neg E[pVq]$ becomes $[pVq]$). ψ can be one of two formulas only: Xp or $[pVq]$, where p, q are Boolean combinations of variables v_i. Let P, Q be the translation of p, q using the corresponding concepts V_i.

- If $\psi = Xp$, we introduce one concept inclusion to $\mathcal{T}_\varphi$:
  $Init \sqsubseteq \underbrace{\forall R.\forall R...\forall R}_{k}.P$
  We need a sequence of k transitions since one transition in the model M_{MD} is translated into k micro-transition in our terminology.
- If $\psi = [pVq]$ we use the auxiliary concept C_φ introduced for this purpose. Intuitively, $[pVq]$ means "*p releases q*". That is, q must hold along an execution path, unless p appears at one stage, in which case q is *released*, and does not have to hold any longer. We introduce the following concept inclusions to $\mathcal{T}_\varphi$.

$$\begin{aligned}(C_\varphi \sqcap \neg P) &\sqsubseteq \forall R.C_\varphi\\ (C_\varphi \sqcap P) &\sqsubseteq \forall R.\neg C_\varphi\\ \neg C_\varphi &\sqsubseteq \forall R.\neg C_\varphi\\ V_1T &\sqsubseteq (\neg C_\varphi \sqcup Q)\end{aligned}$$

  The first three concept inclusions above record the behavior of the concept P. C_φ holds in a state if and only if P has never held on the path leading to this state. The forth inclusion guarantees that on all the 'real' execution states (where V_1T holds), if P has not appeared until the previous state ($\neg C_\varphi$) then Q must hold.

In total, $\mathcal{T}_\varphi$ will consist of at most 4 concept inclusions, with one of them possibly of size k, and the rest of constant size. $\mathcal{T}_{MD,\varphi} = \mathcal{T}_k \cup \mathcal{T}_{MD}^{type} \cup \mathcal{T}_\varphi$ is therefore of size linear in k.

The following proposition relates the consistency of the concept $Init$ with respect to $\mathcal{T}_{MD,\varphi}$ to the satisfaction of φ in the model M_{MD}. Its proof will be given in a full version of the paper.

Proposition 6. *Let MD denote a model description for a model M_{MD}, and let φ be a Basic CTL specification. Then each of the following holds.*

1. *If φ is of type A and* not negated, *then $M_{MD} \models \varphi$ iff $\mathcal{T}_k \cup \mathcal{T}_{MD}^A \cup \mathcal{T}_\varphi \models$ init consistent.*
2. *If φ is of type E and* not negated, *then $M_{MD} \models \varphi$ iff $\mathcal{T}_k \cup \mathcal{T}_{MD}^E \cup \mathcal{T}_\varphi \models$init consistent.*
3. *If φ is of type A and* negated, *then $M_{MD} \models \varphi$ iff $\mathcal{T}_k \cup \mathcal{T}_{MD}^A \cup \mathcal{T}_\varphi \not\models$init consistent.*
4. *If φ is of type E and* negated, *then $M_{MD} \models \varphi$ iff $\mathcal{T}_k \cup \mathcal{T}_{MD}^E \cup \mathcal{T}_\varphi \not\models$init consistent.*

3.1 An Example

We illustrate our method by an example. Consider the model description

$$MD = (I, [\langle v_1 \wedge v_2, v_3\rangle, \langle \neg v_2, v_1 \wedge \neg v_1\rangle, \langle \neg v_1, v_1\rangle])$$

over $V = \{v_1, v_2, v_3\}$ with $I = \{v_1 = 0, v_2 = 1, v_3 = 0\}$. Figure 1 draws the states and transitions of the Kripke structure M_{MD} described by MD. Let the formula to be verified be

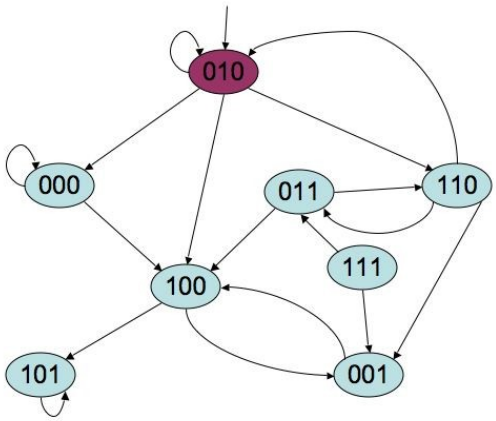


Fig. 1. A Kripke structure for MD

$\varphi = AG(\neg v_1 \vee \neg v_2 \vee \neg v_3)$. Note that $M_{MD} \models \varphi$, as can be seen in Figure 1, since the state $(1,1,1)$ can never be reached from the initial state. The terminology T^{φ}_{MD} derived from MD and φ will use the primitive concepts $\{V_1, V_2, V_3, V_1N, V_2N, V_3N, V_1T, V_2T, V_3T\}$ and the primitive role R. By the construction given in section 3, the set of concept inclusions will be the following.

The Construction of $\mathcal{T}_k$

- Concept inclusions ensuring control over the turn to make a move
  $V_1T \sqsubseteq (\forall R.V_2T)$ $\quad \neg V_1T \sqsubseteq (\forall R.\neg V_2T)$
  $V_2T \sqsubseteq (\forall R.V_3T)$ $\quad \neg V_2T \sqsubseteq (\forall R.\neg V_3T)$
  $V_3T \sqsubseteq (\forall R.V_1T)$ $\quad \neg V_3T \sqsubseteq (\forall R.\neg V_1T)$
- Concept inclusions to propagate the values of V_i
  $((\neg V_3T) \sqcap V_1) \sqsubseteq (\forall R.V_1)$ $\quad ((\neg V_3T) \sqcap (\neg V_1)) \sqsubseteq (\forall R.(\neg V_1))$
  $((\neg V_3T) \sqcap V_2) \sqsubseteq (\forall R.V_2)$ $\quad ((\neg V_3T) \sqcap (\neg V_2)) \sqsubseteq (\forall R.(\neg V_2))$
  $((\neg V_3T) \sqcap V_3) \sqsubseteq (\forall R.V_3)$ $\quad ((\neg V_3T) \sqcap (\neg V_3)) \sqsubseteq (\forall R.(\neg V_3))$
- Concept inclusions to propagate the values of V_iN
  $((\neg V_1T) \sqcap V_1N) \sqsubseteq (\forall R.V_1N)$ $\quad ((\neg V_1T) \sqcap (\neg V_1N)) \sqsubseteq (\forall R.(\neg V_1N))$
  $((\neg V_2T) \sqcap V_2N) \sqsubseteq (\forall R.V_2N)$ $\quad ((\neg V_2T) \sqcap (\neg V_2N)) \sqsubseteq (\forall R.(\neg V_2N))$
  $((\neg V_3T) \sqcap V_3N) \sqsubseteq (\forall R.V_3N)$ $\quad ((\neg V_3T) \sqcap (\neg V_3N)) \sqsubseteq (\forall R.(\neg V_3N))$
- Concept inclusions to move V_i equal to V_iN whenever V_1T holds
  $(V_1T \sqcap V_1N) \sqsubseteq V_1$ $\quad (V_1T \sqcap (\neg V_1N)) \sqsubseteq (\neg V_1)$
  $(V_1T \sqcap V_2N) \sqsubseteq V_2$ $\quad (V_1T \sqcap (\neg V_2N)) \sqsubseteq (\neg V_2)$
  $(V_1T \sqcap V_3N) \sqsubseteq V_3$ $\quad (V_1T \sqcap (\neg V_3N)) \sqsubseteq (\neg V_3)$

The Construction of $\mathcal{T}^{type}_{MD}$ We need to determine the *type* of φ. Note that $AG(p) = A[false\ Vp]$, thus φ can be written as $\varphi = A[false\ V(\neg v_1 \vee \neg v_2 \vee \neg v_3)]$. The type of φ is then A. We proceed to build $\mathcal{T}^{A}_{MD}$:

- The initial condition
  $INIT \sqsubseteq (\neg V_1 \sqcap V_2 \sqcap \neg V_3 \sqcap V_1T \sqcap \neg V_2T \sqcap \neg V_3T)$

- Computation of V_1
  $(V_1T \sqcap V_1 \sqcap V_2) \sqsubseteq (\exists R.(\neg V_1 N))$
  $(V_1T \sqcap (\neg(V_1 \sqcap V_2)) \sqcap V_3) \sqsubseteq (\exists R.V_1 N)$
  $(V_1T \sqcap (\neg(V_1 \sqcap V_2)) \sqcap (\neg V_3)) \sqsubseteq ((\exists R.V_1 N) \sqcap (\exists R.(\neg V_1 N)))$
- Computation of V_2
  $(V_2T \sqcap V_2) \sqsubseteq ((\exists R.V_2 N) \sqcap (\exists R.(\neg V_2 N)))$
  $(V_2T \sqcap (\neg V_2)) \sqsubseteq (\exists R.(\neg V_2 N))$
- Computation of V_3
  $(V_3T \sqcap V_1) \sqsubseteq (\exists R.V_3 N)$ $(V_3T \sqcap (\neg V_1)) \sqsubseteq (\exists R.(\neg V_3 N))$

The Construction of $\mathcal{T}_\varphi$ Since $\varphi = A[false\ V(\neg v_1 \vee \neg v_2 \vee \neg v_3)]$, then according to the translation in section 3, we need to introduce a concept C_φ, in order to record the behavior of $\perp$ (the translation of *false*). However, the behavior of $\perp$ cannot change along the execution path, and we get that $\neg C_\varphi$ is always false. Thus we can omit the concept C_φ altogether, and add only the concept inclusion: $V_1T \sqsubseteq (\neg V_1 \sqcup \neg V_2 \sqcup \neg V_3)$.

By Proposition 6 we get that $M_{MD} \models \varphi$ if and only if $\mathcal{T}_k \cup \mathcal{T}^A_{MD} \cup \mathcal{T}_\varphi \models$ *Init* consistent.

4 Summary

In this paper we have shown that model checking the basic modalities of CTL can be performed using DL reasoning. First experiments using the DL reasoner "Racer" [7] were conducted, with reassuring results. In the future, we plan to test our method on real models from the hardware industry. Several other related directions seem interesting for further investigation. Firstly, it seems possible, using additional calls to "Racer", to handle full CTL model checking. Secondly, we are interested in implementing symbolic model checking algorithms in $\mathcal{ALC}$ and applying these different model checking approaches to available test beds.

Acknowledgements

The authors gratefully acknowledge the support for this result provided by Nortel networks Ltd. and by NSERC Canada.

References

1. S. Ben-David, C. Eisner, D. Geist, and Y. Wolfsthal. Model checking at IBM. *Formal Methods in System Design*, 22(2):101–108, 2003.
2. A. Biere, A. Cimatti, E. Clarke, and Y. Zhu. Symbolic model checking without bdds. In *TACAS'99*, 1999.
3. R. Bryant. Graph-based algorithms for boolean function manipulation. In *In IEEE Transactions on Computers*, volume c-35 no. 8.
4. E. Clarke and E. Emerson. Design and synthesis of synchronization skeletons using branching time temporal logic. In *Proc. Workshop on Logics of Programs*, LNCS 131, pages 52–71. Springer-Verlag, 1981.
5. E. M. Clarke, O. Grumberg, and D. Peled. *Model Checking*. The MIT Press, 2000.
6. F. Copty, L. Fix, R. Fraer, E. Giunchiglia, G. Kamhi, A. Tacchella, and M. Y. Vardi. Benefits of bounded model checking at an industrial setting. In *CAV'01*, july 2001.
7. V. Haarslev and R. Moller. Racer system description. In *International Joint Conference on Automated Reasoning (IJCAR'2001)*, volume 2083.
8. I. Horrocks. The FaCT system. pages 307–312, 1998.
9. I. Horrocks and U. Sattler. Decidability of $\mathcal{SHIQ}$ with complex role inclusion axioms. *Artificial Intelligence*, 160(1–2):79–104, Dec. 2004.
10. K. McMillan. Symbolic model checking, 1993.
11. K. Yorav, S. Katz, and R. Kiper. Reproducing synchronization bugs with model checking. In *CHARME*, 2001.

Towards Mobile Reasoning

Thomas Kleemann

http://www.kleemann-online.com
tomkl@kleemann-online.com

Abstract. Highly optimized reasoning support for Description Logics (DLs) has been developed during the past years. This paper presents our efforts to develop a reasoner suitable for mobile devices, namely mobile phones.
This might become important upon the convergence of the mobile industry and the web, especially the semantic web. Details of the implementation and current limitations provide explanations to the differences in performance in comparison to one of the well-known desktop reasoners.

1 Motivation

Mobile phones have become an ubiquitous companion for many users. There are western countries with more mobile contracts than inhabitants. The mobile industry is pushing users to adopt modern 3G services, i.e. to use the mobile phone as a client to messaging and news services. At the dawn of the semantic web we expect these messages to be semantically annotated. The device should be able to sort out unwanted messages and reduce the penetration of the user to messages of interest. We suggested a user profile and matchmaking in [6, 8] that copes with SPAM annotations. More DL based approaches to matchmaking can be found in [9, 10].

Unlike [15] we consider privacy a major issue. Due to the nature of user profiles, we would like to encourage users to keep their private data on their personal device. This requires the mobile device to decide the matchmaking without support of external reasoners. As a consequence we developed the reasoner Pocket KRHyper [7] that runs on a mobile phone or similar devices.

The remainder of this paper presents some details of the software and hardware environment, explains how the DL reasoning is performed, finally gives some figures on performance compared to RacerPro 1.9[1] that is one of the highly developed desktop reasoners.

2 Environment

The environment of mobile phones poses several restrictions on the development of the reasoner. Resources like computing power, memory and energy are tightly limited. Although phones offer hundreds of MB for the storage of pictures, sounds, and ringtones the memory offered to applications is restricted to a few hundreds kB. This is a hard limitation imposed by the manufacturer, that cannot be configured. Even

[1] http://www.racer-systems.com

worse this small memory is used for the application code, the application data like the ontology and the working set of the current computation. Consequently whenever we had the choice between memory consumption or additional computation, we went for the additional computation. Computing power is reduced by a factor of 50 to 100 compared to an average desktop PC.

The widest spread language on phones is JAVA 2 micro edition (J2ME)[2]. J2ME is tailored for devices with limited resources. It offers a language comparable to standard JAVA 1.1.8. Unfortunately all modern container classes of JAVA2, that would ease development and speed up execution, are missing.

3 Reasoning

The reasoning is based on a first-order model generating reasoner implementing the hyper-tableaux calculus [1]. The use of a first-order tableaux may be a little surprise, but turns out to be viable, as can be seen in section 4. The reasoning task offers a timelimit and throws *Exceptions* upon timing out or running into memory shortages. Thus the application may react appropriately. The reasoner comes as a library to be integrated into J2ME applications (midlets).

3.1 Translation

To use the FO reasoner for DL purposes the knowledge base is transformed into a set of possibly disjunctive clauses. The approach is relational and preserves the open world assumption. Although the result is a logic program, the restrictions of DLP [3] do not apply.

The knowledge base is considered to be a finite set of axioms $C \sqsubseteq D$ and $C \equiv D$, where C, D are concepts of the DL ALC extended by inverse and transitive roles and role hierarchies.

The transformation into sets of clauses introduces subconcepts to reduce the complexity of the concept expression or axiom. As an example the axiom $\exists R.C \sqsubseteq \forall S.\forall T.D$ is decomposed into $\exists R.C \sqsubseteq sub_i$ and $sub_i \sqsubseteq \forall S.sub_j$ and $sub_j \sqsubseteq \forall T.D$ to comply with the transformation primitives. Table 1 gives the transformation primitives in abstract DL syntax, a corresponding first order formula, and the generated clauses.

The clauses marked with * share variables in a disjunctive head, they are not range-restricted. As an extension to previous work [12] Pocket KRHyper now handles these clauses by generating all ground instances up to the given term weight. Doing so often causes timeouts of the prover due to the limited resources. Clauses marked with ** are suitable for reasoning tasks in acyclic terminologies. Decidability commonly requires the tableau procedure to engage a blocking technique. The blocking techniques found in [4] may be adapted to the transformation as shown in [2].

[2] http://java.sun.com/j2me/

description logic	first order formula	clauses
$C \sqcap D \sqsubseteq E$	$\forall x.C(x) \wedge D(x) \rightarrow E(x)$	e(x) :- c(x), d(x).
$C \sqcup D \sqsubseteq E$	$\forall x.C(x) \vee D(x) \rightarrow E(x)$	e(x) :- c(x). e(x) :- d(x).
$C \sqsubseteq \neg D$	$\forall x.C(x) \rightarrow \neg D(x)$	false :- c(x), d(x).
$\exists R.C \sqsubseteq D$	$\forall x \forall y.R(x,y) \wedge C(y) \rightarrow D(x)$	d(x) :- c(y), r(x,y).
$\forall R.C \sqsubseteq D$	$\forall x.(\forall y.R(x,y) \rightarrow C(y)) \rightarrow D(x)$	d(x); r(x, f_{R-C}(x)). * d(x) :- c(f_{R-C}(x)).
$C \sqsubseteq D \sqcap E$	$\forall x.C(x) \rightarrow D(x) \wedge E(x)$	e(x) :- c(x). d(x) :- c(x).
$C \sqsubseteq D \sqcup E$	$\forall x.C(x) \rightarrow D(x) \vee E(x)$	e(x); d(x) :- c(x).
$\neg C \sqsubseteq D$	$\forall x.\neg C(x) \rightarrow D(x)$	c(x); d(x). *
$C \sqsubseteq \exists R.D$	$\forall x.C(x) \rightarrow (\exists y.R(x,y) \wedge D(y))$	d(f_{R-D}(x)) :- c(x). r(x, f_{R-D}(x)) :- c(x). **
$C \sqsubseteq \forall R.D$	$\forall x.C(x) \rightarrow (\forall y.R(x,y) \rightarrow D(y))$	d(y) :- c(x), r(x,y).
$R \sqsubseteq S$	$\forall x \forall y.R(x,y) \rightarrow S(x,y)$	s(x,y) :- r(x,y)
$R^- \equiv S$	$\forall x \forall y.R(x,y) \leftrightarrow S(y,x)$	s(y,x) :- r(x,y). r(x,y) :- s(y,x).
R^+	$\forall x \forall y \forall z.R(x,y) \wedge R(y,z) \rightarrow R(x,z)$	r(x,z) :- r(x,y), r(y,z).

Table 1. Translation Primitives

The effective test for satisfiability of a concept C inserts a single fact $C(a)$ into the knowledge base. C is satisfiable if the Pocket KRHyper finds a model for the knowledge base. A refutation indicates the unsatisfiability of the concept. Subsumption is reduced to satisfiability.

3.2 Optimizations

Common to DL reasoners are many optimizations [5]. In the following section some counterparts within the FO reasoner are presented.

Backjumping. Backjumping is a technique to reduce the unnecessary evaluation of branches, that do not contribute to the clash in a branch. In hyper-tableaux the 'hyper' eventually tracks back over multiple branching points to avoid the same.

Semantic Branching. This optimization is derived from DPLL. As hyper-tableaux is also a descendant from DPLL a corresponding technique known as complement splitting is in use. Semantic branching adds the complement of already inspected branches to subsequent branches. This avoids the redundant inspection of clashed terms. Different from semantic branching the complements of subsequent branches are added to the branches, that are inspected first. Because of the symmetry the effect is the same for cases where all branches are closed.

Boolean Constraint Propagation. The current mobile implementation of the hyper-tableaux tries to choose the disjunctive clause for the next expansion, that will immediately clash one of its branches. This is similar to BCP.

Lazy Unfolding. Beyond these optimizations a behavior similar to lazy unfolding is introduced by the transformation step. All subexpressions are named, and the transformation step tries to identify with already known subexpressions. The FO tableaux will expand these subexpressions only if no clash has been found prior to expansion.

Absorption. Absorption is left to the preprocessing of the knowledge base. It absorbs general concept inclusions (GCI) into the definition of concepts. For a DL tableaux this reduces the number of disjunctive branches. Here absorption reduces the number of cases, that require the enumeration of all ground instances, see section 4 for an example.

Reducing Memory Consumption. Currently the J2ME environment does not offer an preinstalled XML-parser. So we decided to use the lisp-like KRSS [11] syntax. The lisp-parser is significantly smaller than an XML-parser.

The clausal knowledge base is partitioned. The knowledge base keeps track of these parts by setting a mark at each such step, so it is possible to revert it to the state it had before an update or query. This allows for the addition of clauses generated for query processing and a subsequent removal of them. The repeated transformation of unchanged DL axioms to clauses is avoided. The query related clauses are added and retracted as required by the query processing.

Any form of indexing or caching requires additional memory. Because the memory constraints of the used platforms are tight, we dropped most of these enhancements. Consequently the current solution does not scale (wrt. the size of the model to search) as well as desktop reasoners.

3.3 Engineering

The development of a reasoning engine is a challenging and error-prone task. To cope with these difficulties we adopted the idea of unit testing. Unit testing automates the execution of (parts of) your software on predefined tests with known results. The ease of use and high degree of automation enables the early detection of software glitches and sideeffects of changes throughout the whole development cycle.

The tests are divided into a first-order part and a DL part. For the first-order reasoning kernel we used a subset of the TPTP [13]. These clausal problems are fed into the reasoner and evaluated by the FO tableaux. The result of the reasoning process is compared to the known result.

Starting with some 'benchmarks' taken from the DIG page[3] many own tests, that focus on specific implementation details, were added to test the DL part. The results in the following section are derived from scaled tests. The automation of the tests includes the comparison of predefined results with the actual result of the reasoner.

[3] http://dl.kr.org/dig/

test	knowledge base	comments
1	$C \equiv C_1; C_i \sqsubseteq C_{i+1}$	test the propagation of individuals and the ability to handle long branches
2	$C \equiv C_1; C_i \sqsubseteq C_{i+1}; C_{m+1} \sqsubseteq \bot$	like test1 but not satisfiable; test refutations on long branches
3	$C \equiv \exists R.C_1; C_i \sqsubseteq \exists R.C_{i+1}$	test the handling of role successors
4	$C \equiv \exists R.C_1; C_i \sqsubseteq \exists R.C_{i+1}; C_{m+1} \sqsubseteq \bot$	like test 3, but with a refutation at the end of the branch
5	$C \equiv \exists R.C_1; \exists R.C_i \sqsubseteq \exists R.C_{i+1}$	test the handling of role successors and concept inclusions
6	$C \equiv \exists R.C_1; \exists R.C_i \sqsubseteq \exists R.C_{i+1}; \exists R.C_{m+1} \sqsubseteq \bot$	like test 5 with a closing inclusion
7	$C \equiv \exists R.C_1; \exists R.C_i \sqsubseteq \exists R.C_{i+1}; \top \sqsubseteq \forall R.\neg C_{m+1}$	like test 6 but with differently formulated closing, that derives from a range restriction
8	$C \equiv \exists R.C_1; \exists R.C_i \sqsubseteq \exists R.C_{i+1}; \top \sqsubseteq \forall R.\bot$	similar to test 7 but branches may be closed prior to the evaluation of the inclusions
9	$C \equiv \exists R.C_0; \exists R.C_i \sqsubseteq \exists R.C_{i+1}; \top \sqsubseteq \forall R.\bot$	this time the inclusions are not related to the definition of C. Still not satisfiable.
10	$C \equiv \exists R.C_0; C_{i+1} \sqcap \forall R.C_{i+2} \sqsubseteq \exists R.C_i; C_{i+1} \sqsubseteq \exists R.C_{i+2}$	this test demonstrates the effect of not range-restricted clauses for FO model generators
11	$C \equiv \exists R.C_0; C_{i+1} \sqsubseteq \exists R.C_{i+2} \sqcap (\exists R.\neg C_{i+2} \sqcup \exists R.C_i)$	Absorption in a preprocessing step solves this problem
12	$C \equiv \exists R_1.C_1; R_i \sqsubseteq R_{i+1}$	test handling of role hierarchies
13	$C \equiv \exists R_1.C_1; R_i \sqsubseteq R_{i+1}; \top \sqsubseteq \forall R_{m+1}.\bot$	test refutation at the end of role hierarchies

Table 2. Testsuite

4 Comparison

Based on the test cases for DL reasoning a generator of tests was developed. These tests are used in multiple sizes wrt. the number of axioms to offer a view into the scalability of the reasoning solution. In the primary application of Pocket KRHyper all models or refutations were computed in short branches of a tableaux (less than 25 proof steps) because the applications ontology was about 200 concepts large but not so deeply nested.

To the best of my knowledge there are no competing reasoners on mobile phones. So I decided to compare the performance with RacerPro 1.9. The JAVA-library is compiled for the desktop version of JAVA (J2SE) without changes in the code. Consequently all indexing and caching is still missing even though the desktop environment would allow for the additional space. The integration of RacerPro into the automated test routine uses its socket interface. The syntax of the test cases is compatible due to the use of KRSS.

Out of more than 50 tests the following section describes about a dozen tests to give an idea of the characteristics of Pocket KRHyper. The tests presented here are limited to satisfiability tests with satisfiable *and* unsatisfiable concepts. This is suitable to describe the behavior of subsumption tasks with positive and negative outcome. Checking for subsumption and non-subsumption is not a trivial task for all first-order reasoners, who are possibly better suited to find refutations than models [14].

4.1 Testsuite

All tests are scaled to the sizes $m \in \{2,5,10,20,50\}$. Table 2 lists the tests, where $i \in \{1,2,...m\}$ for the corresponding size. *C* possibly indexed are concepts, *R* possibly indexed are roles. All 65 tests check for the satisfiability of concept *C*. A JAVA application runs all tests with Pocket KRHyper and RacerPro, compares the results and collects the execution time in ms for the reasoning tasks. Some comments in Table 2 indicate the targeted feature of the test.

The automated execution of these tests led to quite stable results wrt. the time needed. Surprisingly only Racer had some glitches, where every now and then but reproducible single tests performed quite bad beyond the time-limit of 1500ms. In the calculation of the graphical representation (see Fig. 1) of the runtimes the median of five runs was taken to eliminate this phenomenon. The times for the FO reasoner include the transformation into clauses. Table 3 gives exact values, — indicates a timeout. All times are in milliseconds taken on a contemporary personal computer running WindowsXP with Java Runtime 1.5 (J2SE). Reasoners are Pocket KRHyper via the library and RacerPro 1.9 via TCP-Socket. Due to limitations in precision all runtimes below 10ms are presented as $\leq$10ms.

4.2 Evaluation

Some weaknesses of the first-order approach are quite obvious. Test 10 is never solved within the timelimit. This is a consequence of the necessary enumeration of ground instances. The only solution to this problem is an improved preprocessing as test 11 indicates. As one might expect both tests are performing well with Racer. The only surprise is the increased runtime for test 10 with size 5 compared to the same test in size 10. An additional weakness is the scalability for tests 3 and 4. Following the transformation steps, a lot of functions are used. Pocket KRHyper tries to reduce the amount of storage and compares all new facts with existing ones in the current branch. This comparison is carried out by a unification method. This method suffers from the deeply nested functions and a lack of caching of intermediate results. In

m	test / reasoner	1	2	3	4	5	6	7	8	9	10	11	12	13
m=2	racer	20	10	10	10	10	10	20	10	20	10	10	10	10
	pKrh	30	$\leq$ 10	$\leq$ 10	$\leq$ 10	$\leq$ 10	$\leq$ 10	$\leq$ 10	$\leq$ 10	$\leq$ 10	—	$\leq$ 10	$\leq$ 10	$\leq$ 10
m=5	racer	20	10	20	20	20	30	20	10	20	90	20	10	20
	pKrh	$\leq$10	$\leq$ 10	20	30	10	11	$\leq$ 10	$\leq$ 10	$\leq$ 10	—	$\leq$ 10	$\leq$ 10	$\leq$ 10
m=10	racer	20	30	20	20	30	30	30	30	20	50	30	30	30
	pKrh	$\leq$10	$\leq$ 10	90	70	30	20	40	$\leq$ 10	30	—	$\leq$ 10	$\leq$ 10	$\leq$ 10
m=20	racer	50	40	40	40	50	50	50	71	40	90	60	40	41
	pKrh	30	$\leq$ 10	701	681	100	100	130	$\leq$ 10	$\leq$ 10	—	$\leq$ 10	20	20
m=50	racer	91	100	90	90	90	100	100	90	90	241	170	90	90
	pKrh	60	20	—	—	981	1001	1402	20	20	—	140	40	60

Table 3. Runtimes in comparision of Pocket KRHyper and RacerPro 1.9

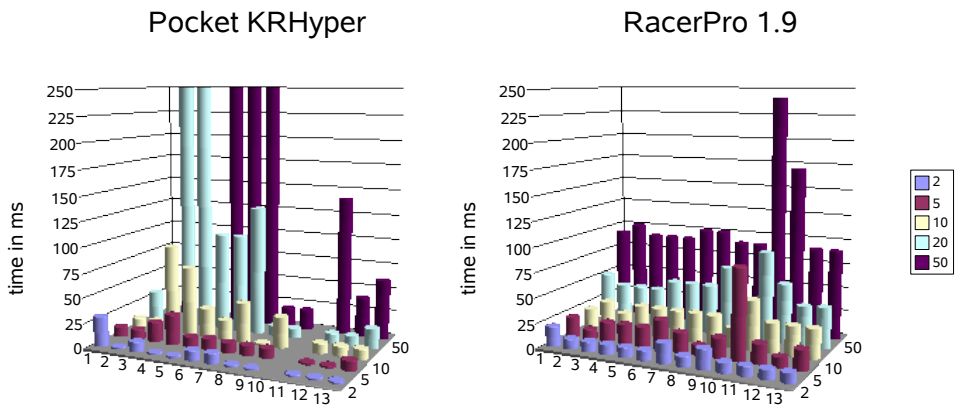


Fig. 1. Runtimes in milliseconds, max. 250ms, timelimit 1500ms

contrast Racer scales well for these two problems up to sizes between 100 and 200. Only above these limits Racer performs bad.

Besides these weaknesses the first-order model generation approach gives competitive runtimes compared to a well established DL reasoner. For small test cases (up to 10) and special cases (e.g. role hierarchies: test 12 and 13) the performance is usually better than Racer. This is not to speak of superior performance, because Racer is designed for a richer DL and has to maintain all indexes and caches, that will speed up subsequent queries into the same TBox. One of Pocket KRHyper s nice features is the symmetry of satisfiable and non-satisfiable problems or respectively subsumption testing. The lack of scalability is most of all a consequence of the dropped indexing. That was chosen deliberately because of the tightly limited memory in mobile phones.

Tests 8 and 9 demonstrate the *local* character of the FO tableaux and the transformation. The additional axioms in the knowledge base do not significantly influence the time for answering the satisfiability. This is an important property for the suitability into a resource scarce environment.

5 Conclusion

In this paper some details about the Pocket KRHyper approach to DL reasoning were presented. Although the mobile reasoner was successful in its original application a more thorough testing and comparison was due. Out of many tests 13 were described in detail. For many other features like inverse roles or explicit disjuncts the comparison revealed similar results. Overall the developed reasoner proved to be a useful tool. The need to look into very specific details of the implementation and not into the average capability or performance to solve problems motivated the disregard of existing benchmarks.

Of course Racer is not the only competing system[4]. To ease our purposes Racer supported the same syntax and I consider this a good basis for further developments. Especially because Racer is going to implement the SWRL rule language I will revisit the comparison. Rules are a natural extension for the first-order reasoner. In fact merely an interface has to be added. That was one of the reasons not to specialize the reasoning procedure for DL purposes. Furthermore the transformation is likely to include more features like normalizations and absorption in future versions. The current version of Pocket KRHyper is available for download at sourceforge[5].

Finally I would like to thank Racer-Systems for granting a license for the current version. Compared to version 1.7 many improvements make a difference in this comparison.

References

1. Peter Baumgartner. Hyper Tableaux — The Next Generation. Technical Report 32–97, Universität Koblenz-Landau, 1997.
2. Peter Baumgartner, Ulrich Furbach, Margret Gross-Hardt, and Thomas Kleemann. Model based deduction for database schema reasoning. In Susanne Biundo, Thom Frühwirth, and Günther Palm, editors, *KI 2004: Advances in Artificial Intelligence*, volume 3238, pages 168–182. Springer Verlag, 2004.
3. Benjamin N. Grosof, Ian Horrocks, Raphael Volz, and Stefan Decker. Description logic programs: Combining logic programs with description logic. In *Proc. of the Twelfth International World Wide Web Conference (WWW 2003)*, pages 48–57. ACM, 2003.
4. Volker Haarslev and Ralf Möller. Expressive ABox Reasoning with Number Restrictions, Role Hierarchies, and Transitively Closed Roles. In *KR2000: Principles of Knowledge Representation and Reasoning*, pages 273–284. Morgan Kaufmann, 2000.
5. I. Horrocks and P. F. Patel-Schneider. Optimising description logic subsumption. *Journal of Logic and Computation*, 9(3):267–293, 1999.
6. Thomas Kleemann and Alex Sinner. Description logic based matchmaking on mobile devices. In Joachim Baumeister and Dietmar Seipel, editors, *Proc. of 1st Workshop on Knowledge Engineering and Software Engineering - KESE2005, Koblenz*, ISSN 1860-4471, pages 37–48, 2005.
7. Thomas Kleemann and Alex Sinner. Krhyper - in your pocket, system description. In Robert Nieuwenhuis, editor, *Automated Deduction - CADE-20: 20th International Conference on Automated Deduction*, volume 3632 of *Lecture Notes in Computer Science*, pages 452–458. Springer Verlag, 2005.
8. Thomas Kleemann and Alex Sinner. User profiles and matchmaking on mobile phones. In Oscar Bartenstein, editor, *Proc. of 16th International Conference on Applications of Declarative Programming and Knowledge Management INAP2005, Fukuoka*, 2005.
9. Lei Li and Ian Horrocks. A software framework for matchmaking based on semantic web technology. In *Proc. of the Twelfth International World Wide Web Conference (WWW'2003)*, pages 331–339. ACM, 2003.
10. Tommaso Di Noia, Eugenio Di Sciascio, Francesco M. Donini, and Marina Mongiello. Abductive matchmaking using description logics. In *Proc. of IJCAI'03*, pages 337–342, 2003.
11. Peter F. Patel-Schneider and B. Swartout. Description-logic knowledge representation system specification, November 1993.
12. A. Sinner and T. Kleemann. Krhyper - in your pocket, system description. In *Proc. of Conference on Automated Deduction, CADE-20*. Springer Verlag, 2005.
13. G. Sutcliffe and C.B. Suttner. The TPTP Problem Library: CNF Release v1.2.1. *Journal of Automated Reasoning*, 21(2):177–203, 1998.
14. Dmitry Tsarkov and Ian Horrocks. DL reasoner vs. first-order prover. In *Proc. of the 2003 Description Logic Workshop (DL 2003)*, volume 81 of *CEUR (http://ceur-ws.org/)*, pages 152–159, 2003.
15. Wolfgang Wahlster. Smartweb: Mobile applications of the semantic web. In *GI Jahrestagung (1)*, pages 26–27, 2004.

[4] see http://www.cs.man.ac.uk/˜ sattler/reasoners.html for a comprehensive list

[5] http://sourceforge.net/projects/mobilereasoner

Part III
Posters

Euclidian Roles in Description Logics

Giorgos Stoilos and Giorgos Stamou
National and Technical University of Athens
Department of Electrical and Computer Engineering

Abstract

In the current paper we investigate the role of Euclidian roles in Description Logics.

1 Introduction

The last years much research effort has been spend towards increasing the expressiveness of Description Logics with respect to what can be said about roles. For example, in [2] the Description Logic $\mathcal{RIQ}$ is extended with several role axioms, like reflexive and irreflexive role axioms, disjoint role axioms and simple negation on roles. These extensions has motivated us to investigate possibilities of extending Description Logics with other role axioms. In the following we investigate the extension of DLs with Euclidian role axioms.

Euclidian roles are widely used in Modal Logics [1]. Formally, a role R is called Euclidian iff for each x, y, z, $R(x, y)$ and $R(x, z)$ imply $R(y, z)$. Syntactically in order for a DL language $\mathcal{L}$ to include Euclidian role axioms we have to extend the RBox to allow expressions of the form $\mathsf{Eucl}(R)$. The semantics of such axioms are derived quite straightforwardly. Given an interpretation $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$, $\mathcal{I}$ satisfies $\mathsf{Eucl}(R)$ if for all $x, y, z \in \Delta^{\mathcal{I}}$, $\{\langle x, y\rangle, \langle x, z\rangle\} \subseteq R^{\mathcal{I}} \rightarrow \langle y, z\rangle \in R^{\mathcal{I}}$. As with transitive role axioms, in order to efficiently handle Euclidian roles we have to understand the Euclidian closure of a relation R and handle properly value restrictions, $\forall R.C$. In Figure 1 (a) we can see the Euclidian closure of a relation R. As we can see the models of such role axioms include a root node, while all other nodes are interconnected with each other plus with themselves. That is because from $R(x, y) \wedge R(x, y)$ we derive $R(y, y)$. It is important to note here that the inverse of a Euclidian role is not necessarily Euclidian. An interesting question that is raised about Euclidian roles is if the can or can't be used in qualified number restrictions. We remind here that if transitive roles are used in QCRs then the resulting logic is undecidable. It turns out that similarly if

Euclidian roles are used in number restrictions, then the language is undecidable. More interestingly, if the inverse of a Euclidian role is used in a QCR, then the language is again undecidable, although the inverse of a Euclidian role is not necessarily Euclidian. The proof of this claim uses a reduction from the classical domino problem. Figure 1 (b), (c) and (d) illustrate the visualization of the grid and role hierarchy, where S^{ε}_{ij} are Euclidian roles.

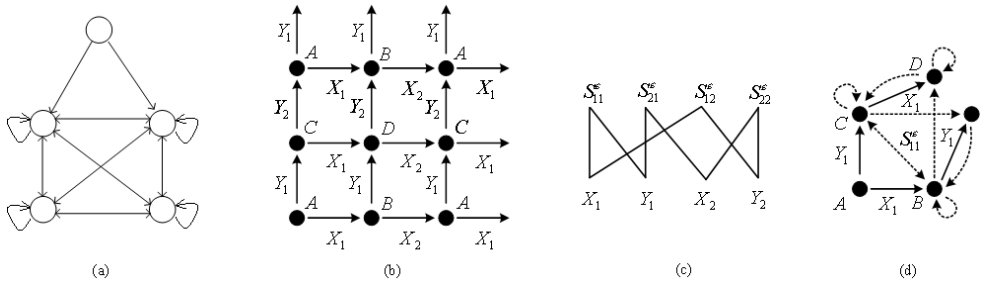


Figure 1: The (a) Euclidean closure, (b) grid, (c) role hierarchy and (d) coincidence enforcing

As it happens with many role axioms of $\mathcal{SRIQ}$ [2], it comes to no surprise that Euclidian roles are a syntactic sugar for $\mathcal{SRIQ}$. More precisely, an axiom $\mathsf{Eucl}(R)$ can be represented by the RIA $R^{-}R \sqsubseteq R$. This means that if $\langle x, y\rangle \in (R^{-})^{\mathcal{I}}$ (or $\langle y, x\rangle \in R^{\mathcal{I}}$) and $\langle y, z\rangle \in R^{\mathcal{I}}$, then $\langle x, z\rangle \in R^{\mathcal{I}}$, which capture the semantics of Euclidian roles. Moreover, in [2] the definition of simple roles is also extended to cover the new expressive means of the language. As it was expected, according to the definition in [2], a role R that is defined by the Euclidianity RIA is not simple, since it is defined in terms of itself and its inverse, hence Euclidian roles are not simple and cannot participate in QCRs. Additionally, an inverse role R^{-} is simple if R is [2], which agrees with our comment about the inverses of Euclidian roles.

References

[1] J.Y. Halpern and Y.Moses. A guide to completeness and complexity for modal logics of knowledge and belief. *Artificial Intelligence*, 54(3):319–379, 1992.

[2] I. Horrocks, O. Kutz, and U. Sattler. The irresistible $\mathcal{SRIQ}$. Proc. of the International Workshop on OWL: Experiences and Directions, 2005.

Experiences with Load Balancing and Caching for Semantic Web Applications

Alissa Kaplunova, Atila Kaya, Ralf Möller
Hamburg University of Technology (TUHH)
al.kaplunova|at.kaya|r.f.moeller@tu-harburg.de

1 Introduction

In our work we consider applications which generate queries w.r.t. many different knowledge bases. We presuppose that for a particular KB there exists many possible query servers. In order to successfully build applications that exploit these KB servers, an appropriate middleware is required. In particular, if there are many servers for a specific KB, the middleware is responsible for managing request dispatching and load balancing. Load balancing must be accompanied by middleware-side caching in order to reduce network latency.

In our view the KB servers we consider are managed by different organizations. Therefore, DL applications used in some company need some gateway inference server that provides local caching (in the intranet) to: (i) reduce external communication and (ii) avoid repetitive external server access operations in case multiple intranet applications pose the same queries.

In our case study we investigate a server for answering OWL-QL$^-$ queries[1]. This server (called RacerManager) acts as a proxy that delegates queries to back-end DL reasoners (RacerPro servers) that manage the KB mentioned in the query and load KBs on demand. Compared to previous versions, the functionality of RacerManager has been substantially enhanced. We address the problems of load balancing and caching strategies in order to exploit previous query results (possibly produced by different users of the local site). Caching is investigated in the presence of incrementally answered OWL-QL$^-$ queries. In addition, the effects of concurrent query executions on multiple (external) inference servers and corresponding transmissions of multiple partial result sets for queries are studied.

2 OWL-QL$^-$ Server as a Middleware

Reasoning over ontologies with a large number of individuals in ABoxes is a big challenge for existing reasoners. To deal with this problem, RacerPro supports iterative query answering, where clients may request partial result sets in the form of tuples. For iterative query answering, RacerPro can be configured to compute the next tuples on demand (lazy mode). Moreover, it can be instructed to return cheap (easily inferable) tuples first.

Although these configuration options enable the reasoner to achieve significant performance improvements for a single client, this effect decreases in scenarios where multiple clients pose queries concurrently. In fact, a single RacerPro instance cannot process several client requests in parallel. Thus, as long as RacerPro is processing a clients request, which usually includes activities such as parsing the query, reading the corresponding knowledge base, classifying it, finding requested number of answer tuples and returning them, all other clients have to wait in a queue.

[1] OWL-QL$^-$ stands for OWL-QL with distinguished variables only.

Motivated by the concurrency problem, our OWL-QL$^-$ server is implemented to act as a load-balancing middleware between clients and multiple RacerPro instances. We chose a common n-tier architecture as the base layout. RacerManager can initialize and manage an array of RacerPro instances. Multiple clients can use the web service offered by RacerManager to send their OWL-QL$^-$ queries concurrently. With respect to the states of the managed RacerPro instances and a naive load-balancing strategy (similar to round-robin), RacerManager dispatches the queries to RacerPro instances. More precisely, given a query, which requires some ontology, RacerManager prefers RacerPro instances, which already worked on this ontology. Before a OWL-QL$^-$ query is send to a reasoner instance, it is translated to the new Racer Query Language (nRQL) by the translator module. Preliminary test results showed that, the proposed architecture prevents clients from blocking each other, as it is the case if multiple clients interact with a single reasoner.

Additionally, irrespective of load balancing and query dispatching, a client may benefit from the caching mechanism offered by RacerManager. In case he sends a query, which has been posed before, answer tuples are delivered directly from the cache. If the client requires more tuples than available in the cache, only the missing number of tuples are requested from an appropriate RacerPro instance. The cache expiration can be set to an arbitrary duration or turned off. In the latter case, the cache will never be cleared.

3 Features and Limitations

When developing the OWL-QL$^-$ server, our main goal was to ensure scalability and high performance. Therefore we only take into account OWL-QL features, that could be efficiently implemented in the middleware tier. In fact, some features such as may-bind variables implemented by the middleware using rolling-up techniques and disjunctive queries would result in excessive computing and increased communication with reasoners. In our opinion, they should be supported by reasoners directly.

We presuppose that for many Semantic Web applications, conjunctive queries with must-bind (distinguished) variables will be enough. Therefore, our current implementation supports such queries. Furthermore, it handles queries that include a premise (if-then queries). The server does not support scenarios where multiple knowledge bases are to be used to answer a query or where a knowledge base is not specified by the client. Furthermore, RacerManager does not provide proofs about the reasoning made.

4 Future Work

In our future work we will conduct comprehensive experiments in order to empirically evaluate RacerManager. Moreover, we will compare alternative strategies to determine more efficient load balancing and cache usage algorithms.

It is obvious, that in some situations the single RacerManager server may itself become a bottleneck. One possible solution is to replicate it and to set up a HTTP or hardware dispatcher in front of RacerManager instances.

From our point of view, there are some open questions regarding the implementation of the OWL-QL standard which we will address in the future, e.g.: (i) How should queries without a KB reference be handled in different scenarios? (ii) In which scenarios should clients share results obtained by other clients? (iii) Is expiration of query results important for applications? (iv) Are subscription services required?

How sensor data interpretation could benefit from description logics: Position paper

Sylvia Melzer, Ralf Möller
Hamburg University of Technology, Germany
{sylvia.melzer, r.f.moeller}@tuhh.de

In many areas the interest in real-time and content-based data dissemination is increasing. Sensor data might be sent to an interested client automatically (via publish/subscribe facilities) or might be downloaded on demand by sending a query (pull model). Data dissemination and data access are important aspects in a sensor-based environment and both have a wide influence on the architectual design for an individual sensor data supply. Even simple sensors provide individual records of data by supporting web service calls (maybe via a proxy). In this scenario web service architectures are faced with high data rates, large profile population, variable query life span and high result volume. Technical details and an implementation of a service-based publish/subscribe communication architecture is described in detail in [2].

Content-based publish/subscribe systems support the required flexibility to interpret complex queries. Figure 1 illustrates a publish/subscribe system for sensor data information.

Sensors deliver sensor data like multimedia data to the broker system which is a service. Based on the content of information and the user profiles (called demand

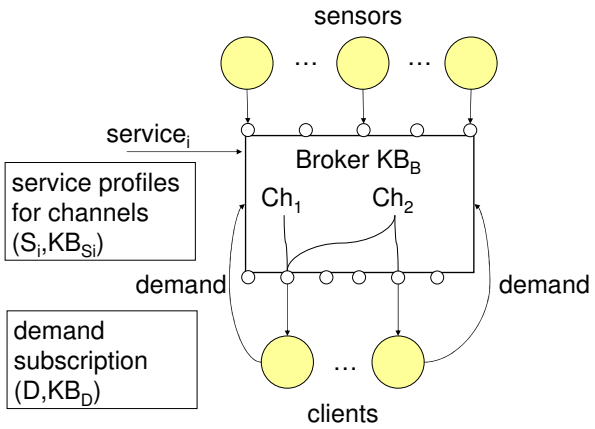


Figure 1: publish/subscribe system

subscriptions, or demands D for short) the broker system delivers information to the client. Clients subscribe to the service with their demand profiles. After low-level filtering, aggregating, and combining raw sensor data, the broker component provides certain sensor data information channels (Ch_i). Channels are described with some profiles (called service profiles (S)) as well. Both parts, services and demands are described with tuples (S_i, KB_{S_i}) or (D_i, KB_{D_i}). All knowledge bases $KB_{S/D}$ refer to some broker-specific background knowledge base KB_B.

If a certain demand subscription is sent to the broker, the problem is to find relevant channel-based services such that match with the demand specification. In the literature, several formalizations of this process are discussed [1, 3]:

- Satisfiability of the conjunction $S_i \sqcap D_i$ of supply and demand description with respect to $KB_{S_i} \cup KB_{D_i} \cup KB_B$.
- Subsumption $S_i \sqsubseteq D_i$ between supply and demand descriptions w.r.t. a background ontology $KB_{S_i} \cup KB_{D_i} \cup KB_B$.

In our scenario, the situation is slightly more complicated because some demands can only be fulfilled if some channels are combined (see the combination of Ch_1 and Ch_2 for the left client in Figure 1). In addition, maybe some filtering has to be done at client-side because there is no perfect match with existing services (channels) and incoming demands. In this context a feedback process (i.e. output data is fed back to the input of the system), currently discussed in BOEMIE[1], proves that clients get more relevant and enriched information because of the evolution of knowledge bases and the fusion of information from different channels in a bootstrapping fashion. In the e-commerce literature, service and demand adaptions have already been investigated in terms of service negotiation. According to this view, abduction and contraction operators can be used to configure supplies such that they match modified demands. This theory also provides a foundation for channel fusion (service modification) and a posteriori client-side filtering (demands taylored towards available services).

References

[1] Lei Li and Ian Horrocks. A software framework for matchmaking based on semantic web technology. *Int. J. of Electronic Commerce*, 8(4):39–60, 2004.

[2] Sylvia Melzer. A content-based publish-subscribe architecture for individualized sensor data supply (in German). Master thesis, Hamburg University of Technology, January 2006.

[3] David Trastour, Claudio Bartolini, and Javier Gonzalez-Castillo. A semantic web approach to service description for matchmaking of services. In *Proceedings of the International Semantic Web Working Symposium (SWWS)*, 2001.

[1]BOEMIE (Bootstrapping Ontology Evolution with Multimedia Information Extraction) is an IST 6th Framework Programme Project (FP6-027538)

Cost-efficient web service composition for processes with distributed retrieval queries: Position paper

Irma Sofia Espinosa Peraldi and Ralf Möller
Hamburg University of Technology
Software Systems Institute (STS)
Hamburg, Germany
sofia.espinosa@tuhh.de, r.f.moeller@tuhh.de

1 Problem description

In the near future, description logic reasoning services will be offered as web services and initial versions of, for instance, OWL QL web services have already been developed (e.g., [1]). Due to the high acceptance of web service technologies to support processes across heterogeneous architectures, the amount of web service offering similar functionality is proliferating. Thus, the problem of selecting a particular service binding in order to minimize resource consumption in a web service composition has emerged. In particular we investigate optimization strategies for web service compositions that define *large-scale inference processes* with *query correlation*. Due to the business logic behind the process, there is an implicit knowledge about the enquired domain, the processing order of the queries and if a query subsumes another. Furthermore, the web service partners of such a composition provide reasoning services over (possibly heterogeneous) ontologies. Thus, the optimization criteria we are interested in is very much service-dependent, where reasoning tasks with query subsumption, optimization techniques with Abox indexing and KB availability are being considered.

2 Approach

The optimization problem is expressed in terms of a *configuration problem* in which the web service partners (DL reasoners) become objects to be combined such that they satisfy the given criteria. The optimization algorithm was designed such that it can handle multiple criteria expressed as constraints, which

are hard criteria that can not be relaxed, and objectives with no total order, of which a minimum or maximum number should not be violated. Furthermore, the considered criteria have dynamic values, due to the exposure of the web services partners to other (competing) calls for instance retrieval. Thus, it is not possible to a priori determine an execution plan. Therefore, we pursued an approach proposed by [3] using the notion of Pareto dominance, for local optimization, where for each invocation step in the composition, the set of given web service partners are being compared against the multiple objectives along the dimensions. A relaxation process is applied in case of over constraining. For details on the implementation see [4].

3 Contribution

Until now no solution has been presented to optimize large-scale inference processes (represented as web service compositions), where on the one side, the correlation of the process's queries is being considered, and on the other side, reasoning tasks and optimization techniques offered by several DL reasoning services (representing the web service partners of the composition), is being used as criteria for optimizing resource consumption. In the current implementation only RacerPro[2] reasoners can be used to obtain required information about loaded Tboxes and Aboxes, about Abox index structures being computed, about whether a new query is subsumed by another query already answered by some server etc. However, since web service architecture are more and more accepted, current proposals for DL reasoner communication languages such as DIG might be extended with corresponding facilities in the future.

References

[1] J. Galinski, A. Kaya, and R. Möller. Development of a server to support the formal semantic web query language OWL-QL. In I. Horrocks, U. Sattler, and F. Wolter, editors, *Proc. International Workshop on Description Logics*, 2005.

[2] Racer Systems GmbH & Co. KG. Racerpro 1.9. WWW page, March 2005. http://www.racer-systems.com (10.03.06).

[3] D. Navinchandra. *Exploration and innovation in Design. Towards a Computational Model*. Springer-Verlag, 1 edition, 1991.

[4] I. S. Espinosa Peraldi. Cost-efficient web service compositions. Master thesis, TU Hamburg-Harburg, June 2005.

Static Knowledge Representation of Multi-Agent System's Specification by Description Logics

Haiyan Che[1], Jigui Sun[1,2] and Haibo Yu[1]
[1]College of Computer Science and Technology, Jilin University
Changchun 130012, China
{chehy,jgsun,yuhb}@jlu.edu.cn

[2]Key Laboratory of Symbolic Computation and
Knowledge Engineering of Ministry of Education, Jilin University
Changchun 130012, China

Abstract

Modularity and rigor are two key elements for multi-agent technology. Hong Zhu's multi-agent system (MAS) development method provides proper language facilities supporting modularity. To enhance this method with rigor advocates a DL method to map the specification of MAS into a DL TBox. Thus, we can use the existing DL reasoners and systems to verify and validate some system's properties.

1 Introduction

Agent technology has been predicted to be the next generation mainstream computation paradigm. However, lack of rigor and language facilities directly supporting modularity and abstract mechanisms hamper the wide-scale adoption of it. Hong Zhu et al. contribute on the research of such language facilities. They advocate a model driven method for MAS development, which designs and implements CAMLE (Caste-centric Agent-oriented Modeling Language and Environment) [1] providing tools for constructing graphic MAS models, automatically checking consistency between various views and models, and automatically transforming these models into formal specifications in SLABS [2]. To enhance Hong Zhu's method with rigor, we propose a DL method to map the specification of MAS written in SLABS into a DL TBox. Taking advantage of DL's decidability and its existing reasoners and systems, we can perform certain reasonings about the system's properties.

2 Static Knowledge Representation with Description Logic $\mathcal{ALCNIF}_{reg}$

The most important language facilities in SLABS are caste, agent and scenario. A MAS is regarded as a set of agents, and castes are the classifiers of agents, or the roles agents play. They define templates of the structure and behavior characteristics of agents. Scenarios are defined as the behavior patterns of the agents in an agent's environment, perceiving which the agent can decide its action rather than driven by message communications. To represent the static part of the MAS's specification we choose $\mathcal{ALCNIF}_{reg}$, which extends $\mathcal{ALC}$ with unqualified number restrictions ($\mathcal{N}$), inverse roles ($\mathcal{I}$), agreement ($\mathcal{F}$), and the composition of roles (reg).Based on the translation method from object-oriented model to DLs proposed by Calvanese et al. [3] we define a map from SLABS to DLs, which maps each caste and agent definition to corresponding concept, and depicts the instance relationship between agent and caste by the concept constructor of role agreement.

3 Conclusion

This paper advocates a DL method to formalize the specification and perform certain reasonings about the properties of MAS. $\mathcal{ALCNIF}_{reg}$ is chose to represent the static part of the specification and the satisfiability of concepts w.r.t. acyclic TBoxes is decidable with the complexity of NExpTime-complete [4]. To make our method practicable we need extend basic DL to represent dynamic knowledge as our further work.

References

[1] Zhu, H. and Shan, L., Caste-Centric Modelling of Multi-Agent Systems: The CAMLE Modelling Language and Automated Tools, in Beydeda, S. and Gruhn, V. (eds) Model-driven Software Development, Research and Practice in Software Engineering, Vol. II, Springer (2005) 57-89

[2] Zhu, H., "SLABS: A Formal Specification Language for Agent-Based Systems", Int. J. of Software Engineering and Knowledge Engineering 11(5) (2001) 529-558

[3] Diego Calvanese, Maurizio Lenzerini, and Daniele Nardi. Unifying class-based representation formalisms. J. of Artificial Intelligence Research, 11 (1999) 199-240

[4] F. Baader, D. McGuinness, D. Nardi, and P. Patel-Schneider. The Description Logic Handbook: Theory, Implementation and Applications, Cambridge University Press (2003)

A Tableaux-based Mobile DL Reasoner

F. Müller and M. Hanselmann and T. Liebig and O. Noppens
Ulm University, D-89069 Ulm, Germany
{felix.mueller|michael.hanselmann}@uni-ulm.de

This is a preliminary experience report about designing and implementing a tableaux-based DL reasoner suitable to run on a mobile device. For now the reasoning system only offers pure TBox subsumption. Currently there is no preprocessing for efficient taxonomy computation nor ABox support. The reasoner supports the DIG standard for client communication.

Nowadays most mobile devices are capable of running Java programs using the Micro Edition of the Java 2 platform (J2ME). In order to gain valuable experience in the course of developing a mobile DL reasoner we took the most ambiguous challenge by choosing the greatest common environment, namely pure J2ME, on very limited devices, namely mobile phones.

Our approach is based on the well-known tableaux algorithm which is used in most modern DL reasoning systems. This has the advantage of starting with a relatively simple base algorithm which can be successively extended in order to support additional language constructs. The mobile reasoner currently can handle $\mathcal{ALCN}$ KBs with unique definitions.

Experience has shown that a wide range of optimization techniques typically improve the performance of a naive tableaux implementation by orders of magnitude [1]. The most important ones are lazy unfolding, absorption and dependency directed backtracking. Not all of them are relevant within the current implementation due to the restricted expressivity of the supported language (e. g. absorption can be dropped in absence of GCIs).

Within our mobile setting of very restricted memory and processing resources it is not only important to utilize those optimizations but to implement them in a resource-saving way. In this respect, the right choice of lean data structures is of exceptional importance. Our solution consists of an array based index model in order to implement an extended lazy unfolding technique called *normalizing* and *naming/tagging* (e.g. see [2]) very efficiently. Naming means that all concept expressions and sub-expressions are recursively indexed with an unambiguous identifier. In order to save memory we use integers as identifiers for concepts as well as expressions in contrast to strings or even Java objects. For example, an expression $D \sqcap \forall r.E$ may be successively named as follows: $3 \rightarrow (D \sqcap \forall r.E)$,

$4 \rightarrow D$, and $5 \rightarrow \forall r.E$ (E will then be named within a recursive call). Note that before introducing a new name the algorithm has to look-up whether this expression has already been named. A negated expression will receive a negative integer. All named TBox axioms, sub-axioms and elements have an associated integer matrix encoding their definition. Consider the TBox axiom $A \equiv D \sqcap \forall r.E$ and $6 \rightarrow E, 1 \rightarrow A$. As a result of the naming phase the following integer arrays (among others) are generated: for 1 (resp. A): $\boxed{k_n}\boxed{4}\boxed{5}$ where k_n is an integer which encodes that the following entries are conjunctively connected. 5 ($\forall r.E$) in turn is represented with $\boxed{k_m}\boxed{6}$ where k_m encodes a universal quantification over a particular role, namely r (we use a special bit encoding in order to map language constructor, role, etc. into one integer). It can easily be seen that this reflects the original axiom by replacing the identifiers with their given expressions.

The encoding from above is of advantage for detection of obvious syntactical clashes even between complex expressions which is known to be one of the most effective optimizations [1]. For example, we can use a fast integer operation in order to detect A and $\neg A$ within a tableaux node simply by locally sorting it and checking whether there are two succeeding elements whose sum is 0.

Our implementation successfully runs using the freely available Sun Wireless Toolkit emulator software as well as on real mobile phones. In order to test our reasoner we added a DIG/1.1 interface which allows to use Protégé as a front-end within the emulator setting. For linking Protégé with the reasoner running on a real phone we developed a small desktop application which serves as a proxy. More precisely, the proxy receives HTTP DIG messages from a client and will pass those via Bluetooth to the mobile reasoner.

To our knowledge, there is no related work except Pocket KRHyper [3] a port of a subset of the hyper tableau calculus to a mobile phone. Pocket KRHyper covers unfoldable $\mathcal{ALCI}$ TBoxes extended by role hierarchies using a variant of the KRSS syntax. As far as can be seen within the log files of the system, FaCT++ seems to utilize a similar internal naming approach.

References

[1] Ian Horrocks. Applications of Description Logics: State of the Art and Research Challenges. In *Proc. of the 13th Int. Conf. on Conceptual Structures (ICCS'05)*, pages 78–90, 2005.

[2] Ian Horrocks and Peter Patel-Schneider. Optimising Description Logic Subsumption. *Journal of Logic and Computation*, 9(3):267–293, June 1999.

[3] Alex Sinner and Thomas Kleemann. KRHyper – In Your Pocket. In *Proc. of the International Conference on Automated Deduction (CADE-20)*, volume 3632 of *LNCS*, pages 452–458. Springer Verlag, 2005.

Computing Maximally Satisfiable Terminologies for the Description Logic $\mathcal{ALC}$ with GCIs

Kevin Lee and Thomas Meyer
NICTA and University of New South Wales, Sydney, Australia
{kevin.lee, thomas.meyer}@nicta.com.au

Jeff Z. Pan
University of Aberdeen, Aberdeen, UK
jpan@csd.abdn.ac.uk

1 Introduction

Existing description logic reasoners provide the means to detect logical errors in ontologies, but lack the capability to resolve them. We present a tableau-based algorithm for computing maximally satisfiable terminologies in $\mathcal{ALC}$. Our main contribution is the ability of the algorithm to handle GCIs, using a refined blocking condition that ensures termination is achieved at the right point during the expansion process. Our work is closely related to that of [1], which considered the same problem for assertional (Abox) statements only, and [2], which deals only with unfoldable terminologies for $\mathcal{ALC}$.

2 Reasoning with Unsatisfiability

The algorithm receives as input a Tbox $\mathcal{T}$ and a concept name A in $\mathcal{ALC}$, and returns as output the maximally A-satisfiable subsets (A-MSSs) of $\mathcal{T}$. Each sentence in $\mathcal{T}$ is labelled with a unique propositional atom, hence $\mathcal{T}$ will be a set of labelled axioms of the form $(C \sqsubseteq D)^p$ and $(C \doteq D)^p$. Our algorithm starts by creating an Abox $\mathcal{A}$ containing only the labelled assertion $A(x)^\top$, where x is an individual name. It then proceeds by continuously applying the expansion rules below, first to $\mathcal{A}$, and then to the Aboxes created subsequently, until none of the rules are applicable. Note that clash-detection is not a condition for termination. Also, we assume that all concept assertions are in negation normal form. The expansion rules make use of the following abbreviations and definitions: $\mathcal{A} \oplus C(x)^\phi$ stands for: $(\mathcal{A} \setminus \{C(x)^\psi\}) \cup \{C(x)^{\phi \vee \psi}\}$ if $C(x)^\psi \in \mathcal{A}$, and $\mathcal{A} \cup \{C(x)^\phi\}$ otherwise. Similarly, $\mathcal{A} \oplus R(x,y)^\phi$ stands for: $(\mathcal{A} \setminus \{R(x,y)^\psi\}) \cup \{R(x,y)^{\phi \vee \psi}\}$ if $R(x,y)^\psi \in \mathcal{A}$, and $\mathcal{A} \cup \{R(x,y)^\phi\}$ otherwise. We refer to a labelled concept assertion $C(x)^\phi$ as $\mathcal{A}$-*insertable* iff, whenever there is a ψ such that $C(x)^\psi \in \mathcal{A}$, then

$\sqcap$-rule	**if** $C_1 \sqcap C_2(x)^\phi \in \mathcal{A}$, and either $C_1(x)^\phi$ or $C_2(x)^\phi$ is $\mathcal{A}$-insertable, **then** $\mathcal{A}' := (\mathcal{A} \oplus C_1(x)^\phi) \oplus C_2(x)^\phi$.
$\sqcup$-rule	**if** $C_1 \sqcup C_2(x)^\phi \in \mathcal{A}$, and both $C_1(x)^\phi$ and $C_2(x)^\phi$ are $\mathcal{A}$-insertable, **then** $\mathcal{A}' := \mathcal{A} \oplus C_1(x)^\phi$, $\mathcal{A}'' := \mathcal{A} \oplus C_2(x)^\phi$.
$\exists$-rule	**if** $\exists R.C(x)^\phi \in \mathcal{A}$, x is not blocked, and either $R(x,y)^\phi$ or $C(y)^\phi$ is $\mathcal{A}$-insertable, **then** $\mathcal{A}' := (\mathcal{A} \oplus R(x,y)^\phi) \oplus C(y)^\phi$, where y is a new individual name and $y > y'$ for all individual names y' in $\mathcal{A}$.
$\forall$-rule	**if** $\{\forall R.C(x)^\phi, R(x,y)^\psi\} \subseteq \mathcal{A}$, and $C(y)^{\phi \wedge \psi}$ is $\mathcal{A}$-insertable, **then** $\mathcal{A}' := \mathcal{A} \oplus C(y)^{\phi \wedge \psi}$.
$\sqsubseteq$-rule	**if** $(C \sqsubseteq D)^\phi \in \mathcal{T}$, $\neg C \sqcup D(x)^\phi$ is $\mathcal{A}$-insertable for some individual name x, **then** $\mathcal{A}' := \mathcal{A} \oplus \neg C \sqcup D(x)^\phi$.
$\doteq$-rule	**if** $(C \doteq D)^\phi \in \mathcal{T}$, $(\neg C \sqcup D) \sqcap (C \sqcup \neg D)(x)^\phi$ is $\mathcal{A}$-insertable for some individual name x, **then** $\mathcal{A}' := \mathcal{A} \oplus (\neg C \sqcup D) \sqcap (C \sqcup \neg D)(x)^\phi$.

$\phi \not\models \psi$. After constructing an expansion tree using the above expansion rules, we compute a propositional formula called the *clash-resolve* formula as follows: Suppose $\mathcal{A}_1, \ldots, \mathcal{A}_n$ are the complete Aboxes obtained from the expansion. A particular clash $\{C(x)^{\phi_1}, \neg C(x)^{\phi_2}\} \subseteq \mathcal{A}_i$ is expressed by the propositional formula $\phi_1 \wedge \phi_2$. Now suppose there are k_i clashes in $\mathcal{A}_i$, and let $\psi_{i,1}, \ldots, \psi_{i,k_i}$ be the formulas expressing all the clashes in $\mathcal{A}_i$. The clash-resolve formula associated with $\mathcal{T}$ and A is: $\bigvee_{i=1}^{n} \bigwedge_{j=1}^{k_i} \neg\psi_{i,j}$. To compute the A-MSSs, simply find the prime implicants of the clash-resolve formula. Each prime implicant is of the form $(\neg p_1 \wedge \ldots \wedge \neg p_m)$, and the corresponding A-MSS can be found by removing from the original set of axioms $\mathcal{T}$ the axioms whose labels are $p_1, \ldots, p_m$.

Our result also show that classical blocking does not always block correctly at the right point, hence it will not always yield the desired results. The reason is that the labels associated with sentences are not taken into account when blocking is performed. Therefore, we define the *refined* blocking condition as follows: An individual y is *blocked* by x iff $y > x$ and for every $C(y)^\psi \in \mathcal{A}$, it is the case that $C(x)^\phi \in \mathcal{A}$ for some ϕ such that $\psi \models \phi$.

3 Conclusion and Future Work

We presented an algorithm for computing maximally satisfiable terminologies for description logic represented in $\mathcal{ALC}$. Unlike the existing algorithms, our proposed algorithm could also handle GCIs. We outlined some limitations with the classic subset blocking in the labelled satifiablity algorithm and addressed these issues by proposing a refined blocking condition. For future work, we will investigate on extending our algorithm to more expressive description logics, such as $\mathcal{SI}$ and $\mathcal{ALCN}$.

References

[1] Franz Baader and Bernhard Hollunder. Embedding Defaults into Terminological Knowledge Representational Formalisms. *Journal of Automated Reasoning*, 14:149–180, 1995.

[2] Stefan Schlobach. Diagnosing terminologies. In *Proceedings of AAAI05*, pages 670–675, 2005.

Pellet System Description

Evren Sirin and Bijan Parsia
Maryland Information and Network Dynamics Lab,
8400 Baltimore Avenue, College Park, MD, 20740 USA
evren@cs.umd.edu, bparsia@isr.umd.edu

The description logic $\mathcal{SHOIN}(\mathcal{D})$ has attracted considerable interest as the foundation of the W3C standard Web Ontology Language variant, OWL-DL. Pellet is a sound and complete tableau reasoner for $\mathcal{SHOIN}(\mathcal{D})$ and incorporates a number of key features such as conjunctive ABox query, axiom pinpointing, rules, $\mathcal{E}$-connection reasoning, and novel optimizations for nominals. In this paper we summarize Pellet's features and special capabilities.

Novel Optimizations Pellet implements most of the state of the art optimization techniques provided in the DL literature, e.g. *absorption*, *dependency-directed backjumping*, etc. In addition, there are also many novel optimization techniques implemented in Pellet especially to deal with nominals and large ABoxes. See [3] for details about these techniques.

Axiom pinpointing Axiom pinpointing is a non-standard DL inference service that provides a *justification* for any arbitrary entailment derived by a reasoner from an OWL-DL knowledge base. Given a KB and any of its logical consequences, the axiom pinpointing service determines the premises in the KB that are sufficient for the entailment to hold. The justification is useful for understanding the output of the reasoner, which is key for many tasks, such as ontology debugging, design and evolution. See [2] for more details about axiom pinpointing service in Pellet.

Conjunctive ABox query Query answering is yet another important feature for Semantic Web. Pellet includes a query engine that answers conjunctive ABox queries. In the presence of *non-distinguished variables*, the well-known "rolling-up" technique is used to answer tree-shaped queries. If there are only distinguished variables then every query atom can be answered in isolation and arbitrary shaped queries can be handled. See [4] for an explanation of the *query simplification* and *query reordering* techniques implemented in Pellet.

$\mathcal{E}$-Connections $\mathcal{E}$-Connections are a framework for combining several families of decidable logics. In [1] we have proposed tableau algorithms for different $\mathcal{E}$-Connection languages involving Description Logics so that an ontology can be refer to another ontology without fully importing the contents of that ontology. Pellet has been extended with tableau-based decision procedures for $\mathcal{E}$-Connection languages involving combinations of $\mathcal{SHOIN}(\mathcal{D})$ ontologies [1].

Rules Pellet has support for $\mathcal{AL}$-log (Datalog + $\mathcal{SHOIN}(\mathcal{D})$) via a coupling with a Datalog reasoner. It incorporates the traditional $\mathcal{AL}$-log algorithm and a new precompilation technique that is incomplete but more efficient. Pellet also has an experimental implementation of a direct tableau algorithm for a DL-safe rules extension to $\mathcal{SHOIN}(\mathcal{D})$ [5]. Preliminary empirical results have been encouraging and we think that the DL-safe implementation is practical for small to mid-sized ontologies esp. when the full expressivity of $\mathcal{SHOIN}(\mathcal{D})$ is needed.

Usability Pellet provides many different interfaces. Pellet has a command line interface, an interactive web form, DIG protocol support, and programmatic interface with bindings to Jena and OWL-API libraries. It is used in conjunction with ontology editors Swoop, Protege, and TopBraid Composer.

See the Pellet Web page at `http://www.mindswap.org/2003/pellet` for downloads and more detailed information about the system including the performance evaluation and comparison with other systems.

References

[1] B. Cuenca Grau, B. Parsia, and E.Sirin. Combining OWL ontologies using $\mathcal{E}$-connections. *Journal of Web Semantics*, 4(1), January 2005.

[2] A. Kalyanpur, B. Parsia, B. Cuenca-Grau, and E. Sirin. Axiom pinpointing: Finding (precise) justifications for arbitrary entailments in OWL-DL, 2006. Available at http://www.mindswap.org/papers/TracingTechReport.pdf.

[3] E. Sirin, B. Cuenca-Grau, and B. Parsia. From wine to water: Optimizing description logic reasoning for nominals. In *International Conf. on the Principles of Knowledge Representation and Reasoning (KR-2006)*, 2006.

[4] E. Sirin and B. Parsia. Optimizations for answering conjunctive abox queries: First results. In *Proc. of the Int. Description Logics Workshop (DL 2006)*, 2006.

[5] B. Parsia V. Kolovski and E. Sirin. Extending the $\mathcal{SHOIQ}$(D) tableaux with dl-safe rules: First results. In *Proc. of the Int. Description Logics Workshop (DL 2006)*, 2006.

Topological Reasoning in Basic Description Logics

Matteo Cristani, Nicoletta Gabrielli and Paolo Torelli
Dipartimento di Informatica
Università di Verona
Strada Le Grazie 15, 37134 Verona (ITALY)

1 Introduction

In the recent past, several scholars have shown interest in the development of an integration between general classification reasoning, as typically performed in Description Logic frameworks [DL2003], and Spatial Reasoning [CohnH01], usually carried out in a constraint-based context. Two approaches have been carried out in the recent past. One is based upon the extension $\mathcal{ALCRP}(\mathcal{D})$ of $\mathcal{ALC}$ with concrete domains, where the spatial reasoning capabilities of the framework are deployed by means of standard topological interpretation of rational numbers [HaLM1999]. The other one, instead, is based on the extension $\mathcal{ALCI}_{\mathcal{RCC}}$ [Wess2002] of $\mathcal{ALC}$ with role formation operators that are limited in scope to the definition of an algebraic-logic framework for spatial reasoning quite well-known in the reference literature and usually referred to as the *Region Connection Calculus* [Rand1989, Rand1992]. The goal of both these approaches is the representation of the topological properties and relationships between spatial objects that are in fact elements of the domain of the interpretation.

This work aims at representing the *topological properties of the sets of elements* that correspond to the interpretations of the concepts defined in a $\mathcal{ALC}$ terminology. In particular we introduce a rewriting operator that can be proved to behave as a Kuratowski closure. With this closure operator we define the notion of *connection* between concepts. The definition of a closure operator induces a topology [Kell1975] on the domain of the interpretation in which concepts can be open or closed and therefore they can be connected or not. The notion of connection between concepts deals with the knowledge ontology, in fact we aspect that for example the concept of *forest* will be not connected with the concept of *car*.

The introduction of rewriting operators in a description logic cannot interfere with the complexity of decision algorithms, so that if we introduce a rewriting

operator in a description logic $\mathcal{L}$ that is decidable, then $\mathcal{L}$ remains decidable. However, a rewriting operator has its own computational cost, which is the complexity of applying the operator to generic descriptions of $\mathcal{L}$. The complexity of the rewriting operator is therefore worth computing.

In particular, we obtain three results. First, the extended case complexity - in which the Kuratowski operator is introduced in $\mathcal{ALCC}_{\mathcal{U}}$ - is ExpTime, hence remaining decidable. The second result is that we have the same average case complexity as $\mathcal{ALCC}_{\mathcal{U}}$. The third aspect is that complexity is theoretically scalable with the operators.

References

[DL2003] F. Baader, D. Calvanese, D. McGuinness, D. Nardi, P. Patel-Schneider. *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press, 2003.

[CohnH01] A. G. Cohn, S. M. Hazarika, *Qualitative Spatial Representation and Reasoning: An Overview.*, In *Fundamenta Informaticae* volume 46, number 1-2, 2001, pp. 1-29

[HaLM1999] V. Haarslev, C. Lutz, R. Möller. *A Description Logic with Concrete Domains and a Role-forming Predicate Operator*. Journal of Logic Computation, Jun 1999; number 9, pp. 351-384.

[Kell1975] J. L. Kelley. *General Topology*. Springer-Verlag, 1975.

[Rand1989] D. A. Randell, A. G. Cohn. *Modelling topological and metrical properties of physical processes*. In *Proceedings of the 1st International Conference on Knowledge Representation and Reasoning (KR-89)*, May 1989, pp. 55-66. Morgan Kaufmann, Toronto, ON, Canada

[Rand1992] D. A. Randell, Z. Cui, A. G. Cohn. *A spatial logic based on regions and connection*. In *Proceedings of the 3rd International Conference on Principles of Knowledge Representation and Reasoning (KR-92)*, October 1992, pp. 165-176. Morgan Kaufmann, Cambridge, MA, USA.

[Wess2002] M. Wessel. *Qualitative Spatial Reasoning with the $\mathcal{ALCI}_{\mathcal{RCC}}$ Family - First Results and Unanswered Questions*. University of Hamburg, Computer Science Department, 2002 - 2003

The New ICom

– Demo Paper –

Pablo Fillottrani, Enrico Franconi, Sergio Tessaris
Free University of Bozen-Bolzano, Italy
lastname@inf.unibz.it

ICom is an advanced CASE tool, which allows the user to design multiple UML class diagrams with inter- and intra-model constraints. Complete logical reasoning is employed by the tool to verify the specification, infer implicit facts, devise stricter constraints, and manifest any inconsistency.

For the ontology creation and maintenance tasks, ICom interface supports ontology engineers in engineering ontologies that meets clear and measurable quality criteria. Indeed, recently we observe the development of large numbers of ontologies. These ontologies have, however, usually been developed in an ad hoc manner by domain experts, often with only a limited understanding of the semantics of ontology languages. The result is that many ontologies are of low quality—they make poor use of the languages in which they are written and do not accurately capture the author's rich knowledge of the domain. This problem becomes even more acute as ontologies are maintained and extended over time, often by multiple authors. Poor quality ontologies usually require localised "tuning" in order to achieve the desired results within applications. This leads to further degradation in their overall quality, increases the brittleness of the applications that use them, and makes interoperability and reuse difficult or impossible. To overcome these problems tools are needed which support the design and the development of the basic infrastructure for building, merging, and maintaining ontologies.

The Ontology Editor works on *projects*, which may contain one or more UML class diagram. The diagrams are referred as *models*. Multiple projects can be opened at the same time, but objects cannot be moved across them. Only one project is visible at a time and the editing of each project is independent. The user can switch between different projects using the tabs at the bottom of the project area.

Figure 1 shows the main window of the Ontology Editor editing a single model. Classes are represented by boxes and n-ary associations by diamonds. Associations have so-called *association classes* specifying their name and attributes. Isa relationships are represented as arrows with a disc in the middle (e.g. see `Enterprise` and `Business_Organization`).

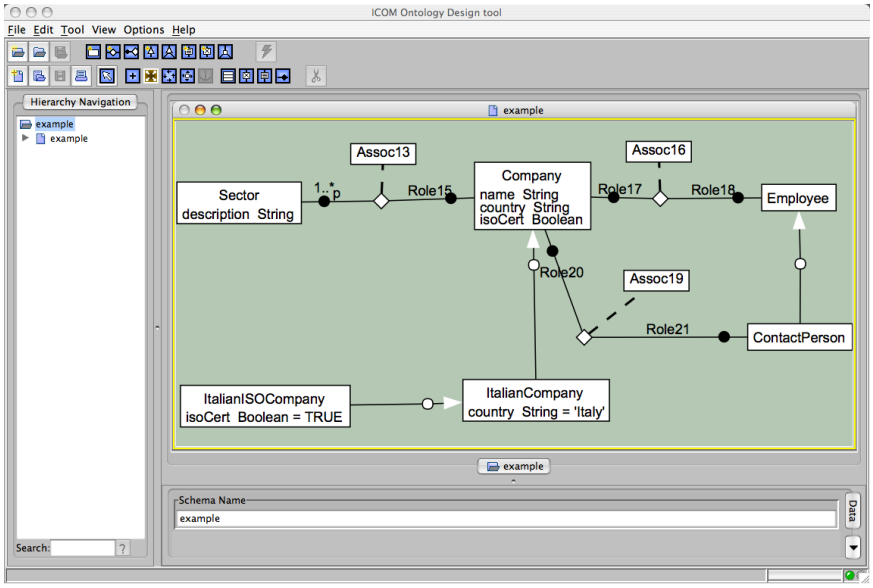


Figure 1: Ontology Editor main window

Although the Ontology Editor can be used as a standalone modelling tool, exploiting its full capabilities require the coupling of the system with any DIG enabled Description Logic reasoner [3, 1]. The leverage of automated reasoning to support the domain modelling is enabled by a precise semantic definition of all the elements of the class diagrams. The diagrams and inter-model constraints are internally translated into a class-based logic formalism. The same underlying logic enables the use of a *view definition language* to specify additional contraints, not captured at the diagram level. The next section introduces this logic-based modelling language.

After the verification process, the system provides the user with a visual account of the deductions by modifying the appearance of the model diagrams in the project. The user can discard the deductions and the entire project will be returned to its original state (and any information about unsatisfiability will be discarded). Alternatively, the equivalence, subsumption association, and role cardinality deductions can be added permanently to the project by committing them.

ICom is a fairly mature project, its first release has been published in 2001 (see [4, 2]). This paper presents the new improved version of the tool. Although the theoretical underpinning is the same, ICom in its current version underwent major changes in several crucial aspects. First of all, the diagrammatic representation is now based on UML rather than Entity Relationship diagrams. The graphical interface has been completely rewritten to improve the usability and intuitiveness of the tool. Interoperability with other tools is a crucial aspect; so, import and export modules have been developed for XMI 2.x [5] and Description Logics based ontology languages.

References

[1] Sean Bechhofer. The dig description logic interface: Dig/1.1. Technical report, University of Manchester, 2003.

[2] Diego Calvanese, Giuseppe De Giacomo, Maurizio Lenzerini, Daniele Nardi, and Riccardo Rosati. Description logic framework for information integration. In *Proc. of the 6th Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR'98)*, pages 2–13, 1998.

[3] Diego Calvanese, Maurizio Lenzerini, and Daniele Nardi. Description logics for conceptual data modeling. In Jan Chomicki and Günter Saake, editors, *Logics for Databases and Information Systems*. Kluwer, 1998.

[4] Enrico Franconi and Gary Ng. The icom tool for intelligent conceptual modelling. In *7th Intl. Workshop on Knowledge Representation meets Databases (KRDB'00)*, 2000.

[5] OMG, 2005. `http://www.omg.org/technology/documents/formal/xmi.htm`.

Index

Artale, Alessandro 97

Baader, Franz 15
Ben-David, Shoham 223
Booth, Richard 253

Calvanese, Diego 51, 62
Che, Haiyan 249
Cristani, Matteo 257
Cuenca Grau, Bernardo 175, 208

De Giacomo, Giuseppe 51
Ding, Yu 143

Eiter, Thomas 62

Fillottrani, Pablo 259
Fokoue, Achille 135
Franconi, Enrico 259

Gabrielli, Nicoletta 257
Gardiner, Tom 167
Glimm, Brite 3

Haarslev, Volker 143, 151, 159
Halaschek-Wiener, Christian 200
Hanselmann, Michael 251
Hladik Jan 74
Horrocks, Ian 3, 167, 175
Hudek, Alexander K. 86

Kalyanpur, Aditya 200, 208
Kang, Dazhou 39, 111
Kaplunova, Alissa 243
Kaya, Atila 243
Kershenbaum, Aaron 135
Kleemann, Thomas 231
Kolovski, Vladimir 192
Kutz, Oliver 175

Lee, Kevin 253
Lembo, Domenico 51
Lenzerini, Maurizio 51
Li, Yanhui 39, 111
Liebig, Thorsten 251
Liu, Hongkai 27
Lu, Jianjiang 39, 111
Lukácsy, Gergely 183
Lutz, Carsten 15, 27, 97

Möller, Ralf 151, 243, 245, 247
Müller, Felix 251
Ma, Li 135
Mellish, Chris 127
Melzer, Sylvia 245
Meyer, Thomas 253
Miličić, Maja 27

Nagy, Zsolt 183
Noppens Olaf 251

Ortiz, Magdalena 62

Pan, Jeff 119, 127, 253
Parsia, Bijan . . . 192, 200, 208, 215, 255
Peñaloza Rafael 74
Peraldi, Irma Sofia Espinosa 247

Rosati, Riccardo 51

Sattler, Ulrike 3, 175
Sirin, Evren 192, 200, 215, 255
Stamou, Giorgos 119, 241
Stoilos, Giorgos 119, 241
Sun, Jigui 249
Suntisrivaraporn, Boontawee 15
Szeredi, Peter 183

Tessaris, Sergio 259
Toman, David 97
Torelli, Paolo 257
Trefler, Richard 223
Tsarkov, Dmitry 167

Weddell, Grant 86, 223
Wessel, Michael 151
Wolter, Frank 27

Xu, Baowen 39, 111

Zuo, Ming 159