Proceedings of the 23rd International Workshop on Description Logics DL2010

May 4-7, 2010
Waterloo, Canada

EDITED BY
VOLKER HAARSLEV, DAVID TOMAN, and GRANT WEDDELL

Preface

Welcome to the 23rd International Workshop on Description Logics, DL 2010, in Waterloo, Canada. 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 48 papers submitted 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:

- From DL to SMT (and back?)
    Roberto Sebastiani (Trento)
- Searching for the Holy Grail.
    Ian Horrocks (Oxford)
- Composing and Inverting Schema Mappings.
    Phokion Kolaitis (UC Santa Cruz and IBM Research - Almaden)

The organizers of the DL 2010 workshop gratefully acknowledge the logistical and financial support of the Fields Institute (`http://www.fields.utoronto.ca/`), and the logistical support and use of facilities provided by the University of Waterloo.

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 beneted from the help of many people at the University of Waterloo, in particular Jeff Pound and Jiewen Wu for helping with the local organization and the DL 2010 WEB site. Finally, we would like to acknowledge that the work of the PC was greatly simplied by using the EasyChair conference management system (`www.easychair.org`) developed by Andrei Voronkov.

<div align="right">

Volker Haarslev, David Toman, and Grant Weddell
2010 PC chairs and organizers

</div>

Organizing Committee

- Workshop chair: Grant Weddell (Waterloo)
- Program chairs: Volker Haarslev (Concordia) and David Toman (Waterloo)

Programme Committee

- Carlos Areces (LORIA, France)
- Alessandro Artale (Free University of Bozen-Bolzano, Italy)
- Meghyn Bienvenu (University of Bremen, Germany)
- Alex Borgida (Rutgers University, USA)
- Andrea Cali (University of Oxford, UK)
- Diego Calvanese (Free University of Bozen-Bolzano, Italy)
- Bernardo Cuenca Grau (University of Oxford, UK)
- Giuseppe De Giacomo (Sapienza Universita di Roma, Italy)
- Achille Fokoue (IBM Research, USA)
- Enrico Franconi (Free University of Bozen-Bolzano, Italy)
- Rajeev Gore (Australian National University, Australia)
- Stijn Heymans (TU Vienna, Austria)
- Pascal Hitzler (Wright State University, USA)
- Ian Horrocks (University of Oxford, UK)
- Ulrich Hustadt (University of Liverpool, UK)
- Yevgeny Kazakov (University of Oxford, UK)
- Boris Konev (University of Liverpool, UK)
- Roman Kontchakov (Birkbeck University of London, UK)
- Thomas Lukasiewicz (University of Oxford, UK)
- Carsten Lutz (University of Bremen, Germany)
- Tommie Meyer (Meraka Institute, South Africa)
- Maja Milicic (University of Manchester, UK)
- Ralf Moeller (Hamburg University of Technology, Germany)
- Boris Motik (University of Oxford, UK)
- Jeff Z. Pan (University of Aberdeen, UK)
- Bijan Parsia (University of Manchester, UK)
- Peter F. Patel-Schneider (Bell Labs Research, USA)
- Riccardo Rosati (Sapienza Universita di Roma, Italy)
- Sebastian Rudolph (University of Karlsruhe, Germany)
- Uli Sattler (University of Manchester, UK)
- Stefan Schlobach (Vrije Universiteit Amsterdam, The Netherlands)
- Luciano Serafini (ITC-IRST, Italy)
- Evren Sirin (Clark & Parsia, USA)
- Viorica Sofronie-Stokkermans (Max-Planck-Institute, Germany)
- Giorgos Stamou (National Technical University of Athens, Greece)
- Umberto Straccia (ISTI-CNR, Italy)
- Anni-Yasmin Turhan (TU Dresden, Germany)
- Frank Wolter (University of Liverpool, UK)
- Misha Zakharyaschev (Birkbeck University of London, UK)

Table of Contents

III   Posters

Part I

Invited Talks

From DL to SMT (and back?)

Roberto Sebastiani

Trento

Satisfiability Modulo Theory (SMT) is the problem of checking the satisfiability of first-order formulas with respect to some background theories. In recent years, SMT has become increasingly popular due to its success in encoding and solving many real-world problems in important applications domains that include formal verification, scheduling and compiler optimization. To this extent, very-efficient SMT solvers have been developed combining the power of SAT solvers with the expressiveness of dedicated decision procedures for several theories of practical interest (including, e.g., the theory linear arithmetic, of arrays, and of bit-vectors). In this talk I will survey my many-year experience in SMT, which initially largely benefitted from ideas coming from my previous work on satisfiability in ALC. I will highlight techniques and ideas which may be of interest to the DL community and hint at some ongoing work in which DL reasoning largely benefits from SAT and SMT techniques.

Searching for the Holy Grail

Ian Horrocks

Oxford

In this talk I will review my personal odyssey from Grail to the Semantic Web and back again: a fifteen-year (and counting) mission to explore strange new worlds; to seek out new logics and new applications; to boldly go where no description logician has gone before. I will try to identify important lessons that I have learned along the way about the theory and practice of logic based knowledge representation (and I will try to avoid further mixing of metaphors), but like any "road movie", the journey should be at least as important as the destination.

Composing and Inverting Schema Mappings

Phokion Kolaitis

UC Santa Cruz and IBM Research - Almaden

Schema mappings are high-level specifications that describe the relationship between two database schemas. Schema mappings constitute the essential building blocks in formalizing the main data inter-operability tasks, including data exchange and data integration. Several different operators on schema mappings have been introduced and studied in considerable depth. Among these, the composition operator and the inverse operator are the most fundamental and prominent ones. The aim of this talk is to present an overview of results about two these operators, and to illustrate their applications to schema evolution.

Part II

Presentations

Temporal Conceptual Modelling with *DL-Lite*

A. Artale,[1] R. Kontchakov,[2] V. Ryzhikov,[1] and M. Zakharyaschev[2]

[1] KRDB Research Centre
Free University of Bozen-Bolzano
I-39100 Bolzano, Italy
lastname@inf.unibz.it

[2] Dept. of Comp. Science and Inf. Sys.
Birkbeck College
London WC1E 7HX, UK
{roman,michael}@dcs.bbk.ac.uk

1 Introduction

Conceptual modelling formalisms such as the Entity-Relationship model (ER) and Unified Modelling Language (UML) have become a *de facto* standard in database design by providing visual means to describe application domains in a declarative and reusable way. On the other hand, both ER and UML turned out to be closely connected with description logics that are underpinned by formal semantics and thus capable of providing services for effective reasoning over conceptual models; see, e.g., [11, 4].

Temporal conceptual data models (TCMs) [18, 25] have been introduced in the context of temporal databases [20, 15, 13]. In this case, apart from the classical constructs—such as inheritance between classes and relationships, cardinality constraints restricting participation in relationships, and disjointness and covering constraints—temporal constructs are used to capture the temporal behaviour of various components of conceptual schemas. Such constructs can be grouped into 3 categories. *Timestamping constraints* discriminate between those classes, relationships and attributes that change over time and those that are time-invariant [28, 18, 16, 6, 25]. *Evolution constraints* control how domain elements evolve over time by 'migrating' from one class to another [19, 23, 26, 25, 3]. We distinguish between *qualitative* evolution constraints describing generic temporal behaviour, and *quantitative* ones specifying the exact moment of migration. *Temporal cardinality constraints* restrict the number of times an instance of a class participates in a relationship. *Snapshot* cardinality constraints do it at each moment of time, while *lifespan* cardinality constraints impose restrictions over the entire existence of the instance as a member of the class [27, 22].

Temporal conceptual data models can be encoded in various temporal description logics (TDLs), which have been designed and investigated since the seminal paper [24] with the aim of understanding the computational price of introducing a temporal dimension in DLs; see [21] for a recent survey. A general conclusion one can draw from the obtained results is that—as far as there is nontrivial interaction between the temporal and DL components—TDLs based on full-fledged DLs like $\mathcal{ALC}$ turn out to be too complex for effective reasoning (see the end of the introduction for details).

The aim of this paper is to tailor 'minimal' TDLs that are capable of representing various aspects of TCMs and investigate their computational behaviour. First of all, as the DL component we choose the 'light-weight' *DL-Lite* logic

$DL\text{-}Lite_{bool}^{\mathcal{N}}$, which was shown to be adequate for capturing conceptual models without relationship inheritance[1] [4], and its fragment $DL\text{-}Lite_{core}^{\mathcal{N}}$ with most primitive concept inclusions, which are nevertheless enough to represent almost all types of constraints (apart from covering). To discuss our choice of the temporal constructs, consider a toy TCM describing a company.

For the timestamping constraint 'employee is a *snapshot class*' (by the standard TCM terminology, such a class never changes in time) one can use the axiom Employee $\sqsubseteq$ ⊡ Employee with the temporal operator ⊡ 'always.' Likewise, the constraint 'manager is a *temporary class*' in the sense that each of its instances must leave the class, the axiom Manager $\sqsubseteq$ ⬦¬Manager is required, where ⬦ means 'some time.' Both of these axioms are regarded as *global*, i.e., applicable to all time points. Note that to express ⬦ using more standard temporal constructs, we need both 'some time in the past' $\diamondsuit_P$ and 'some time in the future' $\diamondsuit_F$: e.g., ⬦ $= \diamondsuit_P \diamondsuit_F$. To encode a snapshot n-ary relationship, one can reify it into a snapshot class with n auxiliary *rigid*—i.e., time-independent—roles; for a temporary relationship, the reifying class is temporary and the roles are *local* [9, 7]. The qualitative evolution constraints 'each manager was once an employee' and 'a manager will always remain a manager' can be expressed by the axioms Manager $\sqsubseteq \diamondsuit_P$Employee and Manager $\sqsubseteq \Box_F$Manager, while 'an approved project keeps its status until a later date when it actually starts' can be expressed using the 'until' operator: ApprovedProject $\sqsubseteq$ ApprovedProject $\mathcal{U}$ Project. The quantitative evolution constraint 'each project must be finished in 3 years' requires the next-time operator $\bigcirc_F$: Project $\sqsubseteq \bigcirc_F\bigcirc_F\bigcirc_F$FinishedProject. The snapshot cardinality constraint 'an employee can work on at most 2 projects at each moment of time' can be expressed as Employee $\sqsubseteq \leq 2$ worksOn, while the lifespan constraint 'over the whole career, an employee can work on at most 5 projects' requires temporal operators on roles: Employee $\sqsubseteq \leq 5$ ⬦worksOn. Note that 'temporalised' roles of the form ⬦R and ⊡R are always rigid. To represent a temporal database instance of a TCM, we use assertions like $\bigcirc_P$Manager(bob) for 'Bob was a manager last year' and $\bigcirc_F$manages(bob, cronos) for 'Bob will manage project Cronos next year.' As usual, n-ary tables are represented via reification.

These considerations lead us to TDLs based on $DL\text{-}Lite_{bool}^{\mathcal{N}}$ and $DL\text{-}Lite_{core}^{\mathcal{N}}$ and interpreted over the flow of time $(\mathbb{Z}, <)$, in which (1) the future and past temporal operators can be applied to concepts; (2) roles can be declared local or rigid; (3) the 'undirected' temporal operators 'always' and 'some time' can be applied to roles; (4) the concept inclusions (TBox axioms) are global and the database (ABox) assertions are specified to hold at particular moments of time.

To our surprise, the most expressive TDL based on $DL\text{-}Lite_{bool}^{\mathcal{N}}$ and featuring all of (1)–(4) turns out to be undecidable. As follows from the proof of Theorem 5 below, it is a subtle interaction of functionality constraints on temporalised roles with the next-time operator and full Booleans on concepts that causes undecidability. This 'negative' result motivates consideration of various fragments of our full TDL by restricting not only the DL but also the temporal component. The table below illustrates the expressive power of the resulting fragments in the context of TCMs. We also note that both $DL\text{-}Lite_{bool}^{\mathcal{N}}$ and $DL\text{-}Lite_{core}^{\mathcal{N}}$ with global

---

[1] $DL\text{-}Lite_{bool}^{\mathcal{N}}$ with relationship inclusions regains the full expressive power of $\mathcal{ALC}$.

axioms can capture snapshot cardinality constraints, while lifespan cardinality constraints require temporalised roles of the form $\diamondsuit R$ and $\boxdot R$.

concept temporal operators	timestamping	evolution	
		qualitative	quantitative
$\mathcal{U}/\mathcal{S}$	+	+	+
$\Box_{F/P}, \bigcirc_{F/P}$	+	+	+
$\Box_{F/P}$	+	+	−
$\boxast, \bigcirc_{F/P}$	+	−	+
$\boxast$	+	−	−

The next table summarises the complexity results obtained in this paper for satisfiability of temporal knowledge bases formulated in our TDLs.

concept temporal operators	local & rigid roles only		temporalised roles
	$DL\text{-}Lite^{\mathcal{N}}_{bool}$	$DL\text{-}Lite^{\mathcal{N}}_{core}$	$DL\text{-}Lite^{\mathcal{N}}_{bool}$
$\mathcal{U}/\mathcal{S}$	PSPACE Thm. 1	PSPACE [8]	undec. Thm. 5
$\Box_{F/P}, \bigcirc_{F/P}$	PSPACE Thm. 2 (ii)	NP Thm. 3	undec. Thm. 5
$\Box_{F/P}$	NP Thm. 2 (i)	NP [8]	?
$\boxast, \bigcirc_{F/P}$	PSPACE Thm. 2 (ii)	NP Thm. 3	undec. Thm. 5
$\boxast$	NP Thm. 2 (i)	NLOGSPACE Thm. 4	NP Thm. 6

Apart from the undecidability result of Theorem 5, quite surprising is NP-completeness of the temporal extension of $DL\text{-}Lite^{\mathcal{N}}_{core}$ with the operators $\Box_F$ and $\bigcirc_F$ (and their past counterparts) on concepts provided by Theorem 3. Indeed, if full Booleans are available, even the propositional temporal logic with these operators is PSPACE-complete. Moreover, if the 'until' operator $\mathcal{U}$ is available in the temporal component, disjunction is expressible even with $DL\text{-}Lite^{\mathcal{N}}_{core}$ as the underlying DL, and the logic becomes PSPACE-complete [8]. In all other cases, the complexity of TDL reasoning coincides with the maximal complexity of reasoning in the component logics (despite nontrivial interaction between them, as none of our TDLs is a fusion of its components). It is also of interest to observe the dramatic increase of complexity caused by the addition of $\bigcirc_F$ to the logic in the lower right corner of the table (from NP to undecidability).

To put this paper in the more general context of temporal description logics, we note first that our TDLs extend those in [8] with the past-time operators $\mathcal{S}$, $\Box_P$, $\diamondsuit_P$, $\bigcirc_P$ over $\mathbb{Z}$ (which are essential for capturing timestamping constraints), universal modalities $\boxast$ and $\diamondsuit$, and temporalised roles. Temporal operators on *DL-Lite* axioms and concepts in the presence of rigid roles were investigated in [7], where it was shown that the resulting temporalisations of $DL\text{-}Lite^{\mathcal{N}}_{bool}$ and $DL\text{-}Lite^{\mathcal{N}}_{horn}$ are ExpSpace-complete. Temporal extensions of the standard DL $\mathcal{ALC}$ feature the following computational behaviour: $\mathcal{ALC}$ with temporal operators on axioms, rigid concepts and roles is 2ExpTime-complete [10]. It is ExpSpace-complete if temporal operators on concepts and axioms are allowed but no rigid or temporalised roles are available [17], and ExpTime-complete if the language allows only temporalised concepts and global axioms [24, 2]. Finally, the 'undirected' temporal operators $\boxast$ and $\diamondsuit$ on concepts and roles together with global axioms result in a 2ExpTime-complete extension of $\mathcal{ALC}$ [9].

2   Temporal DLs based on *DL-Lite*$_{bool}^{\mathcal{N}}$

The TDL $T_{\mathcal{US}}DL\text{-}Lite_{bool}^{\mathcal{N}}$ is based on $DL\text{-}Lite_{bool}^{\mathcal{N}}$ [1, 5], which, in turn, extends $DL\text{-}Lite_{\sqcap,\mathcal{F}}$ [12] with full Booleans over concepts and cardinality restrictions over roles. The language of $T_{\mathcal{US}}DL\text{-}Lite_{bool}^{\mathcal{N}}$ contains *object names* $a_0, a_1, \ldots$, *concept names* $A_0, A_1, \ldots$, *local role names* $P_0, P_1, \ldots$ and *rigid role names* $G_0, G_1, \ldots$. *Roles* R, *basic concepts* B and *concepts* C are defined as follows:

$$
\begin{aligned}
S \ &::=\ P_i \ \mid\ G_i, \qquad\qquad R \ ::=\ S \ \mid\ S^-, \\
B \ &::=\ \bot \ \mid\ A_i \ \mid\ \geq q\, R, \\
C \ &::=\ B \ \mid\ \neg C \ \mid\ C_1 \sqcap C_2 \ \mid\ C_1\, \mathcal{U}\, C_2 \mid\ C_1\, \mathcal{S}\, C_2,
\end{aligned}
$$

where $q \geq 1$ is a natural number (the results obtained below do not depend on whether q is given in unary or binary). A $T_{\mathcal{US}}DL\text{-}Lite_{bool}^{\mathcal{N}}$ *interpretation* is a function $\mathcal{I}$ on the integers $\mathbb{Z}$ (the intended flow of time):

$$
\mathcal{I}(n)\ =\ \big(\Delta^{\mathcal{I}}, a_0^{\mathcal{I}}, \ldots, A_0^{\mathcal{I}(n)}, \ldots, P_0^{\mathcal{I}(n)}, \ldots, G_0^{\mathcal{I}(n)}, \ldots\big),
$$

where $\Delta^{\mathcal{I}}$ is a nonempty set, the (constant) domain of $\mathcal{I}$, $a_i^{\mathcal{I}} \in \Delta^{\mathcal{I}}$, $A_i^{\mathcal{I}(n)} \subseteq \Delta^{\mathcal{I}}$ and $P_i^{\mathcal{I}(n)}, G_i^{\mathcal{I}(n)} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$ with $G_i^{\mathcal{I}(n)} = G_i^{\mathcal{I}(m)}$, for $i \in \mathbb{N}$ and $n, m \in \mathbb{Z}$. We adopt the unique name assumption according to which $a_i^{\mathcal{I}} \neq a_j^{\mathcal{I}}$, for $i \neq j$, although our complexity results would not change if we dropped it, apart from the NLogSpace bound of Theorem 4, which would increase to NP [5]. The role and concept constructs are interpreted in $\mathcal{I}$ as follows:

$$
(S^-)^{\mathcal{I}(n)} = \{(y,x) \mid (x,y) \in S^{\mathcal{I}(n)}\}, \ \ \bot^{\mathcal{I}(n)} = \emptyset, \ \ (\neg C)^{\mathcal{I}(n)} = \Delta^{\mathcal{I}} \setminus C^{\mathcal{I}(n)},
$$

$$
(C_1 \sqcap C_2)^{\mathcal{I}(n)} = C_1^{\mathcal{I}(n)} \cap C_2^{\mathcal{I}(n)}, \ \ (\geq q\, R)^{\mathcal{I}(n)} = \big\{x \mid \sharp\{y \mid (x,y) \in R^{\mathcal{I}(n)}\} \geq q\big\},
$$

$$
(C_1 \, \mathcal{U} \, C_2)^{\mathcal{I}(n)} = \bigcup_{k>n}\big(C_2^{\mathcal{I}(k)} \cap \bigcap_{n<m<k} C_1^{\mathcal{I}(m)}\big),
$$

$$
(C_1 \, \mathcal{S} \, C_2)^{\mathcal{I}(n)} = \bigcup_{k<n}\big(C_2^{\mathcal{I}(k)} \cap \bigcap_{n>m>k} C_1^{\mathcal{I}(m)}\big).
$$

Note that our *until* and *since* operators are 'strict' (i.e., do not include the current moment). We also use the temporal operators $\Diamond_F$ ('some time in the future'), $\Diamond_P$ ('some time in the past'), $\diamondsuit$ ('some time'), their duals $\Box_F$, $\Box_P$ and $\boxdot$, $\bigcirc_F$ ('next time') and $\bigcirc_P$ ('previous time'), which are all expressible by means of $\mathcal{U}$ and $\mathcal{S}$, e.g., $\Diamond_F C = \neg\bot\mathcal{U}C$, $\Box_F C = \neg\Diamond_F\neg C$, $\bigcirc_F C = \bot\mathcal{U}C$, $\diamondsuit C = \Diamond_F\Diamond_P C$ and $\boxdot C = \Box_F\Box_P C$. (Other standard abbreviations we use include $C_1 \sqcup C_2$, $\exists R$ and $\top = \neg\bot$.) Apart from full $T_{\mathcal{US}}DL\text{-}Lite_{bool}^{\mathcal{N}}$, we consider a few of its sublanguages allowing only some of the (definable) temporal operators mentioned above:

- $T_{FP}DL\text{-}Lite_{bool}^{\mathcal{N}}$, which allows only $\Diamond_F C$, $\Diamond_P C$ and their duals (but no $\bigcirc_F C$ or $C_1\,\mathcal{U}\,C_2$), and its extension $T_{FPX}DL\text{-}Lite_{bool}^{\mathcal{N}}$ with $\bigcirc_F C$ and $\bigcirc_P C$;
- $T_U DL\text{-}Lite_{bool}^{\mathcal{N}}$, allowing only $\diamondsuit C$ and $\boxdot C$, and its extension $T_{UX}DL\text{-}Lite_{bool}^{\mathcal{N}}$ with $\bigcirc_F C$ and $\bigcirc_P C$.

    A *TBox*, $\mathcal{T}$, in any of our languages $\mathcal{L}$ is a finite set of *concept inclusions* (CIs) of the form $C_1 \sqsubseteq C_2$, where the C_i are $\mathcal{L}$-concepts. An *ABox*, $\mathcal{A}$, consists

of assertions of the form $\bigcirc^n B(a)$ and $\bigcirc^n S(a, b)$, where B is a basic concept, S a (local or rigid) role name, a,b object names and $\bigcirc^n$, for $n \in \mathbb{Z}$, is a sequence of n operators $\bigcirc_F$ if $n \geq 0$ and $|n|$ operators $\bigcirc_P$ if $n < 0$. Taken together, the TBox $\mathcal{T}$ and ABox $\mathcal{A}$ form the *knowledge base* (KB) $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ in $\mathcal{L}$.

The *truth-relation* is defined as usual: $\mathcal{I} \models C_1 \sqsubseteq C_2$ iff $C_1^{\mathcal{I}(n)} \subseteq C_2^{\mathcal{I}(n)}$, for all $n \in \mathbb{Z}$, that is, we interpret concept inclusions globally, $\mathcal{I} \models \bigcirc^n B(a)$ iff $a^{\mathcal{I}} \in B^{\mathcal{I}(n)}$, and $\mathcal{I} \models \bigcirc^n S(a, b)$ iff $(a^{\mathcal{I}}, b^{\mathcal{I}}) \in S^{\mathcal{I}(n)}$. We call $\mathcal{I}$ a *model* of a KB $\mathcal{K}$ and write $\mathcal{I} \models \mathcal{K}$ if $\mathcal{I} \models \alpha$ for all α in $\mathcal{K}$. If $\mathcal{K}$ has a model then it is said to be *satisfiable*. A concept C (role R) is *satisfiable* w.r.t. $\mathcal{K}$ if there are a model $\mathcal{I}$ of $\mathcal{K}$ and $n \in \mathbb{Z}$ such that $C^{\mathcal{I}(n)} \neq \emptyset$ (respectively, $R^{\mathcal{I}(n)} \neq \emptyset$). Clearly, the concept and role satisfiability problems are equivalent to KB satisfiability.

Our first result states that the satisfiability problem for $T_{\mathcal{US}} DL\text{-}Lite_{bool}^{\mathcal{N}}$ KBs is as complex as satisfiability in propositional temporal logic *LTL*.

Theorem 1. *Satisfiability of $T_{\mathcal{US}} DL\text{-}Lite_{bool}^{\mathcal{N}}$ KBs is* PSPACE-*complete.*

The proof is by a two-step (non-deterministic polynomial) reduction to *LTL*. First, we reduce satisfiability of a $T_{\mathcal{US}} DL\text{-}Lite_{bool}^{\mathcal{N}}$ KB $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ to satisfiability in the one-variable first-order temporal logic in a way similar to [8]. For each basic concept B ($\neq \bot$), we take a fresh *unary* predicate $B^*(x)$ and encode $\mathcal{T}$ as

$$\mathcal{T}^{\dagger} = \bigwedge_{C_1 \sqsubseteq C_2 \in \mathcal{T}} \boxdot \forall x \left(C_1^*(x) \to C_2^*(x) \right),$$

where the C_i^* are the results of replacing each B with $B^*(x)$ ($\sqcap$ with $\wedge$, etc.). We assume that $\mathcal{T}$ contains CIs of the form $\geq q\, R \sqsubseteq\, \geq q'\, R$, for $\geq q\, R, \geq q'\, R$ in $\mathcal{T}$ such that $q > q'$ and there is no q'' with $q > q'' > q'$ and $\geq q''\, R$ in $\mathcal{T}$. We also assume that $\mathcal{T}$ contains $\geq q\, R \equiv \boxdot \geq q\, R$ if $\geq q\, R$ occurs in $\mathcal{T}$, for a rigid role R (i.e., for G_i or G_i^-). To take account of the fact that roles are *binary* relations, we add to $\mathcal{T}^{\dagger}$ the following formula, for each role name S:

$$\varepsilon_S = \boxdot \left(\exists x\, (\exists S)^*(x) \leftrightarrow \exists x\, (\exists S^-)^*(x) \right)$$

(which says that at each moment of time the domain of S is nonempty iff its range is nonempty). The ABox $\mathcal{A}$ is encoded by a conjunction $\mathcal{A}^{\dagger}$ of ground atoms of the form $\bigcirc^m B^*(a)$ and $\bigcirc^n (\geq q\, R)^*(a)$ in the same way as in [8]. Thus, $\mathcal{K}$ is satisfiable iff the formula

$$\mathcal{K}^{\dagger} = \mathcal{T}^{\dagger} \wedge \bigwedge_S \varepsilon_S \wedge \mathcal{A}^{\dagger}$$

is satisfiable. The second step of our reduction is based on the observation that if $\mathcal{K}^{\dagger}$ is satisfiable then it can be satisfied in a model such that

(R) if $(\exists S)^*(x)$ is true at some moment (on some domain element) then it is true at all moments of time (perhaps on different domain elements).

Indeed, if $\mathcal{K}^{\dagger}$ is satisfied in $\mathcal{I}$ then it is satisfied in the disjoint union $\mathcal{I}^*$ of all $\mathcal{I}^n$, $n \in \mathbb{Z}$, obtained from $\mathcal{I}$ by shifting its time line n moments forward. It follows

from **(R)** that $\mathcal{K}^\dagger$ is satisfiable iff there is a set Σ of role names such that

$$
\mathcal{K}^{\dagger\Sigma} \;\; = \;\; \mathcal{T}^\dagger \;\; \wedge \;\; \bigwedge_{S\in\Sigma} \big((\exists S)^*(d_S) \wedge (\exists S^-)^*(d_{S^-})\big) \;\; \wedge \\
\bigwedge_{S\notin\Sigma} \boxdot \forall x \,\neg\big((\exists S)^*(x) \vee (\exists S^-)^*(x)\big) \;\; \wedge \;\; \mathcal{A}^\dagger
$$

is satisfiable, where the d_S are fresh constants (informally, the roles in Σ are nonempty at some moment, whereas all other roles are always empty). Finally, as $\mathcal{K}^{\dagger\Sigma}$ contains no existential quantifiers, it can be regarded as an *LTL*-formula because all the universal quantifiers can be instantiated by all the constants in the formula, which results only in a polynomial blow-up of $\mathcal{K}^{\dagger\Sigma}$.

This reduction can also be used to obtain complexity results for the fragments of $T_{\mathcal{US}}DL\text{-}Lite^{\mathcal{N}}_{bool}$ mentioned above. Using the well-known facts that satisfiability in the fragments of *LTL* with $\Diamond_F/\Diamond_P$ and with $\diamondsuit$ is NP-complete, and that the extension of any of these fragments with $\bigcirc_F/\bigcirc_P$ becomes PSPACE-complete again, we obtain:

Theorem 2. (i) *Satisfiability of $T_{FP}DL\text{-}Lite^{\mathcal{N}}_{bool}$ and $T_U DL\text{-}Lite^{\mathcal{N}}_{bool}$ KBs is* NP-*complete.* (ii) *For $T_{FPX}DL\text{-}Lite^{\mathcal{N}}_{bool}$ and $T_{UX}DL\text{-}Lite^{\mathcal{N}}_{bool}$ KBs, satisfiability is* PSPACE-*complete.*

3   Temporal DLs based on $DL\text{-}Lite^{\mathcal{N}}_{core}$

So far, to decrease complexity we have restricted the expressive power of the temporal component of $T_{\mathcal{US}}DL\text{-}Lite^{\mathcal{N}}_{bool}$. But the underlying DL $DL\text{-}Lite^{\mathcal{N}}_{bool}$ also has some natural fragments of lower complexity [5]. In this section, we consider the simplest of them known as $DL\text{-}Lite^{\mathcal{N}}_{core}$ and containing only CIs of the form $B_1 \sqsubseteq B_2$ and $B_1 \sqcap B_2 \sqsubseteq \bot$, where the B_i are basic concepts. Satisfiability of $DL\text{-}Lite^{\mathcal{N}}_{core}$ KBs is NLOGSPACE-complete.

Let $T_{\mathcal{US}}DL\text{-}Lite^{\mathcal{N}}_{core}$ be the fragment of $T_{\mathcal{US}}DL\text{-}Lite^{\mathcal{N}}_{bool}$ with CIs of the form $D_1 \sqsubseteq D_2$ and $D_1 \sqcap D_2 \sqsubseteq \bot$, where the D_i are defined by the rule:

$$
D \quad ::= \quad B \quad | \quad B_1\,\mathcal{U}\,B_2 \,| \quad B_1\,\mathcal{S}\,B_2.
$$

By restricting D_1 and D_2 to concepts of the form

$$
D \quad ::= \quad B \quad | \quad \Diamond_F B \quad | \quad \Diamond_P B \quad | \quad \Box_F B \quad | \quad \Box_P B
$$

we obtain $T_{FP}DL\text{-}Lite^{\mathcal{N}}_{core}$. These restrictions do not improve the complexity of reasoning: satisfiability of $T_{\mathcal{US}}DL\text{-}Lite^{\mathcal{N}}_{core}$ KBs is PSPACE-complete, while for $T_{FP}DL\text{-}Lite^{\mathcal{N}}_{core}$ it is NP-complete [8].

What is really surprising and nontrivial is that extending $T_{FP}DL\text{-}Lite^{\mathcal{N}}_{core}$ with the next- and previous-time operators does not increase the complexity; cf. Theorem 2 (ii). More formally, define $T_{FPX}DL\text{-}Lite^{\mathcal{N}}_{core}$ by restricting D_1 and D_2 to concepts of the form:

$$
D \quad ::= \quad B \quad | \quad \Diamond_F B \quad | \quad \Diamond_P B \quad | \quad \Box_F B \quad | \quad \Box_P B \quad | \quad \bigcirc_F B \quad | \quad \bigcirc_P B,
$$

and let $T_{UX}DL\text{-}Lite^{\mathcal{N}}_{core}$ be the logic with the D_i of the form:

$$
D \quad ::= \quad B \quad | \quad \diamondsuit B \quad | \quad \boxdot B \quad | \quad \bigcirc_F B \quad | \quad \bigcirc_P B.
$$

Theorem 3. *Satisfiability of $T_{FPX}\,DL\text{-}Lite_{core}^{\mathcal{N}}$ and $T_{UX}\,DL\text{-}Lite_{core}^{\mathcal{N}}$ KBs is* NP-*complete.*

We present only a sketch of the proof here; the full proof can be found at `http://www.dcs.bbk.ac.uk/~roman/papers/dl10-full.pdf`.

In a way similar to the proof of Theorem 1, one can (non-deterministically and polynomially) reduce satisfiability of a $T_{FPX}\,DL\text{-}Lite_{core}^{\mathcal{N}}$ KB to satisfiability of an *LTL*-formula $\varphi = \bigwedge_i \boxdot(E_i \vee E_i') \wedge \psi$, where the E_i and E_i' are of the form $p, \diamond_F p, \diamond_P p, \square_F p, \square_P p, \bigcirc_F p, \bigcirc_P p$ or a negation thereof, and ψ is a conjunction of formulas of the form $\bigcirc^n p$, p a propositional variable. Let Γ be the set of all subformulas of φ of the form $\diamond_F p$, $\diamond_P p$, $\square_F p$ or $\square_P p$. It should be clear that if φ is satisfied in an interpretation then the flow of time can be partitioned into $|\Gamma| + 1$ intervals $I_0, \ldots, I_{|\Gamma|}$ such that, for each $\gamma \in \Gamma$ and each I_i, γ is true at *some* point in I_i iff γ is true at *every* point in I_i. The existence of such intervals can be expressed by certain syntactic conditions on their 'states,' the most crucial of which is satisfiability of a formula of the form

$$\chi \quad = \quad \Psi \wedge \square^{\leq m}\Phi \wedge \bigcirc^m(\Psi' \wedge \bigcirc\Psi''),$$

for $\Phi = \bigwedge_i(D_i \vee D_i')$, with each of the D_i and D_i' being a literal L (a propositional variable or its negation) or $\bigcirc L$, conjunctions Ψ, Ψ' and Ψ'' of literals, and $m \geq 0$, where $\bigcirc^n\Psi$ is the result of attaching n operators $\bigcirc$ to each literal in Ψ and $\square^{\leq m}\Phi = \bigwedge_{0 \leq i \leq m} \bigcirc^i\Phi$. Intuitively, m is the number of distinct states in an interval I_i, Ψ and Ψ' are the first and the last states in I_i, Ψ'' is the first state of the next interval I_{i+1}, and Φ a set of binary clauses that describe possible transitions between the states. Let $cons_\Phi^m(\Psi)$ be the set of all literals L that are true at the moment $m \geq 0$ in every model of $\Psi \wedge \square^{\leq m}\Phi$. As the formula $\Psi \wedge \square^{\leq m}\Phi$ is essentially a 2CNF, one can compute $cons_\Phi^m(\Psi)$ inductively as follows:

$$cons_\Phi^0(\Psi) = \{L \mid \Phi \cup \Psi \models L\},$$
$$cons_\Phi^m(\Psi) = \{L \mid \Phi \models L' \to \bigcirc L, L' \in cons_\Phi^{m-1}(\Psi)\} \cup \{L \mid \Phi \models L\}.$$

For each L, construct a non-deterministic finite automaton $\mathfrak{A}_L = (Q, Q_0, \sigma, F_L)$ over the alphabet $\{0\}$ that accepts 0^m iff $L \in cons_\Phi^m(\Psi)$. Define the states in Q to be all the literals from χ, the set of initial states $Q_0 = cons_\Phi^0(\Psi)$, the accepting states $F_L = \{L\}$, and the transition relation

$$\sigma = \{(L'', L') \mid \Phi \models L'' \to \bigcirc L'\} \cup \{(L', L') \mid \Phi \models L'\}.$$

Then a state L is reachable in m σ-steps from a state in Q_0 iff $L \in cons_\Phi^m(\Psi)$, and so $\mathfrak{A}_L$ is as required. Every such $\mathfrak{A}_L$ can be converted into an equivalent automaton in the Chrobak normal form [14] using Martinez's algorithm [29], which gives rise to M_L-many arithmetic progressions $a_1^L + b_1^L\mathbb{N}, \ldots, a_{M_L}^L + b_{M_L}^L\mathbb{N}$, where $a + b\mathbb{N} = \{a + bn \mid n \in \mathbb{N}\}$, such that

(A_1) $M_L, a_i^L, b_i^L \leq |\Phi \cup \Psi|^2$, for $1 \leq i \leq M_L$, and
(A_2) $L \in cons_\Phi^m(\Psi)$ iff $m \in \bigcup_{i=1}^{M_L}(a_i^L + b_i^L\mathbb{N})$.

Satisfiability of χ can now be established by a polynomial-time algorithm which checks whether the following three conditions hold:

1. $p, \neg p \in cons_\Phi^n(\Psi)$, for no variable p and no $0 \le n \le m + 1$;
2. $\neg L \notin cons_\Phi^m(\Psi)$, for all literals $L \in \Psi'$;
3. $\neg L \notin cons_\Phi^{m+1}(\Psi)$, for all literals $L \in \Psi''$.

To verify 1, we check, for each variable p, whether the linear Diophantine equations $a_i^p + b_i^p x = a_j^{\neg p} + b_j^{\neg p} y$, for $1 \le i \le M_p$ and $1 \le j \le M_{\neg p}$, have a solution (x_0, y_0) such that $0 \le a_i^p + b_i^p x_0 \le m + 1$. Set $a = b_i^p$, $b = -b_j^{\neg p}$ and $c = a_j^{\neg p} - a_i^p$, which gives us the equation $ax + by = c$. If $a \ne 0$ and $b \ne 0$ then, by Bézout's lemma, it has a solution iff c is a multiple of the greatest common divisor d of a and b, which can be checked in polynomial time using the Euclidean algorithm (provided that the numbers are encoded in unary, which can be assumed in view of $(\mathbf{A_1})$). Moreover, if the equation has a solution, then the Euclidean algorithm also gives us a pair (u_0, v_0) such that $d = au_0 + bv_0$, in which case all the solutions of the above equation form the set $\left\{ \left((cu_0 + bk)/d, (cv_0 - ak)/d \right) \mid k \in \mathbb{Z} \right\}$. Thus, it remains to check whether a number between 0 and $m + 1$ is contained in $a_i^p + b_i^p(a_j^{\neg p} - a_i^p)u_0/d + b_i^p b_j^{\neg p}/d\mathbb{N}$. The case $a = 0$ or $b = 0$ is left to the reader. To verify condition 2, we check, for each $L \in \Psi'$, whether m belongs to one of $a_i^{\neg L} + b_i^{\neg L}\mathbb{N}$, for $1 \le i \le M_L$, which can be done in polynomial time. Condition 3 is analogous. This gives us the NP upper bound for the logics mentioned in Theorem 3. The lower bound can be proved by reduction of the 3-colourability problem to satisfiability of $T_{UX} DL\text{-}Lite_{core}^{\mathcal{N}}$ KBs.

Theorem 3 shows that $T_{FPX} DL\text{-}Lite_{core}^{\mathcal{N}}$ can be regarded as a good candidate for representing temporal conceptual data models. Although not able to express covering constraints, $T_{FPX} DL\text{-}Lite_{core}^{\mathcal{N}}$ still appears to be a reasonable compromise compared to the full PSPACE-complete logic $T_{FPX} DL\text{-}Lite_{bool}^{\mathcal{N}}$.

By restricting the temporal constructs to the undirected universal modalities ⊞ and ⧫, we obtain an even simpler logic:

Theorem 4. *Satisfiability of $T_U DL\text{-}Lite_{core}^{\mathcal{N}}$ KBs is* NLOGSPACE-*complete.*

The proof of the upper bound is by embedding into the universal Krom fragment of first-order logic.

4 Temporal DLs with Temporalised Roles

As we have seen before, in order to express lifespan cardinalities, temporal operators on roles are required. Modalised roles are known to be 'dangerous' and very difficult to deal with when temporalising expressive DLs such as $\mathcal{ALC}$ [17, Section 14.2]. To our surprise, even in the case of *DL-Lite*, temporal operators on roles may cause undecidability (while rigid roles are 'mostly harmless'). Denote by $T_X^{\mathcal{R}} DL\text{-}Lite_{bool}^{\mathcal{N}}$ the fragment of $T_{\mathcal{US}} DL\text{-}Lite_{bool}^{\mathcal{N}}$ with $\bigcirc_F$ as the only temporal operator over concepts and with roles R of the form

$$R \quad ::= \quad S \quad | \quad S^- \quad | \quad \Diamond R \quad | \quad \boxdot R.$$

The extensions of $\Diamond R$ and $\boxdot R$ in an interpretation $\mathcal{I}$ are defined as follows:

$$(\Diamond R)^{\mathcal{I}(n)} = \bigcup\nolimits_{k \in \mathbb{Z}} R^{\mathcal{I}(k)} \quad \text{and} \quad (\boxdot R)^{\mathcal{I}(n)} = \bigcap\nolimits_{k \in \mathbb{Z}} R^{\mathcal{I}(k)}.$$

Theorem 5. *Satisfiability of $T_X^{\mathcal{R}}$DL-Lite$_{bool}^{\mathcal{N}}$ KBs is undecidable.*

The proof is by reduction of the $\mathbb{N} \times \mathbb{N}$-tiling problem: given a finite set T of tile types $t = (up(t), down(t), left(t), right(t))$, decide whether T can tile the $\mathbb{N} \times \mathbb{N}$-grid. We assume that the tiles use k colours numbered from 1 to k.

We construct a $T_X^{\mathcal{R}}$DL-Lite$_{bool}^{\mathcal{N}}$ KB $\mathcal{K}_T$ such that $\mathcal{K}_T$ is satisfiable iff T tiles $\mathbb{N} \times \mathbb{N}$. The temporal dimension clearly provides us with one of the two axes of the grid. The other axis is constructed from the domain elements: let R be a role such that $\geq 2 \Diamond\!\!\!* R \sqsubseteq \bot$ and $\geq 2 \Diamond\!\!\!* R^- \sqsubseteq \bot$. In other words, if xRy at some moment of time then there is no $y' \neq y$ with xRy' at any moment of time (and the same for R^-). We can generate an infinite sequence of the domain elements by saying that $\exists R^- \sqcap \bigcirc_F \exists R^-$ is nonempty and $\exists R^- \sqcap \bigcirc_F \exists R^- \sqsubseteq \exists R \sqcap \bigcirc_F \exists R$. (The reason for generating the R-arrows at two consecutive moments of time will become apparent below.) It should be also noted that the produced sequence may in fact be either a finite loop or an infinite sequence of distinct elements.

Now, let t be a fresh concept name, for each $t \in T$, and let tile types be disjoint, i.e., $t \sqcap t' \sqsubseteq \bot$ for $t \neq t'$. After the double R-arrows we place the first column of tiles, and every $k + 1$ moments afterwards we place a column of tiles that matches the colours of the previous column:

$$\exists R^- \sqcap \bigcirc_F \exists R^- \sqsubseteq \bigsqcup\nolimits_{t \in T} \bigcirc_F \bigcirc_F t, \qquad t \sqsubseteq \bigsqcup\nolimits_{right(t)=left(t')} \bigcirc_F^{k+1} t', \quad \text{for each } t \in T.$$

It remains to ensure that the tiles are arranged in a proper grid and have matching top-bottom colours. It is for this purpose that we have (i) used the double R-arrows to generate the sequence of domain elements, and (ii) placed the columns of tiles every $k + 1$ moments of time (not every moment). Consider the following CIs, for $t \in T$ and $1 \leq i \leq k$:

$$t \sqsubseteq \neg \exists R^-, \quad t \sqsubseteq \neg \bigcirc_F^i \exists R^- \ \ (\text{if } i \neq down(t)) \quad \text{and} \quad t \sqsubseteq \bigcirc_F^{up(t)} \exists R.$$

The first two CIs ensure that between any two tiles $k + 1$ moments apart there may be only one incoming R-arrow. This, in particular, means that after the double R-arrows no other two consecutive R-arrows are possible, and thus the proper $\mathbb{N} \times \mathbb{N}$-grid is ensured. Moreover, the exact position of the incoming R-arrow is uniquely determined by the *down*-colour of the tile, which in conjunction with the last CI guarantees that this colour matches the tile below. The following picture illustrates the construction:
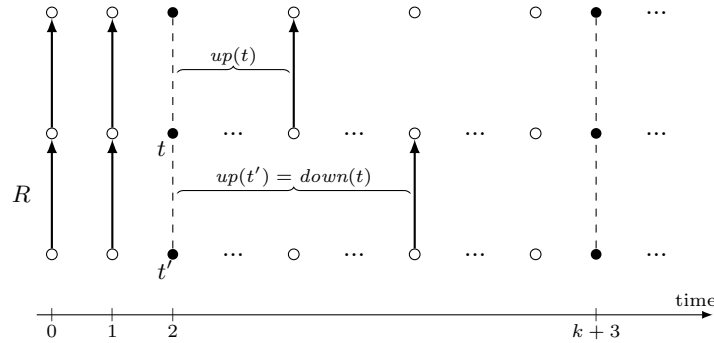
Note that the next-time operator $\bigcirc_F$ is heavily used in the encoding above. If we replace it with $\lozenge\!\!\!\lozenge$ and $\boxdot$ on concepts, then reasoning in the resulting logic $T_U^{\mathcal{R}} DL\text{-}Lite_{bool}^{\mathcal{N}}$ becomes much simpler:

Theorem 6. *Satisfiability of $T_U^{\mathcal{R}} DL\text{-}Lite_{bool}^{\mathcal{N}}$ KBs is* NP*-complete.*

This result is proved using a modification of the quasimodel construction from [7, 8]: we show that a KB is satisfiable iff there exists a *quasimodel* of polynomial size. In the *types* of our quasimodels, concepts $\geq q\,R$, $\geq q\,\lozenge\!\!\!\lozenge\,R$ and $\geq q\,\boxdot\,R$ reflect the number of R-successors of the element required, respectively, in the current moment of time, 'sometime' ($\lozenge\!\!\!\lozenge\,R$-successors) and 'always' ($\boxdot\,R$-successors). In order to deal with temporalised roles, we have to introduce the following conditions on quasimodels: (i) the numbers of $\lozenge\!\!\!\lozenge\,R$-successors and $\boxdot\,R$-successors in types do not change along a *run* (in other words, temporalised roles are rigid roles); (ii) the number of R-successors in every type is sandwiched between the number of $\boxdot\,R$- and the number of $\lozenge\!\!\!\lozenge\,R$-successors; ($iii$) if there is a run with more $\lozenge\!\!\!\lozenge\,R$-successors than $\boxdot\,R$-successors, then there is a run with more $\lozenge\!\!\!\lozenge\,R^-$-successors than $\boxdot\,R^-$-successors; (iv) in each run with more $\lozenge\!\!\!\lozenge\,R$-successors than $\boxdot\,R$-successors, not all R-successors are $\boxdot\,R$-successors, and not all $\lozenge\!\!\!\lozenge\,R$-successors are R-successors at all moments of time. Special conditions are also required for the runs on the objects in the ABox. Full details can be found at `http://www.dcs.bbk.ac.uk/~roman/papers/dl10-full.pdf`.

5   Conclusion

From the complexity-theoretic point of view, the best candidates for reasoning about TCMs appear to be $T_{FPX} DL\text{-}Lite_{core}^{\mathcal{N}}$ and $T_{FPX} DL\text{-}Lite_{bool}^{\mathcal{N}}$: the former is NP-complete and the latter PSPACE-complete. Moreover, we believe that the reduction of $T_{FPX} DL\text{-}Lite_{core}^{\mathcal{N}}$ to *LTL* in the proof of Theorem 3 can be done deterministically, in which case one can use standard *LTL* provers for TCM reasoning. We also believe that $T_{FPX} DL\text{-}Lite_{core}^{\mathcal{N}}$ extended with temporalised roles can be decidable, which remains one of the most challenging open problems. But it seems to be next to impossible to reason in an effective way about all TCM constraints without any restrictions.

References

1. A. Artale, D. Calvanese, R. Kontchakov, and M. Zakharyaschev. *DL-Lite* in the light of first-order logic. In *Proc. of AAAI*, 2007.
2. A. Artale, E. Franconi, F. Wolter, and M. Zakharyaschev. A temporal description logic for reasoning about conceptual schemas and queries. In *Proc. of JELIA*, 2002.
3. A. Artale, C. Parent, and S. Spaccapietra. Evolving objects in temporal information systems. *Annals of Mathematics and Artificial Intelligence*, 50:5–38, 2007.
4. A. Artale, D. Calvanese, R. Kontchakov, V. Ryzhikov, and M. Zakharyaschev. Reasoning over extended ER models. In *Proc. of ER*, 2007.
5. A. Artale, D. Calvanese, R. Kontchakov, and M. Zakharyaschev. The *DL-Lite* family and relations. *J. of Artificial Intelligence Research (JAIR)*, 36:1–69, 2009.

6. A. Artale and E. Franconi. Foundations of temporal conceptual data models. In *Conceptual Modeling: Foundations and Applications*, vol. 5600 of *LNCS*. 2009.

7. A. Artale, R. Kontchakov, C. Lutz, F. Wolter, and M. Zakharyaschev. Temporalising tractable description logics. In *Proc. of TIME*, 2007.

8. A. Artale, R. Kontchakov, V. Ryzhikov, and M. Zakharyaschev. *DL-Lite* with temporalised concepts, rigid axioms and roles. In *Proc. of FroCoS*, 2009.

9. A. Artale, C. Lutz, and D. Toman. A description logic of change. In *Proc. of IJCAI*, 2007.

10. F. Baader, S. Ghilardi, and C. Lutz. LTL over description logic axioms. In *Proc. of KR*, 2008.

11. D. Berardi, D. Calvanese, and G. De Giacomo. Reasoning on UML class diagrams. *Artificial Intelligence*, 168:70–118, 2005.

12. D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, and R. Rosati. Tractable reasoning and efficient query answering in description logics: The *DL-Lite* family. *J. of Automated Reasoning*, 39:385–429, 2007.

13. J. Chomicki and D. Toman. Temporal relational calculus. In *Encyclopedia of Database Systems*, pages 3015–3016. Springer, 2009.

14. M. Chrobak. Finite automata and unary languages. *Theor. Comput. Sci.*, 47:149–158, 1986.

15. C. Date, H. Darwen, and N. Lorentzos. *Temporal Data and the Relational Model*. Morgan Kaufmann, 2002.

16. M. Finger and P. McBrien. Temporal conceptual-level databases. In *Temporal Logics—Mathematical Foundations and Computational Aspects*. Oxford University Press, 2000.

17. D. Gabbay, A. Kurucz, F. Wolter, and M. Zakharyaschev. *Many-Dimensional Modal Logics: Theory and Applications*. Elsevier, 2003.

18. H. Gregersen and J. Jensen. Temporal Entity-Relationship models—a survey. *IEEE TKDE*, 11, 1999.

19. G. Hall and R. Gupta. Modeling transition. In *Proc. of ICDE*, 1991.

20. C. Jensen and R. Snodgrass. Temporally enhanced database design. In *Advances in Object-Oriented Data Modeling*. MIT Press, 2000.

21. C. Lutz, F. Wolter, and M. Zakharyaschev. Temporal description logics: A survey. In *Proc. of TIME*, 2008.

22. P. McBrien, A. Seltveit, and B. Wangler. An Entity-Relationship model extended to describe historical information. In *Proc. of CISMOD*, 1992.

23. A. Mendelzon, T. Milo, and E. Waller. Object migration. In *Proc. of PODS*, 1994.

24. K. Schild. Combining terminological logics with tense logic. *Proc. of EPIA*, 1993.

25. S. Spaccapietra, C. Parent, and E. Zimanyi. *Conceptual Modeling for Traditional and Spatio-Temporal Applications—The MADS Approach*. Springer, 2006.

26. J. Su. Dynamic constraints and object migration. *Theoretical Computer Science*, 184:195–236, 1997.

27. B. Tauzovich. Towards temporal extensions to the entity-relationship model. In *Proc. of ER*. Springer, 1991.

28. C. Theodoulidis, P. Loucopoulos, and B. Wangler. A conceptual modelling formalism for temporal database applications. *Information Systems*, 16:401–416, 1991.

29. A. To. Unary finite automata vs. arithmetic progressions. *Inf. Process. Lett.*, 109:1010–1014, 2009.

MASTRO at Work: Experiences on Ontology-based Data Access

Domenico Fabio Savo[1], Domenico Lembo[1], Maurizio Lenzerini[1],
Antonella Poggi[1], Mariano Rodriguez-Muro[2], Vittorio Romagnoli[3],
Marco Ruzzi[1], Gabriele Stella[3]

[1] SAPIENZA Università di Roma	[2] Free University of Bozen-Bolzano	[3] Banca Monte dei Paschi di Siena
lastname@dis.uniroma1.it	rodriguez@inf.unibz.it	*firstname.lastname*@banca.mps.it

Abstract. We report on an experimentation of Ontology-based Data Access (OBDA) carried out in a joint project with SAPIENZA University of Rome, Free University of Bolzano, and Monte dei Paschi di Siena (MPS), where we used MASTRO for accessing, by means of an ontology, a set of data sources of the actual MPS data repository. By both looking at these sources, and by interviews with domain experts, we designed both the ontology representing the conceptual model of the domain, and the mappings between the ontology and the sources. The project confirmed the importance of several distinguished features of *DL-Lite*$_{A,Id}$ to express the ontology and has shown very good performance of the MASTRO system in all the reasoning tasks, including query answering, which is the most important service required in the application.

1 Introduction

While the amount of data stored in current information systems continuously grows, turning these data into information is still one of the most challenging tasks for Information Technology. The task is complicated by the proliferation of data sources both in single organizations, and in open environments. Specifically, the information systems of medium and large organizations are typically constituted by several, independent, and distributed data sources, and this poses great difficulties with respect to the goal of accessing data in a unified and coherent way. Such a unified access is crucial for getting useful information out of the system, as well as for taking decision based on them. This explains why organizations spend a great deal of time and money for the understanding, the governance, the curation, and the integration of data stored in different sources [7].

The following are some of the reasons why a unified access to data sources is problematic.

– Despite the fact that the initial design of a collection of data sources (e.g., a database) is adequate, corrective maintenance actions tend to re-shape the data sources into a form that often diverges from the original conceptual structure.
– It is common practice to change a data repository so as to adapt it both to specific application-dependent needs, and to new requirements. The result is that data sources often become data structures coupled to a specific application (or, a class of applications), rather than application-independent databases.

    – The data stored in different sources tend to be redundant, and mutually inconsistent, mainly because of the lack of central, coherent and unified data management tasks.

    In principle, there are two alternative solutions to the above problems. One solution is the re-engineering of the information system, i.e., the design of a new, coherent, and unified data repository serving all the applications of the organization [8], and replacing the original data sources. This approach is unfeasible in many situations, due to cost and organization problems. The other solution is to create a new stratum of the information system, co-existing with the data sources, according to the "data integration" paradigm [1]. Such new stratum is constituted by (i) a global (also called "mediated") schema, representing the unified structure presented to the clients, and (ii) the mapping relating the source data with the elements in global schema. There are two methods for realizing such stratum, called materialized and virtual. In the materialized approach, called data warehousing, the global schema is populated with concrete data deriving from the sources. In the virtual approach, data are not moved, and queries posed to the system are answered by suitably accessing the sources [9]. The latter approach, which is the one referred to in this work, is preferable in a dynamic scenario, where sources may be updated frequently, and clients want to use up-to-date information.

    In current data integration tools the global schema is expressed in terms of a logical database model, e.g. the relational data model [1]. It is well-known that the abstractions and the constructs provided by this kind of data models are influenced by implementation issues. It follows that the global schema represents a sort of unified data structure accommodating the various data at the sources, and the client, although freed from physical aspects of the source data (where they are, and how they can be accessed), is still exposed to issues concerning how data are packed into specific structures.

    To overcome these problems, we recently proposed the notion of *ontology-based data integration*, also called *ontology-based data access* (OBDA) [14,12][1], whose basic idea is to express the global schema as an ontology, i.e., a conceptual specification of the application domain. With this idea, the integrated view that the system provides to information consumers is not merely a data structure accommodating the various data at the sources, but a semantically rich description of the relevant concepts and relationships in the domain of interest, with the mapping acting as the reconciling mechanism between the conceptual level and the data sources. Besides this characteristic, OBDA also exploits reasoning on the ontology in computing the answers to queries, thus (at least partially) overcoming possible incompleteness that may be present in the data.

    In this paper we report on an experimentation of OBDA carried out in a joint project by Banca Monte dei Paschi di Siena (MPS)[2], Free University of Bozen-Bolzano, and SAPIENZA Università di Roma, where we used MASTRO [13] for accessing, by means of an ontology, a set of data sources from the actual MPS data repository. MASTRO is an OBDA system extending the QUONTO[3] reasoner, which is based on, *DL-Lite$_{A,Id}$* [2],

---

[1] The two terms have very similar meaning. We tend to use the term "ontology-based data integration" in scenarios where the data sources are heterogenous (i.e., managed by different data management systems), and distributed, which is not the case in the project described here.

[2] MPS is one of the main banks, and the head company of the third banking group in Italy (see http://english.mps.it/).

[3] http://www.dis.uniroma1.it/quonto

one of the logics of the *DL-Lite* family [4]. The OBDA scenario refers to a set of 12 relational data sources, collectively containing about 15 million tuples. By both looking at these sources, and by interviews with domain experts, we designed both the ontology representing the conceptual model of the domain, and the mapping between the ontology and the sources. The ontology comprises 79 concepts and 33 roles, and is expressed in terms of approximately 600 *DL-Lite*$_{A,Id}$ axioms. The relationships between the ontology and the sources are expressed in terms of about 200 mapping assertions. The results of the experimentation can be summarized as follows.

1) In the context of the MPS scenario, OBDA has indeed addressed many of the data access issues mentioned before. The system provides the users with the possibility of querying the data sources by means of the conceptual model of the domain, and this opens up the possibility for a variety of users of extracting information from a set of data sources that previously were accessed through specific applications.

2) The project confirmed the importance of several distinguished features of *DL-Lite*$_{A,Id}$, namely, identification constraints, and epistemic queries. Both features are missing in the standard ontology language OWL 2. In particular, we believe that the absence of identification constraints in OWL 2 may hamper the usefulness of such language in ontology-based data access.

3) MASTRO has shown very good performance in all the reasoning tasks, including query answering, which is the most important service required in the application. This has been achieved by specific optimizations designed within this project of the MASTRO query answering algorithm, in particular concerning the phase of unfolding the query against the mapping.

4) The experience in this project has shown that OBDA can be used for checking the quality of data sources. There are basically two kinds of data quality problems that our system is able to detect, one related to unexpected incompletenesses in the data sources, and the other one related to inconsistencies present in the data. The OBDA system designed for the MPS scenario has been able to provide useful information in order to improve both aspects of data quality.

5) Our work has pointed out the importance of the ontology itself, as a precious documentation tool for the organization. Indeed, the ontology developed in our project is adopted in MPS as a specification of the relevant concepts in the organization.

6) The OBDA system serves also as an inspiration for devising new data governance tasks. Relying on OBDA services, queries such as "how is a certain concept (e.g., *customer*) represented in a specific data source (e.g., table $GZ0005$)?" can now be answered, simply by exploiting both the ontology and the mappings designed in the project, and the query reformulation capability of MASTRO.

The paper is organized as follows. Section 2 presents a brief description of MASTRO. Sections 3 illustrates the scenario of our experimentation. Section 4 presents the ontology and the mapping. Section 5 illustrates the use of MASTRO in the scenario. Section 6 concludes the paper.

2   The MASTRO system

MASTRO is an OBDA system jointly developed at the SAPIENZA University of Rome and Free University of Bozen-Bolzano. MASTRO allows for the definition of

DL-Lite$_{A,Id}$ [2] ontologies connected through semantic mappings to external independent relational databases storing data to be accessed. Thus, differently from other approaches to ontology definition and reasoning [10,6,11], the extensional level of the ontology, namely, the instances of concepts and roles, are not explicitly asserted and possibly managed by a DBMS, but are specified by mapping assertions describing how they can be retrieved from the data at the sources. In the following we briefly sketch the architecture of the system, distinguishing between "Ontology Definition Module", "Mapping Manager", "Data Source Manager", and "Reasoner".

The *Ontology Definition Module* provides mechanisms for the specification of the ontology as a *DL-Lite*$_{A,Id}$ TBox. *DL-Lite*$_{A,Id}$ is a Description Logic (DL) belonging to the *DL-Lite* family, which adopts the Unique Name Assumption, and provides all the constructs of OWL 2 QL[4], a tractable profile of OWL 2, plus functionality and identification assertions, with the limitation that these kind of assertions cannot involve sub-roles. These last features, while enhancing the expressive power of the logics, do not endanger the efficiency of both intensional reasoning, and query answering. In other words, the computational complexity of these tasks is the same as in OWL 2 QL, namely PTIME with respect to the size of the TBox, and LOGSPACE in the size of the data at the sources.

The *Mapping Manager* supports the definition of mapping assertions relating the data at the sources to the concepts in the ontology. The mapping assertions supported by MASTRO are a particular form of GAV mappings [9]. More specifically, a mapping assertion is an expression of the form $\psi \rightsquigarrow \varphi$ where ψ is an arbitrary SQL query over the database, and φ is a *DL-Lite*$_{A,Id}$ conjunctive query without existential variables. As described in [12], data extracted by means of query ψ are used, together with suitable Skolem functions, to build the logic terms representing the object identifiers, thus solving the impedance mismatch problem between data at the sources and instances of the ontology. The Mapping Manager interacts with the *Data Source Manager*, which is in charge of the communication with the underlying relational sources, providing transparent access to a wide range of both commercial and freeware relational DBMSs[5].

Finally, the *Reasoner* exploits both the TBox and the mapping assertions in order to (i) check the satisfiability of the whole knowledge base, and (ii) compute the answer to the queries posed by the users. Such module is based on QUONTO, a reasoner for the *DL-Lite* family that uses query rewriting as a main processing technique. The two main run-time services provided by the reasoner are query answering, and consistency check. The MASTRO process to answer conjunctive queries (CQs) is inspired by the one implemented in the QUONTO system. First, the query posed by the user over the ontology is reformulated in terms of the inclusion assertions expressed among concepts and roles; second, such rewriting is *unfolded* according to the mapping assertions in order to generate an SQL query which can be directly issued over the relational data source. It can be shown that the answers to such an SQL query are exactly the answers logically implied by the whole knowledge base [2]. As a further powerful feature, MASTRO is able to answer EQL (Epistemic Query Language) queries [3], i.e., first-order logic queries over the ontology interpreted under an epistemic semantics. Finally, MASTRO provides the

---

[4] http://www.w3.org/TR/owl2-profiles/
[5] No relational sources can be accesses by means of suitable wrapping tools

consistency check capability. By virtue of the characteristics of *DL-Lite$_{A,Id}$*, MASTRO reduces consistency checking to verifying whether queries generated for disjointness assertions, functionality assertions, identification constraints and EQL constraints return an empty result. To this aim, a boolean query is automatically generated for every such construct and then rewritten, unfolded, and evaluated over the database.

3   Case study: The domain of experimentation

The data of interest in our case study are those exploited by MPS personnel for risk estimation in the process of granting credit to bank customers. A customer may be a person, an ordinary company, or an holding company. Customers are ranked with respect to their credit worthiness, which is established considering various circumstances and credit/debit positions of customers. In addition to customer information, data of interest regard company groups to which customers belong, and business relations between bank customers (in particular, fifteen different kinds of such relations are relevant).

Hereinafter, such groups of customers will be called *Clusters of Connected Customers (CCCs)*. A 15 million tuple database, stored in 12 relational tables managed by the IBM DB2 RDBMS, has been used as data source collection in the experimentation. Figure 1 shows a summary of the data sources. Such data sources are managed by a specific application. The application is in charge of guaranteeing data integrity (in fact, the

Source name	Source Decription	Source size
GZ0001	Data on customers	3.463.083
GZ0002	Data on juridical connections between customers	157.280
GZ0003	Data on guarantee connection between customers	1.270.333
GZ0004	Data on economical connections between customers	104.033
GZ0005	Data on corporation connections between customers	1.021.779
GZ0006	Data on patrimonial connections between customers	809.321
GZ0007	Data on company groups	55.362
GZ0012	Customers loan information	5.966.948
GZ0015	Data on monitoring and reporting procedures	1.243
GZ0101	Data on membership of customers into CCCs	2.225.466
GZ0102	Information on CCCs	663.656
GZ0104	Data on bank credit coordinators for juridical CCCs	38.457

Fig. 1. Data sources

underlying database does not force constraints on data). Not only this application performs various updates, but an automatic procedure is executed on a daily basis to examine the data collected in the database so as to identify connections between customers that are relevant for the credit rating calculus. Based on these connections, customers are grouped together to form CCCs. For each cluster, several data are collected that characterize the kinds of connections holding among cluster members (i.e., specifying juridical, economic, or financial aspects of connections).

Data source schemas have undergone many changes in the years, trying to adapt to the changes in the application. The result is a stratification of the data source which causes an extended use of control fields, validity flags, and no longer used attributes in the source schemas. Consequently, an increasing effort for the management of the data sources is required, which has to be completely entrusted to the management applications rather than the domain experts. The aim of the experimentation has been to prove the validity of the OBDA approach in all cases in which companies need to access efficiently their information assets.

4   Case study: ontology, mapping, and methodology

The process that led us to realize the OBDA system for the MPS case study has been carried out essentially in two main phases: in the first one, we have developed the ontology, whereas in the second one we have specified the mapping between the ontology and the data sources.

To be as much independent as possible from the actual source database, in the first phase we carried out an in-depth analysis of the business domain following a top-down approach. Therefore, after identifying the central concepts and the main relations between them, we iteratively refined the ontology, being supported in each development cycle by the experts from MPS. The top-down approach turned out to be fundamental for the success of the entire project, since in this way we were able to avoid that the data model provided by the schema of the data sources could affect the definition of the ontology, thus achieving complete separation between the conceptual layer and the logical/physical layer of the system. In fact, further information on the model coming from the analysis of the sources has been exploited only towards the end of the design process, in order to refine the realized ontology.

The final ontology comprises 79 concepts, 33 roles, 37 concept attributes, and is expressed in terms of about 600 *DL-Lite$_{A,Id}$* axioms, including 30 identification constraints (IDCs), plus 20 EQL constraints (EQLCs). Basically, the ontology is constructed around the concepts *Customer*, *CompanyGroup*, *CCC*, and various kinds of relations existing between customers (cf. Section 3).

In the following, we report on a series of modeling issues we dealt with during the ontology definition phase. First, we observe that in the domain we have analyzed, several properties of individuals depend on time. It has been therefore necessary in the ontology to take trace of the changes of such properties, maintaining the information on the validity periods associated with each such change. Even though from a very abstract point of view, such properties might be considered roles or attributes, to properly model the temporal dimension, each such role or attribute needs to be in fact *reified* in the ontology. A timestamp attribute has been associated to each concept introduced by the reification process, together with a suitable identification constraint ensuring that no two instances of each such concept refer to the same period of time.

Example 1. The membership of a customer in a cluster of connected customers is a time-dependent notion which is associated with a validity period. A crucial requirement is that a customer is not member of two clusters at the same time. In the ontology, this is modeled by the following assertions.

1. $\exists inGrouping \sqsubseteq Customer$
2. $\exists inGrouping^{-} \sqsubseteq Grouping$
3. $\exists relativeTo \sqsubseteq Grouping$
4. $\exists relativeTo^{-} \sqsubseteq CCC$
5. $Grouping \sqsubseteq \exists inGrouping^{-}$
6. $Grouping \sqsubseteq \exists relativeTo$
7. $(\mathsf{funct}\ relativeTo)$
8. $(\mathsf{funct}\ inGrouping^{-})$
9. $Grouping \sqsubseteq \delta(timestamp)$
10. $(\mathsf{id}\ Grouping\ inGrouping^{-}, timestamp)$

The concept *Grouping* can be seen as the reification of the notion of membership of a customer in a CCC. Assertions (1) – (8) realize reification. Assertion (9) imposes that a *timestamp* is associated to each instance of *Grouping*. Finally, assertion (10) is the IDC imposing that no two distinct instances of $Grouping$ exist that are connected to the same pair constituted by a value for the attribute $timestamp$ and an object filler for $inGrouping^{-}$, thus specifying that a customer is never grouped at the same time in two CCCs.

Identification constraints turned out to be an essential modeling construct, not only for correctly modeling the temporal dimension through reification, but also for expressing important integrity constraints over the ontology that could not be captured otherwise, as shown next in Example 2.

Example 2. Two types of clusters of connected customers are of interest represented by the concepts *JuridicalCCC* and *EconomicCCC*, respectively. Consider then the following identification constraint on *JuridicalCCC*.

$$(id\ JuridicalCCC\ timestamp,\ relativeTo^{-} \circ\ ?actualGrupping \circ\ inGrouping^{-} \circ$$
$$inMembership \circ\ ?Holding \circ\ hasMembership^{-})$$

Such constraint specifies that no two distinct instances of *JuridicalCCC* exist that are connected to the same pair constituted by a value for $timestamp$ and an object filler for the $path\ relativeTo^{-}\circ?actualGrupping \circ\ inGrouping^{-} \circ inMembership\circ?Holding \circ\ hasMembership^{-}$. Intuitively, the path navigates through the roles of the ontology, using the construct $?C$ to test that the path passes through instances of C. Since the role $hasMembership$ is typed in the ontology by the concept *CompanyGroup*, the identification constraint actually says that for a certain timestamp no two juridical CCCs exists that are connected via the above path to the same company group.

Globally, we have specified more than 30 IDCs in the ontology. None of these presently correspond to integrity constraints at the data sources. This is because, as it is usual in practice, very few integrity constraints are explicitly asserted at the sources. Thus, our ontology plays an important role in representing business rules not explicitly reflected in the data repository of the organization.

EQLCs turned out to be another important means for correct domain modeling. Such constraints indeed permit to overcome some expressiveness limitations of *DL-Lite*$_{A,Id}$, without causing any computational blow up. Indeed, EQLCs are interpreted according to a suitable semantic approximation (cf. Section 2). In this experimentation we have heavily used EQLCs to express, e.g., hierarchy completeness and other important business constraints, otherwise not expressible in our ontology.

Example 3. An important constraint we want to force on the ontology is that for every customer which has a guarantor for a loan we have to know the amount of bank credit provided to the customer. This is specified through the following EQLC, which is expressed in SparSQL, a query language presented in [5] based on SPARQL and SQL:

```
EQLC( verify not exists (
   SELECT withGuarantor.cus, withGuarantor.t
   FROM sparqltable( SELECT ?cus ?t
                     WHERE{ ?cus :isLinked ?link.
                            ?link rdf:type 'GuaranteeRelations'.
                            ?link :timestamp ?t}) withGuarantor
   WHERE (withGuarantor.cus, withGuarantor.t) NOT IN (
         SELECT withCredit.cus, withCredit.t
         FROM sparqltable( SELECT ?cus ?amnt ?t
                     WHERE{ ?cus :hasLoan ?loan.
                      ?loan :creditAmount ?amnt.
                      ?loan :timestamp ?t }) withCredit )))
```

The above constraint says that no customer cus exists, such that cus is connected to an instance of the concept $GuaranteeRelations$ at the time t, and cus has not a "known" $creditAmount$ at the same time t. It is worth noticing that OWL 2, despite its expressiveness, does not allow for expressing the above constraint.

Let us now turn our attention to mapping specification. The mapping specification phase has required a complete understanding and an in-depth analysis of the data sources, which highlighted some modeling weaknesses present in the source database schema: various modifications stratified in the years over the original data schema have partially transformed the data sources, which now reveal some problems related to redundancy, inconsistency, and incompleteness in the data. Localizing the right data to be mapped to ontology constructs has thus required the definition of fairly complex mapping assertions, as shown in Example 4.

Example 4. Consider the following mapping assertion specifying how to construct instances of *JuridicalCCC* using data returned by an SQL query accessing both the table $GZ0102$, which contains information about CCCs, and the table $GZ0007$, which contains information about the company groups.

```
SELECT id_cluster, timestamp_val FROM GZ0102, GZ0007
WHERE GZ0102.validity_code = 'T' AND GZ0102.id_cluster <> 0
    AND GZ0007.validity_code = 'T' AND GZ0007.id_group <> 0
    AND GZ0102.id_cluster = GZ0007.id_group
⤳  JuridicalCCC(ccc(id_cluster, timestamp_val))
```

From the data source analysis it turned out that each CCC that has an identifier ($GZ0102.id\_cluster$) coinciding with the identifier of a company group ($GZ0007.id\_group$) is a juridical CCC. Such a property is specified in the SQL query in the mapping through the join between $GZ0102$ and $GZ0007$ ($GZ0102.id\_cluster = GZ0007.id\_group$). Notice that invalid tuples (those with $validity\_code$ different from 'T') and meaningless tuples (those with $id\_cluster$ or id_group equal zero) are excluded from the selection. The query returns pairs of $id\_cluster$ and $timestamp\_val$, which are used as arguments of the function $ccc()$ to build logic terms representing objects that are instances of *JuridicalCCC*, according to the method described in [12].

The mapping specification phase has produced around 200 mapping assertions, many of which are quite involved. Their design has been possible by a deep understanding of the tables involved, their attributes, and the values they store. We initially tried to automate this process with the help of current tools for automatic mapping generation, but, due to the complexity of extracting the right semantics of the source tables, we failed. This is in line with our past experience on mapping design: the bulk of the work in mapping specification has to be essentially carried out manually.

5 The system at work

In this section we discuss the actual use of MASTRO in the MPS scenario. As a general comment, we remark that the OBDA system we designed for this scenario allowed to overcome many of the data access problems we have discussed in the previous sections. In particular, querying the data sources through the conceptual view provided by the ontology enabled various kinds of users, not necessarily experts of the application managing data at the sources, to profitably access such data. In what follows, we concentrate on two crucial aspects of our experience: the use we made of MASTRO to check the quality of the data sources, and the impact that certain characteristics of

the MPS scenario have had on the evolution of the system in terms of its tuning and optimizations.

As mentioned in the introduction, we faced two main issues concerning the quality of the data sources, namely incompleteness and inconsistency in the data at the sources. Detecting data incompleteness has been possible by exploiting the MASTRO query answering services, and more precisely, by inspecting the rewriting and the unfolding that MASTRO produces in the query answering process. Let us see this on an example. To retrieve from the data sources the identification codes of all company groups, MPS operators simply use a single SQL query projecting out the id_code from the table *GZ0007*, which contains information about company groups. Surprisingly, using the ontology to obtain all company codes, we actually get a larger answer set, by posing over the ontology the obvious corresponding query $q(y) \leftarrow CompanyGroup(x), id\_code(x, y)$. The reason for such a difference in the answers resides in the fact that the query that MAS-TRO asks to the source database, and that is automatically produced by the rewriting and unfolding procedures of MASTRO, is much more complex than the query used by the MPS operators. By reasoning over the ontology, and exploiting the mapping assertions, MASTRO accesses all the source tables that store codes of company groups, and this set of tables does not in fact contain only the codes of company groups that occur in table *GZ0007*. Such a result showed that some foreign key dependencies constraining the identification codes stored in the table *GZ0007* were in fact missing in the source database, and that such a table should not been considered complete with respect to such information.

We turn now to data inconsistency issues. In *DL-Lite$_{A,Id}$*, inconsistencies are caused by data that violate the assertions of the ontology, specifically disjointness assertions, functionality constraints, identification constraints, and EQL constraints. Also, causes of inconsistencies can be easily localized by retrieving the minimal set of data that produce each single violation. We actually modified the classical consistency check of MASTRO in order to identify the offending data, in particular exploiting the feature of answering EQL queries (cf. Section 2) and their ability to express negation. Consider for example the relation $linkedTo$, which is declared to be inverse functional (i.e., (funct $linkedTo^-$)). In order to detect the violation of such constraint and the guilty data, we use the following EQL query:

```
SELECT testview.l, testview.c1, testview.c2
FROM sparqltable (SELECT ?l ?c1 ?c2
                WHERE{?c1:linkedTo?l. ?c2:linkedTo?l}) testview
WHERE testview.c1 <> testview.c2
```

Switching our attention to the performance of the system, there are two sources of complexity to be considered in the query answering and consistency checking services provided in MASTRO, the query *reformulation* and query *unfolding* procedures. Reformulation introduces complexity since it may produce an exponential number of queries to be answered. Nevertheless, in the case of the MPS ontology, this potential drawback did not occur. Indeed, in most cases, the number of queries produced by this step was small (between 1 and 25). In contrast, the query unfolding step presented challenges that led to several important improvements in MASTRO, briefly discussed below.

In complex scenarios, such as the one we considered in our experimentation, we found that the most critical aspect for performance is what we call *query structure*, i.e.,

the form of the SQL queries issued to the source database. Query structure is characterized by the specific technique used to produce SQL queries out of queries formulated over TBox predicates ($\mathcal{T}$-predicates).

In MASTRO, query unfolding is based on the use of *SQL views* over the source database. More specifically, the mapping is first pre-processed so as to have only assertions in which the query over the ontology contains just one predicate (splitting). Then, all assertions referring to the same $\mathcal{T}$-predicate are combined together in order to have one SQL view, which we call $\mathcal{T}$-*view*, for each predicate. Essentially, the view is obtained taking the union of the SQL queries occurring in the left-hand side of the assertions, and pushing the construction of logic terms representing instances of concepts and roles in the view itself. Unfolding a query specified over $\mathcal{T}$-predicates amounts therefore to simply unfold each query atom with the corresponding $\mathcal{T}$-view. For example, if the split mapping assertions for the role *linkedTo* are

m_1: `SELECT .... WHERE cd_tp = 503` $\rightsquigarrow linkedTo(\mathbf{cus}(idcus), \mathbf{link}(linkid))$
m_2: `SELECT .... WHERE cd_tp = 501` $\rightsquigarrow linkedTo(\mathbf{cus}(idcus), \mathbf{link}(linkid))$

then, the following view, `linkedto_Tview`, is associated to the *linkedTo* predicate:

```
SELECT 'cus('||idcus||')' as term1, 'link('||linkid||')' as term2
FROM (SELECT .... WHERE cd_tp = 503) view_m1
UNION
SELECT 'cus'(||idcus||')' as term1, 'link('||linkid||')' as term2
FROM (SELECT ....  WHERE cd_tp = 501) view_m2
```

Notice that in the `SELECT` clause we build logical terms by means of simple SQL string concatenation operations, indicated with the `||` operator. Then, the query $q(X) \leftarrow linkedTo(X, Y)$ is unfolded into `SELECT term1 FROM linkedto_Tview`.

Despite its simplicity, we found out that, in scenarios characterized by a high volume of data and complex and numerous mapping assertions, this approach fail, due to low performance of the generated queries. For example, in our test cases, queries with a single atom that involve database relations with high volume of data often required several minutes to be answered. More complex queries, with more than 2 atoms and involving also big relations, would often require hours or would even not terminate. The reason for this bad performance is in the limitations of DBMS query planners in handling subqueries in the `FROM` clause, and joins between terms representing objects, rather than directly on database values. What we observed is that, in order to deal with subqueries, query planners rely on a process called query *flattening*, in which the query planner attempts to rephrase a query with subqueries into a new query with no subqueries. If the query planner is not successful in this attempt, e.g., due to the complexity of the subqueries, it will resort to subquery materialization, an extremely expensive operation when the volume of data is high.

In order to avoid materialization and joins between object terms, and in general, to increase the chances of the query planner to produce a *good plan*, we devised a strategy that led us to produce queries that are as simple as possible with respect to subqueries. This led us to adopt what we call an $\mathcal{M}$-*view* approach to unfolding. In this approach, we build simpler views, one for each SQL query in the split mapping assertions, and we associate *all* of them to the corresponding $\mathcal{T}$-predicate. For example, in the previous case we would define the two views below

```
view_m1 = SELECT .... WHERE cd_tp = 503
view_m2 = SELECT .... WHERE cd_tp = 501
```

and the unfolding of the query $q(X) \leftarrow linkedTo(X, Y)$ would be as follows

```
SELECT 'cus('||idcus||')' FROM view_m1
UNION
SELECT 'cus('||idcus||')' FROM view_m2
```

Notice that in this case, the construction of the object term `'cus('||idcus||')'` is in the external `SELECT` clause and is not pushed into the views in the `FROM` clause.

What is important to note here is the exchange of simplicity of the unfolding procedure for simplicity of the structure of the queries being generated (i.e., less nesting in the subqueries) and a new exponential growth in the amount of queries sent to the database, e.g., now for every *linkedTo* atom in a query, we will produce an SQL query taking the union of at least two queries, one where we only use `view_m1` and one with `view_m2`. Although this growth could seem problematic, we have found that the increase in the performance of executing each individual query pays off the increase in the number of queries to be executed. Moreover, since these queries are independent, we can use parallelism in query execution to improve performance even more.
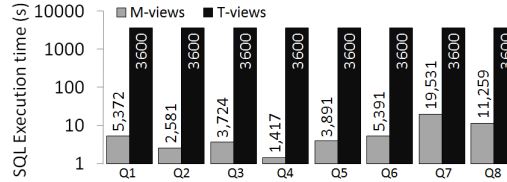
Fig. 2. $\mathcal{M}$-views vs. $\mathcal{T}$-views using an execution timeout of 1hr.

To give an idea of the effectiveness of the described optimizations, we present in Figure 2 the data about the execution of a collection of 8 representative queries (the units of the vertical axis are seconds). These queries are all of interest to MPS, and challenging in terms of number of atoms, complexity of the unfolding and the volume of data accessed.

6 Conclusions

From the point of view of MPS, the project has provided very useful results in various areas of interest:

− Data integration, providing the capability of accessing application data in a unified way, by means of queries written at a logical/conceptual level by end-users not necessarily acquainted with the characteristics of the application;

− Database quality improvement, providing tools for monitoring the actual quality of the database, both at an intensional and an extensional level;

− Knowledge sharing, providing, with the ontology-based representation of the application domain, an efficient means of communicating and sharing knowledge and information throughout the company.

The plan is to continue the experience by extending the work to other MPS applications, with the idea that the ontology-based approach could result in a basic step for the future IT architecture evolution, oriented towards Service-oriented architectures and Business Process Management.

References

1. P. A. Bernstein and L. Haas. Information integration in the enterprise. *Comm. of the ACM*, 51(9):72–79, 2008.
2. D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, A. Poggi, M. Rodriguez-Muro, and R. Rosati. Ontologies and databases: The *DL-Lite* approach. In S. Tessaris and E. Franconi, editors, *Reasoning Web Summer School 2009*, volume 5689 of *LNCS*, pages 255–356. Springer, 2009.
3. D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, and R. Rosati. EQL-Lite: Effective first-order query processing in description logics. In *Proc. of IJCAI 2007*, pages 274–279, 2007.
4. D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, and R. Rosati. Tractable reasoning and efficient query answering in description logics: The *DL-Lite* family. *J. of Automated Reasoning*, 39(3):385–429, 2007.
5. C. Corona, E. D. Pasquale, A. Poggi, M. Ruzzi, and D. F. Savo. When OWL met DL-Lite... In *SWAP-08*, 2008.
6. J. Dolby, A. Fokoue, A. Kalyanpur, L. Ma, E. Schonberg, K. Srinivas, and X. Sun. Scalable grounded conjunctive query evaluation over large and expressive knowledge bases. In *Proc. of ISWC 2008*, volume 5318 of *LNCS*, pages 403–418. Springer, 2008.
7. L. M. Haas. Beauty and the beast: The theory and practice of information integration. In *Proc. of ICDT 2007*, volume 4353 of *LNCS*, pages 28–43. Springer, 2007.
8. J. Henrard, D. Roland, A. Cleve, and J.-L. Hainaut. Large-scale data reengineering: Return from experience. In *WCRE '08: Proceedings of the 2008 15th Working Conference on Reverse Engineering*, pages 305–308. IEEE Computer Society, 2008.
9. M. Lenzerini. Data integration: A theoretical perspective. In *Proc. of PODS 2002*, pages 233–246, 2002.
10. C. Lutz, D. Toman, and F. Wolter. Conjunctive query answering in the description logic $\mathcal{EL}$ using a relational database system. In *Proc. of IJCAI 2009*, pages 2070–2075, 2009.
11. H. Pérez-Urbina, B. Motik, and I. Horrocks. Tractable query answering and rewriting under description logic constraints. *J. of Applied Logic*, 2009. To appear.
12. A. Poggi, D. Lembo, D. Calvanese, G. De Giacomo, M. Lenzerini, and R. Rosati. Linking data to ontologies. *J. on Data Semantics*, X:133–173, 2008.
13. A. Poggi, M. Rodriguez, and M. Ruzzi. Ontology-based database access with DIG-Mastro and the OBDA Plugin for Protégé. In K. Clark and P. F. Patel-Schneider, editors, *Proc. of OWLED 2008 DC*, 2008.
14. M. Rodriguez-Muro, L. Lubyte, and D. Calvanese. Realizing ontology based data access: A plug-in for Protégé. In *Proc. of IIMAS 2008*, pages 286–289. IEEE CS Press, 2008.

Justification Masking in OWL

Matthew Horridge[1], Bijan Parsia[1], Ulrike Sattler[1]

The University of Manchester, UK

Abstract. This paper presents a discussion on the phenomena of masking in the context of justifications for entailments. Various types of masking are introduced and a definition for each type is given.

1 Introduction

Many open source and commercial ontology development tools such as Protégé-4, Swoop, The NeOn Toolkit and Top Braid Composer use justifications [5] as a kind of explanation for entailments in ontologies. A *justification* for an entailment, also known as a *MinA* [1, 2], or a *MUPS* [11] if specific to explaining why a class name is unsatisfiable, is a minimal subset of an ontology that is sufficient for the given entailment to hold. More precisely, a justification is taken to be a subset minimal set of axioms that supports an entailment. Justifications are a popular in the OWL world and, as the widespread tooling support shows, have been used in preference to full blown proofs for explaining why an entailment follows from a set of axioms.

However, despite the popularity of justifications, they suffer from several problems. Some of these problems, namely issues arising from the potential superfluity of axioms in justifications, were highlighted in [3]. Specifically, while all of the axioms in a justification are needed to support the entailment in question, there may be *parts* of these axioms that are not required for the entailment to hold. For example, consider $\mathcal{J} = \{A \sqsubseteq \exists R.B, Domain(R, C), C \sqsubseteq D \sqcap E\}$ which entails $A \sqsubseteq D$. While $\mathcal{J}$ is a justification for $A \sqsubseteq D$, and *all axioms* are required to support this entailment, there are *parts* of these axioms that are superfluous as far as the entailment is concerned: In the first axiom the filler of the existential restriction is superfluous, in the third axiom the conjunct E is superfluous for the entailment.

An important phenomenon related to superfluity has become known as *justification masking*. Recalling that there may be several justifications for an entailment, which may but do not have to overlap, masking refers to the case where the number of justifications for an entailment does not reflect the number of reasons for that entailment. For example, consider $\mathcal{J} = \{A \sqsubseteq \exists R.C \sqcap \forall R.C, D \equiv \exists R.C\}$ which entails $A \sqsubseteq D$. Clearly, $\mathcal{J}$ is a justification for $A \sqsubseteq D$. It is also noticeable that there are superfluous parts in this justification. Moreover, there are two distinct reasons why $\mathcal{J} \models A \sqsubseteq D$, the first being $\{A \sqsubseteq \exists R.C, \exists R.C \sqsubseteq D\}$ and the second being $\{A \sqsubseteq \exists R.\top \sqcap \forall R.C, \exists R.C \sqsubseteq D\}$. The work presented in the paper describes how masking can occur within a justification, over a set of justifications, and over a set of justifications and axioms outside justifications. The main

problems identified with masking are (i) it can hamper understanding—not all reasons for an entailment may be salient to a person trying to understand the entailment, and (ii) it can hamper the design or choice of a repair plan—not all reasons for an entailment may be obvious, and if the plan consists of weakening and removing parts of axioms it may not actually result in a successful repair of the ontology in question.

In [3] *laconic* and *precise* justifications were presented as a tool for dealing with the problems of superfluity and masking. However, while the basic *intuitions* of masking were presented in [3], and it was shown that laconic justifications could be used as a tool for working with masking, only two types of masking where discussed. This paper presents a comprehensive analysis of the different types of masking, provides a characterisation of masking, and lays down definitions and an analysis for the various types of masking.

2   Preliminaries

The work presented in this paper focuses on OWL 2. OWL 2 [8] is the latest standard in ontology languages from the World Wide Web Consortium. An OWL 2 ontology roughly corresponds to a $\mathcal{SROIQ}(D)$ [4] knowledge base. For the purposes of this paper, an *ontology* is regarded as a finite set of $\mathcal{SROIQ}$ axioms $\{\alpha_0, \ldots, \alpha_n\}$. An axiom is of the form of $C \sqsubseteq D$ or $C \equiv D$, where C and D are (possibly complex) concept descriptions, or $S \sqsubseteq R$ or $S \equiv R$ where S and R are (possibly inverse or complex) roles.

It should be noted that OWL contains a significant amount of syntactic sugar, such as $\mathrm{DisjointClasses}(C, D)$, $\mathrm{FunctionalObjectProperty}(R)$ or $\mathrm{Domain}(R, C)$. However, these axioms can be represented using sub-class and sub-property axioms.

Justifications are a popular form of explanation in the OWL world. A justification for an entailment η in an ontology $\mathcal{O}$, such that $\mathcal{O} \models \eta$ is a minimal subset of that entails η.

Definition 1 (Justification). $\mathcal{J}$ *is a justification for* $\mathcal{O} \models \eta$ *if* $\mathcal{J} \subseteq \mathcal{O}$, $\mathcal{J} \models \eta$ *and for all* $\mathcal{J}' \subsetneq \mathcal{J}$ $\mathcal{J}' \not\models \eta$.

By a slight abuse of notation, the nomenclature used in this paper also refers to a minimally entailing set of axioms (that is not necessarily a subset of an ontology) as a justification.

Much of the work presented in the remainder of the paper uses the "well known" structural transformation — δ. This transformation takes a set of axioms and flattens out each axiom by introducing names for sub-concepts, transforming the axioms into an equi-satisfiable set of axioms. The structural transformation was first described in Plaisted and Greenbaum [10], with a version of the rewrite rules for description logics given in [9].

In what follows, $\mathcal{A}$ is the ABox of an ontology, $\mathcal{R}$ is the RoleBox, and $\mathcal{T}$ is the TBox. A is an atomic concept in the signature of $\mathcal{O}$, A_D and A'_D are fresh concept names that are not in the signature of $\mathcal{O}$. C_i and D are arbitrary

concepts, excluding $\top$, $\bot$ and literals of the form X or $\neg X$ where X is not in the signature of $\mathcal{O}$, $\mathbf{C}$ is a possibly empty disjunction of arbitrary concepts. $C \equiv D$ is syntactic sugar for $C \sqsubseteq D$ and $D \sqsubseteq C$, as is $=nR.D$ for $\geq nR.D \sqcap \leq nR.D$. Domain and range axioms are GCIs so that $Domain(R, C)$ means $\exists R.\top \sqsubseteq C$, and $Range(R, C)$ means $\top \sqsubseteq \forall R.C$. The negation normal form of D is $\mathsf{nnf}(D)$. The structural transformation δ is defined as follows:

$$\delta(\mathcal{O}) := \bigcup_{\alpha \in \mathcal{R} \cup \mathcal{A}} \delta(\alpha) \cup \bigcup_{C_1 \sqsubseteq C_2 \in \mathcal{T}} \delta(\top \sqsubseteq \mathsf{nnf}(\neg C_1 \sqcup C_2))$$

$$\delta(D(a)) := \delta(\top \sqsubseteq \neg\{a\} \sqcup \mathsf{nnf}(D))$$

$$\delta(\top \sqsubseteq \mathbf{C} \sqcup D) := \delta(\top \sqsubseteq A'_D \sqcup \mathbf{C}) \cup \bigcup_{i=1}^{i=n} \delta(A'_D \sqsubseteq D_i) \text{ for } D = \prod_{i=1}^{i=n} D_i$$

$$\delta(\top \sqsubseteq \mathbf{C} \sqcup \exists R.D) := \delta(\top \sqsubseteq A_D \sqcup \mathbf{C}) \cup \{A_D \sqsubseteq \exists R.A'_D\} \cup \delta(A'_D \sqsubseteq D)$$

$$\delta(\top \sqsubseteq \mathbf{C} \sqcup \forall R.D) := \delta(\top \sqsubseteq A_D \sqcup \mathbf{C}) \cup \{A_D \sqsubseteq \forall R.A'_D\} \cup \delta(A'_D \sqsubseteq D)$$

$$\delta(\top \sqsubseteq \mathbf{C} \sqcup {\geq} nR.D) := \delta(\top \sqsubseteq A_D \sqcup \mathbf{C}) \cup \{A_D \sqsubseteq {\geq} nR.A'_D\} \cup \delta(A'_D \sqsubseteq D)$$

$$\delta(\top \sqsubseteq \mathbf{C} \sqcup {\leq} nR.D) := \delta(\top \sqsubseteq A_D \sqcup \mathbf{C}) \cup \{A_D \sqsubseteq {\leq} nR.A'_D\} \cup \delta(A'_D \sqsubseteq D)$$

$$\delta(A'_D \sqsubseteq D) := A'_D \sqsubseteq D \text{ (If } D \text{ is of the form } A \text{ or } \neg A)$$

$$\delta(A'_D \sqsubseteq D) := \delta(\top \sqsubseteq \neg A'_D \sqcup D) \text{ (If } D \text{ is } not \text{ of the form } A \text{ or } \neg A)$$

$$\delta(\beta) := \beta \text{ for any other axiom}$$

The transformation ensures that concept names that are in the signature of $\mathcal{O}$ only appear in axioms of the form $X \sqsubseteq A$ or $X \sqsubseteq \neg A$, where X is some concept name *not* occurring in the signature of $\mathcal{O}$. Note that the structural transformation does not use structure sharing. For example, given $\top \sqsubseteq C \sqcup \exists R.C$, two new names are introduced, one for each use of C, to give $\{\top \sqsubseteq X_0 \sqcup X_1, X_0 \sqsubseteq C, X_1 \sqsubseteq \exists R.X_2, X_2 \sqsubseteq C\}$. The preclusion of structure sharing ensures that the different positions of C are captured.

The definition of laconic justifications uses the notion of the length of an axiom. Length is defined as follows: For X, Y a pair of concepts or roles, A a concept name, and R a role, the length of an axiom is defined as follows:

$$|X \sqsubseteq Y| := |X| + |Y|, \quad |X \equiv Y| := 2(|X| + |Y|),$$

where

$$|\top| = |\bot| := 0,$$
$$|A| = |\{i\}| = |R| = |R^-| := 1,$$
$$|\neg C| := |C|$$
$$|C \sqcap D| = |C \sqcup D| := |C| + |D|$$
$$|\exists R.C| = |\forall R.C| = |{\geq} nR.C| = |{\leq} nR.C| := |R| + |C|$$

It should be noted that this definition is slightly different from the usual definition, but it allows cardinality axioms such as $A \sqsubseteq {\leq} 2R.C$ to be weakened to $A \sqsubseteq {\leq} 3R.C$ without increasing the length of the axiom.

In what follows the standard definition of deductive closure is used, and $\mathcal{O}^\star$ is used to denote the deductive closure of $\mathcal{O}$.

Definition 2. *$\mathcal{J}$ is a laconic justification for η over $\mathcal{O}$ if:*

1. $\mathcal{J}$ is a justification in $\mathcal{O}^\star$.
2. $\delta(\mathcal{J})$ is a justification in $(\delta(\mathcal{O}))^\star$
3. For each $\alpha \in \delta(\mathcal{J})$ there is no α' such that
    (a) α' is weaker than α $(\alpha \models \alpha'$ but $\alpha' \not\models \alpha)$
    (b) $|\alpha'| \leq |\alpha|$
    (c) $(\delta(\mathcal{J}) \setminus \{\alpha\}) \cup \delta(\alpha')$ is a justification for η

Intuitively, a laconic justification is a justification whose axioms do not contain any superfluous parts and all of whose parts are are weak as possible.

3 Intuitions about Masking

The basic notion of masking is that when taken on their own, the weakest parts of axioms in a justification may combine together with other parts of axioms within the justification or external to the justification to reveal further reasons that are not directly represented by the set of regular justifications, and do not directly have a one-to-one "correspondence" with the set of regular justifications.

We define four important types of masking: *Internal Masking*, *Cross Masking*, *External Masking* and *Shared Cores*. The intuitions behind these types of masking are explained below.

Internal Masking Internal masking refers to the phenomena where there are multiple reasons within a single justification as to why the entailment in question holds. An example of internal masking is shown below.

$$\mathcal{O} = \{A \sqsubseteq B \sqcap \neg B \sqcap C \sqcap \neg C\} \models A \sqsubseteq \bot$$

There is a single regular justification for $\mathcal{O} \models A \sqsubseteq \bot$, namely $\mathcal{O}$ itself. However, within this justification there are, intuitively, two reasons as to why $\mathcal{O} \models A \sqsubseteq \bot$, the first being $\{A \sqsubseteq B \sqcap \neg B\}$ and the second being $\{A \sqsubseteq C \sqcap \neg C\}$.

Cross Masking Intuitively, cross masking is present within a set of justifications for an entailment when parts of axioms from one justification combine with parts of axioms from another justification in the set to produce new reasons for the given entailment. For example, consider the following ontology.

$$\mathcal{O} = \{A \sqsubseteq B \sqcap \neg B \sqcap C$$
$$A \sqsubseteq D \sqcap \neg D \sqcap \neg C\} \models A \sqsubseteq \bot$$

There are two justifications for $\mathcal{O} \models A \sqsubseteq \bot$, namely $\mathcal{J}_1 = \{A \sqsubseteq B \sqcap \neg B \sqcap C\}$ and $\mathcal{J}_2 = \{A \sqsubseteq D \sqcap \neg D \sqcap \neg C\}$. However, part of the axiom in $\mathcal{J}_1$, namely $A \sqsubseteq C$ may combine with part of the axiom in $\mathcal{J}_2$, namely $A \sqsubseteq \neg C$ to produce a further reason: $\mathcal{J}_3 = \{A \sqsubseteq C, A \sqsubseteq \neg C\}$.

External Masking While internal masking and cross masking take place over a set of "regular" justifications for an entailment, external masking involves parts of axioms from a regular justification combining with parts of axioms from an ontology (intuitively the axioms outside of the set of regular justifications) to produce further reasons for the entailment in question. Consider the example below,

$$\mathcal{O} = \{A \sqsubseteq B \sqcap \neg B \sqcap C$$
$$A \sqsubseteq \neg C\} \models A \sqsubseteq \bot$$

There is just one justification for $\mathcal{O} \models A \sqsubseteq \bot$, however, although $A \sqsubseteq \neg C$ intuitively plays a part in the unsatisfiability of A it will never appear in a justification for $\mathcal{O} \models A \sqsubseteq \bot$. When $\mathcal{O}$ is taken into consideration, there are two salient reasons for $A \sqsubseteq \bot$, the first being $\{A \sqsubseteq B \sqcap \neg B\}$ and the second being $\{A \sqsubseteq C, A \sqsubseteq \neg C\}$

Shared Core Masking Finally, two justifications share a *core* if after stripping away the superfluous parts of axioms in each justification the justifications are essentially structurally equal. Consider the example below,

$$\mathcal{O} = \{A \sqsubseteq B \sqcap \neg B \sqcap C$$
$$A \sqsubseteq B \sqcap \neg B\} \models A \sqsubseteq \bot$$

There are two justifications for $\mathcal{O} \models \eta$, $\mathcal{J}_1 = \{A \sqsubseteq B \sqcap \neg B \sqcap C\}$ and $\mathcal{J}_2 = \{A \sqsubseteq B \sqcap \neg B\}$. However, $\mathcal{J}_1$ can be reduced to the laconic justification $\{A \sqsubseteq B \sqcap \neg B\}$ (since C is irrelevant for the entailment), which is structurally equal to $\mathcal{J}_2$. With regular justifications, it appears that there are more reasons for the entailment, when in fact each justification boils down to the same reason.

3.1   Masking Due to Weakening

The above intuitions have been illustrated using simple propositional examples. However, it is important to realise that masking is not just concerned with boolean parts of axioms. *Weakest parts* of axioms must also be taken into consideration. For example, consider

$$\mathcal{O} = \{A \sqsubseteq\ \geq 2R.C$$
$$A \sqsubseteq\ \geq 1R.D$$
$$C \sqsubseteq \neg D\} \models A \sqsubseteq\ \geq 2R$$

There is one regular justification for $\mathcal{O} \models A \sqsubseteq\ \geq 2R$ namely, $\mathcal{J}_1 = \{A \sqsubseteq\ \geq 2R.C\}$. However, there are intuitively two reasons for this entailment. The first is described by the justification obtained as a weakening of $\mathcal{J}_1$, and is $\mathcal{J}_2 = \{A \sqsubseteq\ \geq 2.R\}$. The second is obtained by weakening the first axiom in $\mathcal{O}$ and combining it with the second and third axioms in $\mathcal{O}$ to give $\{A \sqsubseteq\ \geq 1R.C, A \sqsubseteq\ \geq 1R.D, C \sqsubseteq \neg D\}$.

Of course, masking due to weakening can occur in internal masking, cross masking, external masking and shared cores.

3.2 Summary on Intuitions

As can be seen from the above examples, the *basic idea* is that when the weakest parts of axioms in a justification, set of justifications or an ontology are taken into consideration, there can be multiple reasons for an entailment that are otherwise not exposed with regular justifications. These reasons take the form of laconic justifications—justifications whose axioms do not contain any superfluous parts and whose parts are as weak as possible. With internal masking, cross masking and external masking, there are more laconic justifications (by some measure) than there are regular justifications. With shared cores there are fewer laconic justifications (by some measure) than there are regular justifications.

3.3 Detecting Masking

Given the above link between masking, weakest parts of axioms and laconic justifications, it may seem fruitful to use laconic justifications as a mechanism for detecting masking. Specifically, it may seem like a good idea to count laconic justifications for the entailment in question. However, this is a flawed intuition and several problems prevent laconic justification counting being used *directly* as a masking detection mechanism. We begin by noting that there may be an *infinite* number of laconic justifications for an entailment.

Lemma 1 (Number of Laconic Justifications). *Let $\mathcal{S}$ be a set of $\mathcal{SROIQ}$ axioms such that $\mathcal{S} \models \eta$. In general, there may be an infinite number of laconic justifications over $\mathcal{S}$ for $\mathcal{S} \models \eta$.*

Proof: Consider an ontology $\mathcal{O}$ such that $\mathcal{O} \models A \sqsubseteq \bot$. Since laconic justifications may be drawn from the deductive closure of an ontology it is possible to construct an infinite set of justifications for the unsatisfiability of A of the form $\{A \sqsubseteq\ \geq nR.\top, A \sqsubseteq\ \leq (n-1)R.\top\}$.

The Promiscuity of the Deductive Closure The first problem is that, in general, there can be an infinite number of laconic justifications for a given entailment (Lemma 1). The notion of counting the number of laconic justifications over a set of axioms and comparing this to the number of regular justifications over the same set of axioms is therefore useless when it comes to detecting and defining masking. Even if the logic used did not result in an infinite number of laconic justifications, the effects of splitting and syntactic equivalence could result in miscounting. For example, consider $\mathcal{J}_1 = \{A \sqsubseteq B \sqcap C, B \sqcap C \sqsubseteq D\}$, where $\mathcal{J}_1$ is in itself laconic, however another justification $\mathcal{J}_2 = \{A \sqsubseteq B, A \sqsubseteq C, B \sqcap C \sqsubseteq D\}$ can be obtained, which is also laconic. Clearly, masking is not present in $\mathcal{J}_1$, but there are more laconic justifications than there are regular justifications.

Preferred Laconic Justifications Another approach might be to count the number of *preferred laconic justifications*, which are laconic justifications that

are made up of axioms which come from a filter on the deductive closure of a set of axioms. The notion of preferred laconic justifications was introduced in [3], where a filter called $\mathcal{O}^+$ is used to compute justifications that bear a syntactic resemblance to the axioms from which they are derived. Unfortunately, this idea is sensitive to the definition of the filter. Different filters, for different applications, may give different answers and false positives. While a particular filter could be verified to behave correctly and perhaps be used as an optimisation for detecting masking in an implementation, this mechanism is not appropriate for defining masking.

Preservation of Positional Information Another problem is that structural information can be lost with laconic justifications. Consider the $\{A \sqsubseteq B \sqcap (C \sqcap B)\}$ as a justification for $A \sqsubseteq B$. Masking is clearly present within this justification. If $B_{@1}$ denotes the first occurrence of B, and $B_{@2}$ denotes the second occurrence of B then A is a subclass of B because of two reasons: $A \sqsubseteq B_{@1}$ and $A \sqsubseteq B_{@2}$. However, this positional information is lost in all laconic justifications for $A \sqsubseteq B$. In essence, syntax is crucial when it comes to masking.

Splitting is Not Enough While syntax is very important when considering masking, it does not suffice to consider syntax alone. The example of masking due to weakening shows that simply splitting a set of axioms $\mathcal{S}$ into their constituent parts, using the structural transformation $\delta(\mathcal{S})$, and then examining the justifications for the entailment with respect $\delta(\mathcal{S})$ is not enough to capture this notion of masking. Weakenings of the split axioms must be considered in any mechanism that is used to detect masking.

4   Masking Defined

With the above intuitions and desiderata in mind the notion of masking can be made more concrete. The basic idea is to pull apart the axioms in a justification, set of justifications and an ontology, compute constrained weakenings of these parts (inline with the definition of laconic justifications), and then to check for the presence and number of laconic justifications within the set of regular justifications for an entailment with respect to these parts and their weakenings.

4.1   Parts and Their Weakenings

We first define a function $\delta^+(\mathcal{S})$, which maps a set of axioms $\mathcal{S}$ to a set of axioms composed from the union of $\delta(\mathcal{S})$ with the constrained weakenings of axioms in $\delta(\mathcal{S})$. The weakenings of axioms is constrained in that for an axiom $\alpha \in \delta(\mathcal{S})$, a weakening α' of α is contained in $\delta^+(\mathcal{S})$ only if α' is no longer than α—i.e. the weakening does not introduce any extra parts.

Definition 3 (δ^+). *For a set of $\mathcal{SROIQ}$ axioms, $\mathcal{S}$,*

$$\delta^+(\mathcal{S}) := \delta(\mathcal{S}) \cup \{\alpha' \mid \exists \alpha \in \delta(\mathcal{S}) \ s.t. \ \alpha \models \alpha' \ and \ \alpha' \not\models \alpha \ and \ |\delta(\alpha')| = 1)\}$$

Lemma 2 (δ^+justificatory finiteness). *For a finite set of axioms $\mathcal{S}$, the set of justifications for an entailment in $\delta^+(\mathcal{S})$ is finite.*

Proof: $\delta^+(S)$ is composed of the set of axioms in $\delta(\mathcal{S})$, which is finite, plus a possibly infinite set of axioms taken from the deductive closure of *each axiom* in $\delta(\mathcal{S})$. For a $\mathcal{SROIQ}$ axiom α, every axiom α' in $\delta(\alpha)$ must either be one of the following forms:

$$\top \sqsubseteq X_i \sqcup X_j$$
$$X_i \sqsubseteq A$$
$$X_i \sqsubseteq \neg A$$
$$X_i \sqsubseteq \exists R.X_j$$
$$X_i \sqsubseteq \forall R.X_j$$
$$X_i \sqsubseteq \{o\}$$
$$X_i \sqsubseteq \exists R.\mathsf{Self}$$
$$X_i \sqsubseteq\, \geq nR.X_j$$

in which case the set of axioms in $\delta^+(\alpha)$ is finite since the set of weakenings (in accordance with the definition of δ^+) of α' is finite. Or, α' is of the form:

$$X_i \sqsubseteq\, \leq nR.X_j$$

in which case there is an infinite number of weakenings of α' in $\delta^+(\alpha)$ since $A \sqsubseteq\, \leq (n+1)R.C$ is weaker than $A \sqsubseteq\, \leq nR.C$ for any $n \geq 0$. If justifications are made up solely of the axioms of the form corresponding to the first set then the set of justifications is clearly finite. If justifications contain axioms of the second form $X_i \sqsubseteq\, \leq nR.X_j$ then there is a finite upper bound m for n, where there are no justifications containing an axiom of the from $X_i \sqsubseteq\, \leq kR.X_j$ for some $k > m$. This is because, for values of k, where k is equal to the maximum number in $\leq$ restrictions in the closure of $\mathcal{S}$, or more, $X_i \sqsubseteq\, \leq kR.X_j$ is too weak to participate in a justification, and this follows as a straight forward consequence of $\mathcal{SROIQ}$'s model theory [4]. $\square$

Next, a function which filters out laconic justifications for an entailment from a set of justifications for the entailment is defined:

Definition 4 (Laconic Filtering). *For a set of axioms $\mathcal{S} \models \eta$, $laconic(\mathcal{S}, \eta)$ is the set of justifications for $\mathcal{S} \models \eta$ that are laconic over $\mathcal{S}$.*

Notice that because of Lemma 2, the set of justifications $laconic(\mathcal{S}, \eta)$ is finite.

4.2   Masking Definitions

With the definition of δ^+ and the definition of laconic filtering in hand, the various types of masking can now be defined.

Definition 5 (Internal Masking). *For a justification $\mathcal{J}$ for $\mathcal{O} \models \eta$, internal masking is present within $\mathcal{J}$ if*

$$\left| laconic(\delta^+(\mathcal{J}), \eta) \right| > 1$$

Lemma 3. *Internal masking is not present within a laconic justification.*

Proof: Assume that $\mathcal{J}$ is a laconic justification for η and that internal masking is present within $\mathcal{J}$. This means that there either must be (i) at least two laconic justifications for $\delta^+(\mathcal{J}) \models \eta$, i.e. there exists some $\mathcal{J}_1, \mathcal{J}_2 \subsetneq \delta^+(\mathcal{J})$ where $\mathcal{J}_1 \neq \mathcal{J}_2$ and are both laconic. However, since $\mathcal{J}$ itself is laconic this violates condition 2 of Definition 2, or (ii) there is a non-length increasing weakening of one or more axioms in $\delta(\mathcal{J})$ that yields $\delta(\mathcal{J})'$. However since $\mathcal{J}$ is laconic this violates conditions 3a and 3b of Definition 2.   □

Let $\mathcal{O} \models \eta$ and $\mathcal{J}_1, \ldots, \mathcal{J}_n$ be the set of all justifications for $\mathcal{O} \models \eta$. Cross masking and External masking are then defined as follows:

Definition 6 (Cross Masking). *For two justifications $\mathcal{J}_i$ and $\mathcal{J}_j$, cross masking is present within $\mathcal{J}_i$ and $\mathcal{J}_j$ if*

$$\left| laconic\big(\delta^+(\mathcal{J}_i \cup \mathcal{J}_j), \eta\big) \right| > \big(\left| laconic\big(\delta^+(\mathcal{J}_i), \eta\big) \right| + \left| laconic\big(\delta^+(\mathcal{J}_j), \eta\big) \right| \big)$$

Definition 7 (External Masking). *External masking is present if*

$$\left| laconic(\delta^+(\mathcal{O}), \eta) \right| > \left| laconic(\delta^+(\bigcup_{i=1}^{i=n} \mathcal{J}_i), \eta) \right|$$

Definition 8 (Shared Cores). *Two justifications $\mathcal{J}_i$ and $\mathcal{J}_j$ for $\mathcal{O} \models \eta$, share a core if there is a justification $\mathcal{J}_i' \in laconic(\delta^+(\mathcal{J}_i), \eta)$ and a justification $\mathcal{J}_j' \in laconic(\delta^+(\mathcal{J}_j), \eta)$ and a renaming ρ of terms not in $\mathcal{O}$ such that $\rho(\mathcal{J}_i') = \mathcal{J}_j'$.*

5   Examples

The issue of masking is indeed a real world problem with realistic ontologies. For example, external masking is present in the DOLCE ontology. The entailment quale ⊑ region has a single justification:

$$\{\textsf{quale} \equiv \textsf{region} \sqcap \exists\, \textsf{atomicPartOf.region}\}$$

However, there are further justifications that are externally masked by this regular justification. There are three laconic justifications, the first being

$$\{\textsf{quale} \sqsubseteq \textsf{region}\}$$

which is directly obtained as a weaker form of the regular justification. More interestingly, there are two additional laconic justifications:

$$\{\textsf{quale} \sqsubseteq \exists\textsf{atomicPartOf.region}$$
$$\textsf{atomicPartOf} \sqsubseteq \textsf{partOf}$$
$$\textsf{partOf} \sqsubseteq \textsf{part}^-$$
$$\textsf{region} \sqsubseteq \forall\textsf{part.region}\}$$

and also

$$\{\mathsf{quale} \sqsubseteq \mathsf{atomicPartOf.region}$$
$$\mathsf{atomicPartOf} \sqsubseteq \mathsf{atomicPart}^-$$
$$\mathsf{atomicPart} \sqsubseteq \mathsf{part}$$
$$\mathsf{region} \sqsubseteq \forall \mathsf{part.region}\}$$

Both of these justifications represent reasons for the entailment which are never seen with regular justifications due to the presence of external masking.

A real ontology about pathway interactions[1] contains an unsatisfiable class called "Phosphate Acceptor". There are 32 regular justifications for this class being unsatisfiable. However, upon examination, these 32 justifications share a single core. When trying to understand the reason for the unsatisfiable class, the succinctness of the core provides a dramatic improvement in terms of usability.

6   Implementation Issues

The main focus of this paper has been to pin down the notions and types of masking. At this stage no attention has been paid to the practicalities of detection masking. However, the definitions for the various types of masking make use of the well known structural transformation δ—δ^+ must be computed from δ. Naturally, this raises the question of performance and scalability, since many reasoners rely on the structure of axioms in real world ontologies for several key optimisations. Normalising the axioms in an ontology using the structural transformation, i.e. converting axioms to clausal form, raises the possibility of negating these optimisations. While more investigation work needs to be done, some preliminary experiments indicate that it is feasible to detect internal masking and cross masking. It is expected that an algorithm that transforms an ontology in an incremental manner, using techniques similar to those presented in [3] for computing laconic justifications, could provide a practical mechanism for detecting external masking.

7   Related Work

Various groups [6, 7, 11] have concentrated their efforts on what can be thought of as fine-grained justifications. In particular, Kalyanpur et al. [6, 5] presented work on fine-grained justifications, where axioms were split into smaller axioms in order to obtain a more "precise" justification. This work discusses the reasons for fine-grained justifications, one of which corresponds to the notion of external masking presented here. However, no precise definitions of masking were given in this work.

---

[1] http://owl.cs.manchester.ac.uk/repository/download?ontology=http://purl.org/NET/biopax-obo/examples/reaction.owl (courtesy Alan Ruttenberg)

8   Conclusions

This paper has presented a discussion on the phenomenon of justification masking. The notion and types of masking have been discussed and defined. These definitions basically identify the parts of axioms in a justification, over a set of justifications and an ontology, weaken the parts and then look for the number of laconic justifications that are present in the set of justifications over the axioms that represent these weakened parts.

References

1. Franz Baader and Bernhard Hollunder. Embedding defaults into terminological knowledge representation formalisms. *Journal of Automated Reasoning*, 14(1):149–180, 1995.
2. Franz Baader, Rafael Peñaloza, and Boontawee Suntisrivaraporn. Pinpointing in the description logic $\mathcal{EL}$. In Joachim Hertzberg, Michael Beetz, and Roman Englert, editors, *KI 2007: Advances in Artificial Intelligence, 30th Annual German Conference on AI, KI 2007, Osnabrück, Germany*, volume 4667 of *Lecture Notes in Computer Science*, pages 52–67. Springer, September 2007.
3. Matthew Horridge, Bijan Parsia, and Ulrike Sattler. Laconic and precise justifications in OWL. In *ISWC 08 The International Semantic Web Conference 2008, Karlsruhe, Germany*, 2008.
4. Ian Horrocks, Oliver Kutz, and Ulrike Sattler. The even more irresistible $\mathcal{SROIQ}$. In Patrick Doherty, John Mylopoulos, and Christopher A. Welty, editors, *The 10th International Conference on Principles of Knowledge Representation and Reasoning (KR 2006), Lake District, United Kingdom*, pages 57–67. AAAI Press, June 2006.
5. Aditya Kalyanpur. *Debugging and Repair of OWL Ontologies*. PhD thesis, The Graduate School of the University of Maryland, 2006.
6. Aditya Kalyanpur, Bijan Parsia, and Bernardo Cuenca Grau. Beyond asserted axioms: Fine-grain justifications for OWL-DL entailments. In Bijan Parsia, Ulrike Sattler, and David Toman, editors, *DL 2006, Lake District, U.K.*, volume 189 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2006.
7. Joey Sik Chun Lam, Derek H. Sleeman, Jeff Z. Pan, and Wamberto Weber Vasconcelos. A fine-grained approach to resolving unsatisfiable ontologies. *Journal of Data Semantics*, 10:62–95, 2008.
8. Boris Motik, Peter F. Patel-Schneider, and Bijan Parsia. OWL 2 Web Ontology Language structural specification and functional style syntax. W3C Recommendation, W3C – World Wide Web Consortium, October 2009.
9. Boris Motik, Rob Shearer, and Ian Horrocks. Optimized reasoning in description logics using hypertableaux. In *Proc. of the 21st Int. Conf. on Automated Deduction (CADE-21)*, volume 4603 of *Lecture Notes in Artificial Intelligence*, pages 67–83. Springer, 2007.
10. David A. Plaisted and Steven Greenbaum. A structure-preserving clause form translation. *Journal of Symbolic Computation*, 1986.
11. Stefan Schlobach and Ronald Cornet. Non-standard reasoning services for the debugging of description logic terminologies. In *IJCAI International Joint Conference on Artificial Intelligence*, 2003.

$\mathcal{EL}$-Concepts go Second-Order: Greatest Fixpoints and Simulation Quantifiers

Carsten Lutz[1], Robert Piro[2], and Frank Wolter[2]

[1] Department of Computer Science, University of Bremen, Germany
[2] Department of Computer Science, University of Liverpool, UK
clu@uni-bremen.de,{Robert.Piro,Wolter}@liverpool.ac.uk

1 Introduction

The well-known description logic (DL) $\mathcal{ALC}$ is usually regarded as the *basic DL* that comprises all Boolean concept constructors and from which all expressive DLs are derived by admitting additional concept constructors. The fundamental role of $\mathcal{ALC}$ is largely due to the fact that it is very well-behaved regarding its logical, model-theoretic, and computational properties. This good behavior can, in turn, be explained nicely by the fact that $\mathcal{ALC}$-concepts can be characterized exactly as the bisimulation invariant fragment of first-order logic (FO) in the sense that an FO formula is invariant under bisimulation if, and only if, it is equivalent to an $\mathcal{ALC}$-concept [22, 13, 16]. In particular, invariance under bisimulation explains the tree-model property of $\mathcal{ALC}$ as well as its favorable computational properties [24]. In the mentioned characterization, the condition that $\mathcal{ALC}$ is a fragment of FO is much less important than its bisimulation invariance. In fact, $\mathcal{ALC}\mu$, the extension of $\mathcal{ALC}$ with fixpoint operators, is not a fragment of FO, but inherits almost all important properties of $\mathcal{ALC}$ [8, 12]. Similar to $\mathcal{ALC}$, $\mathcal{ALC}\mu$'s fundamental role (in particular in its formulation as the modal mu-calculus) can be explained by the fact that $\mathcal{ALC}\mu$-concepts can be characterized exactly as the bisimulation invariant fragment of monadic second-order logic (MSO) [14, 8]. Indeed, from a purely theoretical viewpoint it is hard to explain why $\mathcal{ALC}$ rather than $\mathcal{ALC}\mu$ forms the logical underpinning of current ontology language standards; the facts that mu-calculus concepts can be hard to grasp and that, despite the same theoretical complexity, efficient reasoning in $\mathcal{ALC}\mu$ is more challenging than in $\mathcal{ALC}$ are probably the only reasons for the limited interest in $\mathcal{ALC}\mu$ compared to $\mathcal{ALC}$.

In recent years, the development of very large ontologies and the use of ontologies to access instance data has led to a revival of interest in *tractable* DLs. The main examples are $\mathcal{EL}$ [5] and DL-Lite [9], the logical underpinnings of the OWL profiles OWL2 EL and OWL2 QL, respectively. In contrast to $\mathcal{ALC}$, a satisfactory characterization of the expressivity of such DLs is still missing, and a first aim of this paper is to fill this gap for $\mathcal{EL}$. To this end, we characterize $\mathcal{EL}$ as a maximal fragment of FO that is preserved under *simulations* and has *finite minimal models*. Note that preservation under simulations alone would characterize $\mathcal{EL}$ with disjunctions, and the existence of minimal models reflects the "Horn-aspect" of $\mathcal{EL}$.

The second and main aim of this paper, however, is to introduce and investigate two equi-expressive extensions of $\mathcal{EL}$ with greatest fixpoints, $\mathcal{EL}^\nu$ and $\mathcal{EL}^{\nu+}$, and to

prove that they stand in a similar relationship to $\mathcal{EL}$ as $\mathcal{ALC}\mu$ to $\mathcal{ALC}$. To this end, we prove that $\mathcal{EL}^\nu$ (and therefore also $\mathcal{EL}^{\nu+}$, which admits mutual fixpoints and is exponentially more succinct than $\mathcal{EL}^\nu$) can be characterized as a maximal fragment of MSO that is preserved under *simulations* and has *finite minimal models*. Similar to $\mathcal{ALC}\mu$, $\mathcal{EL}^\nu$ and $\mathcal{EL}^{\nu+}$ inherit many good properties of $\mathcal{EL}$, the most interesting being that *reasoning with general concept inclusions (GCIs) is still tractable* and that the same type of algorithm can be used. Thus, in contrast to $\mathcal{ALC}\mu$, the development of practical decision procedures is no obstacle to using $\mathcal{EL}^\nu$.

Moreover, $\mathcal{EL}^{\nu+}$ has a number of very useful properties that $\mathcal{EL}$ and most of its extensions are lacking. To begin with, we show that in $\mathcal{EL}^{\nu+}$ *least common subsumers (LCS)* w.r.t. general TBoxes always exist and can be computed in polynomial time (for a bounded number of concepts). This result can be regarded as an extension of similar results for least common subsumers w.r.t. *classical* TBoxes in $\mathcal{EL}$ with greatest fixpoint semantics in [1]. Similarly, in $\mathcal{EL}^{\nu+}$ *most specific concepts* always exist and can be computed in linear time; a result which also generalizes [1]. Secondly, we show that $\mathcal{EL}^{\nu+}$ has the *Beth definability property* with explicit definitions being computable in polytime and of polynomial size. It has been convincingly argued in [21, 20] that this property is of great interest for structuring TBoxes and for ontology based data access. Another application of $\mathcal{EL}^{\nu+}$ is demonstrated in [15], where the succinct representations of definitions in $\mathcal{EL}^{\nu+}$ are used to develop polytime algorithms for decomposing certain general $\mathcal{EL}$-TBoxes.

To prove these result and provide a better understanding of the modeling capabilities of $\mathcal{EL}^{\nu+}$ we show that it has the same expressive power as extensions of $\mathcal{EL}$ by means of *simulation quantifiers*, a variant of second-order quantifiers that quantifies "modulo a simulation of the model"; in fact, the relationship between simulation quantifiers and $\mathcal{EL}^{\nu+}$ is somewhat similar to the relationship between $\mathcal{ALC}\mu$ and bisimulation quantifiers [11]. Proofs are omitted for brevity and the reader is referred to www.csc.liv.ac.uk/~frank/publ/publ.html.

2   Preliminaries

Let $\mathsf{N_C}$ and $\mathsf{N_R}$ be countably infinite and mutually disjoint sets of concept and role names. $\mathcal{EL}$-*concepts* are built according to the rule

$$C := \quad A \quad | \quad \top \quad | \quad \bot \quad | \quad C \sqcap D \quad | \quad \exists r.C,$$

where $A \in \mathsf{N_C}$, $r \in \mathsf{N_R}$, and C, D range over $\mathcal{EL}$-concepts[3]. An $\mathcal{EL}$-*concept inclusion* takes the form $C \sqsubseteq D$, where C, D are $\mathcal{EL}$-concepts. A *general $\mathcal{EL}$-TBox* $\mathcal{T}$ is a finite set of $\mathcal{EL}$-concept inclusions. An *ABox assertion* is an expression of the form $A(a)$ or $r(a, b)$, where a, b are from a countably infinite set of individual names $\mathsf{N_I}$, $A \in \mathsf{N_C}$, and $r \in \mathsf{N_R}$. An *ABox* is a finite set of ABox assertions. By $\mathsf{Ind}(\mathcal{A})$ we denote the set of individual names in $\mathcal{A}$. An $\mathcal{EL}$-*knowledge base* (KB) is a pair $(\mathcal{T}, \mathcal{A})$ that consists of an $\mathcal{EL}$-TBox $\mathcal{T}$ and an ABox $\mathcal{A}$.

---

[3] In the literature, $\mathcal{EL}$ is typically defined without $\bot$. The sole purpose of including $\bot$ here is to simplify the formulation of some results.

The semantics of $\mathcal{EL}$ is based on interpretations $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$, where the *domain* $\Delta^{\mathcal{I}}$ is a non-empty set, and $\cdot^{\mathcal{I}}$ is a function mapping each concept name A to a subset $A^{\mathcal{I}}$ of $\Delta^{\mathcal{I}}$, each role name r to a binary relation $r^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$, and each individual name a to an element $a^{\mathcal{I}}$ of $\Delta^{\mathcal{I}}$. The interpretation $C^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$ of $\mathcal{EL}$-concepts C in an interpretation $\mathcal{I}$ is defined in the standard way [6]. We will often make use of the fact that $\mathcal{EL}$-concepts can be regarded as formulas in FO (and, therefore, MSO) with unary predicates from $\mathsf{N_C}$, binary predicates from $\mathsf{N_R}$, and exactly one free variable [6]. We will often not distinguish between $\mathcal{EL}$-concepts and their translations into FO/MSO.

We now introduce $\mathcal{EL}^{\nu}$, the extension of $\mathcal{EL}$ with greatest fixpoints and the main language studied in this paper. $\mathcal{EL}^{\nu}$-*concepts* are defined like $\mathcal{EL}$-concepts, but additionally allow the greatest fixpoint constructor $\nu X.C$, where X is from a countably infinite set of *(concept) variables* $\mathsf{N_V}$ and C an $\mathcal{EL}^{\nu}$-concept. A variable is *free* in a concept C if it occurs in C at least once outside the scope of any ν-constructor that binds it. An $\mathcal{EL}^{\nu}$-concept is *closed* if it does not contain any free variables. An $\mathcal{EL}^{\nu}$-*concept inclusion* takes the form $C \sqsubseteq D$, where C, D are are closed $\mathcal{EL}^{\nu}$-concepts. The semantics of the greatest fixpoint constructor is as follows, where $\mathcal{V}$ is an *assignment* that maps variables to subsets of $\Delta^{\mathcal{I}}$ and $\mathcal{V}[X \mapsto W]$ denotes $\mathcal{V}$ modified by setting $\mathcal{V}(X) = W$:

$$(\nu X.C)^{\mathcal{I},\mathcal{V}} = \bigcup \{W \subseteq \Delta^{\mathcal{I}} \mid W \subseteq C^{\mathcal{I},\mathcal{V}[X \mapsto W]}\}$$

We will also consider an extended version of the ν-constructor that allows to capture mutual recursion. It has been considered e.g. in [10, 23] and used in a DL context in [19]; it can be seen as a variation of the fixpoint equations considered in [8]. The constructor has the form $\nu_i X_1 \cdots X_n.C_1, \ldots, C_n$ where $1 \leq i \leq n$. The semantics is defined by setting $(\nu_i X_1 \cdots X_n.C_1, \ldots, C_n)^{\mathcal{I},\mathcal{V}}$ to

$$\bigcup \{W_i \mid \exists W_1, \ldots, W_{i-1}, W_{i+1}, \ldots, W_n \text{ s.t. for } 1 \leq j \leq n: \\ W_j \subseteq C_j^{\mathcal{I},\mathcal{V}[X_1 \mapsto W_1, \ldots, X_n \mapsto W_n]}\}$$

We use $\mathcal{EL}^{\nu+}$ to denote $\mathcal{EL}$ extended with this mutual greatest fixpoint constructor. Clearly, $\nu X.C \equiv \nu_1 X.C$, thus every $\mathcal{EL}^{\nu}$-concept is equivalent to an $\mathcal{EL}^{\nu+}$-concept. Conversely, we have the following result, where the first part follows from [8]. The length of a concept C is defined as the number of occurrences of symbols in it.

Proposition 1. *For every $\mathcal{EL}^{\nu+}$-concept, one can construct an equivalent $\mathcal{EL}^{\nu}$-concept of at most exponential size. Moreover, there is a sequence of $\mathcal{EL}^{\nu+}$-concepts $C_0, C_1, \ldots$ such that C_i is of length $p(i)$, p a polynomial, whereas the shortest $\mathcal{EL}^{\nu}$-concept equivalent to C_i is of length at least 2^i.*

By extending the translation of $\mathcal{EL}$-concepts into FO in the obvious way, one can translate closed $\mathcal{EL}^{\nu+}$-concepts into an MSO formula with one free first-order variable. We will often not distinguish between $\mathcal{EL}^{\nu+}$-concepts and their translation into MSO.

3 Characterizing $\mathcal{EL}$ using simulations

The purpose of this section is to provide a model-theoretic characterization of $\mathcal{EL}$ as a fragment of FO that is similar in spirit to the well-known characterization of $\mathcal{ALC}$

as the bisimulation-invariant fragment of FO. To this end, we first characterize $\mathcal{EL}^{\sqcup}$, the extension of $\mathcal{EL}$ with the disjunction constructor $\sqcup$, as the fragment of FO that is preserved under simulation. Then we characterize the fragment $\mathcal{EL}$ of $\mathcal{EL}^{\sqcup}$ using, in addition, the existence of minimal models. A *pointed interpretation* is a pair $(\mathcal{I}, d)$ consisting of an interpretation $\mathcal{I}$ and $d \in \Delta^{\mathcal{I}}$. A *signature* Σ is a set of concept and role names.

Definition 1 (Simulations). Let $(\mathcal{I}_1, d_1)$ and $(\mathcal{I}_2, d_2)$ be pointed interpretations and Σ a signature. A relation $S \subseteq \Delta^{\mathcal{I}_1} \times \Delta^{\mathcal{I}_2}$ is a Σ-*simulation* between $(\mathcal{I}_1, d_1)$ and $(\mathcal{I}_2, d_2)$, in symbols $S : (\mathcal{I}_1, d_1) \leq_{\Sigma} (\mathcal{I}_2, d_2)$, if $(d_1, d_2) \in S$ and the following conditions hold:

1. for all concept names $A \in \Sigma$ and all $(e_1, e_2) \in S$, if $e_1 \in A^{\mathcal{I}_1}$ then $e_2 \in A^{\mathcal{I}_2}$;
2. for all role names $r \in \Sigma$, all $(e_1, e_2) \in S$, and all $e_1' \in \Delta^{\mathcal{I}_1}$ with $(e_1, e_1') \in r^{\mathcal{I}_1}$, there exists $e_2' \in \Delta^{\mathcal{I}_2}$ such that $(e_2, e_2') \in r^{\mathcal{I}_2}$ and $(e_1', e_2') \in S$.

If such an S exists, then we also say that $(\mathcal{I}_2, d_2)$ Σ-*simulates* $(\mathcal{I}_1, d_1)$ and write $(\mathcal{I}_1, d_1) \leq_{\Sigma} (\mathcal{I}_2, d_2)$.

If $\Sigma = \mathsf{N_C} \cup \mathsf{N_R}$, then we omit Σ and use the term *simulation* to denote Σ-simulations and $(\mathcal{I}_1, d_1) \leq (\mathcal{I}_2, d_2)$ stands for $(\mathcal{I}_1, d_1) \leq_{\Sigma} (\mathcal{I}_2, d_2)$. It is well-known that the description logic $\mathcal{EL}$ is intimately related to the notion of a simulation, see for example [4, 17]. In particular, $\mathcal{EL}$-concepts are preserved under simulations in the sense that if $d \in C^{\mathcal{I}}$ for an $\mathcal{EL}$-concept C and $(\mathcal{I}_1, d_1) \leq_{\Sigma} (\mathcal{I}_2, d_2)$, then $d_2 \in C^{\mathcal{I}_2}$. This observation, which clearly generalizes to $\mathcal{EL}^{\sqcup}$, illustrates the (limitations of the) modeling capabilities of $\mathcal{EL}/\mathcal{EL}^{\sqcup}$. We now strengthen it to an exact characterization of the expressive power of these logics relative to FO.

Let $\varphi(x)$ be an FO-formula (or, later, MSO-formula) with one free variable x. We say that $\varphi(x)$ is *preserved under simulations* if, and only if, for all $(\mathcal{I}_1, d_1)$ and $(\mathcal{I}_2, d_2)$, $\mathcal{I}_1 \models \varphi[d_1]$ and $(\mathcal{I}_1, d_1) \leq (\mathcal{I}_2, d_2)$ implies $\mathcal{I}_2 \models \varphi[d_2]$.

Theorem 1. *An FO-formula $\varphi(x)$ is preserved under simulations if, and only if, it is equivalent to an $\mathcal{EL}^{\sqcup}$-concept.*

To characterize $\mathcal{EL}$, we add a central property of Horn-logics on top of preservation under simulations. Let $\mathcal{L}$ be a set of FO (or, later, MSO) formulas, each with one free variable. We say that $\mathcal{L}$ *has (finite) minimal models* if, and only if, for every $\varphi(x) \in \mathcal{L}$ there exists a (finite) pointed interpretation $(\mathcal{I}, d)$ such that for all $\psi(x) \in \mathcal{L}$, we have $\mathcal{I} \models \psi[d]$ if, and only if, $\forall x.(\varphi(x) \rightarrow \psi(x))$ is a tautology.

Theorem 2. *The set of $\mathcal{EL}$-concepts is a maximal set of FO-formulas that is preserved under simulations and has minimal models (equivalently: has finite minimal models): if $\mathcal{L}$ is a set of FO-formulas that properly contains all $\mathcal{EL}$-concepts, then either it contains a formula not preserved under simulations or it does not have (finite) minimal models.*

We note that de Rijke and Kurtonina have given similar characterizations of various non-Boolean fragments of $\mathcal{ALC}$. In particular, Theorem 1 is rather closely related to results proved in [16] and would certainly have been included in the extensive list of characterizations given there had $\mathcal{EL}$ already been as popular as it is today. In contrast, the novelty of Theorem 2 is that it makes the Horn character of $\mathcal{EL}$ explicit through minimal models while the characterizations of disjunction-free languages in [16] are based on simulations that take sets (rather than domain-elements) as arguments.

4 Simulation quantifiers and $\mathcal{EL}^\nu$

To understand and characterize the expressive power and modeling capabilities of $\mathcal{EL}^\nu$, we introduce three distinct types of simulation quantifiers and show that, in each case, the resulting language has the same expressive power as $\mathcal{EL}^\nu$.

Simulating interpretations. The first language $\mathcal{EL}^{si}$ extends $\mathcal{EL}$ by the concept constructor $\exists^{sim}(\mathcal{I}, d)$, where $(\mathcal{I}, d)$ is a finite pointed interpretation in which only finitely many $\sigma \in \mathsf{N_C} \cup \mathsf{N_R}$ have a non-empty interpretation $\sigma^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$. The semantics of $\exists^{sim}(\mathcal{I}, d)$ is defined by setting for all interpretations $\mathcal{J}$ and $e \in \Delta^{\mathcal{J}}$,

$$e \in (\exists^{sim}(\mathcal{I}, d))^{\mathcal{J}} \text{ iff } (\mathcal{I}, d) \leq (\mathcal{J}, e).$$

Example 1. Let $\mathcal{I}$ consist of one point d such that $(d, d) \in r^{\mathcal{I}}$. Then $e \in (\exists^{sim}(\mathcal{I}, d))^{\mathcal{J}}$ iff there is an infinite r-chain starting at e in $\mathcal{I}$, i.e., there exist $e_0, e_1, e_2, \ldots$ such that $e = e_0$ and $(e_i, e_{i+1}) \in r^{\mathcal{J}}$ for all $i \geq 0$.

To attain a better understanding of the constructor $\exists^{sim}$, it is interesting to observe that *every $\mathcal{EL}^{si}$-concept is equivalent to a concept of the form* $\exists^{sim}(\mathcal{I}, d)$.

Lemma 1. *For every $\mathcal{EL}^{si}$-concept C one can construct, in linear time, an equivalent concept of the form* $\exists^{sim}(\mathcal{I}, d)$.

Proof. By induction on the construction of C. If $C = A$ for a concept name A, then let $\mathcal{I} = (\{d\}, \cdot^{\mathcal{I}})$, where $A^{\mathcal{I}} = \{d\}$ and $\sigma^{\mathcal{I}} = \emptyset$ for all symbols distinct from A. Clearly, A and $\exists^{sim}(\mathcal{I}, d)$ are equivalent. For $C_1 = \exists^{sim}(\mathcal{I}_1, d_1)$ and $C_2 = \exists^{sim}(\mathcal{I}_2, d_2)$ assume that $\Delta^{\mathcal{I}_1} \cap \Delta^{\mathcal{I}_2} = \{d_1\} = \{d_2\}$. Then $\exists^{sim}(\mathcal{I}_1 \cup \mathcal{I}_2, d_1)$ is equivalent to $C_1 \sqcap C_2$. For $C = \exists r. \exists^{sim}(\mathcal{I}, d)$ construct a new interpretation $\mathcal{I}'$ by adding a new node e to $\Delta^{\mathcal{I}}$ and setting $(e, d) \in r^{\mathcal{I}'}$. Then $\exists^{sim}(\mathcal{I}', e)$ and C are equivalent. $\square$

We will show that there are polynomial translations between $\mathcal{EL}^{si}$ and $\mathcal{EL}^{\nu+}$. When using $\mathcal{EL}^\nu$ in applications and to provide a translation from $\mathcal{EL}^{\nu+}$ to $\mathcal{EL}^{si}$, it is convenient to have available a "syntactic" simulation operator.

Simulating models of TBoxes. The second language $\mathcal{EL}^{st}$ extends $\mathcal{EL}$ by the concept constructor $\exists^{sim}\Sigma.(\mathcal{T}, C)$, where Σ is a finite signature, $\mathcal{T}$ a general TBox, and C a concept. To admit nestings of $\exists^{sim}$, the concepts of $\mathcal{EL}^{st}$ are defined by simultaneous induction; namely, $\mathcal{EL}^{st}$-concepts, concept inclusions, and general TBoxes are defined as follows:

- every $\mathcal{EL}$-concept, concept inclusion, and general TBox is an $\mathcal{EL}^{st}$-concept, concept inclusion, and general TBox, respectively;
- if $\mathcal{T}$ is a general $\mathcal{EL}^{st}$-TBox, C an $\mathcal{EL}^{st}$-concept, and Σ a finite signature, then $\exists^{sim}\Sigma.(\mathcal{T}, C)$ is an $\mathcal{EL}^{st}$-concept;
- if C, D are $\mathcal{EL}^{st}$-concepts, then $C \sqsubseteq D$ is a $\mathcal{EL}^{st}$-concept inclusion;
- a general $\mathcal{EL}^{st}$-TBox is a finite set of $\mathcal{EL}^{st}$-concept inclusions.

The semantics of $\exists^{sim}\Sigma.(\mathcal{T}, C)$ is as follows:

$d \in (\exists^{sim}\Sigma.(\mathcal{T}, C))^{\mathcal{I}}$ iff there exists $(\mathcal{J}, e)$ such that $\mathcal{J}$ is a model of $\mathcal{T}$, $e \in C^{\mathcal{J}}$ and $(\mathcal{J}, e) \leq_{\Gamma} (\mathcal{I}, d)$, where $\Gamma = (\mathsf{N_C} \cup \mathsf{N_R}) \setminus \Sigma$.

Example 2. Let $\mathcal{T} = \{A \sqsubseteq \exists r.A\}$ and $\Sigma = \{A\}$. Then $\exists^{sim} \Sigma.(\mathcal{T}, A)$ is equivalent to the concept $\exists^{sim}(\mathcal{I}, d)$ defined in Example 1.

We will later exploit the fact that $\exists^{sim} \Sigma.(\mathcal{T}, C)$ is equivalent to $\exists^{sim} \Sigma \cup \{A\}.(\mathcal{T}', A)$, where A is a fresh concept name and $\mathcal{T}' = \mathcal{T} \cup \{A \sqsubseteq C\}$. Another interesting (but subsequently unexploited) observation is that we can w.l.o.g. restrict Σ to singleton sets since

$$\exists^{sim}(\{\sigma\} \cup \Sigma).(\mathcal{T}, C) \equiv \exists^{sim}\{\sigma\}.(\emptyset, \exists^{sim} \Sigma.(\mathcal{T}, C))$$
$$\exists^{sim}\emptyset.(\mathcal{T}, C) \equiv \exists^{sim}\{B\}.(\mathcal{T}, C)$$

where B is a concept name that does not occur in $\mathcal{T}$ and C.

Simulating models of KBs. The third language $\mathcal{EL}^{sa}$ extends $\mathcal{EL}$ by the concept constructor $\exists^{sim} \Sigma.(\mathcal{T}, \mathcal{A}, a)$, where a is an individual name in the ABox $\mathcal{A}$, $\mathcal{T}$ is a TBox, and Σ a finite signature. More precisely, we define $\mathcal{EL}^{sa}$-concepts, concept inclusions, general TBoxes, and KBs, by simultaneous induction as follows:

- every $\mathcal{EL}$-concept, concept inclusion, general TBox, and KB is an $\mathcal{EL}^{sa}$-concept, concept inclusion, general TBox, and KB, respectively;
- if $(\mathcal{T}, \mathcal{A})$ is a general $\mathcal{EL}^{sa}$-KB, a an individual name in $\mathcal{A}$, and Σ a finite signature, then $\exists^{sim} \Sigma.(\mathcal{T}, \mathcal{A}, a)$ is an $\mathcal{EL}^{sa}$-concept;
- if C, D are $\mathcal{EL}^{sa}$-concepts, then $C \sqsubseteq D$ is an $\mathcal{EL}^{sa}$-concept inclusion;
- a general $\mathcal{EL}^{sa}$-TBox is a finite set of $\mathcal{EL}^{sa}$-concept inclusions;
- an $\mathcal{EL}^{sa}$-KB is a pair $(\mathcal{T}, \mathcal{A})$ consisting of a general $\mathcal{EL}^{sa}$-TBox and an ABox.

The semantics of $\exists^{sim} \Sigma.(\mathcal{T}, \mathcal{A}, a)$ is as follows:

$d \in (\exists^{sim} \Sigma.(\mathcal{T}, \mathcal{A}, a))^{\mathcal{I}}$ iff there exists a model $\mathcal{J}$ of $(\mathcal{T}, \mathcal{A})$ such that $(\mathcal{J}, a^{\mathcal{J}}) \leq_{\Gamma} (\mathcal{I}, d)$, where $\Gamma = (\mathsf{N_C} \cup \mathsf{N_R}) \setminus \Sigma$.

Example 3. Let $\mathcal{T} = \emptyset$, $\mathcal{A} = \{r(a, a)\}$, and $\Sigma = \emptyset$. Then $\exists^{sim} \Sigma.(\mathcal{T}, \mathcal{A}, a)$ is equivalent to the concept $\exists^{sim}(\mathcal{I}, d)$ defined in Example 1.

Let $\mathcal{L}_1, \mathcal{L}_2$ be sets of concepts. We say that $\mathcal{L}_2$ is *polynomially at least as expressive as* $\mathcal{L}_1$, in symbols $\mathcal{L}_1 \leq_p \mathcal{L}_2$, if for every $C_1 \in \mathcal{L}_1$ one can construct in polynomial time a $C_2 \in \mathcal{L}_2$ such that C_1 and C_2 are equivalent. We say that $\mathcal{L}_1, \mathcal{L}_2$ are *polynomially equivalent*, in symbols $\mathcal{L}_1 \equiv_p \mathcal{L}_2$, if $\mathcal{L}_1 \leq_p \mathcal{L}_2$ and $\mathcal{L}_2 \leq_p \mathcal{L}_1$.

Theorem 3. *The languages $\mathcal{EL}^{\nu+}$, $\mathcal{EL}^{si}$, $\mathcal{EL}^{st}$, and $\mathcal{EL}^{sa}$ are polynomially equivalent.*

We provide sketches of proofs of $\mathcal{EL}^{si} \leq_p \mathcal{EL}^{\nu+}$, $\mathcal{EL}^{\nu+} \leq_p \mathcal{EL}^{st}$, $\mathcal{EL}^{st} \leq_p \mathcal{EL}^{sa}$, and $\mathcal{EL}^{sa} \leq_p \mathcal{EL}^{si}$.

$\mathcal{EL}^{si} \leq_p \mathcal{EL}^{\nu+}$. By Lemma 1, considering $\mathcal{EL}^{si}$-concepts of the form $\exists^{sim}(\mathcal{I}, d)$ is sufficient. Each such concept is equivalent to the $\mathcal{EL}^{\nu+}$-concept $\nu_\ell d_1 \cdots d_n.C_1, \ldots, C_n$, where $\Delta^{\mathcal{I}} = \{d_1, \ldots, d_n\}$ is regarded as a set of concept variables, $d = d_\ell$, and

$$C_i = \bigsqcap \{A \mid d_i \in A^{\mathcal{I}}\} \sqcap \bigsqcap \{\exists r.d_j \mid (d_i, d_j) \in r^{\mathcal{I}}\}.$$

$\mathcal{EL}^{\nu+} \leq_p \mathcal{EL}^{st}$. Let C be a closed $\mathcal{EL}^{\nu+}$-concept. An equivalent $\mathcal{EL}^{st}$-concept is constructed by replacing each subconcept of C of the form $\nu_\ell X_1, \ldots, X_n.C_1, \ldots, C_n$ with

an $\mathcal{EL}^{st}$-concept, proceeding from the inside out. We assume that for every variable X that occurs in the original $\mathcal{EL}^{\nu+}$-concept C, there is a concept name A_X that does not occur in C. Now $\nu_\ell X_1, \ldots, X_n.C_1, \ldots, C_n$ (which potentially contains free variables) is replaced with the $\mathcal{EL}^{st}$-concept

$$\exists^{sim}\{A_{X_1}, \ldots, A_{X_n}\}.(\{A_{X_i} \sqsubseteq C_i^\downarrow \mid 1 \le i \le n\}, A_{X_\ell})$$

where $C_i^\downarrow$ is obtained from C_i by replacing every variable X with the concept name A_X.

$\mathcal{EL}^{st} \le_p \mathcal{EL}^{sa}$. Let C be an $\mathcal{EL}^{st}$-concept. As already observed, we may assume that D is a concept name in all subconcepts $\exists^{sim}\Sigma.(\mathcal{T}, D)$ of C. Now replace each $\exists^{sim}\Sigma.(\mathcal{T}, A)$ in C, proceeding from the inside out, by $\exists^{sim}\Sigma.(\mathcal{T}, \mathcal{A}, a)$, where $\mathcal{A} = \{A(a)\}$. The resulting concept is equivalent to C.

$\mathcal{EL}^{sa} \le_p \mathcal{EL}^{si}$. To prove this inclusion, we make use of *canonical models* for $\mathcal{EL}^{sa}$-KBs, similar to those used for $\mathcal{EL}$ in [5]. In particular, canonical models for $\mathcal{EL}^{sa}$ can be constructed by an extension of the algorithm given in [5], see the full version for details.

Theorem 4 (Canonical model). *For every satisfiable $\mathcal{EL}^{sa}$-KB $(\mathcal{T}, \mathcal{A})$, one can construct in polynomial time a model $\mathcal{I}_{\mathcal{T}, \mathcal{A}}$ of $(\mathcal{T}, \mathcal{A})$ with $|\Delta^{\mathcal{I}_{\mathcal{T}, \mathcal{A}}}|$ bounded by twice the size of $(\mathcal{T}, \mathcal{A})$ and such that for every model $\mathcal{J}$ of $(\mathcal{T}, \mathcal{A})$, we have $(\mathcal{I}_{\mathcal{T}, \mathcal{A}}, a^{\mathcal{I}_{\mathcal{T}, \mathcal{A}}}) \le (\mathcal{J}, a^{\mathcal{J}})$ for all $a \in \mathsf{Ind}(\mathcal{A})$.*

To prove $\mathcal{EL}^{sa} \le_p \mathcal{EL}^{si}$, it suffices to show that any outermost occurrence of a concept of the form $\exists^{sim}\Sigma.(\mathcal{T}, \mathcal{A}, a)$ in an $\mathcal{EL}^{sa}$-concept C can be replaced with the equivalent $\mathcal{EL}^{si}$-concept $\exists^{sim}(\mathcal{I}_{\mathcal{T}, \mathcal{A}}^\Sigma, a)$, where $\mathcal{I}_{\mathcal{T}, \mathcal{A}}^\Sigma$ denotes $\mathcal{I}_{\mathcal{T}, \mathcal{A}}$ except that all $\sigma \in \Sigma$ are interpreted as empty sets. First let $d \in (\exists^{sim}\Sigma.(\mathcal{T}, \mathcal{A}, a))^{\mathcal{J}}$. Then there is a model $\mathcal{I}'$ of $(\mathcal{T}, \mathcal{A})$ such that $(\mathcal{I}', a^{\mathcal{I}'}) \le_\Sigma (\mathcal{J}, d)$. By Theorem 4, $(\mathcal{I}_{\mathcal{T}, \mathcal{A}}, a^{\mathcal{I}_{\mathcal{T}, \mathcal{A}}}) \le (\mathcal{I}', a^{\mathcal{I}'})$. Thus, by closure of simulations under composition, $(\mathcal{I}_{\mathcal{T}, \mathcal{A}}^\Sigma, a) \le_\Sigma (\mathcal{J}, d)$ as required. The converse direction follows from the condition that $\mathcal{I}_{\mathcal{T}, \mathcal{A}}$ is a model of $(\mathcal{T}, \mathcal{A})$. This finishes our proof sketch for Theorem 3.

It is interesting to note that, as a consequence of the proofs of Theorem 3, for every $\mathcal{EL}^{\nu+}$-concept there is an equivalent $\mathcal{EL}^{\nu+}$-concept of polynomial size in which the greatest fixpoint constructor is not nested, and similarly for $\mathcal{EL}^{st}$, $\mathcal{EL}^{sa}$. An important consequence of the existence of canonical models, as granted by Theorem 4, is that reasoning in our family of extensions of $\mathcal{EL}$ is tractable.

Theorem 5 (Tractable reasoning). *Let $\mathcal{L}$ be any of the languages $\mathcal{EL}^\nu$, $\mathcal{EL}^{\nu+}$, $\mathcal{EL}^{si}$, $\mathcal{EL}^{st}$, or $\mathcal{EL}^{sa}$. Then KB consistency, subsumption w.r.t. TBoxes, and the instance problem can be decided in* PTIME.

Proof. By Theorem 3, it suffices to concentrate on $\mathcal{L} = \mathcal{EL}^{sa}$. Consistency can be decided in PTIME by the algorithm that constructs the canonical model. Subsumption can be polynomially reduced in the standard way to the instance problem. Finally, by Theorem 4, we can decide the instance problem as follows: to decide whether $(\mathcal{T}, \mathcal{A}) \models C(a)$, where we can w.l.o.g. assume that $C = A$ for a concept name A, we check whether $(\mathcal{T}, \mathcal{A})$ is inconsistent or $a^{\mathcal{I}_{\mathcal{T}, \mathcal{A}}} \in A^{\mathcal{I}_{\mathcal{T}, \mathcal{A}}}$. Both can be done in PTIME. □

5   Characterizing $\mathcal{EL}^{\nu}$ using simulations

When characterizing $\mathcal{EL}$ as a fragment of first-order logic in Theorem 2, our starting point was the observation that $\mathcal{EL}$-concepts are preserved under simulations and that $\mathcal{EL}$ is a Horn logic, thus having finite minimal models. The same is true for $\mathcal{EL}^{\nu}$: first, $\mathcal{EL}^{\nu}$-concepts are preserved under simulations, as $\mathcal{EL}^{si}$ is obviously preserved under simulations and, by Theorem 3, every $\mathcal{EL}^{\nu}$-concept is equivalent to an $\mathcal{EL}^{si}$-concept. And second, a finite minimal model of an $\mathcal{EL}^{\nu}$-concept C can be constructed by taking the canonical model $\mathcal{I}_{\mathcal{T},\mathcal{A}}$ from Theorem 4 for $\mathcal{T} = \{A \sqsubseteq C\}$ and $\mathcal{A} = \{A(a)\}$. As required, we then have $\models C \sqsubseteq D$ iff $(\mathcal{T},\mathcal{A}) \models D(a)$ iff $a \in D^{\mathcal{I}_{\mathcal{T},\mathcal{A}}}$, for all $\mathcal{EL}^{\nu}$-concepts D. However, $\mathcal{EL}^{\nu}$ is clearly not a fragment of FO. Instead, it relates to *MSO* in exactly the way that $\mathcal{EL}$ related to FO.

Theorem 6. *The set of $\mathcal{EL}^{\nu}$-concepts is a maximal set of MSO-formulas that is preserved under simulations and has finite minimal models: if $\mathcal{L}$ is a set of MSO-formulas that properly contains all $\mathcal{EL}^{\nu}$-concepts, then either it contains a formula not preserved under simulations or it does not have finite minimal models.*

Proof. Assume that $\mathcal{L} \supseteq \mathcal{EL}^{\nu}$ is preserved under simulations and has finite minimal models. Let $\varphi(x) \in \mathcal{L}$. We have to show that $\varphi(x)$ is equivalent to an $\mathcal{EL}^{\nu}$-concept. To this end, take a finite minimal model of φ, i.e., an interpretation $\mathcal{I}$ and a $d \in \Delta^{\mathcal{I}}$ such that for all $\psi(x) \in \mathcal{L}$ we have that $\forall x.(\varphi(x) \rightarrow \psi(x))$ is valid iff $\mathcal{I} \models \psi[d]$. We will show that φ is equivalent to (the MSO translation of) $\exists^{sim}(\mathcal{I}, d)$. We may assume that $\exists^{sim}(\mathcal{I}, d) \in \mathcal{L}$. Since $d \in (\exists^{sim}(\mathcal{I}, d))^{\mathcal{I}}$, we thus have that $\forall x.(\varphi(x) \rightarrow \exists^{sim}(\mathcal{I}, d)(x))$ is valid. Conversely, assume that $d' \in (\exists^{sim}(\mathcal{I}, d))^{\mathcal{J}}$ for some interpretation $\mathcal{J}$. Then $(\mathcal{I}, d) \leq (\mathcal{J}, d')$. We have $(\mathcal{I}, d) \models \varphi[d]$. Thus, by preservation of $\varphi(x)$ under simulations, $\mathcal{J} \models \varphi[d']$. Thus $\forall x.(\exists^{sim}(\mathcal{I}, d)(x) \rightarrow \varphi(x))$ is also valid.   □

A number of closely related characterizations remain open. For example, we conjecture that an extension of Theorem 1 holds for $\mathcal{EL}^{\nu,\sqcup}$ and MSO (instead of $\mathcal{EL}$ and FO). Also, it is open whether Theorem 6 still holds if finite minimal models are replaced by arbitrary minimal models.

6   Applications and Logical Properties

The μ-calculus is considered to be extremely well-behaved regarding its expressive power and logical properties. The aim of this section is to take a brief look at the expressive power of its $\mathcal{EL}$-analogues $\mathcal{EL}^{\nu}$ and $\mathcal{EL}^{\nu+}$. In particular, we show that $\mathcal{EL}^{\nu+}$ is more well-behaved than $\mathcal{EL}$ in a number of respects. Throughout this section, we will not distinguish between the languages previously proved polynomially equivalent.

To begin with, we construct the *least common subsumer* (LCS) of two concepts w.r.t. a general $\mathcal{EL}^{\nu+}$-TBox (the generalization to more than two concepts is straightforward). Given a general $\mathcal{EL}^{\nu+}$-TBox $\mathcal{T}$ and concepts C_1, C_2, a concept C is called the *LCS of C_1, C_2 w.r.t. $\mathcal{T}$ in $\mathcal{EL}^{\nu+}$* if

  – $\mathcal{T} \models C_i \sqsubseteq C$ for $i = 1, 2$;

- if $\mathcal{T} \models C_i \sqsubseteq D$ for $i = 1, 2$ and D a $\mathcal{EL}^{\nu+}$-concept, then $\mathcal{T} \models C \sqsubseteq D$.

It is known that, in $\mathcal{EL}$, the LCS does not always exist [1].

Example 4. In $\mathcal{EL}$, the LCS of A, B w.r.t.

$$\mathcal{T} = \{A \sqsubseteq \exists\mathsf{has\_parent}.A, B \sqsubseteq \exists\mathsf{has\_parent}.B\}$$

does not exist. In $\mathcal{EL}^{\nu}$, however, the LCS of A,B w.r.t. $\mathcal{T}$ is given by $\nu X.\exists\mathsf{has\_parent}.X$.

To construct the LCS in $\mathcal{EL}^{\nu+}$, we adopt the product construction used in [1] for the case of classical TBoxes with a fixpoint semantics. For interpretations $\mathcal{I}_1$ and $\mathcal{I}_2$, the *product* $\mathcal{I}_1 \times \mathcal{I}_2$ is defined by setting $\Delta^{\mathcal{I}_1 \times \mathcal{I}_2} = \Delta^{\mathcal{I}_1} \times \Delta^{\mathcal{I}_2}$, $(d_1, d_2) \in A^{\mathcal{I}_1 \times \mathcal{I}_2}$ iff $d_i \in A^{\mathcal{I}_i}$ for $i = 1, 2$, and $((d_1, d_2), (d_1', d_2')) \in r^{\mathcal{I}_1 \times \mathcal{I}_2}$ iff $(d_i, d_i') \in r^{\mathcal{I}_i}$ for $i = 1, 2$.

Theorem 7. *Let $\mathcal{T}$ be a general $\mathcal{EL}^{\nu+}$-TBox and C_1 and C_2 be $\mathcal{EL}^{\nu+}$-concepts. Then $\exists^{sim}(\mathcal{I}_{\mathcal{T},C_1} \times \mathcal{I}_{\mathcal{T},C_2}, (d_{C_1}, d_{C_2}))$ is the LCS of C_1, C_2 w.r.t. $\mathcal{T}$ in $\mathcal{EL}^{\nu}$.*

The same product construction has been used in [1] for the case of classical TBoxes with a fixpoint semantics, which, however, additionally require a notion of conservative extension (see Section 7).

Our second result concerns the most specific concept, which plays an important role in the bottom-up construction of knowledge bases and has received quite a bit of attention in the context of $\mathcal{EL}$ [1, 7]. Formally, a concept C is the *most specific concept (MSC)* for an individual a in a knowledge base $(\mathcal{T}, \mathcal{A})$ in $\mathcal{EL}^{\nu+}$ if

- $(\mathcal{T}, \mathcal{A}) \models C(a)$ and
- for every $\mathcal{EL}^{\nu+}$-concept D with $(\mathcal{T}, \mathcal{A}) \models D(a)$, we have $\mathcal{T} \models C \sqsubseteq D$.

In $\mathcal{EL}$, the MSC need not exist, as is witnessed by the KB $(\emptyset, \{\mathsf{has\_parent}(a, a)\})$, where the MSC for a is non-existent.

Theorem 8. *In $\mathcal{EL}^{\nu+}$, the MSC always exists for any a in any KB $(\mathcal{T}, \mathcal{A})$ and is given as $\exists^{sim}\emptyset.(\mathcal{T}, \mathcal{A}, a)$.*

In [1], the MSC in $\mathcal{EL}$-KBs based on classical TBoxes with a fixpoint semantics is defined. The relationship between $\mathcal{EL}^{\nu+}$ and fixpoint TBoxes is discussed in more detail in Section 7.

We now turn our attention to issues of definability and interpolation. From now on, we use $\mathsf{sig}(C)$ to denote the set of concept and role names used in the concept C. A concept C is a Σ-*concept* if $\mathsf{sig}(C) \subseteq \Sigma$. Let $\mathcal{T}$ be a general $\mathcal{EL}^{\nu+}$-TBox, C an $\mathcal{EL}^{\nu+}$-concept and Γ a finite signature.

We start with considering the fundamental notion of a Γ-definition. The question addressed here is whether a given concept can be expressed in an equivalent way by referring only to the symbols in a given signature Γ [21, 20]. Formally, a Γ-concept D is an *explicit Γ-definition* of a concept C w.r.t. a TBox $\mathcal{T}$ if, and only if, $\mathcal{T} \models C \equiv D$ (i.e., C and D are equivalent w.r.t. $\mathcal{T}$). Clearly, explicit Γ-definitions do not always exist in any of the logics studied in this paper: for example, there is no explicit $\{A\}$-definition of B w.r.t. the TBox $\{A \sqsubseteq B\}$. However, it is not hard to show the following using the fact that $\exists^{sim}\Sigma.(\mathcal{T}, C)$ is the most specific Γ-concept that subsumes C w.r.t. $\mathcal{T}$.

Proposition 2. *Let C be an $\mathcal{EL}^{\nu+}$-concept, $\mathcal{T}$ a general $\mathcal{EL}^{\nu+}$-TBox and Γ a signature. There exists an explicit Γ-definition of C w.r.t. $\mathcal{T}$ iff $\exists^{sim}\Sigma.(\mathcal{T},C)$ is such a definition (for $\Sigma = \mathsf{sig}(\mathcal{T},C) \setminus \Gamma$).*

It is interesting to note that if $\mathcal{T}$ happens to be a general $\mathcal{EL}$-TBox and C an $\mathcal{EL}$-concept and there exists an explicit Γ-definition of C w.r.t. $\mathcal{T}$, then the concept $\exists^{sim}\Sigma.(\mathcal{T},C)$ from Proposition 2 is equivalent w.r.t. $\mathcal{T}$ to an $\mathcal{EL}$-concept over Γ. This follows from the fact that $\mathcal{EL}$ has the Beth definability property (see below for a definition) which follows immediately from interpolation results proved for $\mathcal{EL}$ in [15].

The advantage of giving explicit Γ-definitions in $\mathcal{EL}^{\nu+}$ even when $\mathcal{T}$ and C are formulated in $\mathcal{EL}$ is that Γ-definitions in $\mathcal{EL}^{\nu+}$ are of polynomial size while the following example shows that they may be exponentially large in $\mathcal{EL}$.

Example 5. Let $\mathcal{T}$ consist of $A_i \equiv \exists r_i.A_{i+1} \sqcap \exists s_i A_{i+1}$ for $0 \leq i < n$, and $A_n \equiv \top$. Let $\Gamma = \{r_0, \ldots, r_{n-1}, s_0, \ldots, s_{n-1}\}$. Then A_0 has an explicit Γ-definition w.r.t. $\mathcal{T}$ in $\mathcal{EL}$, namely C_0, where $C_i = \exists r_i.C_{i+1} \sqcap \exists s_i.C_{i+1}$ and $C_n = \top$. This definition is of exponential size and it is easy to see that there is no shorter Γ-definition of A_0 w.r.t. $\mathcal{T}$ in $\mathcal{EL}$.

Say that a concept C is *implicitly Γ-defined w.r.t. $\mathcal{T}$* iff $\mathcal{T} \cup \mathcal{T}_\Gamma \models C \equiv C_\Gamma$, where $\mathcal{T}_\Gamma$ and C_Γ are obtained from $\mathcal{T}$ and C, respectively, by replacing each $\sigma \notin \Gamma$ by a fresh symbol σ'. The Beth definability property, which was studied in a DL context in [21, 20], ensures that explicit Γ-definitions always exist when they possibly can.

Theorem 9. *$\mathcal{EL}^{\nu+}$ has the polynomial Beth definability property: for every general $\mathcal{EL}^{\nu+}$-TBox $\mathcal{T}$, concept C, and signature Γ such that C is implicitly Γ-defined w.r.t. $\mathcal{T}$, there is an explicit Γ-definition w.r.t. $\mathcal{T}$, namely $\exists^{sim}(\mathsf{sig}(\mathcal{T},C) \setminus \Gamma).(\mathcal{T},C)$.*

The proof of Theorem 9 relies on $\mathcal{EL}^{\nu}$ having a certain interpolation property. Say that two general TBoxes $\mathcal{T}_1$ and $\mathcal{T}_2$ are *Δ-inseparable* w.r.t. $\mathcal{EL}^{\nu}$ if $\mathcal{T}_1 \models C \sqsubseteq D$ iff $\mathcal{T}_2 \models C \sqsubseteq D$ for all $\mathcal{EL}^{\nu}$-inclusions $C \sqsubseteq D$.

Theorem 10. *Let $\mathcal{T}_1 \cup \mathcal{T}_2 \models C \sqsubseteq D$ and assume that $\mathcal{T}_1$ and $\mathcal{T}_2$ are Δ-inseparable w.r.t. $\mathcal{EL}^{\nu}$ for $\Delta = \mathsf{sig}(\mathcal{T}_1,C) \cap \mathsf{sig}(\mathcal{T}_2,D)$. Then the Δ-concept $F = \exists^{sim}\Sigma.(\mathcal{T}_1,C)$, $\Sigma = \mathsf{sig}(\mathcal{T}_1,C) \setminus \Delta$, is an interpolant of C,D w.r.t. $\mathcal{T}_1, \mathcal{T}_2$; i.e. $\mathcal{T}_1 \models C \sqsubseteq F$ and $\mathcal{T}_2 \models F \sqsubseteq D$.*

We show how Theorem 9 follows from Theorem 10. Assume that $\mathcal{T} \cup \mathcal{T}_\Gamma \models C \equiv C_\Gamma$, where $\mathcal{T}, \mathcal{T}_\Gamma, C, C_\Gamma$ satisfy the conditions of Theorem 9. Then $\mathcal{T}$ and $\mathcal{T}_\Gamma$ are Γ-inseparable and $\Gamma \supseteq \mathsf{sig}(\mathcal{T},C) \cap \mathsf{sig}(\mathcal{T}_\Gamma,C_\Gamma)$. Thus, by Theorem 10, $\mathcal{T} \models \exists^{sim}\Sigma.(\mathcal{T}_\Gamma,C_\Gamma) \sqsubseteq C$ for $\Sigma = \mathsf{sig}(\mathcal{T}_\Gamma,C_\Gamma) \setminus \Gamma$. Now Theorem 9 follows from the fact that $\exists^{sim}\Sigma.(\mathcal{T}_\Gamma,C_\Gamma)$ is equivalent to $\exists^{sim}\Sigma'.(\mathcal{T},C)$ for $\Sigma' = \mathsf{sig}(\mathcal{T},C) \setminus \Gamma$.

In [15], it is shown that $\mathcal{EL}$ also has this interpolation property. However, the advantage of using $\mathcal{EL}^{\nu+}$ is that interpolants are of polynomial size. The decomposition algorithm for $\mathcal{EL}$ given in [15] crucially depends on this property of $\mathcal{EL}^{\nu+}$.

7 Relation to TBoxes with Fixpoint Semantics

There is a tradition of considering DLs that introduce fixpoints at the TBox level instead of at the concept level [18, 19, 2]. In [4], Baader proposes and analyzes such a DL based on $\mathcal{EL}$ and greatest fixpoints. This DL, which we call $\mathcal{EL}^{\mathsf{gfp}}$ from now on, differs from our $\mathcal{EL}^\nu$ in that (i) TBoxes are classical TBoxes rather than sets of GCIs (but cycles are allowed) and (ii) the ν-concept constructor is not present; instead, a greatest fixpoint semantics is adopted for the defined concept names.

On the concept level, $\mathcal{EL}^\nu$ is clearly strictly more expressive than $\mathcal{EL}^{\mathsf{gfp}}$: since fixpoints are introduced at the TBox level, concepts of $\mathcal{EL}^{\mathsf{gfp}}$ coincide with $\mathcal{EL}$-concepts, and thus there is no $\mathcal{EL}^{\mathsf{gfp}}$-concept equivalent to the $\mathcal{EL}^\nu$-concept $\nu X. \exists r. X$. In the following, we show that $\mathcal{EL}^\nu$ is also more expressive than $\mathcal{EL}^{\mathsf{gfp}}$ also on the TBox level, even if we restrict $\mathcal{EL}^\nu$-TBoxes as in $\mathcal{EL}^{\mathsf{gfp}}$. We use the standard notion of logical equivalence, i.e., two TBoxes $\mathcal{T}$ and $\mathcal{T}'$ are *equivalent* iff $\mathcal{T}$ and $\mathcal{T}'$ have precisely the same models. As observed by Schild in the context of $\mathcal{ALC}$ [19], every $\mathcal{EL}^{\mathsf{gfp}}$-TBox $\mathcal{T} = \{A_1 \equiv C_1, \dots, A_n \equiv C_n\}$ is equivalent in this sense to the $\mathcal{EL}^{\nu+}$-TBox $\{A_i \equiv \nu_i X_1, \dots, X_n. C'_1, \dots, C'_n \mid 1 \le i \le n\}$, where each C'_i is obtained from C_i by replacing each A_j with X_j, $1 \le j \le n$. Note that since we are using mutual fixpoints the size of the resulting TBox is polynomial in the size of the original one. In the converse direction, there is no equivalence-preserving translation.

Lemma 2. *For each $\mathcal{EL}^{\mathsf{gfp}}$-TBox, there is an equivalent $\mathcal{EL}^{\nu+}$-TBox of polynomial size, but no $\mathcal{EL}^{\mathsf{gfp}}$-TBox is equivalent to the $\mathcal{EL}^\nu$-TBox $\{A \equiv P \sqcap \nu X. \exists r. X\}$.*

Proof. (sketch) It is not hard to prove that for every $\mathcal{EL}^{\mathsf{gfp}}$-TBox $\mathcal{T}$, defined concept name A in $\mathcal{T}$, and role name r, one of the following holds:

- there is an $m \ge 0$ such that $A \sqsubseteq \exists r^n. \top$ implies $n \le m$ or
- $A \sqsubseteq \exists r^n. B$ for some $n > 0$ and defined concept name B.

However, no such TBox can be equivalent to $A \sqsubseteq \exists r^n. B$ since $\mathcal{T} \models \exists r^n. \top$ for all $n > 0$, but there is no $n > 0$ and defined concept name B with $A \sqsubseteq \exists r^n. B$. □

$\mathcal{EL}^{\mathsf{gfp}}$ and $\mathcal{EL}^\nu$ become equi-expressive if the strict notion of equivalence used above is replaced with one based on conservative extensions, thus allowing the introduction of new concept names that are suppressed from logical equivalence. However, we believe that not having to deal with conservative extensions is an advantage of $\mathcal{EL}^\nu$ over $\mathcal{EL}^{\mathsf{gfp}}$, as conservative extensions tend to make simple definitions somewhat awkward, c.f. the least common subsumers and most specific concepts for $\mathcal{EL}^{\mathsf{gfp}}$ in [3, 4].

References

1. F. Baader. Least common subsumers and most specific concepts in description logics with existential restrictions and terminological cycles. In *Proc. of IJCAI03*, pages 319–324. Morgan Kaufmann, 2003.
2. F. Baader. Using automata theory for characterizing the semantics of terminological cycles. *Annals of Mathematics and Artificial Intelligence*, 18(2–4):175–219, 1996.

3. F. Baader. The instance problem and the most specific concept in the description logic $\mathcal{EL}$ w.r.t. terminological cycles with descriptive semantics. In *Proc. of KI 2003*, volume 2821 of *LNAI*, pages 64–78, 2003. Springer.

4. F. Baader. Terminological cycles in a description logic with existential restrictions. In *Proc. of IJCAI03*, pages 325–330. Morgan Kaufmann, 2003.

5. F. Baader, S. Brandt, and C. Lutz. Pushing the $\mathcal{EL}$ envelope. In *Proc. of IJCAI05*. Morgan Kaufmann, 2005.

6. F. Baader, D. Calvanes, D. McGuiness, D. Nardi, and P. Patel-Schneider. *The Description Logic Handbook: Theory, implementation and applications*. Cambridge Univ. Press, 2003.

7. F. Baader and F. Distel. A finite basis for the set of $\mathcal{EL}$-implications holding in a finite model. In *Proc. of ICFCA8*, volume 4933 of *LNAI*, pages 46–61. Springer, 2008.

8. J. Bradfield and C. Stirling. Modal mu-calculus. In *Handbook of Modal Logic*. Elsevier, 2006.

9. D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, and R. Rosati. Tractable reasoning and efficient query answering in description logics: The *DL-Lite* family. *J. of Automated Reasoning*, 39(3):385–429, 2007.

10. J. W. de Bakker. *Mathematical Theory of Program Correctness*. Prentice-Hall, 1980.

11. T. French. *Bisimulation quantifiers for modal logics*. PhD thesis, University of Western Australia, 2006.

12. G. De Giacomo and M. Lenzerini. Boosting the Correspondence between Description Logics and Propositional Dynamic Logics. In *Proc. of AAAI94*, pages 205-212. AAAI Press, 1994.

13. V. Goranko and M. Otto. Model theory of modal logic. In *Handbook of Modal Logic*. Elsevier, 2007.

14. D. Janin and I. Walukiewicz. On the expressive completeness of the propositional mu-calculus with respect to monadic second order logic. In *Proc. of CONCUR96*, volume 1119 of *PNCS*, pages 263–277. Springer, 1996.

15. B. Konev, C. Lutz, D. Ponomaryov, and F. Wolter. Decomposing description logic ontologies. In *Proceedings of KR10*, 2010.

16. N. Kurtonina and M. de Rijke. Expressiveness of concept expressions in first-order description logics. *Artificial Intelligence*, 107, 1999.

17. C. Lutz and F. Wolter. Deciding inseparability and conservative extensions in the description logic $\mathcal{EL}$. *J. of Symbolic Computation*, 45(2):194–228, 2010.

18. B. Nebel. Terminological cycles: Semantics and computational properties. In *Principles of Semantic Networks*. Morgan Kaufmann, 1991.

19. K. Schild. Terminological cycles and the propositional μ-calculus. In *Proc. of KR'94*, pages 509–520. Morgan Kaufmann, 1994.

20. I. Seylan, E. Franconi, and J. de Bruijn. Effective query rewriting with ontologies over DBoxes. In *Proc. of IJCAI09*, pages 923–925, 2009.

21. B. ten Cate, W. Conradie, M. Marx, and Y. Venema. Definitorially complete description logics. In *Proc. of KR'06*, pages 79–89. AAAI Press, 2006.

22. J. van Benthem. Modal Correspondence Theory. Mathematical Institute, University of Amsterdam, 1976.

23. M. Vardi and P. Wolper. Automata theoretic techniques for modal logics of programs. In *Proc. of STOC84*, pages 446–456. ACM Press, 1984.

24. M. Y. Vardi. Why is modal logic so robustly decidable? In *Descriptive Complexity and Finite Models*, pages 149–184, 1996.

Checking Full Satisfiability of Conceptual Models*

Alessandro Artale, Diego Calvanese, and Angélica Ibáñez-García

KRDB Research Centre
Free University of Bozen-Bolzano, Italy
{artale,calvanese}@inf.unibz.it   yibanez@gmail.com

Abstract. UML class diagrams (UCDs) are the de-facto standard formalism for the analysis and design of information systems. By adopting formal language techniques to capture constraints expressed by UCDs one can exploit automated reasoning tools to detect relevant properties, such as schema and class satisfiability and subsumption between classes. Among the reasoning tasks of interest, the basic one is detecting *full satisfiability* of a diagram, i.e., whether there exists an instantiation of the diagram where *all* classes and associations of the diagram are non-empty and all the constraints of the diagram are respected. In this paper we establish tight complexity results for full satisfiability for various fragments of UML class diagrams. This investigation shows that the full satisfiability problem is ExpTime-complete in the full scenario, NP-complete if we drop ISA between relationships, and NLogSpace-complete if we further drop covering over classes.

1   Introduction

UML (Unified Modeling Language)[1] is the de-facto standard formalism for the analysis and design of information systems. One of the most important components of UML are *class diagrams* (UCDs), which model the domain of interest in terms of objects organized in classes and associations between them (representing relations between class instances). The semantics of UCDs is by now well established, and several works propose to represent it using various kinds of formal languages, e.g., [5,8,7,9,10,4,1,2]. Thus, one can in principle reason on UCDs. The reasoning tasks that one is interested in are, e.g., subsumption between two classes, and satisfiability of a specific class or association in the diagram. Here, we consider *full satisfiability* of a diagram [12], i.e., the fact that there is at least one model of the diagram where each class and association is non-empty. This property is of importance since the presence of some unsatisfiable class or association actually means either that the diagram contains unnecessary information that should be removed, or that there is some modelling error that lead to the loss of satisfiability. In fact, it can be considered as the most fundamental property that should be satisfied by UCDs.

---

[1] http://www.omg.org/spec/UML/

The only work that addressed explicitly the complexity of full satisfiability of UCDs is [12], which includes a classification of UCDs based on *inconsistency triggers*. Each inconsistency trigger is a pattern for recognizing possible inconsistencies of the diagram, based on the interaction between different modelling constraints. [12] introduces various algorithms for checking full satisfiability of UCDs with diverse expressive power, together with an analysis of their computational complexity. Full satisfiability of UCDs is computed in ExpTime in the most general case; in NP if association generalization and multiple and overwriting inheritance of attributes is dropped; and in P if the diagrams are further restricted by forbidding covering constraints. According to the results reported in [12], the complexity of checking full satisfiability of UCDs can be reduced if the value types of the attributes associated to sub-classes are sub-types of the value types for the respective attributes associated to the super-classes. The algorithms handling these *restricted* UCDs are claimed to compute full satisfiability respectively in PSpace (instead of ExpTime) and P (instead of NP).

However, our results show that even when attributes are not considered at all in the UCDs, the complexity of the problem does not change. Indeed this paper shows that the full satisfiability problem is ExpTime-complete in the full scenario, NP-complete if we drop ISA between relationships, and NLogSpace-complete if we further drop covering over classes. Thus, the complexity of full satisfiability coincides in all cases with that of class satisfiability [1]. Our results build on the formalization of UCDs in terms of DLs given in [4,1]. In fact, our upper bounds are an almost direct consequence of the corresponding upper bounds of the corresponding DL formalization. On the other hand, the obtained lower bounds are more involved, and in some cases require a careful analysis of the corresponding proof for class satisfiability. The results presented here hold also for the Entity-Relationship model and other conceptual models.

The rest of the paper is organized as follows. In Section 2, we briefly introduce the DL $\mathcal{ALC}$, on which we base our results, and show that full satisfiability in $\mathcal{ALC}$ is ExpTime-complete. In Sections 3 and 4, we provide our results on full satisfiability of various variants of UCDs.

2 Full Satisfiability in the Description Logic $\mathcal{ALC}$

We start by studying *full satisfiability* for the DL $\mathcal{ALC}$, one of the basic variants of DLs [3]. We first define the notion of *full satisfiability* of a TBox and then we show that it has the same complexity as classical satisfiability for $\mathcal{ALC}$.

Definition 1 (TBox Full Satisfiability). An $\mathcal{ALC}$ TBox $\mathcal{T}$ is said to be *fully satisfiable* if there exists a model $\mathcal{I}$ of $\mathcal{T}$ such that $A^{\mathcal{I}} \neq \emptyset$, for every atomic concept A in $\mathcal{T}$. We say that $\mathcal{I}$ is a *full model* of $\mathcal{T}$.

Lemma 2. *Concept satisfiability w.r.t. $\mathcal{ALC}$ TBoxes can be linearly reduced to full satisfiability of $\mathcal{ALC}$ TBoxes.*

Proof. Let $\mathcal{T}$ be an $\mathcal{ALC}$ TBox and C an $\mathcal{ALC}$ concept. As pointed out in [6], C is satisfiable w.r.t. $\mathcal{T}$ if and only if $C \sqcap A_{\mathcal{T}}$ is satisfiable w.r.t. the TBox $\mathcal{T}_1$

consisting of the single assertion $A_{\mathcal{T}} \sqsubseteq \prod_{C_1 \sqsubseteq C_2 \in \mathcal{T}} (\neg C_1 \sqcup C_2) \sqcap \prod_{1 \leq i \leq n} \forall P_i.\, A_{\mathcal{T}}$, where $A_{\mathcal{T}}$ is a fresh atomic concept and $P_1, \ldots, P_n$ are all the atomic roles in $\mathcal{T}$ and C. In order to reduce the problem to full satisfiability, we extend $\mathcal{T}_1$ to $\mathcal{T}_2 = \mathcal{T}_1 \cup \{A_C \sqsubseteq C \sqcap A_{\mathcal{T}}\}$, with A_C a fresh atomic concept, and prove that

$$C \sqcap A_{\mathcal{T}} \text{ is satisfiable w.r.t. } \mathcal{T}_1 \text{ iff } \mathcal{T}_2 \text{ is fully satisfiable.}$$

($\Rightarrow$) Let $\mathcal{I}$ be a model of $\mathcal{T}_1$ such that $(C \sqcap A_{\mathcal{T}})^{\mathcal{I}} \neq \emptyset$. We construct an interpretation of $\mathcal{T}_2$, $\mathcal{J} = (\Delta^{\mathcal{I}} \cup \{d^{top}\}, \cdot^{\mathcal{J}})$, with $d^{top} \notin \Delta^{\mathcal{I}}$, such that:

$$A_{\mathcal{T}}^{\mathcal{J}} = A_{\mathcal{T}}^{\mathcal{I}}, \qquad A_C^{\mathcal{J}} = (C \sqcap A_{\mathcal{T}})^{\mathcal{I}},$$
$$A^{\mathcal{J}} = A^{\mathcal{I}} \cup \{d^{top}\} \quad \text{for each atomic concept } A \text{ in } \mathcal{T} \text{ and } C,$$
$$P^{\mathcal{J}} = P^{\mathcal{I}} \quad \text{for each atomic role } P \text{ in } \mathcal{T} \text{ and } C.$$

Obviously, the extension of every atomic concept is non-empty in $\mathcal{J}$. Next, we show that $\mathcal{J}$ is a model of $\mathcal{T}_2$, by relying on the fact (easily proved by structural induction) that $D^{\mathcal{I}} \subseteq D^{\mathcal{J}}$, for each subconcept D of concepts in $\mathcal{T}_1$. Then, it is easy to show that $\mathcal{J}$ satisfies the two assertion in $\mathcal{T}_2$:

$$A_{\mathcal{T}}^{\mathcal{J}} = A_{\mathcal{T}}^{\mathcal{I}} \subseteq (\prod_{C_1 \sqsubseteq C_2 \in \mathcal{T}} (\neg C_1 \sqcup C_2) \sqcap \prod_{1 \leq i \leq n} \forall P_i.A_{\mathcal{T}})^{\mathcal{I}}$$
$$\subseteq (\prod_{C_1 \sqsubseteq C_2 \in \mathcal{T}} (\neg C_1 \sqcup C_2) \sqcap \prod_{1 \leq i \leq n} \forall P_i.\, A_{\mathcal{T}})^{\mathcal{J}}$$
$$A_C^{\mathcal{J}} = (C \sqcap A_{\mathcal{T}})^{\mathcal{I}} \subseteq (C \sqcap A_{\mathcal{T}})^{\mathcal{J}}$$

($\Leftarrow$) Conversely, every *full model* $\mathcal{J}$ of $\mathcal{T}_2$ is also a model of $\mathcal{T}_1$ with $(C \sqcap A_{\mathcal{T}})^{\mathcal{J}} \neq \emptyset$, as $A_C^{\mathcal{J}} \subseteq (C \sqcap A_{\mathcal{T}})^{\mathcal{J}}$. $\qquad\square$

Theorem 3. *Full satisfiability of $\mathcal{ALC}$ TBoxes is* EXPTIME-*complete.*

Proof. The EXPTIME membership is straightforward, as deciding full satisfiability of an $\mathcal{ALC}$ TBox $\mathcal{T}$ can be reduced to deciding satisfiability of the TBox $\mathcal{T} \cup \bigcup_{1 \leq i \leq n} \{\top \sqsubseteq \exists P'.A_i\}$, where $A_1, \ldots, A_n$ are all the atomic concepts in $\mathcal{T}$, and P' is a fresh atomic role. The EXPTIME-hardness follows from Lemma 2. $\quad\square$

We now modify the reduction of Lemma 2 so that it applies also to *primitive* $\mathcal{ALC}^-$ TBoxes, i.e., TBoxes that contain only assertions of the form: $A \sqsubseteq B$, $A \sqsubseteq \neg B$, $A \sqsubseteq B \sqcup B'$, $A \sqsubseteq \forall P.\, B$, $A \sqsubseteq \exists P.\, B$, where A, B, B' are atomic concepts, and P is an atomic role.

Theorem 4. *Full satisfiability of primitive $\mathcal{ALC}^-$ TBoxes is* EXPTIME-*complete.*

Proof. The EXPTIME membership follows from Theorem 3. For proving the EXPTIME-hardness, we use a result in [4] showing that concept satisfiability in $\mathcal{ALC}$ can be reduced to atomic concept satisfiability w.r.t. primitive $\mathcal{ALC}^-$ TBoxes. Let $\mathcal{T}^- = \{A_j \sqsubseteq D_j \mid 1 \leq j \leq m\}$ be a primitive $\mathcal{ALC}^-$ TBox, and A_0

an atomic concept. By Lemma 2, we have that A_0 is satisfiable w.r.t. $\mathcal{T}^-$ if and only if the TBox $\mathcal{T}_2'$ containing the assertions

$$A_{\mathcal{T}^-} \sqsubseteq \bigsqcap_{A_j \sqsubseteq D_j \in \mathcal{T}^-} (\neg A_j \sqcup D_j) \sqcap \bigsqcap_{1 \le i \le n} \forall P_i.\, A_{\mathcal{T}^-}, \qquad A_0' \sqsubseteq A_0 \sqcap A_{\mathcal{T}^-},$$

is fully satisfiable, with $A_{\mathcal{T}^-}, A_0'$ fresh atomic concepts. $\mathcal{T}_2'$ is not a primitive $\mathcal{ALC}^-$ TBox, but it is equivalent to the TBox containing the assertions:

$$
\begin{array}{lll}
A_0' \sqsubseteq A_{\mathcal{T}^-} & A_{\mathcal{T}^-} \sqsubseteq \neg A_1 \sqcup D_1 & A_{\mathcal{T}^-} \sqsubseteq \forall P_1.\, A_{\mathcal{T}^-} \\
 & \vdots & \vdots \\
A_0' \sqsubseteq A_0 & A_{\mathcal{T}^-} \sqsubseteq \neg A_m \sqcup D_m & A_{\mathcal{T}^-} \sqsubseteq \forall P_n.\, A_{\mathcal{T}^-},
\end{array}
$$

Finally, to get a primitive $\mathcal{ALC}^-$ TBox, $\mathcal{T}_2^-$, we replace each assertion of the form $A_{\mathcal{T}^-} \sqsubseteq \neg A_j \sqcup D_j$ by $A_{\mathcal{T}^-} \sqsubseteq B_j^1 \sqcup B_j^2$, $B_j^1 \sqsubseteq \neg A_j$, and $B_j^2 \sqsubseteq D_j$, with B_j^1 and B_j^2 fresh atomic concepts, for $j \in \{1, \ldots, m\}$.

We show now that $\mathcal{T}_2'$ is fully satisfiable iff $\mathcal{T}_2^-$ is fully satisfiable:

($\Rightarrow$) Let $\mathcal{I} = (\Delta^\mathcal{I}, \cdot^\mathcal{I})$ be a full model of $\mathcal{T}_2'$. We extend $\mathcal{I}$ to an interpretation $\mathcal{J}$ of $\mathcal{T}_2^-$. Let $\Delta^\mathcal{J} = \Delta^\mathcal{I} \cup \{d^+, d^-\}$, with $\{d^+, d^-\} \cap \Delta^\mathcal{I} = \emptyset$, and define $\cdot^\mathcal{J}$ as follows:

$$
\begin{aligned}
& A_{\mathcal{T}^-}^\mathcal{J} = A_{\mathcal{T}^-}^\mathcal{I}, \quad {A_0'}^\mathcal{J} = {A_0'}^\mathcal{I}, \\
& A^\mathcal{J} = A^\mathcal{I} \cup \{d^+\}, \quad \text{for every other atomic concept } A \text{ in } \mathcal{T}_2', \\
& {B_j^1}^\mathcal{J} = (\neg A_j)^\mathcal{J} \text{ and } {B_j^2}^\mathcal{J} = D_j^\mathcal{J}, \quad \text{for each } A_{\mathcal{T}^-} \sqsubseteq B_j^1 \sqcup B_j^2 \in \mathcal{T}_2^-, \\
& P^\mathcal{J} = P^\mathcal{I} \cup \{(d^+, d^+)\}, \quad \text{for each atomic role } P \text{ in } \mathcal{T}_2^-.
\end{aligned}
$$

It is easy to see that $\mathcal{J}$ is a full model of $\mathcal{T}_2^-$.

($\Leftarrow$) Trivial, since every model of $\mathcal{T}_2^-$ is a model of $\mathcal{T}_2'$.   □

3   Full Satisfiability of UML Class Diagrams

Three notions of UCD satisfiability have been proposed in the literature [13,4,12,11]. First, *diagram satisfiability* refers to the existence of a *model*, i.e., an interpretation that satisfies all constraints expressed by the diagram and where at least one class has a nonempty extension. Second, *class satisfiability* refers to the existence of a model of the diagram where the given class has a nonempty extension. Third, we can check whether there is a model of an UML diagram that satisfies *all* classes and *all* relationships in a diagram. This last notion of satisfiability, referred here as *full satisfiability* and introduced in [12] is thus stronger than diagram satisfiability, since a model of a diagram that satisfies all classes is, by definition, also a model of that diagram.

We adopt the formalization of UCDs in terms of DLs as given in [4,1]. For lack of space we give here only a brief overview of such formalization. Classes are formalized by atomic concepts; and relations by roles. Generalization between
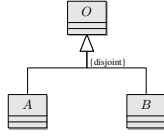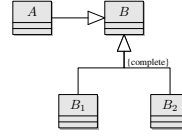
Fig. 1. Encoding of $A \sqsubseteq \neg B$ **Fig. 2.** Encoding of $A \sqsubseteq B_1 \sqcup B_2$
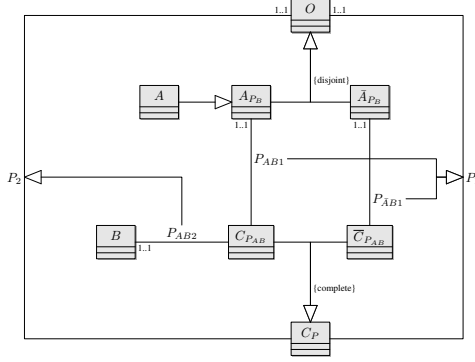
Fig. 3. Encoding of $A \sqsubseteq \forall P.B$ **Fig. 4.** Encoding of $A \sqsubseteq \exists P.B$
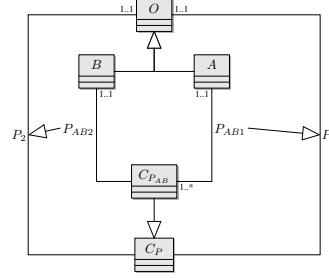
classes (e.g., $C_1 \mathrm{ISA} C_2$) are formalized by concept inclusions ($C_1 \sqsubseteq C_2$); disjointness constrains between two classes C_1 and C_2 by means of axioms of the form $C_1 \sqsubseteq \neg C_2$; and covering constraints by axioms of the form $C \sqsubseteq C_1 \sqcup C_2$. Finally, multiplicity constraints are formalized using qualified number restrictions.

Definition 5 (UML Full Satisfiability). A UCD, $\mathcal{D}$, is *fully satisfiable* if there is an interpretation, $\mathcal{I}$, that satisfies all the constraints expressed in $\mathcal{D}$ and such that $C^{\mathcal{I}} \neq \emptyset$ for every class C in $\mathcal{D}$, and $R^{\mathcal{I}} \neq \emptyset$ for every association R in $\mathcal{D}$. We say that $\mathcal{I}$ is a *full model* of $\mathcal{D}$.

We now address the complexity of full satisfiability for UCDs. For the lower bounds, we use the results presented in Section 2 and reduce full satisfiability of primitive $\mathcal{ALC}^-$ TBoxes to full satisfiability of UCDs. This reduction is based on the ones used in [4,1] for determining the lower complexity bound of schema satisfiability in the extended Entity-Relationship model.

Given a primitive $\mathcal{ALC}^-$ TBox $\mathcal{T}$, construct an UCD $\Sigma(\mathcal{T})$ as follows: for each atomic concept A in $\mathcal{T}$, introduce a class A in $\Sigma(\mathcal{T})$. Additionally, introduce a class O that generalizes (possibly indirectly) all the classes in $\Sigma(\mathcal{T})$ that encode an atomic concept in $\mathcal{T}$. For each atomic role P, introduce a class C_P, which reifies the binary relation P. Further, introduce two functional associations P_1, and P_2 that represent, respectively, the first and second component of P. The assertions in $\mathcal{T}$ are encoded as follows:

- The correspondence of UCDs and DLs gives a straightforward encoding for assertions of the form $A \sqsubseteq B$, $A \sqsubseteq \neg B$, and $A \sqsubseteq B_1 \sqcup B_2$ (see Fig. 1 and Fig. 2).

- For each assertion of the form $A \sqsubseteq \forall P.B$, add the auxiliary classes $C_{P_{AB}}$ and $\overline{C}_{P_{AB}}$, and the associations P_{AB1}, $P_{\bar{A}B1}$, and P_{AB2}, and construct the diagram shown in Fig. 3.
- For each assertion of the form $A \sqsubseteq \exists P.B$, add the auxiliary class $C_{P_{AB}}$ and the associations P_{AB1} and P_{AB2}, and construct the diagram shown in Fig. 4.

Lemma 6. *A primitive $\mathcal{ALC}^-$ TBox $\mathcal{T}$ is fully satisfiable iff the UCD $\Sigma(\mathcal{T})$, constructed as above, is fully satisfiable.*

Proof. ($\Leftarrow$) Let $\mathcal{J} = (\Delta^{\mathcal{J}}, \cdot^{\mathcal{J}})$ be a full model of $\Sigma(\mathcal{T})$. We construct a full model $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ of $\mathcal{T}$ by taking $\Delta^{\mathcal{I}} = \Delta^{\mathcal{J}}$. Further, for every concept name A and for every atomic role P in $\mathcal{T}$, we define respectively $A^{\mathcal{I}} = A^{\mathcal{J}}$ and[2] $P^{\mathcal{I}} = (P_1^-)^{\mathcal{J}} \circ P_2^{\mathcal{J}}$. Let us show that $\mathcal{I}$ satisfies every assertion in $\mathcal{T}$.

($A \sqsubseteq B$, $A \sqsubseteq \neg B$, **and** $A \sqsubseteq B_1 \sqcup B_2$): The statement easily follows from the construction of $\mathcal{I}$.

($A \sqsubseteq \forall P.B$): Let $o \in A^{\mathcal{I}} = A^{\mathcal{J}}$ and $o' \in \Delta^{\mathcal{I}} = \Delta^{\mathcal{J}}$, such that $(o, o') \in P^{\mathcal{I}}$. Since $P^{\mathcal{I}} = (P_1^-)^{\mathcal{J}} \circ P_2^{\mathcal{J}}$, there is $o'' \in \Delta^{\mathcal{J}}$ such that $(o, o'') \in (P_1^-)^{\mathcal{J}}$, and $(o'', o') \in P_2^{\mathcal{J}}$. Then, $o'' \in C_P^{\mathcal{J}} = C_{P_{AB}}^{\mathcal{J}} \cup \overline{C}_{P_{AB}}^{\mathcal{J}}$. We claim that $o'' \in C_{P_{AB}}^{\mathcal{J}}$. Suppose otherwise, then there is a unique $o_1 \in \Delta^{\mathcal{J}}$, such that $(o'', o_1) \in P_{\bar{A}B1}^{\mathcal{J}}$ and $o_1 \in \bar{A}_{P_B}^{\mathcal{J}}$. It follows from $P_{\bar{A}B1}^{\mathcal{J}} \subseteq P_1^{\mathcal{J}}$ and by the multiplicity constraint over C_P, that $o_1 = o$. This rises a contradiction, because $o \in A^{\mathcal{J}} \subseteq A_{P_B}^{\mathcal{J}}$ and, $A_{P_B}^{\mathcal{J}}$ and $\bar{A}_{P_B}^{\mathcal{J}}$ are disjoint. Then $o'' \in C_{P_{AB}}^{\mathcal{J}}$. Further, there is a unique $o_2 \in \Delta^{\mathcal{J}}$ with $(o'', o_2) \in P_{AB2}^{\mathcal{J}}$ and $o_2 \in B^{\mathcal{J}}$. From $P_{AB2}^{\mathcal{J}} \subseteq P_2^{\mathcal{J}}$ and the multiplicity constraint on C_P, it follows that $o_2 = o'$. Thus, we have that $o' \in B^{\mathcal{J}} = B^{\mathcal{I}}$, and therefore, $o \in (\forall P.B)^{\mathcal{I}}$.

($A \sqsubseteq \exists P.B$): Let $o \in A^{\mathcal{I}} = A^{\mathcal{J}}$. Then, there is $o' \in \Delta^{\mathcal{J}}$ such that $(o', o) \in P_{AB1}^{\mathcal{J}}$ and $o' \in C_{P_{AB}}^{\mathcal{J}}$. Then, there is $o'' \in \Delta^{\mathcal{J}}$ with $(o', o'') \in P_{AB2}^{\mathcal{J}}$ and $o'' \in B^{\mathcal{J}} = B^{\mathcal{I}}$. Then, since $P_{AB2}^{\mathcal{J}} \subseteq P_2^{\mathcal{J}}$, $P_{AB1}^{\mathcal{J}} \subseteq P_1^{\mathcal{J}}$ and $P^{\mathcal{I}} = (P_1^-)^{\mathcal{J}} \circ P_2^{\mathcal{J}}$, we can conclude that $(o, o'') \in P^{\mathcal{I}}$.

($\Rightarrow$) Let $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ be a full model of $\mathcal{T}$, and let $role(\mathcal{T})$ be the set of role names in $\mathcal{T}$. Extend $\mathcal{I}$ to a legal instantiation $\mathcal{J} = (\Delta^{\mathcal{J}}, \cdot^{\mathcal{J}})$ of $\Sigma(\mathcal{T})$, by assigning suitable extensions to the auxiliary classes and associations in $\Sigma(\mathcal{T})$. Let $\Delta^{\mathcal{J}} = \Delta^{\mathcal{I}} \cup \Gamma \cup \Lambda$, where: $\Lambda = \biguplus_{A \sqsubseteq \forall P.B \in \mathcal{T}} \{a_{AP_B}, a_{\bar{A}P_B}\}$, such that $\Delta^{\mathcal{I}} \cap \Lambda = \emptyset$, and $\Gamma = \biguplus_{P \in role(\mathcal{T})} \Delta_P$, with:

$$\Delta_P = P^{\mathcal{I}} \cup \bigcup_{A \sqsubseteq \forall P.B \in \mathcal{T}} \{(a_{AP_B}, b), (a_{\bar{A}P_B}, \bar{o})\}$$

with b an arbitrary instance of B, and $\bar{o}$ an arbitrary element of $\Delta^{\mathcal{I}}$. We set $O^{\mathcal{J}} = \Delta^{\mathcal{I}} \cup \Lambda$, $A^{\mathcal{J}} = A^{\mathcal{I}}$ for each class A corresponding to an atomic concept in $\mathcal{T}$, and $C_P^{\mathcal{J}} = \Delta_P$ for each $P \in role(\mathcal{T})$. Additionally, the extensions of the associations P_1 and P_2 are defined as follows:

$$P_1^{\mathcal{J}} = \{((o, o'), o) \mid (o, o') \in C_P^{\mathcal{J}}\}, \qquad P_2^{\mathcal{J}} = \{((o, o'), o') \mid (o, o') \in C_P^{\mathcal{J}}\}.$$

We now show that $\mathcal{J}$ is a full model of $\Sigma(\mathcal{T})$.

---

[2] We use $r_1 \circ r_2$ to denote the composition of two binary relations r_1 and r_2.

1. For the portions of $\Sigma(\mathcal{T})$ due to TBox assertions of the form $A \sqsubseteq B$, $A \sqsubseteq \neg B$, and $A \sqsubseteq B_1 \sqcup B_2$, the statement follows from the construction of $\mathcal{J}$.
2. For each TBox assertion in $\mathcal{T}$ of the form $A \sqsubseteq \forall P.\, B$, let us define

$$
\begin{aligned}
&A_{P_B}^{\mathcal{J}} = A^{\mathcal{I}} \cup \{a_{A_{P_B}}\}, && \bar{A}_{P_B}^{\mathcal{J}} = O^{\mathcal{J}} \setminus A_{P_B}^{\mathcal{J}}, \\
&C_{P_{AB}}^{\mathcal{J}} = \{(o,o') \in C_P^{\mathcal{J}} \mid o \in A_{P_B}^{\mathcal{J}}\}, && \overline{C}_{P_{AB}}^{\mathcal{J}} = \{(o,o') \in C_P^{\mathcal{J}} \mid o \in \bar{A}_{P_B}^{\mathcal{J}}\}, \\
&P_{AB1}^{\mathcal{J}} = \{((o,o'),o) \in P_1^{\mathcal{J}} \mid o \in A_{P_B}^{\mathcal{J}}\}, && P_{\bar{A}B1}^{\mathcal{J}} = \{((o,o'),o) \in P_1^{\mathcal{J}} \mid o \in \bar{A}_{P_B}^{\mathcal{J}}\}, \\
&P_{AB2}^{\mathcal{J}} = \{((o,o'),o') \in P_2^{\mathcal{J}} \mid o \in A_{P_B}^{\mathcal{J}}\}.
\end{aligned}
$$

It is not difficult to see that $\mathcal{J}$ satisfies the fragment of $\Sigma(\mathcal{T})$ as shown in Fig. 3. Further, it is clear that the extension of the classes that encode atomic concepts in $\mathcal{T}$ are non-empty. For the classes A_{P_B}, $\bar{A}_{P_B}$, $C_{P_{AB}}$, and $\overline{C}_{P_{AB}}$ we have that

$$
a_{A_{P_B}} \in A_{P_B}^{\mathcal{J}}, \quad a_{\bar{A}_{P_B}} \in \bar{A}_{P_B}^{\mathcal{J}}, \quad (a_{A_{P_B}},b) \in C_{P_{AB}}^{\mathcal{J}}, \quad (a_{\bar{A}_{P_B}},\bar{o}) \in \overline{C}_{P_{AB}}^{\mathcal{J}}.
$$

For the associations P_1, P_2, P_{AB1}, P_{AB2} and $P_{\bar{A}B1}$ we have that

$$
\begin{aligned}
&((a_{A_{P_B}},b),a_{A_{P_B}}) \in P_{AB1}^{\mathcal{J}} \subseteq P_1^{\mathcal{J}}, && ((a_{\bar{A}_{P_B}},\bar{o}),a_{\bar{A}_{P_B}}) \in P_{\bar{A}B1}^{\mathcal{J}}, \\
&((a_{A_{P_B}},b),b) \in P_{AB2}^{\mathcal{J}} \subseteq P_2^{\mathcal{J}}.
\end{aligned}
$$

3. For each TBox assertion in $\mathcal{T}$ of the form $A \sqsubseteq \exists P.B$, let us define the extensions for the *auxiliary* classes and associations as follows:

$$
\begin{aligned}
C_{P_{AB}}^{\mathcal{J}} &= \{(o,o') \in C_P^{\mathcal{J}} \mid o \in A^{\mathcal{I}} \text{ and } o' \in B^{\mathcal{I}}\}, \\
P_{AB1}^{\mathcal{J}} &= \{((o,o'),o) \in P_1^{\mathcal{J}} \mid (o,o') \in C_{P_{AB}}^{\mathcal{J}}\}, \\
P_{AB2}^{\mathcal{J}} &= \{((o,o'),o') \in P_2^{\mathcal{J}} \mid (o,o') \in C_{P_{AB}}^{\mathcal{J}}\}.
\end{aligned}
$$

We have that $C_{P_{AB}}^{\mathcal{J}} \neq \emptyset$ as there exists a pair $(a,b) \in \Delta_P$ with $a \in A^{\mathcal{I}}$, and $b \in B^{\mathcal{I}}$. Since $C_{P_{AB}}^{\mathcal{J}} \neq \emptyset$, we have that $P_{AB1}^{\mathcal{J}} \neq \emptyset$ and $P_{AB2}^{\mathcal{J}} \neq \emptyset$. $\qquad\square$

Theorem 7. *Full satisfiability of UCDs is* ExpTime*-complete.*

Proof. We establish the upper bound by a reduction to class satisfiability in UCDs, which is known to be ExpTime-complete [4]. Given a UCD $\mathcal{D}$, with classes $C_1, \ldots, C_n$, we construct the UCD $\mathcal{D}'$ by adding to $\mathcal{D}$ a new class $C_\top$ and new associations R_i, for $i \in \{1, \ldots, n\}$. Furthermore, to check that every association is populated we use reification, i.e., we replace each association P in the diagram $\mathcal{D}$ between the classes C_i and C_j (such that neither C_i nor C_j is constrained to participate at least once to P) with a class C_P and two functional associations P_1 and P_2 to represent each component of P. Finally, we add the constraints shown in Fig. 5. Intuitively, we have that if there is a model $\mathcal{I}$ of the extended diagram $\mathcal{D}'$ in which $C_\top^{\mathcal{I}} \neq \emptyset$, then the multiplicity constraint $1..*$ on the association R_P forces the existence of at least one instance o of C_P. By the functionality of P_1 and P_2 there are at least to elements o_i and o_j, such that $o_i \in C_i^{\mathcal{I}}$, $o_j \in C_j^{\mathcal{I}}$, $(o,o_i) \in P_1^{\mathcal{I}}$ and $(o,o_j) \in P_2^{\mathcal{I}}$. Then, one instance of P can be the pair (o_i,o_j). Conversely, if there is a full model $\mathcal{J}$ of $\mathcal{D}$, it is easy to extend it to a model $\mathcal{I}$ of $\mathcal{D}'$ that satisfies $C_\top$.

The ExpTime-hardness follows from Lemma 6 and Theorem 4. $\qquad\square$

Fig. 5. Reducing UML full satisfiability to class satisfiability

4   Full Satisfiability of Restricted UML Class Diagrams

In this section, we investigate the complexity of the full satisfiability problem for two sub-languages: UML_{bool}, which disallows ISA between associations and UML_{ref}, where also completeness between classes is forbidden. By building on the techniques used for the satisfiability proofs in [1], we show that also in this case checking for full satisfiability does not change the complexity of the problem.

We first show that deciding full satisfiability for UML_{bool} diagrams is NP-complete. For the lower bound, we provide a polynomial reduction of the 3SAT problem (which is known to be NP-complete) to full satisfiability of UML_{bool} CDs.

Let an instance of 3SAT be given by a set $\phi = \{c_1, \ldots, c_m\}$ of 3-clauses over a finite set Π of propositional variables. Each clause is such that $c_i = \ell_i^1 \vee \ell_i^2 \vee \ell_i^3$, for $i \in \{1, \ldots, m\}$, where each ℓ_j^k is a literal, i.e., a variable or its negation. We construct an UML_{bool} diagram $\mathcal{D}_\phi$ as follows: $\mathcal{D}_\phi$ contains the classes $C_\phi, C_\top$, one class C_i for each clause $c_i \in \phi$, and two classes C_p and $C_{\neg p}$ for each variable $p \in \Pi$. To describe the constraints imposed by $\mathcal{D}_\phi$, we provide the corresponding DL inclusion assertions, since they are more compact to write than an UCD. For every $i \in \{1, \ldots, m\}$, $j \in \{1, 2, 3\}$, and $p \in \Pi$, we have the assertions

$$
\begin{array}{lll}
C_\phi \sqsubseteq C_\top, & C_i \sqsubseteq C_\top, & C_{l_i^j} \sqsubseteq C_i, \\
C_p \sqsubseteq C_\top, & C_\phi \sqsubseteq C_i, & C_i \sqsubseteq C_{\ell_i^1} \sqcup C_{\ell_i^2} \sqcup C_{\ell_i^3}, \\
C_{\neg p} \sqsubseteq C_\top, & C_\top \sqsubseteq C_p \sqcup C_{\neg p}, & C_{\neg p} \sqsubseteq \neg C_p.
\end{array}
$$

Clearly, the size of $\mathcal{D}_\phi$ is polynomial in the size of ϕ.

Lemma 8. *A set ϕ of 3-clauses is satisfiable if and only if the UML_{bool} class diagram $\mathcal{D}_\phi$, constructed as above, is fully satisfiable.*

Proof. ($\Rightarrow$) Let $\mathcal{J} \models \phi$. Define an interpretation $\mathcal{I} = (\{0, 1\}, \cdot^\mathcal{I})$, with

$$
C_\top^\mathcal{I} = \{0, 1\}
$$

$$
C_\ell^\mathcal{I} = \begin{cases} \{1\}, & \text{if } \mathcal{J} \models \ell \\ \{0\}, & \text{otherwise} \end{cases} \qquad
\begin{array}{l}
C_i^\mathcal{I} = C_{\ell_i^1}^\mathcal{I} \cup C_{\ell_i^2}^\mathcal{I} \cup C_{\ell_i^3}^\mathcal{I}, \quad \text{for } c_i = \ell_i^1 \vee \ell_i^2 \vee \ell_i^3 \\
C_\phi^\mathcal{I} = C_1^\mathcal{I} \cap \cdots \cap C_m^\mathcal{I}.
\end{array}
$$

Clearly, $C^\mathcal{I} \neq \emptyset$ for every class C representing a clause or a literal, and for $C = C_\top$. Moreover, as at least one literal ℓ_i^j in each clause is such that $\mathcal{J} \models \ell_i^j$, then $1 \in C_i^\mathcal{I}$ for every $i \in \{1, \ldots, m\}$, and therefore $1 \in C_\phi^\mathcal{I}$. It is straightforward to check that $\mathcal{I}$ satisfies $\mathcal{T}$.

($\Leftarrow$) Let $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ be a full model of $\mathcal{D}_\phi$. We construct a model $\mathcal{J}$ of ϕ by taking an element $o \in C_\phi^{\mathcal{I}}$, and setting, for every variable $p \in \Pi$, $\mathcal{J} \models p$ if and only if $o \in C_p^{\mathcal{I}}$. Let us show that $\mathcal{J} \models \phi$. Indeed, for each $i \in \{1, \ldots, m\}$, since $o \in C_\phi^{\mathcal{I}}$ and by the generalization $C_\phi \sqsubseteq C_i$, we have that $o \in C_i^{\mathcal{I}}$, and by the completeness constraint $C_i \sqsubseteq C_{\ell_i^1} \sqcup C_{\ell_i^2} \sqcup C_{\ell_i^3}$, there is some $j_i \in \{1, 2, 3\}$ such that $o \in C_{\ell_i^{j_i}}$. If $\ell_i^{j_i}$ is a variable, then $\mathcal{J} \models \ell_i^{j_i}$ by construction, and thus $\mathcal{J} \models c_i$. Otherwise, if $\ell_i^{j_i} = \neg p$ for some variable p, then, by the disjointness constraint $C_{\neg p} \sqsubseteq \neg C_p$, we have that $o \notin C_p^{\mathcal{I}}$. Thus, $\mathcal{J} \models \neg p$, and therefore, $\mathcal{J} \models c_i$. $\square$

Theorem 9. *Full satisfiability of UML_{bool} is* NP-*complete*

Proof. The NP-hardness follows from Lemma 8. To prove the NP upper bound, we reduce full satisfiability to class satisfiability, which, for the case of UML_{bool}, is known to be in NP [1]. We use a similar encoding as the one used in the proof of Theorem 7 (see Fig. 5). $\square$

We turn now to UML_{ref} class diagrams and show that full satisfiability in this case is NLogSpace-complete. We provide a reduction of the REACHABIL-ITY problem on (acyclic) directed graphs, which is known to be NLogSpace-complete (see e.g., [14]) to the complement of full satisfiability of UML_{ref} CDs.

Let $G = (V, E, s, t)$ be an instance of REACHABILITY, where V is a set of vertices, $E \subseteq V \times V$ is a set of directed edges, s is the start vertex, and t the terminal vertex. We construct an UML_{ref} diagram $\mathcal{D}_G$ from G as follows:

- $\mathcal{D}_G$ has two classes C_v^1 and C_v^2, for each vertex $v \in V \setminus \{s\}$, and one class C_s corresponding to the start vertex s.
- For each edge $(u, v) \in E$ with $u \neq s$ and $v \neq s$, $\mathcal{D}_G$ contains the following constraints (again expressed as DL inclusion assertions):

$$C_u^1 \sqsubseteq C_v^1, \qquad\qquad C_u^2 \sqsubseteq C_v^2.$$

- For each edge $(s, v) \in E$, $\mathcal{D}_G$ contains the following constraints:

$$C_s \sqsubseteq C_v^1, \qquad\qquad C_s \sqsubseteq C_v^2.$$

- For each edge $(u, s) \in E$, $\mathcal{D}_G$ contains the following constraints:

$$C_u^1 \sqsubseteq C_s, \qquad\qquad C_u^2 \sqsubseteq C_s.$$

- The classes C_t^1 and C_t^2 are constrained to be disjoint in $\mathcal{D}$, expressed by:

$$C_t^1 \sqsubseteq \neg C_t^2.$$

The following lemma establishes the correctness of the reduction.

Lemma 10. *t is reachable from s in G iff $\mathcal{D}_G$ is not fully satisfiable.*

Proof. ($\Rightarrow$) Let $\pi = v_1, \ldots, v_n$ be a path in G with $v_1 = s$ and $v_n = t$. We claim that the class C_s in the constructed diagram $\mathcal{D}_G$ is unsatisfiable. Suppose otherwise, that there is a model $\mathcal{I}$ of $\mathcal{D}_G$ with $o \in C_s^{\mathcal{I}}$, for some $o \in \Delta^{\mathcal{I}}$. From π, the construction yields a number of generalization constraints in $\mathcal{D}_G$ such that the following holds:

$$C_s^{\mathcal{I}} \subseteq \cdots \subseteq C_t^{1\,\mathcal{I}} \qquad\qquad\qquad C_s^{\mathcal{I}} \subseteq \cdots \subseteq C_t^{2\,\mathcal{I}}$$

From this we obtain that $o \in (C_t^1)^{\mathcal{I}}$ and $o \in (C_t^2)^{\mathcal{I}}$, which violates the disjointness between the classes C_t^1 and C_t^2, in contradiction to $\mathcal{I}$ being a model of $\mathcal{D}_G$. Hence, C_s is unsatisfiable, and therefore $\mathcal{D}_G$ is not fully satisfiable.

($\Leftarrow$) Assume that t is not reachable from s in G. We construct a full model $\mathcal{I}$ of $\mathcal{D}_G$. Let $\Delta^{\mathcal{I}} = \{d_s\} \cup \bigcup_{v \in V \setminus \{s\}} \{d_v^1, d_v^2\}$. Define inductively a sequence of interpretations as follows:

$\mathcal{I}^0 := \left(\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}^0}\right)$, such that:

$$C_s^{\mathcal{I}^0} := \{d_s\}, \quad C_v^{i\,\mathcal{I}^0} := \{d_v^i\}, \, \forall i \in \{1, 2\}, v \in V \setminus \{s\}.$$

$\mathcal{I}^{n+1} := \left(\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}^{n+1}}\right)$, such that:

$$C_s^{\mathcal{I}^{n+1}} := C_s^{\mathcal{I}^n} \cup \bigcup_{(u,s) \in E} \left(C_u^{1\,\mathcal{I}^n} \cup C_u^{2\,\mathcal{I}^n}\right)$$

$$C_v^{i\,\mathcal{I}^{n+1}} := C_v^{i\,\mathcal{I}^n} \cup \bigcup_{(u,v) \in E, \, u \neq s} C_u^{i\,\mathcal{I}^n} \cup \bigcup_{(s,v) \in E} C_s^{\mathcal{I}^n}$$

The definition induces a monotone operator over a complete lattice, and hence it has a fixed point. Let $\mathcal{I}$ be defined by such a fixed point. It is easy to check that $\mathcal{I}$ is such that for all $i \in \{1, 2\}$, and $u, v \in V \setminus \{s\}$ the following holds:

1. For each class C_v^i, we have that $d_v^i \in C_v^{i\,\mathcal{I}}$.
2. $d_s \in C_s^{\mathcal{I}}$.
3. For all $d \in \Delta^{\mathcal{I}}$, $d \in C_u^{i\,\mathcal{I}}$ implies $d \in C_v^{i\,\mathcal{I}}$ iff v is reachable from u in G.
4. For all $d_u^i \in \Delta^{\mathcal{I}}$, $d_u^i \in C_v^{j\,\mathcal{I}}$ for $i \neq j$ iff s is reachable from u in G, and v is reachable from s in G.
5. $d_s \in C_v^{i\,\mathcal{I}}$ iff v is reachable from s in G.

From (1) and (2) we have that all classes in $\mathcal{D}_G$ are populated in $\mathcal{I}$. It remains to show that $\mathcal{I}$ satisfies $\mathcal{D}_G$. A generalization between the classes C_u^i and C_v^i corresponds to the edge $(u, v) \in E$. This means that v is reachable from u in G, and therefore, by (3) we have that $C_u^{i\,\mathcal{I}} \subseteq C_v^{i\,\mathcal{I}}$. A similar argument holds for generalizations involving the class C_s. Furthermore, the classes C_t^1 and C_t^2 are disjoint under $\mathcal{I}$. To show this, suppose that there is an element $d \in \Delta^{\mathcal{I}}$ such that $d \in C_t^{1\,\mathcal{I}} \cap C_t^{2\,\mathcal{I}}$. Then by (5), $d \neq d_s$, as t is not reachable from s. Moreover, $d \neq d_v^i$ for all $i \in \{1, 2\}$ and $v \in V \setminus \{s\}$. Indeed, suppose w.l.o.g. that $i = 1$. Then, by (4), $d_v^1 \in C_t^{2\,\mathcal{I}}$ iff s is reachable from v, and t is reachable from s, which leads to a contradiction. Hence, $C_t^{1\,\mathcal{I}} \cap C_t^{2\,\mathcal{I}} = \emptyset$. $\qquad\square$

Language	Classes			Associations			Complexity
	ISA	disjoint	complete	ISA	multiplicity	refinement	
UML	✓	✓	✓	✓	✓	✓	EXPTIME
UML$_{bool}$	✓	✓	✓	✗	✓	✓	NP
UML$_{ref}$	✓	✓	✗	✗	✓	✓	NLOGSPACE

Table 1. Complexity results for full satisfiability in UML

Theorem 11. *Full-satisfiability of UML$_{ref}$ class diagrams is* NLOGSPACE-*complete.*

Proof. The NLOGSPACE membership follows from the NLOGSPACE membership of class satisfiability [1], and a reduction similar to the one used in Theorem 9. Since NLOGSPACE = CONLOGSPACE (by the Immerman-Szelepcsényi theorem; see, e.g., [14]), and as the above reduction is logspace bounded, it follows that full consistency of UML$_{ref}$ class diagrams is NLOGSPACE-hard. □

5 Conclusions

This paper investigates the problem of *full satisfiability* in the context of UML class diagrams, i.e., whether there is at least one model of the diagram where each class and association is non-empty. Our results (reported in Table 1) show that the complexity of full satisfiability matches the complexity of the classical class diagram satisfiability check. We show a similar result also for the problem of checking the full satisfiability of a TBox expressed in the description logic $\mathcal{ALC}$.

References

1. A. Artale, D. Calvanese, R. Kontchakov, V. Ryzhikov, and M. Zakharyaschev. Reasoning over extended ER models. In *Proc. of the 26th Int. Conf. on Conceptual Modeling (ER 2007)*, volume 4801 of *Lecture Notes in Computer Science*, pages 277–292. Springer, 2007.
2. A. Artale, D. Calvanese, R. Kontchakov, and M. Zakharyaschev. The *DL-Lite* family and relations. *J. of Artificial Intelligence Research*, 36:1–69, 2009.
3. 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.
4. D. Berardi, D. Calvanese, and G. De Giacomo. Reasoning on UML class diagrams. *Artificial Intelligence*, 168(1–2):70–118, 2005.
5. A. Borgida. Description logics in data management. *IEEE Trans. on Knowledge and Data Engineering*, 7(5):671–682, 1995.
6. M. Buchheit, F. M. Donini, and A. Schaerf. Decidable reasoning in terminological knowledge representation systems. *J. of Artificial Intelligence Research*, 1:109–138, 1993.

7. D. Calvanese, M. Lenzerini, and D. Nardi. Unifying class-based representation formalisms. *J. of Artificial Intelligence Research*, 11:199–240, 1999.

8. T. Clark and A. S. Evans. Foundations of the Unified Modeling Language. In D. Duke and A. Evans, editors, *Proc. of the 2nd Northern Formal Methods Workshop*. Springer, 1997.

9. A. Evans, R. France, K. Lano, and B. Rumpe. Meta-modelling semantics of UML. In H. Kilov, editor, *Behavioural Specifications for Businesses and Systems*, chapter 2. Kluwer Academic Publishers, 1999.

10. D. Harel and B. Rumpe. Modeling languages: Syntax, semantics and all that stuff. Technical Report MCS00-16, The Weizmann Institute of Science, Rehovot, Israel, 2000.

11. M. Jarrar and S. Heymans. Towards pattern-based reasoning for friendly ontology debugging. *Int. J. on Artificial Intelligence Tools*, 17(4):607–634, 2008.

12. K. Kaneiwa and K. Satoh. Consistency checking algorithms for restricted UML class diagrams. In *Proc. of the 4th Int. Symp. on Foundations of Information and Knowledge Systems (FoIKS 2006)*, pages 219–239, 2006.

13. M. Lenzerini and P. Nobili. On the satisfiability of dependency constraints in entity-relationship schemata. *Information Systems*, 15(4):453–461, 1990.

14. C. H. Papadimitriou. *Computational Complexity*. Addison Wesley Publ. Co., 1994.

Second-Order Description Logics: Semantics, Motivation, and a Calculus

Simona Colucci[1], Tommaso Di Noia[1], Eugenio Di Sciascio[1], Francesco M. Donini[2], Azzurra Ragone[1]

[1]: SisInfLab & DEE, Politecnico di Bari, Bari, Italy
[2]: DISCOM, Università della Tuscia, Viterbo, Italy

1 Introduction

Eight years ago, Tim Berners-Lee, James Hendler and Ora Lassila published their seminal paper [6] describing the evolution of the current web from a human-processable environment to a machine-processable one. The basic idea was to annotate web resources and give them a machine-processable meaning; the Semantic Web was born. Many efforts have been placed in the last years by the Semantic Web community in the attempt to standardize both the language for representing the content of web resources and the production of annotations/metadata. On the one hand, such efforts successfully led to the affirmation of standard languages for machine-processable representation of web pages content, like the recent W3C recommendation OWL2. On the other hand, it produced the Linked Data initiative: a set of best practices for publishing and connecting data on the Web. These two initiatives are tightly connected. In fact, data published following the Linked Data best practices are interpreted thanks to the ontological layer developed using OWL2. Despite a large effort in annotating and representing the semantic content of a resource (in a semi-automatic way) we see the lack of reasoning engines able to fully exploit such representation power. During the last years highly optimized reasoning engines have been developed for classical deductive reasoning tasks such as subsumption/classification, consistency checking and instance retrieval. At the same time, non-standard reasoning tasks have been proposed in the Description Logics literature as an answer to new issues related to knowledge-based domains especially in retrieval scenarios, ontology design and maintenance and automated negotiation. The most relevant reasoning tasks we may cite are: explanation [18], interpolation [23], concept abduction and concept contraction [10], concept unification [3], concept difference [25], concept similarity [8], concept rewriting [2], negotiation [22], least common subsumer [5], most specific concept [1] and knowledge base completion [4].

For each of the above mentioned tasks a specific algorithmic approach has been proposed and very often only for a particular (sometimes simple) Description Logic. Although the need for such reasoning tasks has been widely recognized, there is not yet a unified view—at least from an algorithmic perspective. Indeed, some of the above mentioned tasks share some properties from a computational point of view and sometimes are very related to each other. Moreover, most of the problems in the cited reasoning tasks are of the form: "Find one or more concept(s) C such that {sentence involving C }" and we are really interested in exhibiting such a concept, not just proving its ex-

istence. In other words, many of the above mentioned reasoning tasks, known as non-standard reasoning, deal with finding—or constructing—a concept. This is the main reason why we refer to such reasoning as *constructive reasoning*. By contrast, "standard" reasoning is about checking some property (true or false) such as subsumption or satisfiability (also query answering can be reduced to instance checking).

In this paper we propose a new second-order framework and a related calculus able to express, in a uniform way, many of the abovementioned constructive reasoning tasks.

The remainder of the paper is structured as follows: in Section 2 we introduce the framework and its formal semantics. Section 3 is devoted to the reformulation of some relevant contructive reasoning tasks in terms of second order formulas. The general calculus is presented in Section 4, before providing a section on "discussion and future directions".

2   Semantics

We denote by $\mathcal{DL}$ a generic Description Logic. Only in order to exemplify our framework, consider the presentation of the DL $\mathcal{SHIQ}$.

Let N_r be a set of role names. A *general role* R can be either a role name $P \in \mathsf{N}_r$, or its inverse, denoted by P^-. We admit a set of *role axioms*, formed by: (1) a *role hierarchy* $\mathcal{H}$, which is a set of role inclusions of the form $R_1 \sqsubseteq R_2$, and (2) a set of transitivity axioms for roles, denoted by $\mathtt{Trans}(R)$. We denote by $\sqsubseteq^*$ the transitive-reflexive closure of $\mathcal{H} \cup \{R^- \sqsubseteq S^- \mid S \sqsubseteq R \in \mathcal{H}\}$. A role S is *simple* if it is not transitive, and for no R such that $R \sqsubseteq^* S$, R is transitive.

In the following syntax for concepts, let A be a generic concept name in a set N_c of concept names.

$$C \longrightarrow \top \mid \bot \mid A \mid \geqslant n\,S.C \mid \leqslant n\,S.C \mid C_1 \sqcap C_2 \mid \neg C \tag{1}$$

We consider the other well-known constructs as abbreviations: $C_1 \sqcup C_2 = \neg(\neg(C_1) \sqcap \neg(C_2))$, $\exists R.C = {\geqslant}\,1\,R.C$, $\forall R.C = {\leqslant}\,0\,R.\neg C$. For computability reasons, only in $\exists R.C, \forall R.C$ the role R can be a general role (*i.e.*, also a transitive role, or a super-role of a transitive role), while in other number restrictions R must be a *simple* role.

Every DL is equipped with a model-theoretic semantics. Again, exemplifying our discussion for $\mathcal{SHIQ}$, an *interpretation* $\mathcal{I}$ is a pair $\langle \Delta^{\mathcal{I}}, \cdot^{\mathcal{I}} \rangle$ where $\Delta^{\mathcal{I}}$ is a set of individuals, and $\cdot^{\mathcal{I}}$ is an interpretation function mapping $\top$ into $\Delta^{\mathcal{I}}$, $\bot$ into $\emptyset$, each concept name $A \in \mathsf{N}_c$ into a subset of $\Delta^{\mathcal{I}}$, and each role name $P \in \mathsf{N}_r$ into a subset of $\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$, and extended to concept and role expressions as follows (let $\sharp\{\ldots\}$ denote the cardinality of a set):

$$\neg C^{\mathcal{I}} = \Delta^{\mathcal{I}} - A^{\mathcal{I}} \tag{2}$$

$$\geqslant n\,R.C^{\mathcal{I}} = \{a \in \Delta^{\mathcal{I}} \mid \sharp\{b \in \Delta^{\mathcal{I}} \mid \langle a,b \rangle \in R^{\mathcal{I}} \wedge b \in C^{\mathcal{I}}\} \geqslant n\} \tag{3}$$

$$\leqslant n\,R.C^{\mathcal{I}} = \{a \in \Delta^{\mathcal{I}} \mid \sharp\{b \in \Delta^{\mathcal{I}} \mid \langle a,b \rangle \in R^{\mathcal{I}} \wedge b \in C^{\mathcal{I}}\} \leqslant n\} \tag{4}$$

$$(C_1 \sqcap C_2)^{\mathcal{I}} = (C_1)^{\mathcal{I}} \cap (C_2)^{\mathcal{I}} \tag{5}$$

$$(P^-)^{\mathcal{I}} = \{\langle b,a \rangle \in \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}} \mid \langle a,b \rangle \in P^{\mathcal{I}}\} \tag{6}$$

As usual, we denote by $C \sqsubseteq D$ the proposition "for every interpretation $\mathcal{I}$ (satisfying role axioms), $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$". We also denote *non-subsumption* by $C \not\sqsubseteq D$, meaning the proposition "there exists an interpretation $\mathcal{I}$ satisfying role axioms such that $C^{\mathcal{I}} \not\subseteq D^{\mathcal{I}}$". Observe that $C \sqsubseteq D$, $C \not\sqsubseteq D$ are propositions (true or false), so they can be combined by $\wedge, \vee$ in a propositional formula Γ. We say that Γ is *true* iff the composition of truth values of subsumptions and non-subsumptions yields *true*.

Second-order Concept Expressions. In order to write second-order formulas, we need a set $\mathsf{N}_x = \{X_0, X_1, X_2, \ldots\}$ of concept variables, which we can quantify over.

A *concept term* is a concept formed according to the rules in (1) plus the rule $C \longrightarrow X$ for $X \in \mathsf{N}_x$. For example, $A \sqcap X_0 \sqcap \forall (P^-).(X_1 \sqcap \exists Q.X_2)$ is a concept term. Although also role variables could be conceived, we do not need them here. We stress the fact that concept terms could be defined starting from the syntax of every Description Logic $\mathcal{DL}$, not just $\mathcal{SHIQ}$. We denote by $\mathcal{DL}_X$ the language of concept terms obtained from $\mathcal{DL}$ by adding concept variables.

We use *general semantics* [15]—also acknowledged as Henkin structures [27]—for interpreting concept variables. In such a semantics, variables denoting unary predicates can be interpreted only by *some subsets* among all the ones in the powerset of the domain $2^{\Delta^{\mathcal{I}}}$—instead, in standard semantics a concept variable could be interpreted as any subset of $\Delta^{\mathcal{I}}$. Note that Baader and Narendran [3] use standard semantics in their paper on concept unification.

Adapting general semantics to our problem, the structure we consider is exactly the sets interpreting concepts in $\mathcal{DL}$. That is, the interpretation $X^{\mathcal{I}}$ of a concept variable X must coincide with the interpretation $E^{\mathcal{I}}$ of some concept $E \in \mathcal{DL}$. Moreover, since we are interested in particular existential second-order formulas, we limit our definition to such formulas.

Definition 1 (General Semantics). *Let $C_1, \ldots, C_m, D_1, \ldots, D_m \in \mathcal{DL}$ be concept terms containing concept variables $X_0, X_1, \ldots, X_n$, and let Γ be a conjunction of concept subsumptions and non-subsumptions, of the form*

$$\Gamma = (C_1 \sqsubseteq D_1) \wedge \cdots \wedge (C_\ell \sqsubseteq D_\ell) \wedge (C_{\ell+1} \not\sqsubseteq D_{\ell+1}) \wedge \cdots \wedge (C_m \not\sqsubseteq D_m) \quad (7)$$

for $1 \leq \ell \leq m$. We say that Γ is satisfiable *in $\mathcal{DL}$ if and only if there exist $n+1$ concepts $E_1, \ldots, E_n \in \mathcal{DL}$ such that, extending the semantics (2)–(6) for each interpretation $\mathcal{I}$, with: $(X_i)^{\mathcal{I}} = (E_i)^{\mathcal{I}}$ for $i = 0, \ldots, n$, it holds that*

1. *for every $j = 1, \ldots, \ell$, and for every interpretation $\mathcal{I}$, $(C_j)^{\mathcal{I}} \subseteq (D_j)^{\mathcal{I}}$ and*
2. *for every $j = \ell+1, \ldots, m$, there exists an interpretation $\mathcal{I}$ s.t. $(C_j)^{\mathcal{I}} \not\subseteq (D_j)^{\mathcal{I}}$.*

Otherwise, Γ is said to be unsatisfiable *in $\mathcal{DL}$. Moreover, we say that the formula*

$$\exists X_0 \cdots \exists X_n.\Gamma \quad (8)$$

is true *in $\mathcal{DL}$ if Γ is satisfiable in $\mathcal{DL}$, otherwise it is* false.

Note that we are considering here only a particular form of closed second-order formulas in Description Logics. This is because we are not interested here in Second-order Description Logics by themselves, but only in their use to express and compute the "constructive" reasoning services of the next section.

3   Modeling Constructive Reasoning Tasks

Hereafter we show how to model some constructive reasoning tasks in trems of formula (8). In this section we only show the constructive formulation of the task and we leave discussion on optimality criteria at the end of the section. The computation of the Most Specific Concept as well as a Knowledge Base Completion could be easily modeled if we allowed in Γ formulas involving an ABox or a TBox.

We introduce the notion of *signature of a concept* that is used in Interpolation and Concept Unification. Given a concept C we define:

$$sign(C)_{\mathsf{N}_c} = \{A \mid A \in \mathsf{N}_c, A \text{ appears syntactically in } C\}$$
$$sign(C)_{\mathsf{N}_r} = \{P \mid P \in \mathsf{N}_r, P \text{ appears syntactically in } C\}$$
$$sign(C) = sign(C)_{\mathsf{N}_c} \cup sign(C)_{\mathsf{N}_r}$$

Least Common Subsumer. A concept $D \in \mathcal{DL}$ is a Common Subsumer of two concepts $C_1, C_2 \in \mathcal{DL}$ if $(C_1 \sqsubseteq D) \wedge (C_2 \sqsubseteq D)$. The Least Common Subsumer (LCS) of C_1, C_2 is the least element w.r.t. $\sqsubseteq$ of the set of concepts which are Common Subsumers of C_1, C_2 and is unique up to equivalence. A concept L is *not* the Least Common Subsumer of C_1, C_2 iff the following formula (of the form (8)) is true in $\mathcal{DL}$:

$$\exists X.(C_1 \sqsubseteq X) \wedge (C_2 \sqsubseteq X) \wedge (X \sqsubseteq L) \wedge (L \not\sqsubseteq X)$$

that is, L is not the LCS if there exists a concept X which is a Common Subsumer, and is strictly more specific than L. By finding a concept satisfying the above formula, and iterating the process, an algorithm for computing the LCS in a sublanguage of $\mathcal{SHIQ}$ has been proposed [12].

Interpolation. Interpolation have been proposed in Description Logics for different purposes. In [23], the computation of an interpolant is used to explain subsumption, if it exists, between two concepts C and D. Konev *et al.* [16] use the notion of interpolation for a TBox $\mathcal{T}$ in order to *forget* part of the vocabulary adopted in $\mathcal{T}$ and reason on a smaller ontology. Seylan *et al.* [24] need the computation of an interpolant between two concepts to rewrite a query in terms of DBox predicates.

Definition 2 (Interpolation). *Given two concepts C and D in $\mathcal{DL}$ such that $C \sqsubseteq D$, an interpolant of C and D is a concept I such that:*

– *$sign(I) \subseteq sign(C) \cup sign(D)$;*
– *both $C \sqsubseteq I$ and $I \sqsubseteq D$.*

Given two concepts C and D such that $C \sqsubseteq D$, the corresponding interpolant satisfies the formula $(C \sqsubseteq X) \wedge (X \sqsubseteq D)$ of the form (7).

Abduction. Abduction in Description Logics has been recognized as an interesting reasoning procedure for a set of heterogeneous tasks [10, 17, 24, 7, 21]. Here we mainly concentrate on Concept Abduction as defined in [10] and Structural Abduction [11] but the formalization can be easily extended to other abductive procedures [13]. Concept Abduction is a straight adaptation of Propositional Abduction.

Definition 3 (Concept Abduction). *Let C, D, be two concepts in $\mathcal{DL}$ where both C and D are satisfiable. A* Concept Abduction Problem *(CAP) is finding a concept $H \in \mathcal{DL}$ such that $C \sqcap H \not\sqsubseteq \bot$, and $C \sqcap H \sqsubseteq D$.*

Every solution H of a CAP satisfies the formula

$$(C \sqcap X \not\sqsubseteq \bot) \wedge (C \sqcap X \sqsubseteq D)$$

Such solutions can be compared by $\sqsubseteq$, preferring the subsumption-maximal ones, since they are the solutions hypothesizing the least. Moreover, a formula of the form (8) can characterize the complement of being subsumption-maximal. A concept H is *not* a subsumption-maximal solution of a CAP iff the formula is true in $\mathcal{DL}$:

$$\exists X.(C \sqcap X \not\sqsubseteq \bot) \wedge (C \sqcap X \sqsubseteq D) \wedge (H \sqsubseteq X) \wedge (X \not\sqsubseteq H)$$

In order to deal with Abduction for expressive Description Logics, a more fine grained definition of Abduction was introduced in [11] with the name of Structural Abduction. The notion of Structural Abduction relies on the notion of *Adbucible Concept* and *Hypotheses List* we report here for the sake of completeness.

Definition 4 (Abducible Concept – Hypotheses List). *Let C and D be two concepts in $\mathcal{DL}$. We define* abducible concept $C^h \doteq H_0 \sqcap Rew(C)$, *where the rewriting $Rew(C)$ is defined recursively as* $Rew(A) = A$; $Rew(\neg A) = \neg A$; $Rew(C_1 \sqcap C_2) = Rew(C_1) \sqcap Rew(C_2)$; $Rew(C_1 \sqcup C_2) = Rew(C_1) \sqcup Rew(C_2)$; $Rew(\exists R.C) = \exists R.(H_{new} \sqcap Rew(C))$; $Rew(\forall R.C) = \forall R.(H_{new} \sqcap Rew(C))$ *where by H_{new} we mean a concept variable not yet appearing in the rewriting. We call* hypotheses list *of C^h the list $\overline{\mathcal{H}} = \langle H_0, H_1, H_2, \ldots \rangle$.*

Definition 5 (Structural Abduction). *Let $C, D \in \mathcal{DL}$, be two concepts where both C and D are satisfiable $C \sqcap D \not\sqsubseteq \bot$. Let $\overline{\mathcal{H}} = \langle H_0, \ldots, H_\ell \rangle$ be the hypotheses list of the abducible concept C^h and $\tilde{\mathcal{A}} = \langle \mathcal{A}_0, \ldots, \mathcal{A}_\ell \rangle$ (for Assumptions) be a list of $\mathcal{DL}$ concept sets. A* Structural Abduction Problem *(SAP) for $\mathcal{DL}$ is finding a list of concepts $\mathcal{H} = \langle H_0, \ldots, H_\ell \rangle$ such that*

$$H_i \in \mathcal{A}_i \text{ for every } i = 0, \ldots, \ell \tag{9}$$

$$\mathcal{T} \not\models \sigma[\overline{\mathcal{H}}/\mathcal{H}](C^h) \sqsubseteq \bot \tag{10}$$

$$\mathcal{T} \models \sigma[\overline{\mathcal{H}}/\mathcal{H}](C^h) \sqsubseteq D \tag{11}$$

We call a SAP General *when $\mathcal{A}_i = \mathcal{DL}$, for every $i = 0, \ldots, \ell$.*

Let C^x be as C^h with X_i in place of H_i for $i = 0, \ldots, \ell$. Then, $H_0, \ldots, H_\ell$ is a solution of a SAP iff it satisfies the formula $(C^x \not\sqsubseteq \bot) \wedge (C^x \sqsubseteq D)$ by letting $(X_i)^{\mathcal{I}} = (H_i)^{\mathcal{I}}$ for every $\mathcal{I}$ and every $i = 0, \ldots, \ell$.

Concept Contraction. Gärdenfors [14] distinguishes three main kinds of belief changes: (i) *expansion*, (ii) *revision*, (iii) *contraction*. Given two concepts C and D such that $C \sqcap D \sqsubseteq \bot$, Concept Contraction is the DL-based version of contraction.

Definition 6 (Concept Contraction). *Let C, D both satisfiable. A* Concept Contraction Problem *(CCP) is finding a pair of concepts $\langle G, K \rangle$ (Give up, Keep) such that $C \equiv G \sqcap K$, and $K \sqcap D \not\sqsubseteq \bot$. We call K a* contraction *of C according to D.*

Every solution $\langle G, K \rangle$ of a CCP satisfies the formula

$$(C \equiv X_0 \sqcap X_1) \wedge (X_1 \sqcap D \not\sqsubseteq \bot)$$

Such solutions can be compared by $\sqsubseteq$, preferring the ones whose G's are subsumption-maximal, since they are the solutions contracting the least. Moreover, a formula of the form (8) can characterize the non-preferred contractions. A pair $\langle G, K \rangle$ is *not* a preferred solution of a CCP iff the following formula is true in $\mathcal{DL}$:

$$\exists X_0 \exists X_1.(C \equiv X_0 \sqcap X_1) \wedge (X_1 \sqcap D \not\sqsubseteq \bot) \wedge (G \sqsubseteq X_0) \wedge (X_0 \not\sqsubseteq G)$$

Concept Unification. Concept Unification [3] between two concepts C and D arises when one wants to rewrite some concept names occurring in C and D in order to make the relation $C \equiv D$ true.

Definition 7. *Let C and D be two concepts in $\mathcal{DL}$ such that $C \not\equiv D$. We define the two sets $\mathcal{X}^C = \{A_i^C \mid i = 1, \ldots, l\}$ and $\mathcal{X}^D = \{A_j^D \mid j = 1, \ldots, m\}$ such that $\mathcal{X}^C \subseteq sign(C)_{\mathsf{N}_c}$ and $\mathcal{X}^D \subseteq sign(D)_{\mathsf{N}_c}$. A Unification Problem is finding the set of rewriting rules $\mathcal{M}$: $A_1^C \rightarrow C_1; \ldots; A_l^C \rightarrow C_l, A_1^D \rightarrow D_1; \ldots; A_m^D \rightarrow D_m$ such that*

$$
\begin{aligned}
sign(C_i) &\subseteq sign(C) \cup sign(D), \ \textit{with } i = 1, \ldots, l \\
sign(D_j) &\subseteq sign(C) \cup sign(D), \ \textit{with } j = 1, \ldots, m \\
C &\equiv_{\mathcal{M}} D
\end{aligned}
$$

The Unification problem is solvable iff the following formula (of the form (8)) is true in $\mathcal{DL}$:

$$\exists A_1^C, \ldots, A_l^C, A_1^D, \ldots, A_m^D.(C \sqsubseteq D) \wedge (D \sqsubseteq C)$$

treating $\mathcal{X}^C, \mathcal{X}^D$ as concept variables interpreted in General Semantics.

Concept Difference. Following the algebraic approaches adopted in classical information retrieval, Concept Difference [25] was introduced as a way to measure concept similarity.

Definition 8. *Let C and D be two concepts such that $C \sqsubseteq D$. The Concept Difference $C - D$ is defined by $max_{\sqsubseteq}\{B \in \mathcal{DL} \text{ such that } D \sqcap B \equiv C\}$.*

We can use a formula of the form (8) to check whether a concept B is *not* a difference between C and D, namely, B is not a Difference iff the formula below is true:

$$\exists X.(C \sqsubseteq D \sqcap X) \wedge (D \sqcap X \sqsubseteq C) \wedge (X \sqsubseteq B) \wedge (B \not\sqsubseteq X)$$

Negotiation. The aim of a negotiation process is to find an agreement between two competitive parties. Both agreement and requirements from the two parties can be represented as (a conjunction of) concepts [22]. Usually, in a negotiation the parties requirements are in conflict with each other. Hence, in order to reach an agreement they have to give up some parts of their requirements. During a negotiation the two parties have to agree on and follow a protocol (*i.e.*, a set of rules that characterize the specific process). Here we define a simple protocol where given the initial requirements W_0^c and W_0^d, if they are in conflict with each other, then the two parties c and d propose a

relaxed version $\overline{W}^c$ and $\overline{W}^d$ of W_0^c and W_0^d. The final aim of the protocol is to satisfy as much as possible both agents with the final agreement. At the first round they relax their requirements keeping the minimal information they want to be satisfied by the final agreement and propose $W_\top^c$ and $W_\top^d$ such that $W_0^c \sqsubseteq W_\top^c$ and $W_0^d \sqsubseteq W_\top^d$. For each following round i they propose least relaxed version of W_0^c and W_0^d which are more specific of the proposals at round $i - 1$ [1]. The protocol stops either if the max number MAX of rounds has been reached or when it does not exist a concept both more specific than the one found at the previous round and less specific than the initial requirements.

Input: concepts W_0^c, W_0^d such that $W_0^c \sqcap W_0^d \sqsubseteq \bot$
Output: the final outcome of the negotiation after n rounds. If c and d reach an agreement
      the returned value is $\langle W_n^c, W_n^d \rangle$, $NULL$ otherwise.

1 **begin**
2    $W^c = W_\top^c$;
3    $W^d = W_\top^d$;
4    $i = 0$; $flag = continue$;
5   **while** $(i < MAX) \wedge (flag == continue)$ **do**
6   **if** $\exists \overline{W}^c, \overline{W}^d.(\overline{W}^c \sqcap \overline{W}^d \not\sqsubseteq \bot) \wedge (\overline{W}^c \sqsubseteq W^c) \wedge (\overline{W}^d \sqsubseteq W^d) \wedge (W_0^c \sqsubseteq \overline{W}^c) \wedge (W_0^d \sqsubseteq \overline{W}^d)$ **then**
7          $W^c = \overline{W}^c$;
8          $W^d = \overline{W}^d$;
9   **else**
10          $flag = stop$;
11       $i = i + 1$;
12   **if** $flag == stop$ **then**
13   **return** $\langle W^c, W^d \rangle$;
14   **else**
15   **return** $NULL$;
16 **end**

Algorithm 1: A simple negotiation protocol

Optimal Solutions. We may classify the above reasoning tasks into two main categories: tasks for which we *just* need to compute a concept (or a set of concepts) as Concept Unification and Interpolation and those for which we need to find a concept (or a set of concepts) according to some minimality/maximality criteria such as LCS, Concept Difference, Concept Abduction, Concept Contraction and Negotiation. In the first case, we have a set of solutions while in the second one we also have a set of sub-optimal solutions to the main problem. As an example, for LCS we have the set of sub-optimal solution represented by "common subsumers". Based on this observation, we may think of a procedure that computes a sub-optimal solution X^i at step i and then

---

[1] The way $\overline{W}^c$ and $\overline{W}^d$ are computed at each step should take into account also agents' preferences. For the sake of conciseness we omit such details.

iteratively computes a better solution X^{i+1} at the next step. The procedure stops (if decidable) when no better solution can be found according to the minimality/maximality criterion. In case the procedure is not decidable, we may decide to iterate for a maximum number of steps. In this case, the procedure returns a sub-optimal solution to the problem. In other words, this means that we may apply a procedure similar to the negotiation protocol described above to other constructive reasoning every time we need to satisfy optimal criteria.

4   A Calculus

Definition 9 (Substitutions).

1. *Let $\mathcal{DL}$ be a Description Logic, $\{i_1, \ldots, i_k\} \subseteq \{0, 1, \ldots, n\}$ be a set of distinct indexes, $X_{i_1}, \ldots, X_{i_k}$ be concept variables, and $D_{i_1}, \ldots, D_{i_k} \in \mathcal{DL}_X$ be concept terms. A* substitution *σ is a set of pairs $\{[X_{i_1}/D_{i_1}], \ldots, [X_{i_k}/D_{i_k}]\}$. A substitution is* ground *if every D_{i_j} contains no variables, i.e., $D_{i_j} \in \mathcal{DL}$.*
2. *For a concept term $C \in (\mathcal{SHIQ})_X$, we inductively define $\sigma(C)$ as $\sigma(X_i) = D_i$, $\sigma(\neg X_i) = \neg(\sigma(D_i))$, $\sigma(A) = A$, $\sigma(C_1 \sqcap C_2) = \sigma(C_1) \sqcap \sigma(C_2)$, $\sigma(\bowtie nR.C) = \bowtie nR.\sigma(C)$ for $\bowtie \in \{\leqslant, \geqslant\}$.*
3. *For concept terms C, D, we define also $\sigma(C \sqsubseteq D) = \sigma(C) \sqsubseteq \sigma(D)$, $\sigma(C \not\sqsubseteq D) = \sigma(C) \not\sqsubseteq \sigma(D)$, and for a boolean conjunction Γ of the form (7), $\sigma(\Gamma)$ is the result of applying σ to every subsumption and non-subsumption statement.*

By using substitutions, a formula of the form (8) is true according to Def.1 if and only if there exists a ground substitution making it valid, as formalized by the theorem below.

Theorem 1. *A formula $\exists X_0 \cdots \exists X_n.\Gamma$ is true in $\mathcal{DL}$ iff there exists a ground substitution $\sigma = \{[X_0/E_0], \ldots, [X_n/E_n]\}$ with $E_0, \ldots, E_n \in \mathcal{DL}$, such that $\sigma(\Gamma)$ is true.*

Observe that since σ is ground, and substitutes every variable in Γ, $\sigma(\Gamma)$ is just a boolean combination of [non-]subsumptions in $\mathcal{SHIQ}$. Observe also that if *standard* semantics is adopted for concept variables [3] instead of Def.1—that is, if $X^{\mathcal{I}}$ can be any subset of $\Delta^{\mathcal{I}}$—then the "only if" part of the above theorem no longer holds, since there can be statements for which $X^{\mathcal{I}}$ is not expressible in the target $\mathcal{DL}$, yielding no substitution. For example, formula $\exists X.(A \sqsubseteq X) \wedge (B \sqsubseteq X) \wedge (\top \not\sqsubseteq X)$ is false in a DL without $\sqcup$ (disjunction), but it would be true in standard semantics: just let for every $\mathcal{I}$, $X^{\mathcal{I}} = A^{\mathcal{I}} \cup B^{\mathcal{I}}$.

We present now a simple calculus, obtained by combining Analytic Tableaux for ordinary concept constructors, and substitutions for concept variables. Then we prove its soundness and completeness. Again, we present the calculus for the DL $\mathcal{SHIQ}$, but only for sake of clarity; the same framework could be adopted for other DLs. We borrow Tableaux rules (T-rules; see below) from well-known results of Tobies [26]. Since inverse roles are present in $\mathcal{SHIQ}$, we use *pairwise blocking* for individuals [26, p.125].

All rules are applicable only if x is *not blocked*. For each $i = 1, ..., n$, $\mathcal{L}_i$ is a branch in τ_i. Rules above the separating line have precedence over rules below it.

⊓**-rule** : **if** $C \sqcap D \in \mathcal{L}_i(x)$,

    then add both C and D to $\mathcal{L}_i(x)$

⊔**-rule** : **if** $C \sqcup D \in \mathcal{L}_i(x)$,

    then add either C or D to $\mathcal{L}_i(x)$

∀**-rule** : **if** $\forall R.C \in \mathcal{L}_i(x)$, and there exists an individual y such that y is an R-successor of x,

    then add C to $\mathcal{L}_i(y)$

⩽**-rule** : **if** $\leqslant n\, S.C \in \mathcal{L}_i(x)$ with $n \geqslant 1$, and
there are $m > n$ S-neighbors (say) $y_1, \ldots, y_m$ of x with $C \in \mathcal{L}_i(y_j)$ for $j = 1, \ldots, m$,
$y, z \in \{y_1, \ldots, y_m\}$ with y being an S-successor of x and not $y \neq z$

    then (1) add $\mathcal{L}_i(y)$ to $\mathcal{L}_i(z)$,
    (2) for every $R \in \mathcal{L}_i(x, y)$ if z is a predecessor of x then add R^- to $\mathcal{L}_i(z, x)$ else add R to $\mathcal{L}_i(x, z)$,
    (3) let $\mathcal{L}_i(x, y) = \emptyset$, and
    (4) for all u with $u \neq y$, set $u \neq z$

∀₊**-rule** : **if** $\forall S.C \in \mathcal{L}_i(x)$, with $\mathtt{Trans}(R)$ and $R \sqsubseteq^* S$, there exists an individual y such that y is an R-successor of x, and $\forall R.C \notin \mathcal{L}_i(y)$,

    then add $\forall R.C$ to $\mathcal{L}_i(y)$

choose-rule : **if** $\bowtie nS.D \in \mathcal{L}_i(x)$, with $\bowtie \in \{\geqslant, \leqslant\}$ and there is an S-neighbor y of x

    then add either D or $\neg D$ to $\mathcal{L}_i(y)$

∃**-rule** : **if** $\exists R.C \in \mathcal{L}_i(x)$, and x has no R-successor y with $C \in \mathcal{L}_i(y)$,

    then pick up a new individual y, add R to $\mathcal{L}(x, y)$, and let $\mathcal{L}_i(y) := \{C\}$

⩾**-rule** : **if** $\geqslant n\, S.C \in \mathcal{L}_i(x)$, and x has not n S-neighbors $y_1, \ldots, y_n$ with $y_\ell \neq y_j$ for $1 \leqslant \ell < j \leqslant n$,

    then create n new successors $y_1, \ldots, y_n$ of x with $\mathcal{L}_i(x, y_\ell) = \{S\}$, $\mathcal{L}_i(y) := \{C\}$, and $y_\ell \neq y_j$, for $1 \leqslant \ell < j \leqslant n$

A branch $\mathcal{L}$ is closed if, for some individual x, either $\bot \in \mathcal{L}(x)$, or $\{A, \neg A\} \subseteq \mathcal{L}(x)$ for some concept name A, or $\leqslant n\, S.C \in \mathcal{L}(x)$ and x has in $\mathcal{L}$ m S-neighbors $y_1, \ldots, y_m$ with $m > n$, with $C \in \mathcal{L}(y_j)$ and $y_i \neq y_j$ for $1 \leqslant i < j \leqslant m$. We call such a situation a *clash*. A tableau is closed if all its branches are closed. A branch is *open* if it is not closed, and no T-rule can be applied to it. A tableau is open if it has at least one open branch.

In order to prove a formula of the form (8), each [non-]subsumption in Γ is associated with a tableau. For a sentence $C_i \sqsubseteq D_i$, the calculus aims at closing the tableau τ_i that starts with the single branch

$$\mathcal{L}_i(a_i) = \{C_i, \neg D_i\} \tag{12}$$

with a_i being an individual. For a sentence $C_i \not\sqsubseteq D_i$, the calculus, starting with τ_i as before, aims at obtaining an open tableau. We call *system* the $n+1$-tuple $\langle \tau_1, \ldots, \tau_m, \sigma \rangle$, made of the n tableaux and the substitution on the variables. The system always starts with $\sigma = \emptyset$. Substitution rules (S-rules) are presented below. We denote the application of the substitution θ to $\langle \tau_1, \ldots, \tau_m, \sigma \rangle$ by $\theta \langle \tau_1, \ldots, \tau_m, \sigma \rangle$ and its result is $\langle \theta(\tau_1), \ldots, \theta(\tau_n), \theta \cup \sigma \rangle$.

All rules are applicable only if $\mathcal{L}$ is *open*, and the substitution is *not σ-blocked*. Rules above the separating line have precedence over rules below it.

$\sigma\top$**-rule** : apply $[X/\top]$ to $\langle \tau_1, \ldots, \tau_m, \sigma \rangle$
σN**-rule** : apply $[X/A]$ to $\langle \tau_1, \ldots, \tau_m, \sigma \rangle$

$\sigma\neg$**-rule** : apply $[X/\neg Y]$ to $\langle \tau_1, \ldots, \tau_m, \sigma \rangle$, where Y denotes a concept variable not appearing in $\langle \tau_1, \ldots, \tau_m, \sigma \rangle$

$\sigma\geqslant$**-rule** : apply $[X/\geqslant m\,R.Y]$ to $\langle \tau_1, \ldots, \tau_m, \sigma \rangle$, where Y denotes a concept variable not appearing in $\langle \tau_1, \ldots, \tau_m, \sigma \rangle$, and if $m > 1$ then R is a simple role

$\sigma\leqslant$**-rule** : apply $[X/\leqslant n\,R.Y]$ to $\langle \tau_1, \ldots, \tau_m, \sigma \rangle$, where Y denotes a concept variable not appearing in $\langle \tau_1, \ldots, \tau_m, \sigma \rangle$, and if $n > 0$ then R is a simple role

---

$\sigma\sqcap$**-rule** : apply $[X/Y_1 \sqcap Y_2]$ to $\langle \tau_1, \ldots, \tau_m, \sigma \rangle$, where Y_1, Y_2 denote concept variables not appearing in $\langle \tau_1, \ldots, \tau_m, \sigma \rangle$

Note that T-rules are applied separately to each branch of each tableau, while S-rules are applied to all branches of all tableaux at the same time.

An S-rule r is *σ-blocked* for $X \in \mathcal{L}_i(x)$ in $\langle \tau_1, \ldots, \tau_m, \sigma \rangle$ if $\langle \tau_1, \ldots, \tau_m, \sigma \rangle$ derives from some $\langle \tau_1', \ldots, \tau_m', \sigma' \rangle$, in which there is some individual x' such that: (i) $X' \in \mathcal{L}_i'(x')$, (ii) $\mathcal{L}_i(x) = \mathcal{L}_i'(x')$, (iii) for every R-successor y of x in $\mathcal{L}_i$, there exists an R-successor y' of x' in $\mathcal{L}_i'$ such that $\mathcal{L}_i(y) = \mathcal{L}_i'(y')$, (iv) for every S, the number of different S-neighbors of x in $\mathcal{L}_i$ is the same as the number of different S-neighbors of x' in $\mathcal{L}_i'$, and (v) Rule r has been applied to $\mathcal{L}_i'$ in $\langle \tau_1', \ldots, \tau_m', \sigma' \rangle$.

Theorem 2 (Soundness). *Let Γ be as in (7). If the calculus of T- and S-rules, starting with τ_i as in (12) and $\sigma = \emptyset$, yields a system $\langle \tau_1, \ldots, \tau_m, \sigma \rangle$ in which τ_i is closed for $i = 1, \ldots, \ell$, and τ_j is open for $j = \ell + 1, \ldots, m$, then there exists a substitution σ' such that $\sigma'(\Gamma)$ is true.*

Proof. Let σ' be σ in which every remaining unsubstituted concept variable is substituted with a different concept name A never appearing in Γ. Since T-rules are sound, each closed tableau τ_i for $i = 1, \ldots, \ell$ is a proof that $\sigma(C_i) \sqsubseteq \sigma(D_i)$, and the same is also a proof for $\sigma'(C_i) \sqsubseteq \sigma'(D_i)$. Moreover, since T-rules are complete, each open tableau τ_j for $j = \ell + 1, \ldots, m$ is a proof that $\sigma(C_j) \not\sqsubseteq \sigma(D_j)$, and the same remains a proof for $\sigma'(C_j) \not\sqsubseteq \sigma'(D_j)$, since remaining variables are substituted by unconstrained concept names. $\qquad\square$

Theorem 3 (Completeness). *Let Γ be as in (7). If there exists a substitution σ such that $\sigma(\Gamma)$ is true, then there is a way of applying T- and S-rules that yields a system $\langle \tau_1, \ldots, \tau_m, \sigma \rangle$ in which τ_i is closed for $i = 1, \ldots, \ell$, and τ_j is open for $j = \ell + 1, \ldots, m$.*

Proof. Since S-rules mimic $\mathcal{SHIQ}$ syntax (1), every ground substitution σ can be reconstructed by repeated applications of S-rules. If one decides to apply all these S-rules at once, one gets a system $\langle \tau_1', \ldots, \tau_m', \sigma' \rangle$ in which each τ_i has one branch $\mathcal{L}_i(a_i) = \{\sigma(C_i), \sigma(\neg D_i)\}$, and $\sigma' = \sigma$. Now since T-rules are sound and complete, their application yields closed tableaux τ_i for $i = 1, \ldots, \ell$, and open tableaux τ_j for $j = \ell + 1, \ldots, m$. □

Soundness and completeness of the above calculus, together with undecidability results for specific problems such as unification in $\mathcal{SHI}$ [28], imply that there are infinite instances in which the calculus does not terminate. However, for specific classes of formulas of the form (8), a termination proof can be devised on the basis of σ-*blocking* [12], which prevents the application of S-rules.

5 Discussion and Future Work

Some related work [3, 12] has been already compared within the technical sections of the paper. In addition, some researchers proposed and studied the use of Higher-order DLs for meta-modeling purposes. More specifically, Pan& Horrocks [20] propose a stratified Higher-order DL (OWL FA) to cope with meta-assertions about concepts and roles; OWL FA is incomparable with any $\mathcal{DL}_X$, since on one side, higher-order assertions can be made, but on the other side, concept variables are not admitted. Motik [19] proves that satisfiability in Higer-order $\mathcal{ALCO}$, which is a fragment of OWL Full, is undecidable; his proof could not be rephrased in $(\mathcal{SHIQ})_X$, since it exploits the feature $\mathcal{O}$ to construct concepts starting from individuals. Motik also proposes a Higher-order DL with two possible semantics, but again, he does not consider concept variables. De Giacomo *et al.* [9] augment a DL with variables that may be interpreted—in a Henkin semantics—as individuals, concepts, and roles at the same time, obtaining a new logic $Hi(\mathcal{DL})$. Also this extension is incomparable with any $\mathcal{DL}_X$, since on one side one can express in $Hi(\mathcal{DL})$ arbitrarily higher-order concepts that are not expressible in $\mathcal{DL}_X$, while in $\mathcal{DL}_X$ one can form complex concept terms that are not allowed in $Hi(\mathcal{DL})$, such as $\exists R.X$.

The innovative potential of the paper mainly lays in the investigation on non-standard reasoning services apparently far from each other under a unifying lens. Such a unification effort paves the way to important generalization results both in the definition and in the solution of problems expressible according to the proposed framework. In particular, on the one hand we exploit the property that many non-standard reasoning services are devoted to the "construction of an objective concept" in order to model all of them as Constructive Reasoning Tasks trough special Second-order sentences in DLs; on the other hand we propose a unified calculus aimed at the design and implementation of a unique system able to solve any non-standard reasoning tasks, whose investigation is object of our current and future research work.

References

1. F. Baader, 'Least common subsumers and most specific concepts in a description logic with existential restrictions and terminological cycles', in *IJCAI 2003*.
2. F. Baader, R. Küsters, and R. Molitor, 'Rewriting concepts using terminologies', in *KR 2000*.
3. F. Baader and P. Narendran, 'Unification of concept terms in description logics', *J. of Symbolic Computation*, **31**, 277–305, (2001).
4. F. Baader and B. Sertkaya, 'Usability issues in description logic knowledge base completion', in *ICFCA-2009*.
5. F. Baader, B. Sertkaya, and A.-Y. Turhan, 'Computing the least common subsumer w.r.t. a background terminology', *J. of Applied Logic*, **5**(3), 392–420, (2007).
6. T. Berners-Lee, J. Hendler, and O. Lassila, 'The semantic web', *Scient. Amer.*, (2001).
7. M. Bienvenu, 'Complexity of abduction in the EL family of lightweight description logics', in *KR 2008*.
8. A. Borgida, T. Walsh, and H. Hirsh, 'Towards measuring similarity in description logics', in *DL 2005*.
9. G. De Giacomo, M. Lenzerini, and R. Rosati, 'On higher-order description logics', in *DL 2009*. CEUR-WS.org.
10. T. Di Noia, E. Di Sciascio, and F. M. Donini, 'Semantic matchmaking as non-monotonic reasoning: A description logic approach', *J. of Artif. Intell. Res.*, **29**, 269–307, (2007).
11. T. Di Noia, E. Di Sciascio, and F. M. Donini, 'Computing information minimal match explanations for logic-based matchmaking', in *WI/IAT 2009*.
12. F. M. Donini, S. Colucci, T. Di Noia, and E. Di Sciascio, 'A tableaux-based method for computing least common subsumers for expressive description logics', in *IJCAI 2009*.
13. C. Elsenbroich, O. Kutz, and U. Sattler, 'A case for abductive reasoning over ontologies', in *OWLED Workshop*.
14. P. Gardenfors, *Knowledge in Flux*, Mit Press, Bradford Book, 1988.
15. L. Henkin, 'Completeness in the theory of types', *J. of Symbolic Logic*, **15**(2), 81–91, (1950).
16. B. Konev, D. Walther, and F. Wolter, 'Forgetting and uniform interpolation in large-scale description logic terminologies', in *IJCAI 2009*.
17. F. Lécué, A. Delteil, and A. Léger, 'Applying abduction in semantic web service composition', in *ICWS 2007*.
18. D. L. McGuinness and A. Borgida, 'Explaining subsumption in description logics', in *IJCAI'95*.
19. B. Motik, 'On the properties of metamodeling in OWL', *J. of Log. and Comp.*, **17**(4), 617–637, (2007).
20. J. Z. Pan and I. Horrocks, 'Owl fa: a metamodeling extension of OWL DL', in *WWW'06*. ACM.
21. I. S. E. Peraldi, A. Kaya, and R. Möller, 'Formalizing multimedia interpretation based on abduction over description logic aboxes', in *DL 2009*.
22. A. Ragone, 'OWL-DL as a power tool to model negotiation mechanisms with incomplete information', in *ISWC/ASWC 2007*.
23. S. Schlobach, 'Explaining subsumption by optimal interpolation', in *JELIA 2004*.
24. I. Seylan, E. Franconi, and J. de Bruijn, 'Effective query rewriting with ontologies over dboxes', in *IJCAI 2009*.
25. G. Teege, 'Making the difference: A subtraction operation for description logics', in *KR'94*.
26. S. Tobies, *Complexity Results and Practical Algorithms for Logics in Knowledge Representation*, Ph.D. dissertation, RWTH Aachen, 2001.
27. J. Väänänen, 'Second-order logic and foundations of mathematics', *The Bulletin of Symbolic Logic*, **7**(4), 504–520, (2001).
28. F. Wolter and M. Zakharyaschev, 'Undecidability of the unication and admissibility problems for modal and description logics', *ACM Trans. on Computational Logic*, **9**, (2008).

Tractable Extensions of the Description Logic $\mathcal{EL}$ with Numerical Datatypes

Despoina Magka, Yevgeny Kazakov, and Ian Horrocks

Oxford University Computing Laboratory
Wolfson Building, Parks Road, OXFORD, OX1 3QD, UK
{despoina.magka,yevgeny.kazakov,ian.horrocks}@comlab.ox.ac.uk

1 Introduction and Motivation

Description logics (DLs) [1] provide a logical foundation for modern ontology languages such as OWL[1] and OWL 2 [2]. $\mathcal{EL}^{++}$ [3] is a lightweight DL for which reasoning is tractable (i.e., can be performed in time that is polynomial w.r.t. the size of the input), and that offers sufficient expressivity for a number of life-sciences ontologies, such as SNOMED CT [4] or the Gene Ontology [5]. Among other constructors, $\mathcal{EL}^{++}$ supports limited usage of datatypes. In DL, datatypes (also called concrete domains) can be used to define new concepts by referring to particular values, such as strings or integers. For example, the concept $\mathsf{Human} \sqcap \exists\mathsf{hasAge}.(<, 18) \sqcap \exists\mathsf{hasName}.(=, \text{"Alice"})$ describes humans, named "Alice", whose age is less than 18. Datatypes are described first by the domain their values can come from and also by the relations that can be used to constrain possible values. In our example, $(<, 18)$ refers to the domain of natural numbers and uses the relation "$<$" to constrain possible values to those less than 18, while $(=, \text{"Alice"})$ refers to the domain of strings and uses the relation "$=$" to constrain the value to "Alice".

In order to ensure that reasoning remains polynomial, $\mathcal{EL}^{++}$ allows only for datatypes which satisfy a condition called p-admissibility [3]. In an nutshell, this condition ensures that the satisfiability of datatype constraints can be solved in polynomial time, and that concept disjunction cannot be expressed using datatype concepts. For example, if we were to allow both $\leq$ and $\geq$ for integers, then we could express $A \sqsubseteq B \sqcup C$ by formulating the axioms $\mathsf{A} \sqsubseteq \exists\mathsf{R}.(\leq, 5)$, $\exists\mathsf{R}.(\leq, 2) \sqsubseteq \mathsf{B}$ and $\exists\mathsf{R}.(\geq, 2) \sqsubseteq \mathsf{C}$. Similarly, we can show that p-admissibility does not allow for both $<$ (or $>$) and $=$. For this reason, the EL Profile of OWL 2, which is based on $\mathcal{EL}^{++}$, admits only equality ($=$) in datatype expressions.

In this paper, we demonstrate how these restrictions can be significantly relaxed without loosing tractability. As a motivating example, consider the following two axioms which might be used, e.g., in a pharmacy-related ontology:

$$\mathsf{Panadol} \sqsubseteq \exists\mathsf{contains}.(\mathsf{Paracetamol} \sqcap \exists\mathsf{mgPerTablet}.(=, 500)) \tag{1}$$

$$\begin{aligned}\mathsf{Patient} \sqcap \exists\mathsf{hasAge}.(<, 6) \sqcap \\ \exists\mathsf{hasPrescription}.\exists\mathsf{contains}.(\mathsf{Paracetamol} \sqcap \exists\mathsf{mgPerTablet}.(>, 250)) \sqsubseteq \bot\end{aligned} \tag{2}$$

---

[1] http://www.w3.org/2004/OWL

Axiom (1) states that the drug Panadol contains 500 mg of paracetamol per tablet, while axiom (2) states that a drug that contains more than 250 mg of paracetamol per tablet must not be prescribed to a patient younger than 6 years old. The ontology could be used, for example, to support clinical staff who want to check whether Panadol can be prescribed to a 3-year-old patient. This can easily be achieved by checking whether concept (3) is satisfiable w.r.t. the ontology:

$$\mathsf{Patient} \sqcap \exists\mathsf{hasAge}.(=, 3) \sqcap \exists\mathsf{hasPrescription}.\mathsf{Panadol} \qquad (3)$$

Unfortunately, this is not possible using $\mathcal{EL}^{++}$, because axioms (1) and (2) involve both equality ($=$) and inequalities ($<$, $>$), and this violates the p-admissibility restriction. In this paper we demonstrate that it is, however, possible to express axioms (1) and (2) and concept (3) in a tractable extension of $\mathcal{EL}$. A polynomial classification procedure can then be used to determine the satisfiability of (3) w.r.t. the ontology by checking if adding an axiom

$$X \sqsubseteq \mathsf{Patient} \sqcap \exists\mathsf{hasAge}.(=, 3) \sqcap \exists\mathsf{hasPrescription}.\mathsf{Panadol}$$

for some new concept name X would entail $X \sqsubseteq \bot$.

Our idea is based on the intuition that equality in (1) and (3) serves a different purpose than inequalities do in (2). Equality in (1) and (3) is used to state a *fact* (the content of a drug and the age of a patient) whereas inequalities in (2) are used to trigger a *rule* (what happens if a certain quantity of drug is prescribed to a patient of a certain age). In other words, equality is used *positively* and inequalities are used *negatively*. It seems reasonable to assume that positive usages of datatypes will typically involve equality since a fact can usually be precisely stated. On the other hand, it seems reasonable to assume that negative occurrences of datatypes will typically involve equality as well as inequalities since a rule usually applies to a range of situations. In this paper, we make a fine-grained study of datatypes in $\mathcal{EL}$ by considering restrictions not only on the kinds of relations included in a datatype, but also on whether the relations can be used positively or negatively. The main contributions of this paper can be summarised as follows:

1. We introduce the notion of a *Numerical Datatype with Restrictions* (*NDR*) that specifies the domain of the datatype, the datatype relations that can be used positively and the datatype relations that can be used negatively.
2. We extend the $\mathcal{EL}$ reasoning algorithm [3] to provide a polynomial reasoning procedure for an extension of $\mathcal{EL}$ with NDRs, where the procedure is sound for any NDR.
3. We introduce the notion of a *safe NDR*, show that every extension of $\mathcal{EL}$ with a safe NDR is tractable and prove that our reasoning procedure is complete for any safe NDR.
4. Finally, we provide a complete classification of safe NDRs for the cases of natural numbers, integers, rationals and reals. Notably, we demonstrate that the numerical datatype restrictions can be significantly relaxed by allowing arbitrary numerical relations to occur negatively—not only equality as currently specified in the OWL 2 EL Profile. As argued earlier, this combination

Table 1. Concept descriptions in $\mathcal{EL}^{\perp}(\mathcal{D})$

NAME	SYNTAX	SEMANTICS
Concept name	C	$C^{\mathcal{I}}$
Top	$\top$	$\Delta^{\mathcal{I}}$
Bottom	$\perp$	$\emptyset$
Conjunction	$C \sqcap D$	$C^{\mathcal{I}} \cap D^{\mathcal{I}}$
Existential restriction	$\exists R.C$	$\{x \in \Delta^{\mathcal{I}} \mid \exists y \in \Delta^{\mathcal{I}} : (x,y) \in R^{\mathcal{I}} \wedge y \in C^{\mathcal{I}}\}$
Datatype restriction	$\exists F.r$	$\{x \in \Delta^{\mathcal{I}} \mid \exists v \in \mathcal{D} : (x,v) \in F^{\mathcal{I}} \wedge r(v)\}$

is of particular interest to ontology engineering, and is thus a strong candidate for the next extension of the EL Profile in OWL 2.

2 Preliminaries

In this section we introduce $\mathcal{EL}^{\perp}(\mathcal{D})$, an extension of $\mathcal{EL}^{\perp}$ [3] with numerical datatypes. In the DL literature datatypes are better known as concrete domains [6]; we call them datatypes to be more consistent with OWL 2 [2]. The syntax of $\mathcal{EL}^{\perp}(\mathcal{D})$ uses a set of *concept names* N_C, a set of *role names* N_R and a set of *feature names* N_F. $\mathcal{EL}^{\perp}(\mathcal{D})$ is parametrised with a *numerical domain* $\mathcal{D} \subseteq \mathbb{R}$ ($\mathbb{R}$ is the set of real numbers). N_C, N_R and N_F are countably infinite sets and, additionally, pairwise disjoint. We call (s, y), where $s \in \{<, \leq, >, \geq, =\}$ and $y \in \mathcal{D}$, a *$\mathcal{D}$-datatype restriction* (or simply a *datatype restriction* if the domain $\mathcal{D}$ is clear from the context). Given a $\mathcal{D}$-datatype restriction $r = (s, y)$ and an $x \in \mathcal{D}$, we say that x satisfies r and we write $r(x)$ iff $(x, y) \in s$, where $s \in \{<, \leq, >, \geq, =\}$ and s is interpreted as the standard relation on real numbers. Table 1 recursively defines concepts in $\mathcal{EL}^{\perp}(\mathcal{D})$, where C and D are concepts, $R \in N_R$, $F \in N_F$ and r is a $\mathcal{D}$-datatype restriction. An *axiom* α *(in $\mathcal{EL}^{\perp}(\mathcal{D})$)* is an expression of the form $C \sqsubseteq D$, where C and D are concepts. An *($\mathcal{EL}^{\perp}(D)-$)ontology* $\mathcal{O}$ is a set of axioms. A concept E is said to *positively (negatively) occur* in an axiom $C \sqsubseteq D$ iff it occurs in D (C). An interpretation of $\mathcal{EL}^{\perp}(\mathcal{D})$ is a pair $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$, where $\Delta^{\mathcal{I}}$ is a non-empty set, the *domain of the interpretation*, and $\cdot^{\mathcal{I}}$ is the *interpretation function*. The interpretation function maps each $A \in N_C$ to a set $A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$, each $R \in N_R$ to a relation $R^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$ and each $F \in N_F$ to a relation $F^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \mathcal{D}$. Note that we do not require the interpretation of features to be functional. In this respect, they correspond to the data properties in OWL 2 [2]. The constructors of $\mathcal{EL}^{\perp}(\mathcal{D})$ are interpreted as indicated in Table 1. *An interpretation $\mathcal{I}$ satisfies an axiom $\alpha = C \sqsubseteq D$ iff $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$* (written $\mathcal{I} \models \alpha$). If $\mathcal{I} \models \alpha$ for every $\alpha \in \mathcal{O}$, then *$\mathcal{I}$ is a model of $\mathcal{O}$* (written $\mathcal{I} \models \mathcal{O}$). If every model $\mathcal{I}$ of $\mathcal{O}$ satisfies the axiom α then we say that $\mathcal{O}$ *entails* α and we write $\mathcal{O} \models \alpha$. We define the *signature of an ontology* $\mathcal{O}$ as the set $sig(\mathcal{O})$ of concept, role and feature names that occur in $\mathcal{O}$. We say that an axiom in $\mathcal{EL}^{\perp}(\mathcal{D})$ is in normal form if it has one of the forms: $A \sqsubseteq B'$ (NF1), $A_1 \sqcap A_2 \sqsubseteq B$ (NF2), $A \sqsubseteq \exists R.B$ (NF3), $\exists R.B \sqsubseteq A$ (NF4), $A \sqsubseteq \exists F.r$ (NF5) or $\exists F.r \sqsubseteq A$ (NF6), where A, A_1, A_2, $B \in N_C^{\top}$, $B' \in N_C^{\top, \perp}$, $R \in N_R$, $F \in N_F$ and r is a $\mathcal{D}$-datatype restriction. The

normalization procedure is the same as for the $\mathcal{EL}^{++}$ [3]; more details can be found in the technical report [7]. ($N_C^\top = N_C \cup \{\top\}$, $N_C^{\top,\perp} = N_C \cup \{\top, \perp\}$).

3 Numerical Datatypes with Restrictions

In this section we introduce the notion of a Numerical Datatype with Restrictions (NDR) which specifies which datatype relations can be used positively and negatively. We then present a sound polynomial consequence-based classification procedure for $\mathcal{EL}^\perp$ extended with NDRs. Finally we prove that the procedure is complete if the NDR satisfies special safety requirements.

Definition 1 (Numerical Datatype with Restrictions). *A* numerical datatype with restrictions (NDR) *is a triple* $(\mathcal{D}, O_+, O_-)$*, where* $\mathcal{D} \subseteq \mathbb{R}$ *is a numerical domain and* $O_+, O_- \subseteq \{<, \leq, >, \geq, =\}$ *is the set of* positive *and, respectively,* negative relations. *An axiom in* $\mathcal{EL}^\perp(\mathcal{D})$ *is an axiom in* $\mathcal{EL}^\perp(\mathcal{D}, O_+, O_-)$ *if for every positive (negative) occurrence of a concept* $\exists F.(s, y)$ *in the axiom,* $s \in O_+$ *(*$s \in O_-$*). An* $\mathcal{EL}^\perp(\mathcal{D}, O_+, O_-)$*-ontology is a set of axioms in* $\mathcal{EL}^\perp(\mathcal{D}, O_+, O_-)$*.*

We are going to describe a classification procedure for $\mathcal{EL}^\perp(\mathcal{D}, O_+, O_-)$, which is closely related to the procedure for $\mathcal{EL}^{++}$ [3]. In order to formulate inference rules for datatypes we need to introduce notation for satisfiability of a datatype restriction and implication between datatype restrictions. For two $\mathcal{D}$-datatype restrictions r_+ and r_-, we write $r_+ \to_\mathcal{D} r_-$ iff $r_+(x)$ implies $r_-(x)$, $\forall x \in \mathcal{D}$. We write $r_+ \to_\mathcal{D} \perp$ iff there is no $x \in \mathcal{D}$ such that $r_+(x)$ holds. In the opposite cases, we write $r_+ \nrightarrow_\mathcal{D} r_-$ and $r_+ \nrightarrow_\mathcal{D} \perp$. We assume that deciding whether $r_+ \to_\mathcal{D} r_-$ and $r_+ \to_\mathcal{D} \perp$ can be done in polynomial time. It is easy to see that this is the case when $\mathcal{D}$ is the set of natural numbers, integers, reals or rationals for the set of relations $\{<, \leq, >, \geq, =\}$.

The classification procedure for $\mathcal{EL}^\perp(\mathcal{D})$ takes as an input an ontology $\mathcal{O}$ whose axioms are in $\mathcal{EL}^\perp(\mathcal{D})$ and in normal form and applies the inference rules in Table 2 to derive new axioms of the form NF1, NF3 and NF5. The rules are applied to already derived axioms and use existence of axioms in $\mathcal{O}$, $r_+ \to_\mathcal{D} \perp$ or $r_+ \to_\mathcal{D} r_-$ as side-conditions. The procedure terminates when no new axiom can be derived. It is easily checked that the procedure runs in polynomial time (there are only polynomially many possible axioms of the form NF1, NF3 and NF5 over $sig(\mathcal{O})$) and that the rules in Table 2 are sound (the conclusions of the rules are logical consequences of their premises).

The completeness proof is based on the canonical model construction similarly as for $\mathcal{EL}^{++}$ [3]. In order to deal with datatypes in the canonical model we introduce a notion of a datatype *constraint*. Intuitively, a constraint specifies which datatype restrictions should hold in a given element of the model and which should not.

Definition 2 (Constraint). *A* constraint *over* $(\mathcal{D}, O_+, O_-)$ *is defined as a pair of sets* (S_+, S_-)*, such that* $S_+ = \{(s_+^1, y_1), \ldots, (s_+^n, y_n)\}$ *with* $s_+^i \in O_+$*,* $S_- = \{(s_-^1, z_1), \ldots, (s_-^m, z_m)\}$ *with* $s_-^j \in O_-$*,* $y_i, z_j \in \mathcal{D}$*,* $(s_+^i, y_i) \nrightarrow_\mathcal{D} (s_-^j, z_j)$

Table 2. Reasoning rules in $\mathcal{EL}^{\perp}(\mathcal{D})$ ($A, B, C, E \in N_C^{\top}$, $C' \in N_C^{\top,\perp}$, $R \in N_R$, $F \in N_F$)

IR1 $\dfrac{}{A \sqsubseteq A}$	**IR2** $\dfrac{}{A \sqsubseteq \top}$	**CR1** $\dfrac{A \sqsubseteq B}{A \sqsubseteq C'} \quad B \sqsubseteq C' \in \mathcal{O}$
CR2 $\dfrac{A \sqsubseteq B \quad A \sqsubseteq C}{A \sqsubseteq D} \quad B \sqcap C \sqsubseteq D \in \mathcal{O}$		**CR3** $\dfrac{A \sqsubseteq B}{A \sqsubseteq \exists R.C} \quad B \sqsubseteq \exists R.C \in \mathcal{O}$
CR4 $\dfrac{A \sqsubseteq \exists R.B \quad B \sqsubseteq C}{A \sqsubseteq D} \quad \exists R.C \sqsubseteq D \in \mathcal{O}$		**CR5** $\dfrac{A \sqsubseteq \exists R.B \quad B \sqsubseteq \perp}{A \sqsubseteq \perp}$

ID1 $\dfrac{}{A \sqsubseteq \perp} \quad A \sqsubseteq \exists F.r_+ \in \mathcal{O} \ , \ r_+ \rightarrow_{\mathcal{D}} \perp$	**CD1** $\dfrac{A \sqsubseteq B}{A \sqsubseteq \exists F.r_+} \quad B \sqsubseteq \exists F.r_+ \in \mathcal{O}$
CD2 $\dfrac{A \sqsubseteq \exists F.r_+}{A \sqsubseteq B} \quad \exists F.r_- \sqsubseteq B \in \mathcal{O} \ , \ r_+ \rightarrow_{\mathcal{D}} r_-$	

and $(s_+^i, y_i) \nrightarrow_{\mathcal{D}} \perp$ for $1 \le i \le n$, $1 \le j \le m$ and $m, n \ge 0$. A constraint (S_+, S_-) over $(\mathcal{D}, O_+, O_-)$ is satisfiable *iff there exists a* solution of (S_+, S_-) *that is a set* $V \subseteq \mathcal{D}$ *such that every* $r_+ \in S_+$ *is satisfied by at least one* $v \in V$ *but no* $r_- \in S_-$ *is satisfied by any* $v \in V$.

Our model construction procedure works only for the cases where we can ensure that every constraint over a numerical domain is satisfiable. This leads us to a notion of safety for an NDR.

Definition 3 (NDR Safety). *Let* $(\mathcal{D}, O_+, O_-)$ *be an NDR.* $(\mathcal{D}, O_+, O_-)$ *is safe iff every constraint over* $(\mathcal{D}, O_+, O_-)$ *is satisfiable.*

Definition 4 (Strong and Weak Convexity). *The NDR* $(\mathcal{D}, O_+, O_-)$ *is strongly convex when for every* $r_+^i = (s_+^i, y_i)$ *and* $r_-^j = (s_-^j, z_j)$, *with* $s_+^i \in O_+$, $s_-^j \in O_-$ *and* $y_i, z_j \in \mathcal{D}$ *($1 \le i \le n$, $1 \le j \le m$), if* $\bigwedge_{i=1}^{n} r_+^i \rightarrow_{\mathcal{D}} \bigvee_{j=1}^{m} r_-^j$, *then there exists an* r_-^j *($1 \le j \le m$) such that* $\bigwedge_{i=1}^{n} r_+^i \rightarrow_{\mathcal{D}} r_-^j$. *($\mathcal{D}, O_+, O_-$) is weakly convex when the implication holds for* $n = 1$.

For example the NDR $(\mathbb{Z}, \{<, >\}, \{=\})$ is weakly convex but not strongly convex. It is weakly convex since the implications $((<, y) \rightarrow_{\mathbb{Z}} \bigvee_{j=1}^{m}(=, z_j))$ and $((>, y) \rightarrow_{\mathbb{Z}} \bigvee_{j=1}^{m}(=, z_j))$ never hold. However, it is not strongly convex: it is $(>, 2) \wedge (<, 5) \rightarrow_{\mathbb{Z}} (=, 3) \vee (=, 4)$, but also $(>, 2) \wedge (<, 5) \nrightarrow_{\mathbb{Z}} (=, 3)$ and $(>, 2) \wedge (<, 5) \nrightarrow_{\mathbb{Z}} (=, 4)$.

Lemma 1. *($\mathcal{D}, O_+, O_-$) is safe iff it is weakly convex.*

Proof. We assume that $(\mathcal{D}, O_+, O_-)$ is not weakly convex and we prove that it is non-safe. Since it is not weakly convex we have that for some $r_+ \rightarrow_{\mathcal{D}} \bigvee_{j=1}^{m} r_-^j$

there exists no r_-^j such that $r_+ \to_{\mathcal{D}} r_-^j$. We define (S_+, S_-), with $S_+ = \{r_+\}$ and $S_- = \{r_-^j\}_{j=1}^m$ and we prove that (S_+, S_-) is not satisfiable. (S_+, S_-) is indeed a constraint because $r_+ \nrightarrow_{\mathcal{D}} \bot$ (otherwise $r_+ \to_{\mathcal{D}} r_-^j$ is true for every r_-^j) and for every r_-^j, $r_+ \nrightarrow_{\mathcal{D}} r_-^j$ (otherwise $r_+ \to_{\mathcal{D}} r_-^j$ is true for at least one r_-^j). Additionally, it is not satisfiable, because from $r_+ \to_{\mathcal{D}} \bigvee_{j=1}^m r_-^j$ there can be found no x such that $r_+(x)$ and $\bigwedge_{j=1}^m \neg r_-^j(x)$.

We prove that if $(\mathcal{D}, O_+, O_-)$ is not safe, then it is not weakly convex. Since it is not safe then there exists a non-satisfiable constraint (S_+, S_-), where $S_+ = \{r_+^i\}_{i=1}^n$ and $S_- = \{r_-^j\}_{j=1}^m$. We have $S_+, S_- \neq \emptyset$ because otherwise a solution for (S_+, S_-) exists. Since (S_+, S_-) is not satisfiable there exists no x for $1 \le i \le n$ such that $r_+^i(x)$ and $\bigwedge_{j=1}^m \neg r_-^j(x)$, or otherwise written, $r_+^i \to_{\mathcal{D}} \bigvee_{j=1}^m r_-^j$. From this and $r_+^i \nrightarrow_{\mathcal{D}} r_-^j$ (from the constraint definition), $(\mathcal{D}, O_+, O_-)$ is not weakly convex. □

Theorem 1 (Completeness). *Let $(\mathcal{D}, O_+, O_-)$ be a safe NDR, let $\mathcal{O}$ be an $\mathcal{EL}^\bot(\mathcal{D}, O_+, O_-)$-ontology containing axioms in normal form and let $\mathcal{O}'$ be the saturation of $\mathcal{O}$ under the rules of Table 2. For every $A, B \in (N_C^\top \cap sig(\mathcal{O}))$, if $\mathcal{O} \models A \sqsubseteq B$, then $A \sqsubseteq B \in \mathcal{O}'$ or $A \sqsubseteq \bot \in \mathcal{O}'$.*

Proof. The proof is analogous to the completeness proof for the $\mathcal{EL}^{++}$ language [3]; we build a canonical model $\mathcal{I}$ for $\mathcal{O}$ using $\mathcal{O}'$ and show that if $A \not\sqsubseteq B \in \mathcal{O}'$ and $A \not\sqsubseteq \bot \in \mathcal{O}'$ then $\mathcal{I} \not\models A \sqsubseteq B$.
For every $A \in N_C$, $F \in N_F$, define $S_+(A, F)$ and $S_-(A, F)$, as follows:

$$S_+(A, F) = \{r_+ \mid A \sqsubseteq \exists F.r_+ \in \mathcal{O}', A \sqsubseteq \bot \notin \mathcal{O}'\} \tag{3}$$

$$S_-(A, F) = \{r_- \mid \exists F.r_- \sqsubseteq B \in \mathcal{O}, A \sqsubseteq B \notin \mathcal{O}'\} \tag{4}$$

We now show that $(S_+(A, F), S_-(A, F))$ is a constraint over $(\mathcal{D}, O_+, O_-)$. First we prove that $r_+ \nrightarrow_{\mathcal{D}} \bot$, $\forall r_+ \in S_+(A, F)$, which is true because otherwise due to rule **ID1** it would be $A \sqsubseteq \bot \in \mathcal{O}'$, in contradiction to the definition of $S_+(A, F)$. Additionally, there is no $r_+ \in S_+(A, F)$ and $r_- \in S_-(A, F)$ such that $r_+ \to_{\mathcal{D}} r_-$, otherwise from $A \sqsubseteq \exists F.r_+ \in \mathcal{O}'$, $\exists F.r_- \sqsubseteq B \in \mathcal{O}$ and **CD2** it would be $A \sqsubseteq B \in \mathcal{O}'$ which contradicts the definition of $S_-(A, F)$. Since $(S_+(A, F), S_-(A, F))$ is a constraint over $(\mathcal{D}, O_+, O_-)$ and $(\mathcal{D}, O_+, O_-)$ is safe, there exists a solution $V(A, F) \subseteq \mathcal{D}$ of $(S_+(A, F), S_-(A, F))$. We now construct the canonical model $\mathcal{I}$:

$$\Delta^{\mathcal{I}} = \{x_A \mid A \in (N_C^\top \cap sig(\mathcal{O})), A \sqsubseteq \bot \notin \mathcal{O}'\} \tag{5}$$

$$B^{\mathcal{I}} = \{x_A \mid x_A \in \Delta^{\mathcal{I}}, A \sqsubseteq B \in \mathcal{O}'\} \tag{6}$$

$$R^{\mathcal{I}} = \{(x_A, x_B) \mid A \sqsubseteq \exists R.B \in \mathcal{O}', x_A, x_B \in \Delta^{\mathcal{I}}\} \tag{7}$$

$$F^{\mathcal{I}} = \{(x_A, v) \mid v \in V(A, F)\} \tag{8}$$

We prove that $\mathcal{I} \models \mathcal{O}$ by showing that $\mathcal{I} \models \alpha$, when α takes one of the NF1-NF6.

NF1 $A \sqsubseteq B$: We need to prove $A^{\mathcal{I}} \subseteq B^{\mathcal{I}}$. Take an $x \in A^{\mathcal{I}}$. By (6), $x = x_C$ such that $C \sqsubseteq A \in \mathcal{O}'$. From $A \sqsubseteq B \in \mathcal{O}$ and since $\mathcal{O}'$ is closed under **CR1**, we have $C \sqsubseteq B \in \mathcal{O}'$. Hence $x = x_C \in B^{\mathcal{I}}$ by (6).

If $B = \bot$, then we need to show that $A^{\mathcal{I}} = \emptyset$. If there exists $x \in A^{\mathcal{I}}$, then by (6) $x = x_C$ such that $C \sqsubseteq A \in \mathcal{O}'$. Since $\mathcal{O}'$ is closed under **CR1** and $A \sqsubseteq \bot \in \mathcal{O}'$, we have $C \sqsubseteq \bot \in \mathcal{O}'$. Thus, $x = x_C \notin \Delta^{\mathcal{I}}$ by (5), which contradicts our assumption that $x \in A^{\mathcal{I}}$.

We examine separately the case when $A = \top$. We have that $x_A \in \Delta^{\mathcal{I}}$ and we need to show that $x_A \in B^{\mathcal{I}}$. From rule **IR2**, we have that $A \sqsubseteq \top \in \mathcal{O}'$. From rule **CR1**, $A \sqsubseteq B \in \mathcal{O}'$; since $x_A \in \Delta^{\mathcal{I}}$ and $A \sqsubseteq B \in \mathcal{O}'$ we get $x_A \in B^{\mathcal{I}}$ by (6).

NF2 $A_1 \sqcap A_2 \sqsubseteq B$: We prove $(A_1 \sqcap A_2)^{\mathcal{I}} \subseteq B^{\mathcal{I}}$. Take an $x \in (A_1 \sqcap A_2)^{\mathcal{I}}$; then, $x \in A_1^{\mathcal{I}}$, $x \in A_2^{\mathcal{I}}$ and by (6) $x = x_A$ for some concept name A such that $A \sqsubseteq A_1 \in \mathcal{O}'$ and $A \sqsubseteq A_2 \in \mathcal{O}'$. Since $A \sqsubseteq A_1 \in \mathcal{O}'$, $A \sqsubseteq A_2 \in \mathcal{O}'$ and $A_1 \sqcap A_2 \sqsubseteq B \in \mathcal{O}$, closure under rule **CR2** gives $A \sqsubseteq B \in \mathcal{O}'$ or $x \in B^{\mathcal{I}}$, by (6).

NF3 $A \sqsubseteq \exists R.B$: We show $A^{\mathcal{I}} \subseteq (\exists R.B)^{\mathcal{I}}$; take an $x \in A^{\mathcal{I}}$. By (6), $x = x_C$ where $C \sqsubseteq A \in \mathcal{O}'$. Since $A \sqsubseteq \exists R.B \in \mathcal{O}$ and $\mathcal{O}'$ is closed under **CR3**, we have $C \sqsubseteq \exists R.B \in \mathcal{O}'$. Since $x_C \in \Delta^{\mathcal{I}}$, we have $C \sqsubseteq \bot \notin \mathcal{O}'$ and, hence, $B \sqsubseteq \bot \notin \mathcal{O}'$ by **CR5**. Thus, $x_B \in \Delta^{\mathcal{I}}$ and $(x_C, x_B) \in R^{\mathcal{I}}$ by (7). Since $B \sqsubseteq B \in \mathcal{O}'$ by **IR1**, we have $x_B \in B^{\mathcal{I}}$ by (6). Thus, $x = x_C \in (\exists R.B)^{\mathcal{I}}$.

NF4 $\exists R.B \sqsubseteq A$: We prove $(\exists R.B)^{\mathcal{I}} \subseteq A^{\mathcal{I}}$; take an $x \in (\exists R.B)^{\mathcal{I}}$. Then, there exists $y \in \Delta^{\mathcal{I}}$ such that $(x, y) \in R^{\mathcal{I}}$ and $y \in B^{\mathcal{I}}$. By (7) and (6) $x = x_C$ and $y = x_D$ such that $C \sqsubseteq \exists R.D \in \mathcal{O}'$ and $D \sqsubseteq B \in \mathcal{O}'$ respectively. Since $\exists R.B \sqsubseteq A \in \mathcal{O}$ and $\mathcal{O}'$ is closed under **CR4**, $C \sqsubseteq A \in \mathcal{O}'$. By (6), $x = x_C \in A^{\mathcal{I}}$.

NF5 $A \sqsubseteq \exists F.r_+$: We show that $A^{\mathcal{I}} \subseteq (\exists F.r_+)^{\mathcal{I}}$; take an $x \in A^{\mathcal{I}}$. By (6), there exists a concept name C such that $x = x_C$ and $C \sqsubseteq A \in \mathcal{O}'$. Since $A \sqsubseteq \exists F.r_+ \in \mathcal{O}$ and $\mathcal{O}'$ is closed under **CD1**, we have $C \sqsubseteq \exists F.r_+ \in \mathcal{O}'$. We use (3) and (4) to build $(S_+(C, F), S_-(C, F))$; we have $r_+ \in S_+(C, F)$. By (8) we have $(x_C, v) \in F^{\mathcal{I}}$ for every $v \in V(C, F)$. Since $r_+ \in S_+(C, F)$, there exists $v \in V(C, F)$ such that v satisfies r_+ and, hence, $x = x_C \in (\exists F.r_+)^{\mathcal{I}}$.

NF6 $\exists F.r_- \sqsubseteq B$: We prove that $(\exists F.r_-)^{\mathcal{I}} \subseteq B^{\mathcal{I}}$; take an $x \in (\exists F.r_-)^{\mathcal{I}}$. By (5), there exists $C \in (N_C^\top \cap sig(\mathcal{O}))$ such that $x = x_C$. By (3) and (4) we construct $(S_+(C, F), S_-(C, F))$. Since $x_C \in (\exists F.r_-)^{\mathcal{I}}$, by (8), there exists $v \in V(C, F)$, such that $r_-(v)$ and $V(C, F)$ is a solution for $(S_+(C, F), S_-(C, F))$. Hence, $r_- \notin S_-(C, F)$, and so, $C \sqsubseteq B \in \mathcal{O}'$ by (4). By $C \sqsubseteq B \in \mathcal{O}'$ and (6), $x_C \in B^{\mathcal{I}}$.

We now show that if $A \sqsubseteq B \notin \mathcal{O}'$ and $A \sqsubseteq \bot \notin \mathcal{O}'$, then $\mathcal{O} \nvDash A \sqsubseteq B$ by proving $\mathcal{I} \nvDash A \sqsubseteq B$ (since $\mathcal{I} \models \mathcal{O}$). $A^{\mathcal{I}} \nsubseteq B^{\mathcal{I}}$ holds, because $x_A \in \Delta^{\mathcal{I}}$ (from $A \sqsubseteq \bot \notin \mathcal{O}'$ and (5)), $x_A \in A^{\mathcal{I}}$ (from $A \sqsubseteq A \in \mathcal{O}'$ using rule **IR1** and by (6)) and $x_A \notin B^{\mathcal{I}}$ (from $A \sqsubseteq B \notin \mathcal{O}'$ and (6)). $\qquad\square$

4 Maximal Safe NDRs for $\mathbb{N}$, $\mathbb{Z}$, $\mathbb{R}$ and $\mathbb{Q}$

In this section we present a full classification of safe NDRs for natural numbers ($0 \in \mathbb{N}$), integers, reals and rationals. Table 3 lists all maximal safe NDRs for $\mathbb{N}$, $\mathbb{Z}$, $\mathbb{R}$ and $\mathbb{Q}$. Due to space constraints we present proofs only for the maximal NDRs of natural numbers, that is NDR_1, NDR_2, NDR_9 and NDR_{10}. For these we show that: (i) they are safe (ii) extending any of them leads to non-safety and (iii) every safe NDR w.r.t. $\mathbb{N}$ is contained in one of the NDR_1, NDR_2, NDR_9 or NDR_{10}. Table 4 presents some basic transformations between (satisfiable) constraints.

Table 3. Maximal safe NDRs for $\mathbb{N}$, $\mathbb{Z}$, $\mathbb{R}$ and $\mathbb{Q}$ where $\mathcal{D}$ is the domain and O_+, O_- is the set of positive and, respectively, negative relations

NDR	$\mathcal{D}$	O_+	O_-
NDR_1	$\mathbb{N}, \mathbb{Z}, \mathbb{R}, \mathbb{Q}$	$\{=\}$	$\{<, \leq, >, \geq, =\}$
NDR_2	$\mathbb{N}, \mathbb{Z}$	$\{>, \geq, =\}$	$\{<, \leq, =\}$
NDR_3	$\mathbb{Z}$	$\{<, \leq, =\}$	$\{>, \geq, =\}$
NDR_4	$\mathbb{R}, \mathbb{Q}$	$\{<, >, \geq, =\}$	$\{<, \leq, =\}$
NDR_5	$\mathbb{R}, \mathbb{Q}$	$\{<, \leq, >, =\}$	$\{>, \geq, =\}$
NDR_6	$\mathbb{Z}$	$\{<, \leq, >, \geq, =\}$	$\{=\}$
NDR_7	$\mathbb{R}, \mathbb{Q}$	$\{<, \leq, >, \geq, =\}$	$\{\leq, =\}$
NDR_8	$\mathbb{R}, \mathbb{Q}$	$\{<, \leq, >, \geq, =\}$	$\{\geq, =\}$
NDR_9	$\mathbb{N}, \mathbb{Z}, \mathbb{R}, \mathbb{Q}$	$\{<, \leq, >, \geq, =\}$	$\{<, \leq\}$
NDR_{10}	$\mathbb{N}, \mathbb{Z}, \mathbb{R}, \mathbb{Q}$	$\{<, \leq, >, \geq, =\}$	$\{>, \geq\}$

Table 4. Transformations $C_1 \Rightarrow C_2$ preserving constraints and their satisfiability for $\mathbb{N}$, where S_-, S_+ and S are sets of datatype restrictions and $y_1 \leq y_2$, $z_1 \leq z_2$

$C_1 = (S \cup S_+^1, S_-), C_2 = (S \cup S_+^2, S_-)$		$C_1 = (S_+, S \cup S_-^1), C_2 = (S_+, S \cup S_-^2)$	
S_+^1	S_+^2	S_-^1	S_-^2
$\{(<, y)\}$	$\{(\leq, y-1)\}$	$\{(<, z)\}$	$\{(\leq, z-1)\}$
$\{(>, y)\}$	$\{(\geq, y+1)\}$	$\{(>, z)\}$	$\{(\geq, z+1)\}$
$\{(\leq, y_1), (\leq, y_2)\}$	$\{(\leq, y_1)\}$	$\{(\leq, z_1), (\leq, z_2)\}$	$\{(\leq, z_2)\}$
$\{(\geq, y_1), (\geq, y_2)\}$	$\{(\geq, y_2)\}$	$\{(\geq, z_1), (\geq, z_2)\}$	$\{(\geq, z_1)\}$
$\{(=, y_1), (\leq, y_2)\}$	$\{(=, y_1)\}$	$\{(=, z_1), (\leq, z_2)\}$	$\{(\leq, z_2)\}$
$\{(\geq, y_1), (=, y_2)\}$	$\{(=, y_2)\}$	$\{(\geq, z_1), (=, z_2)\}$	$\{(\geq, z_1)\}$
		$\{(<, 0)\}$	$\emptyset$

Lemma 2. *Let C_1 and C_2 be as defined in Table 4 and $(\mathbb{N}, O_+, O_-)$ be an NDR. Then (i) C_1 is a constraint over $(\mathbb{N}, O_+, O_-)$ iff C_2 is a constraint over $(\mathbb{N}, O_+, O_-)$ and (ii) if C_1 and C_2 are both constraints over $(\mathbb{N}, O_+, O_-)$, then C_1 is satisfiable iff C_2 is satisfiable.*

Corollary 1. *For $\mathbb{N}$, let NDR_i with $i = 1, 2, 9, 10$. For every $C_1 = (S_+^1, S_-^1)$ over NDR_i there exists a constraint $C_2 = (S_+^2, S_-^2)$ over NDR_i, $y_1, \ldots, y_n \in \mathbb{N}$ and $z_1, \ldots, z_m \in \mathbb{N}$ with m, $n \geq 0$ such that:*

$$S_+^2 \subseteq \{(\leq, y_1), (=, y_2), \ldots, (=, y_{n-1}), (\geq, y_n)\}$$
$$S_-^2 \subseteq \{(\leq, z_1), (=, z_2), \ldots, (=, z_{m-1}), (\geq, z_m)\}$$

where $z_1 < y_1 < \ldots < y_n < z_m$, $z_1 < \ldots < z_m$, $y_i \neq z_j$ ($2 \leq i \leq n-1$, $2 \leq j \leq m-1$) and C_1 over NDR_i is satisfiable iff C_2 over NDR_i is satisfiable.

Lemma 3. NDR_1, NDR_2, NDR_9 *and* NDR_{10} *(all for $\mathbb{N}$) are safe.*

Proof. We prove safety by building a solution V for every (S_+, S_-) over the NDRs; by Corollary 1 we can assume w.l.o.g. the following restrictions:

NDR_1: For S_+ we have that $S_+ \subseteq \{(=,y_1),\ldots,(=,y_n)\}$ and for S_- that $S_- \subseteq \{(\leq,z_1),(=,z_2),\ldots,(=,z_{m-1}),(\geq,z_m)\}$ with $z_1 < y_1 < \ldots < y_n < z_m$, $z_1 < \ldots < z_m$ and $y_i \neq z_j$ $(1 \leq i \leq n, 2 \leq j \leq m-1)$. $V = \{y_1,\ldots,y_n\}$.

NDR_2: For S_+ we have that $S_+ \subseteq \{(=,y_1),\ldots,(=,y_{n-1}),(\geq,y_n)\}$ and for S_- that $S_- \subseteq \{(\leq,z_1),(=,z_2),\ldots,(=,z_m)\}$ with $z_1 < y_1 < \ldots < y_n$, $z_1 < \ldots < z_m$ and $y_i \neq z_j$ $(1 \leq i \leq n-1, 2 \leq j \leq m)$. $V = \{y_1,\ldots,y_{n-1},y_n'\}$, where $y_n' = \max(y_n, z_m) + 1$.

NDR_9: For S_+ we have that $S_+ \subseteq \{(\leq,y_1),(=,y_2),\ldots,(=,y_{n-1}),(\geq,y_n)\}$ and for S_- that $S_- \subseteq \{(\leq,z_1)\}$ with $z_1 < y_1 < \ldots < y_n$. $V = \{y_1,\ldots,y_n\}$.

NDR_{10}: For S_+ we have that $S_+ \subseteq \{(\leq,y_1),(=,y_2),\ldots,(=,y_{n-1}),(\geq,y_n)\}$ and for S_- that $S_- \subseteq \{(\geq,z_1)\}$ with $y_1 < \ldots < y_n < z_1$. $V = \{y_1,\ldots,y_n\}$. $\square$

Lemma 4. *Let* $NDR = (\mathbb{N}, O_+, O_-)$. *If (a), (b) or (c), then* NDR *is non-safe.*

(a) $O_+ \cap \{<,\leq,>,\geq\} \neq \emptyset$, $O_- \cap \{<,\leq\} \neq \emptyset$ *and* $O_- \cap \{>,\geq\} \neq \emptyset$.
(b) $O_+ \cap \{>,\geq\} \neq \emptyset$, $O_- \cap \{>,\geq\} \neq \emptyset$ *and* $\{=\} \subseteq O_-$.
(c) $O_+ \cap \{<,\leq\} \neq \emptyset$ *and* $\{=\} \subseteq O_-$.

Proof. For every of the cases (a)-(c) we provide a counterexample that violates the weak convexity condition and, thus by Lemma 1, safety:
(a): $(<,3) \to_\mathbb{N} (<,1) \vee (\geq,1)$ but $(<,3) \not\to_\mathbb{N} (<,1)$ and $(<,3) \not\to_\mathbb{N} (\geq,1)$. The same counterexample applies when $O_+ \cap \{<,\leq\} \neq \emptyset$, $\{\leq,>\} \subseteq O_-$ and when $O_+ \cap \{<,\leq\} \neq \emptyset$, $\{\leq,\geq\} \subseteq O_-$. For $O_+ \cap \{<,\leq\} \neq \emptyset$, $\{<,>\} \subseteq O_-$ it is $(<,3) \to_\mathbb{N} (<,2) \vee (>,1)$ but $(<,3) \not\to_\mathbb{N} (<,2)$ and $(<,3) \not\to_\mathbb{N} (>,1)$. A similar example can be given for the the cases when $O_+ \cap \{>,\geq\} \neq \emptyset$.
(b): $(>,1) \to_\mathbb{N} (=,2) \vee (\geq,3)$ but $(>,1) \not\to_\mathbb{N} (=,2)$ and $(>,1) \not\to_\mathbb{N} (\geq,3)$
$(>,1) \to_\mathbb{N} (=,2) \vee (>,2)$ but $(>,1) \not\to_\mathbb{N} (=,2)$ and $(>,1) \not\to_\mathbb{N} (>,2)$
$(\geq,1) \to_\mathbb{N} (=,1) \vee (\geq,2)$ but $(\geq,1) \not\to_\mathbb{N} (=,1)$ and $(\geq,1) \not\to_\mathbb{N} (\geq,2)$
$(\geq,1) \to_\mathbb{N} (=,1) \vee (>,1)$ but $(\geq,1) \not\to_\mathbb{N} (=,1)$ and $(\geq,1) \not\to_\mathbb{N} (>,1)$
(c): $(<,3) \to_\mathbb{N} (=,1) \vee (=,2)$ but $(<,3) \not\to_\mathbb{N} (=,1)$ and $(<,3) \not\to_\mathbb{N} (= 2)$
$(\leq,2) \to_\mathbb{N} (=,1) \vee (=,2)$ but $(\leq,2) \not\to_\mathbb{N} (=,1)$ and $(\leq,2) \not\to_\mathbb{N} (= 2)$ $\square$

Lemma 5. NDR_1, NDR_2, NDR_9 *and* NDR_{10} *(all for* $\mathbb{N}$*) are maximal safe, that is if any relation is added to* O_+ *or* O_- *they become non-safe.*

Proof. We examine all cases of adding a new relation:
NDR_1: If any of the $<$, $\leq$, $>$, $\geq$ is added to O_+, then NDR_1 becomes non-safe due to Lemma 4(a).
NDR_2: If $>$ or $\geq$ is added to O_-, then non-safety is due to Lemma 4(b). For adding $<$ or $\leq$ to O_+, non-safety is due to Lemma 4(c).
NDR_9: If $>$ or $\geq$ is added to O_-, then non-safety is due to Lemma 4(a). When $=$ is added to O_- then NDR_9 becomes non-safe due to Lemma 4(c).
NDR_{10}: If $<$ or $\leq$ is added to O_-, then non-safety is due to Lemma 4(a). When $=$ is added to O_- then NDR_{10} becomes non-safe due to Lemma 4(c). $\square$

It remains to demonstrate that every safe NDR for $\mathbb{N}$ is contained in one of the NDR_1, NDR_2, NDR_9 or NDR_{10}. In the following, we assume that O_+^i and O_-^i are defined such that $NDR_i = (\mathbb{N}, O_+^i, O_-^i)$ with $i = 1, 2, 9, 10$.

Lemma 6. *If* $(\mathbb{N}, O_+, O_-)$ *is a safe NDR, then* $O_+ \subseteq O_+^i$ *and* $O_- \subseteq O_-^i$ *for* $i = 1, 2, 9$ *or* 10.

Proof. The proof is by case analysis of possible relations in O_+ and O_-.

Case 1: $O_+ \cap \{<, \leq, >, \geq\} = \emptyset$. In this case, $O_+ \subseteq O_+^1$ and $O_- \subseteq O_-^1$.

Case 2: $O_+ \cap \{<, \leq, >, \geq\} \neq \emptyset$. If $O_- \cap \{<, \leq\} \neq \emptyset$ and $O_- \cap \{>, \geq\} \neq \emptyset$ at the same time, then from Lemma 4(a), the NDR is non-safe. Therefore, we examine two cases: either $O_- \subseteq \{>, \geq, =\}$ or $O_- \subseteq \{<, \leq, =\}$.

Case 2.1: $O_- \subseteq \{>, \geq, =\}$. We distinguish either $O_- \subseteq \{>, \geq\}$ or $\{=\} \subseteq O_-$.

Case 2.1.1: $O_- \subseteq \{>, \geq\} = O_-^{10}$ and $O_+ \subseteq O_+^{10}$.

Case 2.1.2: $\{=\} \subseteq O_-$. By Lemma 4(c) it should be $O_+ \subseteq \{>, \geq, =\} = O_+^2$ otherwise the NDR is non-safe. If $O_- \cap \{>, \geq\} \neq \emptyset$ then the NDR is non-safe by Lemma 4(b); otherwise $O_- = \{=\} \subseteq O_-^2$.

Case 2.2: $O_- \subseteq \{<, \leq, =\} = O_-^2$. If $O_+ \subseteq \{>, \geq, =\}$, then $O_+ \subseteq O_+^2$. Otherwise, $O_+ \cap \{<, \leq\} \neq \emptyset$ and we distinguish whether $O_- \subseteq \{<, \leq\}$ or $\{=\} \in O_-$.

Case 2.2.1: $O_- \subseteq \{<, \leq\} = O_-^9$ and $O_+ \subseteq O_+^9$.

Case 2.2.2: $\{=\} \in O_-$. In this case, the NDR is non-safe by Lemma 4(c).   □

For the cases of integers, reals and rationals the proofs are analogous to the case of natural numbers. The interested reader can find details in the technical report [7]. In the following, we provide a brief explanation for the results. We notice two new maximal safe NDRs w.r.t. $\mathbb{Z}$, namely $\mathsf{NDR_3}$ and $\mathsf{NDR_6}$. The reason is that integers do not have a minimal element such as 0 in the case of naturals. In particular positive occurrences of $<$ (or $\leq$) and negative occurrence of $=$ are no longer dangerous (e.g. $(\leq, 1) \not\twoheadrightarrow_{\mathbb{Z}} (=, 1) \vee (=, 0)$ does not hold anymore). Reals and rationals are examples of dense domains: between every two different numbers there always exists a third one. This property is responsible for new safe NDRs. Specifically, O_+ of $\mathsf{NDR_2}$ and $\mathsf{NDR_3}$ can be extended with $<$ and $>$ respectively because the weak convexity property which did not apply for $\mathbb{Z}$ now applies for $\mathbb{R}$ (e.g. $(<, 5) \not\twoheadrightarrow_{\mathbb{R}} (=, 4) \vee (\leq, 3)$). For the same reason, either $\leq$ or $\geq$ can be added to O_- of $\mathsf{NDR_6}$ (e.g. $(\leq, 5) \not\twoheadrightarrow_{\mathbb{R}} (=, 5) \vee (\leq, 4)$).

5   Related Work and Conclusions

Datatypes have been extensively studied in the context of DLs [3, 6, 8]. Extensions of expressive DLs with datatypes have been examined in depth [6] with the main focus on decidability. Baader, Brandt and Lutz [3] formulated tractable extensions of $\mathcal{EL}$ with datatypes using a p-admissibility restriction for datatypes. A datatype $\mathcal{D}$ is p-admissible if (i) satisfiability and implication of conjunctions of datatype restrictions can be decided in polynomial time, and (ii) $\mathcal{D}$ is convex: if a conjunction of datatype restrictions implies a disjunction of datatype restrictions then it also implies one of its disjuncts [3]. In our case instead of condition (i) we require that implication and satisfiability of just datatype restrictions (not conjunctions) is decidable in polynomial time since we do not consider functional features. Condition (ii) is relaxed to the requirement of safety for NDRs since we take into account not only the domain of the datatypes and the types of

restrictions but also the polarity of their occurrences. The relaxed restrictions allow for more expressive usage of datatypes in tractable languages, as demonstrated by the example given in the introduction. Furthermore, Baader, Brandt and Lutz did not provide a classification of datatypes that are p-admissible; in our case we provide such a classification for natural numbers, integers, rationals and reals. The EL Profile of OWL 2 [2] is inspired by $\mathcal{EL}^{++}$ and restricts all OWL 2 datatypes to satisfy p-admissibility in such a way that only equality can be used. Our result can allow for a significant extension of datatypes in the OWL 2 EL Profile, where in addition inequalities can be used negatively.

Our work is not the only one where the convexity property is relaxed without losing tractability. It has been shown [8] that the convexity requirement is not necessary provided that (i) the ontology contains only concept definitions of the form $A \equiv C$, where A is a concept name, and (ii) every concept name occurs at most once in the left-hand side of the definition. In some applications this requirement can be too restrictive since it disallows the usage of general concept inclusion axioms (GCIs), such as the axiom (2) given in the introduction, which do not cause any problem in our case.

In this work we made a fine-grained analysis of extensions of $\mathcal{EL}$ with numerical datatypes, focusing not only on the types of relations but also on the polarities of their occurrences in axioms. We made a full classification of cases where these restrictions result in a tractable extension for natural numbers, integers, rationals and reals. One practically relevant case for these datatypes is when positive occurrences of datatype expressions can only use equality and negative occurrences can use any of the numerical relations considered. This case was motivated by an example of a pharmacy-related ontology and can be proposed as a candidate for a future extension of the OWL 2 EL Profile. For the cases where the extension is tractable, we provided a polynomial sound and complete consequence-based reasoning procedure, which can be seen as an extension of the completion-based procedure for $\mathcal{EL}$. We think that the procedure can be straightforwardly extended to accommodate other constructors in $\mathcal{EL}^{++}$ such as (complex) role inclusions, nominals, domain and range restrictions and assertions since these constructors do not interact with datatypes [9]. We hope to investigate these extensions in future works.

In future work we also plan to consider other OWL datatypes, such as strings, binary data or date and time, functional features, and to try to extend the consequence-based procedure for Horn $\mathcal{SHIQ}$ [10] with our rules for datatypes. For example, to extend the procedure with functional features, we probably need a notion of "functional safety" for an NDR that corresponds to the strong convexity property (see Definition 4). In order to achieve even higher expressivity for datatypes we shall study how to combine different restrictions on the datatypes occurring in an ontology so that tractability is preserved. For example, using two safe NDRs in a single ontology may result in intractability, as is the case for NDR_1 and NDR_6 for integers (see Table 3). One possible solution to this problem is to specify explicitly which features can be used with which NDRs in order to separate their usage in ontologies.

References

1. Baader, F., Calvanese, D., McGuinness, D.L., Nardi, D., Patel-Schneider, P.F.: The Description Logic Handbook: Theory, Implementation, and Applications. 2nd edn. Cambridge University Press (2007)
2. Grau, B.C., Horrocks, I., Motik, B., Parsia, B., Patel-Schneider, P.F., Sattler, U.: OWL 2: The next step for OWL. J. Web Sem. **6**(4) (2008) 309–322
3. Baader, F., Brandt, S., Lutz, C.: Pushing the $\mathcal{EL}$ Envelope. In: IJCAI, Professional Book Center (2005) 364–369
4. Cote, R., Rothwell, D., Palotay, J., Beckett, R., , Brochu, L.: The systematized nomenclature of human and veterinary medicine. Technical report, Northfield, IL: College of American Pathologists (1993)
5. The Gene Ontology Consortium: Gene Ontology: Tool for the unification of biology. Nature Genetics **25** (2000) 25–29
6. Lutz, C.: Description Logics with Concrete Domains-A Survey. In: Advances in Modal Logic, King's College Publications (2002) 265–296
7. Magka, D., Kazakov, Y., Horrocks, I.: Tractable extensions of the description logic $\mathcal{EL}$ with numerical datatypes. Technical report, Oxford University Computing Laboratory (2010) posted on `http://web.comlab.ox.ac.uk/isg/people/despoina.magka/publications/reports/NDRTechnicalReport.pdf`.
8. Haase, C., Lutz, C.: Complexity of Subsumption in the $\mathcal{EL}$ Family of Description Logics: Acyclic and Cyclic tboxes. In: ECAI. Volume 178., IOS Press (2008) 25–29
9. Baader, F., Brandt, S., Lutz, C.: Pushing the $\mathcal{EL}$ envelope further. In: In Proceedings of the OWLED 2008 DC Workshop on OWL: Experiences and Directions. (2008)
10. Kazakov, Y.: Consequence-driven Reasoning for Horn $\mathcal{SHIQ}$ Ontologies. In: IJCAI. (2009) 2040–2045

Supporting the Development of Data Wrapping Ontologies (Extended Abstract)⋆

Lina Lubyte and Sergio Tessaris

KRDB Research Centre, Free University of Bozen-Bolzano

1 Introduction

The use of a conceptual model or an ontology to wrap and describe relational data sources has been shown to be very effective in several frameworks involving management and access of data, such as information integration through mediated schemata [1], and the Semantic Web [2]. Ontologies provide a conceptual view of the application domain, which is closer to the user perspective, and automated reasoning can be leveraged to support exploration and querying of the underlying data sources.

In this paper we focus on the problem of designing ontologies which describe relational data sources, and whose purpose is to provide a semantically enriched access to the underlying data. We use the term *data wrapping ontologies* to distinguish these ontologies from domain ontologies; whose purpose is to model a domain.

In order to maximise the benefits of using data wrapping ontologies, these should be rich enough to ease their integration with the domain ontology and, at the same time, precisely characterise the data they wrap. Ontologies extracted automatically from data sources (e.g. by analysing the constraints in the logical schema) are faithful representations of the data sources; however, they are usually shallow and with a limited vocabulary. For this reason, they can be used as bootstrap ontologies, and the task of enriching the extracted ontology is crucial in order to build a truly effective ontology-based information access system. The process of enriching an ontology involves at least the introduction of new axioms and/or new terms. While, from a purely ontological viewpoint, an ontology can be arbitrarily modified, we need to bear in mind that the ultimate purpose of the data wrapper is to access the information available from the data sources. This means that newly introduced terms (concepts or roles) should be "backed" by data in the sources; i.e. queries over these terms should be rewritable w.r.t. data sources.

It is easy to provide examples where newly introduced terms will always return empty answers, regardless the actual data contained in the sources (see Section 3). This not necessarily because they are unsatisfiable in the usual model

---

theoretic meaning, but because there is no way of mapping them into the data sources.

In order to ensure that queries over ontologies wrapping data sources provide sensible answers, these ontologies must be carefully handcrafted by taking into account the query answering algorithm. To the best of our knowledge, little or no research has been devoted to the support of the ontology engineer in such a complex and error prone task. Our research is directed to techniques and tools to support this modelling process.

In [3] we introduced the problem and presented some preliminary results. The contribution of this paper is a generalisation of the these results, by providing algorithms to verify term emptiness for a more expressive class of ontology languages (see [4]). In particular, a crucial gain in terms of expressive power of the language adopted in this work is the ability to express inclusions among roles. Moreover we provide a technique to support the user in the "repair" of the empty terms and we present empirical study showing the benefits of our approach.

2   Preliminaries

To formalize ontologies, we use the DL $\mathcal{ELHI}$ [4]. For P an *atomic role*, an $\mathcal{ELHI}$ *basic role* has the form P or P^-. For A an *atomic concept*, an $\mathcal{ELHI}$ *basic concept* has the form $A, \exists R, \exists R.A$ or $B_1 \sqcap B_2$, where R is a basic role. An $\mathcal{ELHI}$ ontology is formalized in terms of a *TBox*, which is a set of *inclusion assertions* of the form $B_1 \sqsubseteq B_2$ or $R_1 \sqsubseteq R_2$, with B_1, B_2 basic concepts and R_1, R_2 basic roles. The actual data instances are instead stored in an *ABox*, that consists of a set of *membership assertions* of the form $A(a)$ or $R(a, b)$, with A an atomic concept, P an atomic role, and a, b constants[1]. An $\mathcal{ELHI}$ *knowledge base* (KB) $\mathcal{K}$ is a tuple $\langle \mathcal{T}, \mathcal{A} \rangle$, where $\mathcal{T}$ is a TBox and $\mathcal{A}$ is an ABox. We assume the "standard" DL semantics, with the *unique name assumption*.

A datalog rule is an expression of the form $\alpha(\boldsymbol{x}) \leftarrow body(\boldsymbol{x}, \boldsymbol{y})$, where $\alpha(\boldsymbol{x})$ is the *head* atom and $body(\boldsymbol{x}, \boldsymbol{y})$ is a set of *body* atoms. A *datalog program* Π is a set of datalog rules. The *extensional database (EDB) predicates* of Π are those that do not occur in the head atom of any rule in Π; all other predicates are called *intentional database (IDB) predicates*. A *datalog query* Q over an $\mathcal{ELHI}$ KB $\mathcal{K}$ is a tuple $\langle Q_\Pi, \Pi \rangle$, where Q_Π is a query predicate and Π is a datalog program whose predicates (except Q_Π) are concept and role names occurring in $\mathcal{K}$. Q is a *conjunctive query* (CQ) if Π contains exactly one rule with Q_Π as its head predicate not occurring in the body. A tuple of constants $\boldsymbol{a}$ is a *certain answer* to a datalog query Q over $\mathcal{K}$ iff $\mathcal{K} \cup \Pi \models Q_\Pi(\boldsymbol{a})$, where Π is considered to be a set of universally quantified implications with the usual first-order semantics. We use $cert(Q, \mathcal{K})$ to denote the set of all certain answers to Q over $\mathcal{K}$.

---

[1] As a matter of fact, an ABox is considered only *virtually*, while the actual data is stored in a relational DBMS and *wrapped* by means of an ontology; see [5].

3 Emptiness of Ontology Terms

The foundation of our technique is the problem of verifying the emptiness of a given term w.r.t. a set of data source terms (i.e. terms "connected" to data sources). Given a Description Logic (DL) theory composed by TBox and ABox over a given vocabulary, we define a subset of the concepts and roles as *data source* terms. Given a TBox, a concept or role term is empty iff the certain answer of the query defined by the term is empty for all possible ABoxes whose assertions are restricted to data source terms. The idea is that data (by means of ABox assertions) can only be associated to data source terms. Clearly the problem is different from classical (un)satisfiability, because we impose a restriction on the kind of allowed ABox assertions. Note that the two problems coincide when all the DL terms are considered as data sources.

To provide an intuition of the reasoning task let us consider a simple example depicted in Figure 1, where the bottom part represents the logical schema, the middle part the data source terms (connected with the relational sources by means of mappings, depicted with dashed arrows) and the top part the enriched fragment of the ontology. It is obvious that any query on Actor would always return empty answer, whatever the data sources may contain; while the concept represented by the same term would be satisfiable. The situation would be different if Actor was also restricted to elements whose range w.r.t. person_role was bound to ActingRole[2]. In this case, there could be instances of the database for which the same query on Actor would return a nonempty answer.
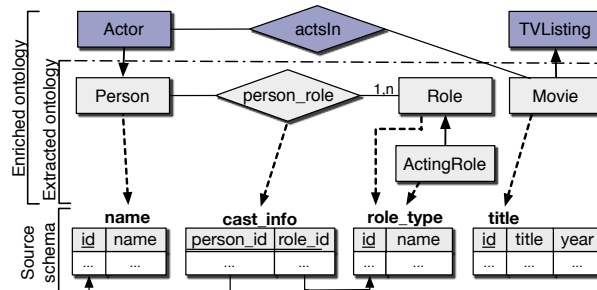
Fig. 1. Example of a simple data wrapper

Let $\Sigma_{\mathcal{DB}}$ denote the subset of terms occurring in $\mathcal{T}$ as "coming" from the data sources, i.e. data source terms. Given such $\Sigma_{\mathcal{DB}}$, a $\Sigma_{\mathcal{DB}}$-*ABox* is an ABox defined over $\Sigma_{\mathcal{DB}}$ only. Given a term η in $\mathcal{T}$, we call a *query for* η a CQ of the form $Q(x) \leftarrow \eta(x)$ (resp., $Q(x,y) \leftarrow \eta(x,y)$), for η an atomic concept (resp., an atomic role) in $\mathcal{T}$. Our goal is to test whether η is empty w.r.t. the data at the

---

[2] In DL terms this corresponds to an inclusion assertion $\exists$person_role.ActingRole $\sqsubseteq$ Actor.

sources, i.e., w.r.t. $\Sigma_{\mathcal{DB}}$. Clearly, such a test should involve the query answering process. That is, to verify emptiness of η, we have to check whether a query for η is empty given a TBox and a $\Sigma_{\mathcal{DB}}$-ABox.

Definition 1. *Let $\mathcal{T}$ be an $\mathcal{ELHI}$ TBox and η a term in $\mathcal{T}$ with query Q for η. Then, η is empty w.r.t. $\Sigma_{\mathcal{DB}}$ iff $cert(Q, \langle \mathcal{T}, \mathcal{A} \rangle) = \emptyset$ for every $\Sigma_{\mathcal{DB}}$-ABox $\mathcal{A}$.*

This defines the problem studied in this paper: given a term η in $\mathcal{T}$ with a CQ Q for η, test whether $cert(Q, \langle \mathcal{T}, \mathcal{A} \rangle) = \emptyset$ for every $\mathcal{A}$ whose assertions are over $\Sigma_{\mathcal{DB}}$ only. Note however that this does not imply that we will be necessarily computing $cert(Q, \langle \mathcal{T}, \mathcal{A} \rangle)$.

It is well known that the problem of computing certain answers in the presence of an incomplete database is often solved via *query rewriting* under constraints. Specifically, from [6] we have that given a conjunctive query Q over an $\mathcal{ELHI}$ KB $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$, we can compute another query Q', a *rewriting* of Q, such that the certain answers of Q over $\mathcal{K}$ and the answers of Q' over $\mathcal{A}$ only coincide, i.e., $cert(Q, \langle \mathcal{T}, \mathcal{A} \rangle) = cert(Q', \mathcal{A})$. Thus, we have the following:

Lemma 1. *Let $\mathcal{T}$ be an $\mathcal{ELHI}$ TBox and η a term in $\mathcal{T}$ with query Q for η. Let Q' be a rewriting of Q. Then, η is empty w.r.t. $\Sigma_{\mathcal{DB}}$ iff $cert(Q', \mathcal{A}) = \emptyset$ for every $\Sigma_{\mathcal{DB}}$-ABox $\mathcal{A}$.*

The above lemma shows that the problem of testing emptiness of a given term amounts to verifying whether the rewriting of its query returns empty answer for every possible $\Sigma_{\mathcal{DB}}$-ABox. We will see later that for this purpose we will not need to compute the actual evaluation, however, we will employ the above relationship as described in the sequel.

4   Testing Emptiness

The rewriting of a CQ over $\mathcal{ELHI}$ KB is a datalog query [6]. Therefore, according to Lemma 1, our problem now comes down to testing emptiness of a query predicate in the rewritten datalog program. The problem of verifying emptiness of datalog predicates has been addressed by Vardi [7], showing that deciding emptiness of IDB predicates can be done in polynomial time. The key idea underlying this result is the observation that a datalog program can be viewed as an infinite union of CQs that, in turn, can be described by means of *expansion trees*. Importantly, [7] shows that we can get rid of variables when building expansion trees, obtaining *skeletons of expansion trees*. Then, an IDB predicate is empty in a datalog program, iff there is no skeleton tree for that predicate having as leaves EDB predicates only. We build our approach on the results of [7], and in particular on the possibility of building finitely labelled trees for IDB predicates.

For a term η with a CQ Q for η in an $\mathcal{ELHI}$ TBox $\mathcal{T}$, we devise our emptiness testing algorithm in four steps: *(i)* rewrite Q using procedure of [6], obtaining a datalog query $Q' = \langle Q_{\Pi}, \Pi \rangle$, *(ii)* add to Π auxiliary rules for making IDB

and EDB predicates explicit, *(iii)* for the resulting Datalog program with a query predicate Q_Π, build an AND-OR skeleton tree for Q_Π, and *(iv)* traverse the obtained tree by marking its nodes as empty/nonempty corresponding to empty/nonempty predicates, and, in turn, to empty/nonempty concepts and roles in $\mathcal{T}$. In the following we will elaborate on steps *(ii)-(iv)*; for details on the rewriting algorithm we refer to [6].

Given a datalog program Π with a query predicate Q_Π resulting from rewriting a CQ for a given term over $\mathcal{T}$, let Π^* denote a datalog program obtained by adding to Π rules of the form:

- $A(x) \leftarrow A(x)$, $P(x,y) \leftarrow P(x,y)$, for every predicate symbol $A, P \in \Pi$ corresponding to an atomic concept and role in $\mathcal{T}$, respectively, such that $A, P \notin \Sigma_{\mathcal{DB}}$ and A, P do not occur among the head atoms of any rule in Π;
- $A(x) \leftarrow A_{db}(x)$, $P(x,y) \leftarrow P_{db}(x,y)$ for every predicate symbol $A, P \in \Pi$ such that $A, P \in \Sigma_{\mathcal{DB}}$.

Note that an auxiliary rule $A(x) \leftarrow A(x)$ is equivalent to a tautology $A(x) \lor \neg A(x)$; thus, from a logical point of view, we do not change the semantics of Π.

The following definition describes the AND-OR skeleton tree that is associated to a datalog program (we assume all rules in Π^* are named).

Definition 2. *Given a datalog program Π^* and an IDB predicate Q_Π in Π^*, the associated* AND-OR skeleton tree for Q_Π in Π^*, *denoted* $\mathsf{tree}(Q_\Pi, \Pi^*)$, *is a labelled tree consisting of alternating levels of and-nodes and or-nodes such that*

- *the root of* $\mathsf{tree}(Q_\Pi, \Pi^*)$ *is a (and-)node labelled by* Q_Π,
- *for every and-node labelled by a predicate R in* $\mathsf{tree}(Q_\Pi, \Pi^*)$ *and for every rule r of Π^* having R as its head predicate, there exists a child* or-node *of R labelled by r,*
- *for every or-node labelled by a rule r in Π^*,* $\mathsf{tree}(Q_\Pi, \Pi^*)$ *has an* and-node *child for every atom g in the body of r, and the label of each such and-node is the predicate symbol of g.*

An and-node labelled by R in $\mathsf{tree}(Q_\Pi, \Pi^*)$ *is a leaf, if either* (i) *it is labelled with an EDB predicate,* (ii) *there are no rules in Π^* having R predicate in the head, or* (iii) *there is some other and-node in* $\mathsf{tree}(Q_\Pi, \Pi^*)$ *labelled by R that has already been expanded; we refer to such node as the* expanded equivalent of R, *denoted $eq(R)$.*

An or-subtree τ in $\mathsf{tree}(Q_\Pi, \Pi^*)$ *is a subtree of* $\mathsf{tree}(Q_\Pi, \Pi^*)$ *such that* (i) *for an and-node $R \in \tau$, τ contains one of the child or-nodes of R in* $\mathsf{tree}(Q_\Pi, \Pi^*)$, (ii) *for an or-node $r \in \tau$, τ contains all children of r in* $\mathsf{tree}(Q_\Pi, \Pi^*)$.

Example 1. Consider the data wrapping ontology from Figure 1. We list below the relevant axioms:

$$\text{Movie} \sqsubseteq \text{TVListing} \qquad \exists \text{actsIn.Actor} \sqsubseteq \text{Movie} \qquad \text{Actor} \sqsubseteq \text{Person}$$
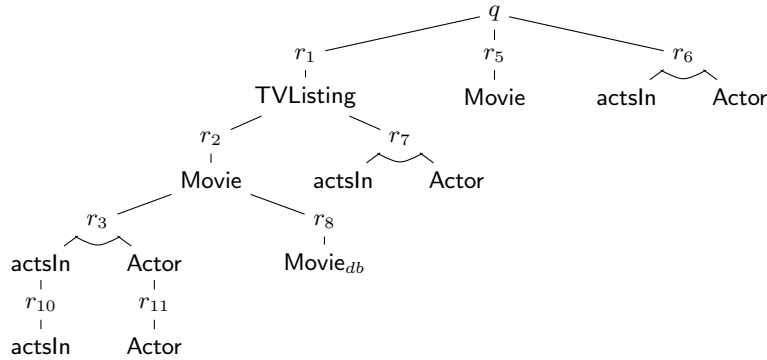
Fig. 2. Skeleton tree corresponding to the Datalog program of Example 1.

As can be seen from the figure, Movie and Person are linked to the data sources, i.e. they are data source terms. Suppose we want to test emptiness of TVListing term in the above ontology. The (partial) datalog program resulting from rewriting the query $q(x) \leftarrow$ TVListing(x) is given below, together with auxiliary rules r_8 through r_{11} to make actsIn and Actor IDB predicates, and Person and Movie EDB predicates.

$r_1 : q(x) \leftarrow$ TVListing(x) $r_7 :$ TVLising$(x) \leftarrow$ actsIn$(y,x),$ Actor(y)

$r_2 :$ TVListing$(x) \leftarrow$ Movie(x) $r_8 :$ Movie$(x) \leftarrow$ Movie$_{db}(x)$

$r_3 :$ Movie$(x) \leftarrow$ actsIn$(y,x),$ Actor(y) $r_9 :$ Person$(x) \leftarrow$ Person$_{db}(x)$

$r_4 :$ Person$(x) \leftarrow$ Actor(x) $r_{10} :$ actsIn$(x,y) \leftarrow$ actsIn(x,y)

$r_5 : q(x) \leftarrow$ Movie(x) $r_{11} :$ Actor$(x) \leftarrow$ Actor(x)

$r_6 : q(x) \leftarrow$ actsIn$(y,x),$ Actor(y)

The AND-OR skeleton tree for this datalog program is shown in Figure 2. Note that the children and-nodes of r_{10}, r_{11}, r_7, r_5 and r_6 are not further expanded, since they have isomorphic nodes that have already been expanded. The tree has 5 distinct or-subtrees, one of them e.g. formed from the path of or-nodes r_1, r_2 and r_8.

It is easy to show that each or-subtree of a given AND-OR skeleton tree corresponds to a skeleton of expansion tree defined in [7]. Therefore, a query predicate Q_Π is empty, iff all or-subtrees of tree(Q_Π, Π^*) are empty.

Definition 3. *Given an AND-OR skeleton tree* tree(Q_Π, Π^*) *for Q_Π in Π^*, an and-node R is empty in* tree(Q_Π, Π^*) *if either* (i) *there is the expanded equivalent of R, $eq(R)$, that is empty in* tree(Q_Π, Π^*); (ii) *R is a leaf in* tree(Q_Π, Π^*), *it is not an EDB predicate, and there is no $eq(R)$ in* tree(Q_Π, Π^*); (iii) *all children or-nodes of R are empty. An or-node r is empty in* tree(Q_Π, Π^*) *if at least one child and-node of r is empty.*

The above definition provides the basis for a procedure for traversing a given AND-OR skeleton tree. While emptiness of Q_Π node can be decided by inspecting leaf nodes only, our algorithm traverses all the tree; this information will

be the main input for suggesting the "repairs" of empty terms, as described in Section 5. We illustrate this process with the following example.

Example 2 (Example 1 continued). We start with actsIn leaf, child of r_{10}, and mark it as empty (it is not an EDB predicate). This makes also its parent r_{10} and, in turn, actsIn and r_3 empty. To decide for Movie, we have to know emptiness of r_8. Movie$_{db}$ is an EDB predicate, so it is nonempty. Consequently, we mark r_8 and Movie as nonempty, which determines non-emptiness for r_2 and then TVListing, r_1 and finally q. Actor leaf, child of r_{11}, is empty as well. Consequently, children of r_7 and r_6 are empty. In contrary, Movie, child of r_5 is marked as nonempty, because its expanded equivalent, child of r_2, is nonempty.

Indeed, we can construct a CQ $q(x) \leftarrow$ Movie$_{db}(x)$ from the AND-OR skeleton tree that witnesses non-emptiness for TVListing.

According to [6] and due to the fact that the input query for a given term has always single atom in its body, we have that the number of rules generated by the rewriting algorithm is exponential w.r.t. $\mathcal{T}$. Given n distinct IDB predicates in Π^*, the size of the AND-OR tree generated from Π^* is at most nm, where m is the maximum number of atoms in the body of a rule in Π^*. Thus, we have the following.

Theorem 1. *Let $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ be a $\mathcal{ELHI}$ KB, η a term in $\mathcal{T}$ and $\Sigma_{\mathcal{DB}}$ set of data source terms. Emptiness of η w.r.t. $\Sigma_{\mathcal{DB}}$ can be decided in time exponential in the size of $\mathcal{T}$.*

Note that the above result is optimal w.r.t. the complexity bounds from [8]: deciding emptiness of a term in $\mathcal{ELI}$ there is shown to be ExpTime-complete.

Finally, we stress the fact that, due to the rewriting algorithm [6], the technique presented in this section is applicable to ontology languages in the full spectrum of DLs from $\mathcal{ELHI}$ to *DL-Lite$_{core}$* [9].

5 Repairing Empty Terms

So far, we have devised a procedure for verifying whether a given term in a data wrapping ontology is empty w.r.t. the database terms at the sources. We now present a method for supporting the repair of empty concepts and roles, consisting of a set of *repairing axioms* that can be seen as guidelines for ontology engineers.

To suggest a repair for an empty term, we naturally resort to the datalog program Π^* and the skeleton tree generated from Π^* by our emptiness testing algorithm. Indeed, the skeleton tree for a term η, by virtue of its construction, contains as nodes all and only *relevant* terms for η: those that contribute or *could* contribute to its non-emptiness. So an intuitive way to fix an empty term is to *focus* on the relevant nodes of its corresponding skeleton tree and to possibly *expand* those nodes by rendering them nonempty. The expansion should obviously be in correspondence with an addition or refinement of a term or/and assertion in the actual ontology.

Given an or-subtree τ in an AND-OR skeleton tree $\mathsf{tree}(Q_\Pi, \Pi^*)$ with all nodes marked, let $\Omega = [\omega_1, \ldots, \omega_n]$ denote the sequence of distinct sets of and-nodes[3] in τ, such that, intuitively, each ω_i contains a set of and-nodes that are empty in $\mathsf{tree}(Q_\Pi, \Pi^*)$ and are grouped in a bottom-up fashion by their depth. The next example illustrates this notion.

Example 3. Suppose rule r_8 was not present in the tree of Figure 4. Hence, TVListing is no longer nonempty. Ω defined above for the or-subtree following r_2, r_3 ancestors of TVListing is the following sequence: $[\{\mathsf{actsIn}, \mathsf{Actor}\}, \{\mathsf{Movie}\},$ $\{\mathsf{TVListing}\}]$. The intuition here is that, in order for TVLisintg to become nonempty, besides rendering TVListing itself nonempty, also Movie or both, actsIn and Actor, if rendered nonempty, would make TVListing nonempty as well. Instead the reason for a depth based ordering is that if both, actsIn and Actor were made nonempty, then the remaining terms in the sequence Movie and TVListing would become nonempty as well.

Thus, for each and-node R in ω, we consider R as a leaf in the tree and examine its possible expansions. In turn, to expand a leaf we need a new rule with its corresponding atom in the head. Given such a rule, we can track down the needed terms and assertions in the ontology and provide those repairs as guidelines to the user. We exploit axioms in the ontology, rather than rules in the program, because, by virtue of [6], not all axioms are in one-to-one correspondence with rules in the computed rewriting.

For a node R corresponding to an atomic concept, say A, our repair service provides the following guidelines. First, it suggests to add an inclusion assertion with A on the right-hand side (line 6). This, from the modeling point of view, results in either defining role typing constraints (or domain and range) for a relationship defined by role P, if such is detected by means of mandatory participation constraints (and similarly if range restriction is given for P). Second, if $A \sqsubseteq B$ is present in $\mathcal{T}$ and B is nonempty, the user is warned with misplaced is-a relationship, i.e., possibly $B \sqsubseteq A$ should have been added instead of $A \sqsubseteq B$. Third, given $A \sqsubseteq B$ in $\mathcal{T}$ such that B has participation constraints to a nonempty role P, the algorithm suggests to assert participation constraints for A to P as well (and similarly if range restriction is known for P). Moreover, given a range concept, say C for P, if C is specialized by some concept D in the ontology, then the suggested axiom for A can also be specialized to D. Finally, the service suggests to assert A as a superclass of some concept B, and as a participating class to some role P, provided both A and P are known to be nonempty in $\mathcal{T}$. When $\mathcal{T}$ is small, such axioms could be included in the set of repairing axioms for every nonempty concept and role in $\mathcal{T}$. Otherwise, the task of selecting appropriate concepts and roles is left to the user.

If a given node R corresponds to a role, say P, the service generates axioms in a similar fashion. First, as before, it warns for misplaced role inclusions, provided such an axiom is present in $\mathcal{T}$. Then, if a root node being considered for repair

---

[3] Two and-nodes are considered distinct if their labels are distinct.

is a concept and not a role[4], then for every nonempty atomic concept A in $\mathcal{T}$ acting as a domain or range of P, the service suggests to add an axiom stating mandatory participation for A to the relationship defined by P (and the same for more specific concepts, as above). Finally, it hints to add an inclusion assertion between roles with P on the right-hand side, i.e. to make P more general than some role S that is nonempty in $\mathcal{T}$.

Note that the set of repairing axioms may also be empty, if there are no nonempty nodes in the tree that can be used for repair. In this case, we suggest to explicitly map to the sources either the actual empty term or any of its relevant terms.

Example 4 (Example 3 continued). Consider a data wrapping ontology from Figure 1 and suppose Movie is not mapped to the sources. To repair actsIn the user will be suggested to assert it as more general than person_role. This is obviously not meaningful, so there is no repair for actsIn. As for Actor, our repair service suggests the following axioms:

$$\begin{array}{ll} \mathsf{Person} \sqsubseteq \mathsf{Actor} & \exists\mathsf{person\_role}.\mathsf{Role} \sqsubseteq \mathsf{Actor} \\ \exists\mathsf{person\_role} \sqsubseteq \mathsf{Actor} & \exists\mathsf{person\_role}.\mathsf{ActingRole} \sqsubseteq \mathsf{Actor} \end{array}$$

6 Evaluation

We have implemented services discussed in Sections 4 and 5 as a plug-in for Protégé 3.3[5] (we are in the process of porting them to Protégé 4) and evaluated their effectiveness with a usability study involving ten external users (see [5] for details).

We used showbiz domain for the study. In particular, for the sources, we used *IMDB* movie database, retrieved using IMDbPY[6]. The wrapping ontology, that we call showbiz, was obtained by first automatically extracting the bootstrap ontology from *IMDB* database together with mappings [10] (21 in total), and then by manually enriching it with terms and assertions to (partly) describe TV programmes. The obtained ontology contained 24 classes and 14 properties.

The subjects were randomly divided into two groups: five subjects without the support for testing emptiness of ontology terms and repairing them (group 1), and five subjects with the support of the above described plug-in (group 2). Then, each subject was given four simple queries over showbiz ontology but having empty answers: e.g. asking for all movies that have a genre, all TV listings and their kinds, etc. Given that, the subjects were asked to add to the ontology new assertions so that the given queries were no longer empty. This involved identifying atoms responsible for query emptiness and repairing the corresponding terms. The subjects in group 2 were additionally asked to fill in a questionnaire concerning their experience using the tool. The goal of this study

---

[4] While our procedure computes repairs for a contributive node, with a root node here we mean the node that one actually aims to repair, as e.g. TVListing in Example 3.

[5] `http://protege.stanford.edu`

[6] `http://imdbpy.sourceforge.net/`

was to compare the time taken and effort needed to complete the task between the two groups, and to evaluate user experience in using the plug-in.

The results of the study are promising. While the assertions added to an ontology in order to arrive to a solution were mostly correct and alike in both groups, the time taken to do it in group 2 was between 2-3 times less than in group 1. Specifically, the average time taken for group 1 was 39 minutes, and 20 minutes for group 2. The average number of changes made to the ontology in order to repair given queries, which we consider to be as key sub-task, for group 1 was 11, and 6 for group 2. The total number of changes needed for all queries was 5. This means that, in average, each subject in group 1 made 5 *erroneous changes* to repair the given queries, while in group 2 – 1 erroneous change.

As mentioned, we have also collected user reactions to the tool. The questionnaire used for this purpose was composed of 10 short statements (e.g., "I found repair guidelines to be adequate"), each accompanied by a 5-point scale of "strongly disagree" (1 point) to "strongly agree" (5 points). Thus, given 5 subjects in group 2, each statement scores to maximum of 25 points. The key aspects, from the usability point of view, are that subjects in group 2 felt that they could effectively identify the reason for query emptiness using the tool (rated a total score of 19) and effectively repair empty terms using the tool (21 points), and strongly agreed that they could identify empty classes/properties and fix them using the tool faster than without it (25 points). Finally, the overall satisfaction of using the plug-in scores to 25.

7   Conclusions

This paper presents a technique for supporting ontology engineers in the development of ontologies for accessing relational data sources. We introduced the notion of emptiness of a given term w.r.t. a DL theory where data can be accessed only through a subset of the concepts and roles (analogously to the EDB/IDB predicates distinction in datalog programs). We have presented an optimal practical algorithm for deciding emptiness of terms in $\mathcal{ELHI}$ ontologies. Moreover, we have shown how the information generated by this algorithm can be exploited in order to support the engineer in "repairing" the ontology. The algorithm presented can be applied in other scenarios, e.g. for optimizing the rewriting by removing rules with empty predicates, or for guiding module extraction based on nonempty terms only (see [8]).

Recently there has appeared a contribution [8], carried out independently, that tackles a very similar problem but comes up in a different context. The authors study the computational complexity for the problem of predicate (and query) emptiness for a wide range of DLs. For the DLs $\mathcal{EL}$ and *DL-Lite*, they provide algorithms for verifying emptiness, taking a different approach from ours. While our algorithm is via translation to emptiness of IDB predicates in datalog, [8] instead uses reduction to standard ABox reasoning. Using their simple technique, emptiness of a term in $\mathcal{EL}$ can be decided in PTime. For $\mathcal{ELI}$, testing emptiness is shown to be already ExpTime-complete, by reduction to

subsumption/instance checking, but no algorithm is provided for this problem. As we mentioned, our practical algorithm is optimal w.r.t. the complexity bounds established there.

Levy [12] defined, in the context of datalog optimization, so-called *irrelevance claims* stating that a formula is irrelevant to a query w.r.t. a knowledge base and proposed algorithms for deciding irrelevance. However, this notion is rather different in nature from the emptiness problem we studied in this paper. In particular, it is a premise of a proof which may or may not be relevant to the deduction of a given formula. Therefore, those techniques cannot be directly applied.

Finally, we refer to the work in [13] as related, where, for queries having answers solely determined by the database predicates (the so-called DBox predicates with closed semantics, as apposed to the ABox), the authors show how to find a rewriting over such predicates. The restriction to determinacy may be however in some cases too strong, as for instance TVListing in Figure 1 is not *determined* by the database predicates but can be (in the classical setting) *rewritten* to a database term.

References

1. Lenzerini, M.: Data integration: A theoretical perspective. In: Proc. of PODS'02, ACM (2002) 233–346
2. Berners-Lee, T., Hendler, J., Lassila, O.: The semantic web. Scientific American (2001)
3. Lubyte, L., Tessaris, S.: Supporting the design of ontologies for data access. In: Workshop Notes of DL'08. (2008)
4. Baader, F., Brandt, S., Lutz, C.: Pushing the $\mathcal{EL}$ envelope. In: Proc. of IJCAI'05. (2005) 364–369
5. Lubyte, L., Tessaris, S.: Supporting the development of data wrapping ontologies. In: Proc. of ASWC'09. (2009) 31–45
6. Pérez-Urbina, H., Motik, B., Horrocks, I.: Tractable query answering and rewriting under description logic constraints. J. of Applied Logic (2009)
7. Vardi, M.Y.: Automata theory for database theoreticians. In: Proc. of PODS'89, ACM (1989) 83–92
8. Baader, F., Bienvenu, M., Lutz, C., Wolter, F.: Query and predicate emptiness in description logics. In: Proc. of KR'10. (2010) To appear.
9. Calvanese, D., Giacomo, G.D., Lembo, D., et al.: Tractable reasoning and efficient query answering in description logics: The DL-Lite family. J. of Automated Reasoning **39**(3) (2007) 385–429
10. Lubyte, L., Tessaris, S.: Automatic extraction of ontologies wrapping relational data sources. In Bhowmick, S., Küng, J., Wagner, R., eds.: Proc. of DEXA 2009. Volume LNCS 5690 of Springer. (2009) 128–142
11. Levy, A.Y.: Irrelevance Reasoning in Knowledge Based Systems. PhD thesis, Stanford University (1993)
12. Seylan, I., Franconi, E., de Bruijn, J.: Effective query rewriting with ontologies over DBoxes. In: Proc. of IJCAI'09. (2009) 923–930

Updating TBoxes in *DL-Lite*

Dmitriy Zheleznyakov, Diego Calvanese, Evgeny Kharlamov[*], and Werner Nutt

KRDB Research Centre
Free University of Bozen-Bolzano, Italy
`zheleznyakov, calvanese, kharlamov, nutt@inf.unibz.it`

Abstract. We study the problem of updates for TBoxes represented in Description Logics of the *DL-Lite* family. *DL-Lite* is at the basis of OWL 2 QL, one of the tractable fragments of OWL 2, the recently proposed revision of the Web Ontology Language. In this paper, we address for the first time the problem of updating TBoxes. We propose some principles that TBox updates should respect. We review known model- and formula-based approaches for updates of logical theories, and exhibit limitations of model-based approaches to handle TBox updates. We propose a novel formula-based approach, and present a polynomial time algorithm to compute TBox updates for $DL\text{-}Lite_{\mathcal{FR}}$. We also study the relationship between propositional logic satisfiability for Horn clauses and computation of TBox updates for *DL-Lite*.

1   Introduction

Ontology languages, and hence Description Logics (DLs), provide excellent mechanisms for representing structured knowledge, and as such they have traditionally been used for modeling at the conceptual level the static and structural aspects of application domains [1]. A family of DLs that has received great attention recently, due to its tight connection with conceptual data models, such as the Entity-Relationship model and UML class diagrams, is the *DL-Lite* family [2]. Such a family of DLs exhibits nice computational properties, in particular when complexity is measured wrt the size of the data stored in the ABox of a DL ontology [2, 3]. It is also at the basis of the tractable profiles of OWL 2, the forthcoming edition of the W3C standard Web Ontology Language.

The reasoning services that have been investigated for the currently used DLs and implemented in state-of-the-art DL reasoners [4], traditionally focus on so-called standard reasoning, both at the TBox level (e.g., TBox satisfiability, concept satisfiability and subsumption wrt a TBox), and at the ABox level (e.g., knowledge base satisfiability, instance checking and retrieval, and more recently query answering) [5, 6]. Recently, however, the scope of ontologies has broadened, and they are now considered to be not only at the basis of the design and development of information systems, but also for providing support in the maintenance and evolution phase of such systems. Moreover, ontologies are considered to be the premium mechanism through which services operating in a Web context can be accessed, both by human users and by other services.

---

[*] The author is co-affiliated with INRIA Saclay, Île-de-France.

Supporting all these activities, makes it necessary to equip DL systems with additional kinds of inference tasks that go beyond the traditional ones, most notably that of *ontology evolution* [7], where new knowledge is incorporated into an existing KB. Two main types of ontology evolution have been considered, namely revision and update [8].

In *revision*, we assume that the new knowledge is certainly true in the real world. Therefore, every model of a revised KB should satisfy this knowledge and should have minimal distance to the old KB, where the notion of distance depends on the application. An important feature of revision is that the distance is defined "globally" and depends on all the models of the old KB. In [9, 10] revision of DL knowledge bases was considered. In *update*, we assume that the new knowledge reflects a change in the real world, and we update every model of the old KB with this new knowledge. Note that update operators, in contrast to revsion operators, work "locally". In our work we focus on ontology update.

A request for an ontology update (or simply *update request*) represents the need of changing an ontology so as to take into account changes that occur in the domain of interest described by the ontology. In general, such a request is represented by a set of formulas denoting those properties that should be true after the change. In the case where the update request causes an undesirable interaction with the knowledge encoded in the ontology, e.g., by causing the ontology or relevant parts of it to become unsatisfiable, the update request cannot simply be added to the ontology. Instead, suitable changes need to be made in the ontology so as to avoid the undesirable interaction, e.g., by deleting parts of the ontology that conflict with the update request. Different choices are possible in general, corresponding to different update semantics, which in turn give rise to different update results [11]. Moreover, it is necessary to understand whether the desired update result can be represented at all as a KB in the DL at hand.

Previous work on updates in the context of DL ontologies has addressed ABox (or instance level) update [12, 13], where the update request consists of a set of ABox assertions. In [12] the problem is studied for DLs of the *DL-Lite* family, while [13] considers the case of expressive DLs. Both works show that it might be necessary to extend the ontology language with additional features/constructs in order to guarantee that the updated ontology can be represented.

Instead, the problem of TBox level update has not been considered before. In this paper we take first steps at filling this gap. Specifically, for the case of DLs of the *DL-Lite* family, we study the problem of updating a TBox with a set of TBox assertions. We address first the issue of which semantics to adopt for TBox updates, and specify some general principles that updates should respect. This leads us to argue that none of the previously proposed semantics [14–17], neither model-based nor formula-based is totally appropriate: either too many formulas need to be thrown out in the result of the update, or such a result is not representable as a *DL-Lite* TBox. Hence, we propose an alternative formula-based semantics, called *Bold Semantics* , and provide polynomial time algorithms to compute it for various members of the *DL-Lite* family (we restrict the attention to those DLs of the *DL-Lite* family that exhibit polynomial time TBox reasoning, specifically we consider only the case where the interaction between functionality assertions and role inclusions is restricted). The task at the core of our algorithm is the problem of checking *full satisfiability* of a *DL-Lite* TBox, i.e., whether
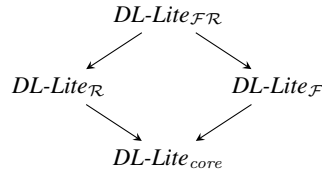
Fig. 1. *DL-Lite* hierarchy.

all atomic concepts and roles are (simultaneously) satisfiable. We provide a novel algorithm for this problem that is based on a reduction to reasoning in propositional binary Horn theories. This gives us also an alternative proof technique that is not based on the Chase for tractability of TBox reasoning in *DL-Lite*.

2   Preliminaries

Description Logics (DLs) [18] are knowledge representation formalisms, tailored for representing the domain of interest in terms of *concepts* and *roles*. In DLs, complex concept and role expressions (or simply, concepts and roles) are obtained starting from atomic concepts and roles (which are simply names) by applying suitable constructs. Concepts and roles are then used in a DL knowledge base (KB) to model the domain of interest. Specifically, a DL KB $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ is formed by two distinct parts, a *TBox* $\mathcal{T}$ and an *ABox* $\mathcal{A}$. The TBox $\mathcal{T}$ represents the *intensional-level* of the KB, that is, the general knowledge. The ABox provides information on the *instance-level* of the KB. In this paper we focus on a family of DLs called *DL-Lite* [2], that corresponds to one of the tractable fragments of OWL 2, the recently proposed revision of the Web Ontology Language.

   The basic logic of the *DL-Lite* family is *DL-Lite$_{core}$*, which includes constructs that are used in all others logics of the family. These constructs are the following:

$$B ::= A \mid \exists R, \qquad C ::= B \mid \neg B, \qquad R ::= P \mid P^-,$$

where A denotes an *atomic concept*, B a *basic concept*, and C a *general concept*. The symbol P denotes an *atomic role*, and R a *basic role*.

   A *DL-Lite$_{core}$* TBox is a set of *concept inclusion assertions* of the form: $B \sqsubseteq C$, and an ABox is a set of *membership assertions* of the form: $A(a)$, $P(a,b)$.

   The two logics *DL-Lite$_{\mathcal{F}}$* and *DL-Lite$_{\mathcal{R}}$* both extend *DL-Lite$_{core}$*. They have ABoxes of the same form as *DL-Lite$_{core}$*, but their TBoxes are different. A *DL-Lite$_{\mathcal{F}}$* TBox may include *functionality assertions* for roles of the form (funct R). *DL-Lite$_{\mathcal{R}}$* has *role inclusion assertions* of the form $R_1 \sqsubseteq R_2$ (instead of functionality assertions). There are proposals that consider *DL-Lite$_{\mathcal{R}}$* also with role disjointness assertions of the form $R_1 \sqsubseteq \neg R_2$, but we do not take them into account in our paper. Both *DL-Lite$_{\mathcal{F}}$* and *DL-Lite$_{\mathcal{R}}$* have nice computational properties, for example, knowledge base satisfiability has polynomial-time complexity in the size of the TBox and logarithmic-space complexity in the size of the ABox, so-called *data complexity*.

DL-Lite$_{\mathcal{FR}}$ is a hybrid of *DL-Lite$_{\mathcal{F}}$* and *DL-Lite$_{\mathcal{R}}$*. It allows for both functional assertions and role inclusion assertions in its TBox. The use of functionality and role inclusion assertions together may lead to an increase in the complexity of reasoning. A way to avoid this is to introduce the following syntactic restriction: if $R_1 \sqsubseteq R_2$ appears in a TBox, then (funct R_2) is *not* in the TBox. Hence, when talking about *DL-Lite$_{\mathcal{FR}}$* knowledge bases in this paper, we assume they satisfy the syntactic restriction above.

In Figure 1 we list the four logics of the *DL-Lite* family and show the relationships between them in terms of expressiveness. If there is an arrow from a logic X to a logic Y in the figure, it means that the logic Y is more expressive than X.

The semantics of a DL is given in terms of first order interpretations. Let Δ be a fixed countably infinite set. All interpretations that we consider are over the same domain Δ.

An *interpretation* $\mathcal{I}$ is a function $\cdot^{\mathcal{I}}$ that assigns to each concept C a subset $C^{\mathcal{I}}$ of Δ, and to each role R a binary relation $R^{\mathcal{I}}$ over Δ in such a way that $A^{\mathcal{I}} \subseteq \Delta$, $P^{\mathcal{I}} \subseteq \Delta \times \Delta$, $(\neg B)^{\mathcal{I}} = \Delta \setminus B^{\mathcal{I}}$, and

$$(\exists R)^{\mathcal{I}} = \left\{ a \mid \exists a'. (a, a') \in R^{\mathcal{I}} \right\}, \qquad (R^-)^{\mathcal{I}} = \left\{ (a_2, a_1) \mid (a_1, a_2) \in R^{\mathcal{I}} \right\}.$$

An interpretation $\mathcal{I}$ is a *model* of an inclusion assertion $D_1 \sqsubseteq D_2$ if $D_1^{\mathcal{I}} \subseteq D_2^{\mathcal{I}}$. An interpretation $\mathcal{I}$ is a *model* of a functionality assertion (funct R) if R is a partial function over Δ, that is, $\mathcal{I} \models \forall x, y_1, y_2. \left(R^{\mathcal{I}}(x, y_1) \wedge R^{\mathcal{I}}(x, y_2) \right) \rightarrow y_1 = y_2$.

Given an assertion F and an interpretation $\mathcal{I}$, we denote by $\mathcal{I} \models F$ the fact that $\mathcal{I}$ is a model of F. A model $\mathcal{I}$ is a model of a knowledge base (KB) $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ (denoted as $\mathcal{I} \models \mathcal{K}$) if $\mathcal{I}$ is a model of each of the assertions of $\mathcal{T} \cup \mathcal{A}$. A KB is *satisfiable* if it has at least one model. A KB $\mathcal{K}$ logically implies an assertion F, written $\mathcal{K} \models F$, if all models of $\mathcal{K}$ are also models of F. Similarly, a TBox $\mathcal{T}$ *logically implies* an assertion F, written $\mathcal{T} \models F$, if all models of $\mathcal{T}$ are also models of F.

Let $\mathcal{T}$ be a set of TBox assertions. The deductive *closure of $\mathcal{T}$*, denoted $cl(\mathcal{T})$, is the set of all assertions that are entailed by $\mathcal{T}$. Clearly, the closure $cl(\mathcal{T})$ is quadratic in the number of atoms of $\mathcal{T}$ and can be computed in time polynomial wrt the size of $\mathcal{T}$.

3   Understanding TBox Updates

Let $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ be a KB and $\mathcal{U}$ be a set of (TBox or/and ABox) assertions, called an update request. What we want to study is how to "incorporate" the assertions $\mathcal{U}$ into $\mathcal{K}$, that is, to perform an *update* of $\mathcal{K}$. In this paper we consider only updates on the TBox level (*TBox updates*), that is, when $\mathcal{U}$ consists of TBox assertions only.

When dealing with updates, both in the knowledge management and the AI community, it is generally accepted that the updated KB $\mathcal{K}'$, or the *update* for short, should comply with the principle of *Minimality of Change* [11, 17], which states that the knowledge base should change as little as possible if new information is incorporated. There are different approaches to updates, suitable for particular applications, and the current belief is there is no general notion of minimality that will "do the right thing" under all circumstances [17]. A number of candidate semantics for updates have appeared in the literature [14–17]. All these approaches can be classified into two groups: *model-based* and *formula-based*.

Let us first understand what are the requirements for updates of knowledge bases and then review known model- and formula-based approaches.

3.1   Principles of TBox Updates

Let $\mathcal{T}$ be a TBox, B a basic concept, and R a basic role occurring in $\mathcal{T}$. We say that B (resp. R) is *satisfiable in* $\mathcal{T}$ if there is a model $\mathcal{I} \models \mathcal{T}$ of $\mathcal{T}$ such that $B^{\mathcal{I}} \neq \emptyset$ (resp. $R^{\mathcal{I}} \neq \emptyset$). If all the atomic concepts and roles occurring in $\mathcal{T}$ are satisfiable, then we say that $\mathcal{T}$ is *fully satisfiable*. Intuitively, a concept "makes sense" if one can instantiate it and we assume that we update TBoxes that make sense, that is, that are fully-satisfiable.

Satisfiability Preservation.   A TBox update is a modification of a KB on the schema level. Such updates make sense, for example, when a company decides to restructure, say, the sales department, and the update $\mathcal{U}$ consists of new requirements for the department. Our first concern is that updates should not make parts of the schema, or TBox constructs, useless, that is, unsatisfiable. For example, for a basic concept of an enterprise ontology, say the concept *Manager*, we want to reject updates that eliminate managers from the enterprise, that is, that force *Manager* to be unsatisfiable.

Protection.   Our next expectation is that the schema update of the sales department should not affect the schema of, say, the production department. At the same time we do not mind if it affects the schemas of other departments, like for instance accounting. That is, we would like some fragment of $\mathcal{T}$, denoted $\mathcal{T}_p$, to be *protected* from any changes, that is, we would like $\mathcal{T}_p$ to be kept in the KB after the update. Therefore, we *accept* an update request $\mathcal{U}$ only if $\mathcal{T}_p \cup \mathcal{U}$ is fully satisfiable, otherwise we *reject* $\mathcal{U}$.

To sum up these desiderata, we list our *update principles*.

Satisfiability Preservation. Updates should preserve satisfiability of basic concepts and roles.
Protection. Updates should preserve the protected fragment of the KB.

3.2   Model-Based Approach to Semantics

Poggi et al. [19, 12] proposed to use Winslett's semantics to update ABoxes. Let us try to understand whether this approach is suitable for TBox updates.

Under the model-based paradigm, the objects of change are individual models $\mathcal{I}$ of $\mathcal{T}$. For a model $\mathcal{I}$ of $\mathcal{T}$, an update with $\mathcal{U}$ results in a set of models of $\mathcal{U}$. In order to update the entire TBox $\mathcal{T}$ with $\mathcal{U}$, one has to

(i) update every model $\mathcal{I} \models \mathcal{T}$ with $\mathcal{U}$, and then
(ii) take the union of the resulting models.

To define the update formally we recall the following definitions. We say that an interpretation $\mathcal{I}$ *is contained in* $\mathcal{I}'$, written $\mathcal{I} \subseteq \mathcal{I}'$, if for every atomic concept or role symbol S it holds that $S^{\mathcal{I}} \subseteq S^{\mathcal{I}'}$. We write $\mathcal{I} \subsetneq \mathcal{I}'$ if $\mathcal{I} \subseteq \mathcal{I}'$ and not $\mathcal{I}' \subseteq \mathcal{I}$. We denote with $\ominus$ the symmetric difference between sets according to the standard definition.
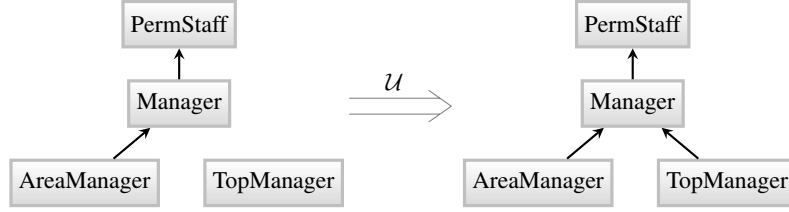
Fig. 2. Updates of ontologies. $\mathcal{U} = \{\mathit{TopManager} \sqsubseteq \mathit{Manager}\}$.

Let $\mathcal{T}_p \subseteq \mathcal{T}$ be the protected fragment of $\mathcal{T}$ and $\mathcal{U}$ an update request accepted for $\mathcal{T}_p$. The *update of an interpretation $\mathcal{I}$ with $\mathcal{U}$ wrt $\mathcal{T}_p$*, denoted $w\text{-}upd_{\mathcal{T}_p}(\mathcal{I}, \mathcal{U})$, where 'w' indicates Winslett's semantics, is the set of interpretations defined as follows:

$$\{\mathcal{I}' \mid \mathcal{I}' \in Mod(\mathcal{T}_p \cup \mathcal{U}), \text{there is no } \mathcal{I}'' \in Mod(\mathcal{T}_p \cup \mathcal{U}) \text{ s.t. } \mathcal{I} \ominus \mathcal{I}'' \subsetneq \mathcal{I} \ominus \mathcal{I}'\}.$$

Then the *update of a TBox $\mathcal{T}$ with $\mathcal{U}$ wrt $\mathcal{T}_p$* is the following set of interpretations:

$$w\text{-}upd_{\mathcal{T}_p}(\mathcal{T}, \mathcal{U}) = \bigcup_{\mathcal{I} \in Mod(\mathcal{T})} w\text{-}upd_{\mathcal{T}_p}(\mathcal{I}, \mathcal{U}).$$

Returning to a user the result of an update as a set of models is not desirable. What we want is to return a KB that describes exactly this set of models. We say that a TBox $\mathcal{T}'$ *represents* the update $w\text{-}upd_{\mathcal{T}_p}(\mathcal{T}, \mathcal{U})$ if $Mod(\mathcal{T}') = w\text{-}upd_{\mathcal{T}_p}(\mathcal{T}, \mathcal{U})$.

Example 1. Consider the TBox $\mathcal{T}$ of an enterprise on the left diagram of Figure 2. In *DL-Lite$_{core}$* the diagram can be written as follows:

$$\mathit{Manager} \sqsubseteq \mathit{PermStaff}, \qquad \mathit{AreaManager} \sqsubseteq \mathit{Manager},$$

where *PermStaff* stand for *Permanent Staff*. The TBox says that every *Manager* belongs to *PermStaff* and every *AreaManager* is a *Manager*. Suppose the TBox is under construction and it was decided to extend it by introducing the inclusion assertion that every *TopManager* is a *Manager*, that is,

$$\mathcal{U} = \{\mathit{TopManager} \sqsubseteq \mathit{Manager}\}.$$

Since there are no disjointness assertions in both $\mathcal{T}$ and $\mathcal{U}$, the update request will be accepted for $\mathcal{T}$, regardless of which fragment is protected, and the desired result of the update is the one in the right diagram of Figure 2. Unfortunately, Winslett's semantics gives an undesirable result.

First, consider the following model $\mathcal{I}$ of $\mathcal{T}$:

$$\mathit{TopManager}^{\mathcal{I}} = \{john\}, \quad \mathit{Manager}^{\mathcal{I}} = \{frank\}, \quad \mathit{PermStaff}^{\mathcal{I}} = \{frank\}.$$

Assume that the protected fragment of $\mathcal{T}$ is empty. Then, according to Winslett's semantics, the update of the model $\mathcal{I}$ contains the following interpretation $\mathcal{I}'$ is in the update of $\mathcal{I}$, which is a model of $\mathcal{U}$ that differs minimally from $\mathcal{I}$:

$$\mathit{TopManager}^{\mathcal{I}'} = \{john\}, \quad \mathit{Manager}^{\mathcal{I}'} = \{john, frank\}, \quad \mathit{PermStaff}^{\mathcal{I}'} = \{frank\}.$$

As one can see, in $\mathcal{I}'$ there is a *Manager*, *john*, who does *not* belong to *PermStaff*. Therefore, the update *w-upd*($\mathcal{T}, \mathcal{U}$) does *not* satisfy the assertion *Manager* $\sqsubseteq$ *PermStaff*.

Second, every *DL-Lite* representation $\mathcal{T}'$ of *w-upd*($\mathcal{T}, \mathcal{U}$) should satisfy the following assertions, which we denote as $\mathcal{T}_0$, that is, $\mathcal{T}_0 \subseteq \mathcal{T}'$:

$$\textit{TopManager} \sqsubseteq \textit{Manager}, \textit{AreaManager} \sqsubseteq \textit{Manager}, \textit{AreaManager} \sqsubseteq \textit{PermStaff}.$$

Is it the case that $\mathcal{T}' = \mathcal{T}_0$? It turns out that not. Consider the following model. Let $\mathcal{I}''$ be an interpretation, where all concepts are empty, except for *Manager*, which contains one individual, say *fred*. It is easy to see that $\mathcal{I}'' \models \mathcal{T}_0$ and $\mathcal{I}'' \models \mathcal{U}$, but it cannot be obtained by minimally changing a model of $\mathcal{T}$. Intuitively, there is no reason for *fred* to have become a *Manager*.

Therefore, there should be some other inclusion assertions in $\mathcal{T}'$, besides the ones of $\mathcal{T}_0$, that forbid the model $\mathcal{I}''$. One can see that these assertions should be entailed by $\mathcal{T} \cup \mathcal{U}$. Otherwise there are models of $\mathcal{T}$ whose update is not expressed by $\mathcal{T}'$. Hence, the only candidate to be included in $\mathcal{T}'$ is *Manager* $\sqsubseteq$ *PermStaff*, but it cannot be in $\mathcal{T}'$, due to the first observation above. Therefore, the update is not expressible in *DL-Lite*. □

We conclude that:
(i) Winslett's semantics cannot be expressed by *DL-Lite* TBoxes.
(ii) The principle of minimal change at the level of interpretations forces one to give up important assertions at the TBox level (in Example 1, we gave up the assertion *Manager* $\sqsubseteq$ *PermStaff*).

We consider this situation as unsatisfactory. Hence, we next examine the formula-based approach to updates and their notion of minimality.

3.3 Formula-Based Approach to Semantics

The key notion in this approach is the one of a *maximal non-contradicting set* of formulas, which we introduce now.

Let $\mathcal{T}$ be a TBox and $\mathcal{U}$ be an update request that is accepted for $\mathcal{T}_p$. We define a *maximal non-contradicting set of formulas for $\mathcal{T}$ and $\mathcal{U}$*, denoted by $\mathcal{T}_m$, as a set of TBox assertions that satisfies the conditions:

(i) $\mathcal{T} \models \mathcal{T}_m$,
(ii) $\mathcal{T}_m \cup \mathcal{U}$ is fully satisfiable,
(iii) the set $\mathcal{T}_m$ is maximal (wrt set inclusion) among the sets that satisfy *(i)* and *(ii)*, that is, there is no $\hat{\mathcal{T}}$ satisfying *(i)* and *(ii)* such that $\mathcal{T}_m \subset \hat{\mathcal{T}}$.

Intuitively, $\mathcal{T}_m$ keeps as many TBox assertions as possible that are entailed by $\mathcal{T}$ and do not conflict with $\mathcal{U}$.

Obviously, the set $\mathcal{T}_m$ is not unique. We denote the set of all such $\mathcal{T}_m$ for $\mathcal{T}$ and $\mathcal{U}$ as $\mathcal{M}(\mathcal{T}, \mathcal{U})$. There are two main approaches to construct updates $\mathcal{T}'$ of $\mathcal{T}$ with $\mathcal{U}$ based on $\mathcal{T}_m$ [11, 17].

WIDTIO. The first approach is called *When In Doubt Throw It Out*, or *WIDTIO* for short. It suggests to add to $\mathcal{U}$ the intersection of all $\mathcal{T}_m$-s, as on the left of Equation 1:

$$\mathcal{T}' = \mathcal{U} \cup \bigcap_{\mathcal{T}_m \in \mathcal{M}(\mathcal{T}, \mathcal{U})} \mathcal{T}_m, \qquad \mathcal{T}' = \mathcal{U} \cup \{ \bigvee_{\mathcal{T}_m \in \mathcal{M}(\mathcal{T}, \mathcal{U})} (\bigwedge_{\phi \in \mathcal{T}_m} \phi) \}. \qquad (1)$$

Cross-Product. According to this approach, one adds to $\mathcal{U}$ the disjunction of all $\mathcal{T}_m$-s, viewing each $\mathcal{T}_m$ as the conjunction of its assertions, as on the right of Equation 1.

Example 2. Consider the *DL-Lite* ontology from Example 1 (Figure 2) and the update request $\mathcal{U} = \{AreaManager \sqsubseteq \neg PermStaff\}$. It is easy to see that $\mathcal{U} \cup \mathcal{T}$ is not fully satisfiable and in order to resolve the conflict one can drop either $Manager \sqsubseteq PermStaff$ or $AreaManager \sqsubseteq Manager$. Thus, $\mathcal{M}(\mathcal{T}, \mathcal{U}) = \{\mathcal{T}_m^{(1)}, \mathcal{T}_m^{(2)}\}$, where $\mathcal{T}_m^{(1)} = \{Manager \sqsubseteq PermStaff\}$, and $\mathcal{T}_m^{(2)} = \{AreaManager \sqsubseteq Manager\}$. Let us now consider WIDTIO and Cross-Product semantics. According to the left formula of Equation 1, the TBox under WIDTIO semantics is equal to

$$\mathcal{U} \cup \left(\mathcal{T}_m^{(1)} \cap \mathcal{T}_m^{(2)} \right) = \mathcal{U} \cup \emptyset = \mathcal{U} = \{AreaManager \sqsubseteq \neg PermStaff\}.$$

The TBox under Cross-Product semantics is

$$\mathcal{U} \cup \{(Manager \sqsubseteq PermStaff) \vee (AreaManager \sqsubseteq Manager)\},$$

where we have combined DL notation with First Order Logic notation. □

As one can see from the example above, a disadvantage of the WITDIO approach is that it may lose a lot of assertions entailed by $\mathcal{T}$ that *do not* conflict with $\mathcal{U}$. On the other extreme is the Cross-Product approach that suggests to keep all possible entailed and not conflicting assertions. A drawback of the approach is that the result of the update cannot be represented in *DL-Lite* anymore since it requires disjunction. Another drawback is that the resulting set of formulas may be exponentially large wrt the original TBox.

Therefore, any practical solution should be one where one chooses *some* $\mathcal{T}_m^{(0)}$ among the $\mathcal{T}_m$, where the result of the update is:

$$\mathcal{T}' = \mathcal{U} \cup \mathcal{T}_m^{(0)}.$$

We call this semantics *Bold Semantics.* The question is which $\mathcal{T}_m$ to choose. There are basically three options. Choose *(i)* an arbitrary one, *(ii)* one that has maximal cardinality, *(iii)* one that fulfills some preferences. For all options, the solution is expressible in *DL-Lite*. Note that we rely for this on the fact that in *DL-Lite* the set of assertions entailed by a TBox is finite.

The first option has the advantage that $\mathcal{T}'$ is expressible in *DL-Lite* and can be computed in polynomial time. Figure 3 presents a nondeterministc algorithm that, given a TBox $\mathcal{T}$ and an update request $\mathcal{U}$, returns a set $\mathcal{T}_m \subseteq cl(\mathcal{T})$ that is a maximal non-contradicting set of assertions for $\mathcal{T}$ and $\mathcal{U}$. The algorithm loops at most as many times as there are assertions in $cl(\mathcal{T})$. The number of such assertions is at most quadratic in

> **INPUT:**   sets $\mathcal{T}, \mathcal{U}$ of TBox assertions, $\mathcal{T}_p \subseteq \mathcal{T}$ fully satisfiable with $\mathcal{U}$
> **OUTPUT:**   a set $\mathcal{T}_m \subseteq cl(\mathcal{T})$ of TBox assertions
> **[1]** $\mathcal{T}_m := \mathcal{U} \cup \mathcal{T}_p; \mathcal{S} := cl(\mathcal{T})$
> **[2]**   **repeat**
> **[3]**   **choose some** $\phi \in \mathcal{S}; \mathcal{S} := \mathcal{S} \setminus \{\phi\}$
> **[4]**   **if** $\{\phi\} \cup \mathcal{T}_m$ is fully satisfiable **then** $\mathcal{T}_m := \mathcal{T}_m \cup \{\phi\}$
> **[5]**   **until** $\mathcal{S} = \emptyset$

Fig. 3. Algorithm *NDMax*$(\mathcal{T}, \mathcal{T}_p, \mathcal{U})$ for nondeterministic computation of $\mathcal{T}_m$

the number of atomic concepts and roles. The crucial step is a check for full satisfiability, which is performed once per loop. If the latter test is polynomial in the size of the input, like in *DL-Lite*$_{\mathcal{FR}}$ (see Section 4), then the entire runtime of the algorithm is polynomial. For the second option we showed $\mathcal{T}_m$ computation is NP-hard, but we cannot present the proof due to lack of space. The third option is good as far as one has reasonable preferences either on the concepts or assertions of the TBox, that gives us polynomial time computation.

Example 3. Consider the KB and the update request from Example 2. As it has been mentioned, $\mathcal{M}(\mathcal{T}, \mathcal{U}) = \{\mathcal{T}_m^{(1)}, \mathcal{T}_m^{(2)}\}$. According to the Bold Semantics computed by the algorithm *NDMax*, the result of the update is a TBox $\mathcal{T} = \mathcal{U} \cup \mathcal{T}_m^{(0)}$ for some $\mathcal{T}_m^{(0)} \in \mathcal{M}(\mathcal{T}, \mathcal{U})$. Thus, the result of the update is either $\mathcal{U} \cup \{AreaManager \sqsubseteq Manager\}$ or $\mathcal{U} \cup \{Manager \sqsubseteq PermStaff\}$. In the former case, the ontology makes sense if manageres could be temporary staff, in the latter one, if area managers are not necessarily managers. Selecting one or the other of these two options could be done by the use of preferences. But we do not consider this here.   □

Theorem 4 (Correctness of Semantics). *Bold Semantics satisfies the principles of Satisfiability Preservation and Protection.*

4   Checking Full Satisfiability

Testing full satisfiability is the key operation in computing updates under Bold Semantics. We show that for *DL-Lite*$_{\mathcal{FR}}$ the problem of checking full satisfiability of a TBox can be translated into a problem of propositional Horn logic. The translation can be used as the starting point for the design of efficient algorithms and it provides additional insight as to why full satisfiabilty can be solved in polynomial time for *DL-Lite*$_{\mathcal{FR}}$.

As a first step, we define a translation function ν that translates TBoxes $\mathcal{T}$ into propositional theories $\nu(\mathcal{T})$. For every basic concept B resulting from the signature of $\mathcal{T}$ we introduce a fresh propositional variable v_B and for every basic role R we introduce the two variables $v_{\exists R}$ and $v_{\exists R^-}$ and denote the set of all such variables as $\mathcal{V}_{\mathcal{T}}$. Then $\nu(\mathcal{T})$ consists of all propositional formulas that can be obtained from $\mathcal{T}$ using the translation in Table 1.

Let $\mathcal{V}$ be a set of propositional variables and $\mathcal{F}$ a set of formulas over $\mathcal{V}$. Then we say that $\mathcal{F}$ is *fully satisfiable* (over $\mathcal{V}$) if $\mathcal{F} \cup \{v\}$ is satisfiable for every $v \in \mathcal{V}$.

TBox assertion ϕ	PL formulas $\nu(\phi)$
$B_1 \sqsubseteq B_2$	$v_{B_1} \rightarrow v_{B_2}$
$B_1 \sqsubseteq \neg B_2$	$v_{B_1} \rightarrow \neg v_{B_2}$
$R_1 \sqsubseteq R_2$	$v_{\exists R_1} \rightarrow v_{\exists R_2},\ v_{\exists R_1^-} \rightarrow v_{\exists R_2^-}$

Table 1. Translation of *DL-Lite$_{\mathcal{FR}}$* TBoxes to propositional theories

Theorem 5. *Let $\mathcal{T}$ be a DL-Lite$_{\mathcal{R}}$ TBox. Then $\mathcal{T}$ is fully satisfiable if and only if $\nu(\mathcal{T})$ is fully satisfiable over $\mathcal{V}_{\mathcal{T}}$.*

Proof. The "only if" direction being clear, we only show the "if" direction.

Suppose that $\nu(\mathcal{T})$ is fully satisfiable over $\mathcal{V}_{\mathcal{T}}$. Then for every basic concept B there is a truth assignment α_B for the variables in $\mathcal{V}_{\mathcal{T}}$ such that $\nu(\mathcal{T}) \cup \{v_B\}$ is satisfiable. Intuitively, this can be seen as putting a test individual into B and letting α_B propagate this individual into additional concepts B' so that the inclusions in $\mathcal{T}$ are satisfied.

Now we choose, for every B, a distinct element $d_B \in \Delta$. Moreover, we define a mapping J that maps every basic concept B' to a subset of Δ by defining $J(B') = \{d_B \mid \alpha_B(v_{B'}) = \mathsf{true}\}$. Intuitively, $J(B')$ consists of all the test individuals d_B that ended up in B' by way of their α_B. Note that due to the construction we have that $J(B') \subseteq J(B'')$ whenever $B' \sqsubseteq B'' \in \mathcal{T}$.

We now define an interpretation $\mathcal{I}'$ by setting $A^{\mathcal{I}'} = J(A)$ for every atomic concept A and $P^{\mathcal{I}'} = J(\exists P) \times J(\exists P^-)$ for every atomic role P. That is, $P^{\mathcal{I}'}$ is the Cartesian product of the sets to which J maps the expressions for the domain and range of P. Clearly, in this way we have that $(\exists P)^{\mathcal{I}'} = J(\exists P)$ and $(\exists P^-)^{\mathcal{I}'} = J(\exists P^-)$. This shows that $\mathcal{I}'$ is a model of $\mathcal{T}$ such that $A^{\mathcal{I}'} \neq \emptyset$ and $P^{\mathcal{I}'} \neq \emptyset$ for all atomic A and P. $\qquad\square$

Note that the proof above shows as a byproduct that a fully satisfiable TBox can be fully satisfied by a finite model.

If $\mathcal{T}$ is a *DL-Lite$_{\mathcal{FR}}$*-TBox, we say that an atomic role P is *functional* if $\mathcal{T}$ contains (funct P) or (funct P^-). We say that P *has a subrole* if P or P^- occurs on the right-hand side of some role inclusion.

Lemma 6. *Let $\mathcal{T}$ be a DL-Lite$_{\mathcal{R}}$-TBox and $\mathcal{F}$ a set of functionality assertions. Suppose that no functional role in $\mathcal{T} \cup \mathcal{F}$ has a subrole. Then $\mathcal{T} \cup \mathcal{F}$ is fully satisfiable if $\mathcal{T}$ is fully satisfiable.*

Proof. Let $\mathcal{I}'$ be an intepretation that fully satisfies $\mathcal{T}$. Let D be the set of elements of Δ that are in the interpretation of some atomic concept or role. Without loss of generality we can assume that D is at most countable and that $\Delta \setminus D$ has at least countably many elements. Then there exist countably many sets $D_1, D_2, \ldots \subseteq \Delta$ such that *(i)* every set D_i, $i \in \mathbb{N}$, has the same cardinality as D and *(ii)* the D_i are mutually disjoint.

For every $i \in \mathbb{N}$, let $m_i \colon D \rightarrow D_i$ be a bijection. We use the m_i to extend the interpretations of atomic concepts from D to the union of the D_i. Technically, we define a new interpretation $\mathcal{I}$ by letting $A^{\mathcal{I}} = \bigcup_{i \in \mathbb{N}} m_i(A^{\mathcal{I}'})$ for every atomic concept A. The definition of the role interpretations needs some preparation. For every atomic role P, let $\delta'_P = (\exists P)^{\mathcal{I}'}$ be the domain of P with respect to $\mathcal{I}'$ and $\rho'_P = (\exists P^-)^{\mathcal{I}'}$ be the

range. We define $\delta_P = \bigcup_{i \in \mathbb{N}} m_i(\delta'_P)$ and, similarly, $\rho_P = \bigcup_{i \in \mathbb{N}} m_i(\rho'_P)$. Note that due to our construction both δ_P and ρ_P are countably infinite.

Now, if P is functional, then let $P^{\mathcal{I}}$ be the graph of an arbitrary bijective function from δ_P to ρ_P. Otherwise, let $P^{\mathcal{I}} = \delta_P \times \rho_P$. Clearly, by construction we have that $(\exists P)^{\mathcal{I}} = \delta_P$ and $(\exists P^-)^{\mathcal{I}} = \rho_P$. Hence, $\mathcal{I}$ satisfies all concept inclusions of $\mathcal{T}$ and all functionality assertions.

Moreover, if $R \sqsubseteq R'$ is a role inclusion in $\mathcal{T}$, we have that $\delta'_R \subseteq \delta'_{R'}$ and $\rho'_R \subseteq \rho'_{R'}$, which implies that $\delta_R \subseteq \delta_{R'}$ and $\rho_R \subseteq \rho_{R'}$. Hence, $R^{\mathcal{I}} \subseteq R'^{\mathcal{I}}$, since R' is not functional and therefore $R'^{\mathcal{I}}$ is the Cartesian product of $\delta_{R'}$ and $\rho_{R'}$. This shows that $\mathcal{I}$ is a model of $\mathcal{T} \cup \mathcal{F}$.   □

Recall that in a *DL-Lite*$_{\mathcal{FR}}$ TBox, there can be no role inclusions with a functional role on the right hand side. In addition, we assume that TBoxes do not contain disjointness axioms for roles. Thus, the preceding lemma is applicable.

Theorem 7. *Let $\mathcal{T}$ be a DL-Lite*$_{\mathcal{FR}}$ *TBox. Then $\mathcal{T}$ is fully satisfiable if and only if $\nu(\mathcal{T})$ is fully satisfiable over $\mathcal{V}_{\mathcal{T}}$.*

Since satisfiability of a set of propositional Horn clauses can be checked in linear time, checking full satisfiablity can be done in time quadratic in the size of the clause set. In [2], polynomiality of concept satisfiablity in *DL-Lite*$_{\mathcal{FR}}$ has been proved using the Chase technique. The techniques used for showing Theorem 7 above provide an alternative proof.

5   Conclusion

To the best of our knowledge, our paper presents the first work on updates for DL TBoxes. We tried to understand what are the natural requirements for such updates and proposed two principles: Satisfiability Preservation and Protection. On the basis of these principles, we examined the well-known semantics for updates poposed by Winslett, which has already been applied by Poggi et al. [19] to ABox updates. The approach turned out to be unintuitive and moreover, the TBox languages of the *DL-Lite* family are not closed under such updates. As an alternative, we examined two formula-based approaches to update semantics: WIDTIO and the Product Approach. The former one leads to an inappropriate loss of knowledge, while for the latter update results are not expressible in *DL-Lite*. As a consequence, we proposed a new semantics for TBox updates, Bold Semantics, that satisfies both our principles. We showed that TBoxes resulting from updates under our semantics can be computed in polynomial time for *DL-Lite*$_{\mathcal{FR}}$. Moreover, we exhibited a tight connection between update computation and reasoning with propositional Horn formulas. This connection can be used as the starting point for the design of efficient update algorithms and it provides additional insight as to why TBox reasoning can be solved in polynomial time for *DL-Lite*$_{\mathcal{FR}}$.

References

1. Borgida, A., Brachman, R.J.: Conceptual modeling with description logics. [18] chapter 10 349–372
2. Calvanese, D., De Giacomo, G., Lembo, D., Lenzerini, M., Rosati, R.: Tractable reasoning and efficient query answering in description logics: The *DL-Lite* family. J. of Automated Reasoning **39**(3) (2007) 385–429
3. Artale, A., Calvanese, D., Kontchakov, R., Zakharyaschev, M.: The *DL-Lite* family and relations. J. of Artificial Intelligence Research **36** (2009) 1–69
4. Möller, R., Haarslev, V.: Description logic systems. [18] chapter 8 282–305
5. Haarslev, V., Möller, R.: On the scalability of description logic instance retrieval. J. of Automated Reasoning **41**(2) (2008) 99–142
6. Calvanese, D., De Giacomo, G., Lenzerini, M.: Conjunctive query containment and answering under description logics constraints. ACM Trans. on Computational Logic **9**(3) (2008) 22.1–22.31
7. Flouris, G., Manakanatas, D., Kondylakis, H., Plexousakis, D., Antoniou, G.: Ontology change: Classification and survey. Knowledge Engineering Review **23**(2) (2008) 117–152
8. Katsuno, H., Mendelzon, A.: On the difference between updating a knowledge base and revising it. In: Proc. of the 2nd Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR'91). (1991) 387–394
9. Qi, G., Du, J.: Model-based revision operators for terminologies in description logics. In: Proc. of the 21st Int. Joint Conf. on Artificial Intelligence (IJCAI 2009). (2009) 891–897
10. Flouris, G.: On belief change in ontology evolution. AI Communications—The Eur. J. on Artificial Intelligence **19**(4) (2006)
11. Eiter, T., Gottlob, G.: On the complexity of propositional knowledge base revision, updates and counterfactuals. Artificial Intelligence **57** (1992) 227–270
12. De Giacomo, G., Lenzerini, M., Poggi, A., Rosati, R.: On instance-level update and erasure in description logic ontologies. J. of Logic and Computation, Special Issue on Ontology Dynamics **19**(5) (2009) 745–770
13. Liu, H., Lutz, C., Milicic, M., Wolter, F.: Updating description logic ABoxes. In: Proc. of the 10th Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR 2006). (2006) 46–56
14. Abiteboul, S., Grahne, G.: Update semantics for incomplete databases. In: Proc. of the 11th Int. Conf. on Very Large Data Bases (VLDB'85). (1985)
15. Ginsberg, M.L., Smith, D.E.: Reasoning about action I: A possible worlds approach. Technical Report KSL-86-65, Knowledge Systems, AI Laboratory (1987)
16. Winslett, M.: A model-based approach to updating databases with incomplete information. ACM Trans. on Database Systems **13**(2) (1988) 167–196
17. Winslett, M.: Updating Logical Databases. Cambridge University Press (1990)
18. Baader, F., Calvanese, D., McGuinness, D., Nardi, D., Patel-Schneider, P.F., eds.: The Description Logic Handbook: Theory, Implementation and Applications. Cambridge University Press (2003)
19. De Giacomo, G., Lenzerini, M., Poggi, A., Rosati, R.: On the update of description logic ontologies at the instance level. In: Proc. of the 21st Nat. Conf. on Artificial Intelligence (AAAI 2006). (2006) 1271–1276

Orel: Database-Driven Reasoning for OWL 2 Profiles

Markus Krötzsch, Anees ul Mehdi, and Sebastian Rudolph

Institute AIFB, Karlsruhe Institute of Technology, DE
{mak,ame,sru}@aifb.uni-karlsruhe.de

Abstract. We describe Orel, a reasoning system for an ontology language which subsumes both the EL and the RL profile of the recently standardised web ontology language OWL 2. Orel performs consequence-driven reasoning on the database level which is always sound. It is guaranteed to be complete if the ontology is contained in one of the two profiles. We present the underlying calculus, the core algorithm, and initial evaluation results.

1 Introduction

With the standardisation of the Web Ontology Language OWL 2 in 2009 [1], the development of theoretically well-studied and practically deployable expressive ontology languages for the Semantic Web has reached a new level of maturity. Among various other improvements, the new version of OWL is the first that adequately addresses the trade-off between logical expressivity and scalability that is inherent to formal knowledge representation by specifying additional light-weight language profiles. The three OWL 2 profiles EL, RL, and QL constitute sublanguages which – while still sufficiently expressive for many applications – exhibit a polynomial time complexity for standard reasoning tasks, and are therefore particularly suitable for working with large ontological descriptions [2].

The Orel software that is introduced in this system description provides storage and reasoning services for both OWL EL and RL. The specific features that set it apart from existing implementations are twofold. First, its implementation is tailored toward materialisation of entailments in a persistent storage backend such as a relational database management system (DBMS). Second, it realises a rule-based approach for implementing both OWL RL and OWL EL inferencing in a single polytime algorithm.

Orel's approach to reasoning is to express inference tasks for OWL 2 in terms of inference tasks for the simple rule language datalog [3]. The basis of this method is an entailment-preserving translation of description logics to datalog that has been introduced in [4]. The latter approach has been presented for a hybrid ontology-rule language that includes features which cannot be expressed in OWL 2. This provides an interesting path for extending Orel to also cover some of the expressivity of rule languages like SWRL [5] or RIF-BLD [6], but the present paper focusses on the supported OWL 2 features only.

In Section 2, we discuss Orel's inferencing calculus, and present some optimisations for data-centric processing. Thereafter, in Section 3, we briefly highlight our basic approach for extending this inferencing mechanism to efficient schema inferencing, and

$PhD \sqsubseteq AcademicDegree$ $PostDoc \sqsubseteq \exists has.PhD$ $Graduate \equiv \exists has.AcademicDegree$

$PhD(x) \rightarrow AcademicDegree(x)$ $Graduate(x) \rightarrow has(x, d_{\exists has.AD})$

$PostDoc(x) \rightarrow has(x, d_{\exists has.PhD})$ $Graduate(x) \rightarrow AcademicDegree(d_{\exists has.AD})$

$PostDoc(x) \rightarrow PhD(d_{\exists has.PhD})$ $has(x, y) \wedge AcademicDegree(y) \rightarrow Graduate(x)$

Fig. 1. Example translation to datalog

in Section 4, we recall the general techniques for adapting a rule-based calculus for execution in a relational DBMS. Section 5 provides further details on the implementation and initial evaluation results. We discuss related work in Section 6 and give an outlook to the future development of Orel in Section 7. Orel is free software that can be obtained at `http://code.google.com/p/orel/`.

2 A Data-Driven Approach for Translating OWL into Datalog

The algorithms in [4] extend to a first-order knowledge representation language dubbed ELP that combines features of the description logic $\mathcal{EL}^{++}$ [7], Description Logic Rules [8], and DL-safe Rules [9]. Yet, the expressivity of ELP has been restricted sufficiently to allow for polynomial-time reasoning. Instead of repeating the formal details that can readily be found in [4], we summarise the underlying approach by means of a brief example, and provide more detailed descriptions of the algorithms that are actually implemented in Orel. Throughout this work, we use description logic syntax for concisely expressing the semantics of OWL 2 axioms.

As an example, consider the set of OWL 2 axioms in Fig. 1 (top). Following a strategy as in [4], this knowledge base would be translated into the rule set in Fig. 1 (bottom). These rules are intended to be read as first-order implications based on a standard predicate logic semantics.[1] Note that the translation is faithful regarding the signature: OWL classes are translated into unary predicates, and OWL properties into binary predicates. Thus it is not hard to see how axioms from the original ontology relate to implications in the translated datalog program.

While this translation is straightforward in many cases, a special approach is needed to cover existential expressions as in `ObjectSomeValuesFrom`. Since datalog does not allow existential entailments, auxiliary constants are introduced to represent additional "anonymous" individuals the existence of which is required by the ontology. Please note that only a single constant is introduced for affected class expressions during the translations. This limits the amount of additional individuals that need to be considered, and it is vital to retain polytime complexity.

While the above translation is rather intuitive for the most part, the presented encoding has several practical drawbacks that come to the fore when attempting an actual implementation. In particular, the created rule set may become rather large; it grows linearly with the size of the knowledge base. However, typical LP engines exhibit far

---

[1] Since we are only interested in positive entailments, assuming a non-monotonic semantics for datalog would not lead to different inference results. See [3] for details.

$$
\begin{array}{rcl|rcl}
A(n) & \mapsto & \mathtt{inst}(n,\hat{A}) & A \sqsubseteq C & \mapsto & \mathtt{subClass}(\hat{A},\hat{C}) \\
R(n,m) & \mapsto & \mathtt{triple}(n,R,m) & A \sqcap B \sqsubseteq C & \mapsto & \mathtt{subIntersect}(\hat{A},\hat{B},\hat{C}) \\
\exists R.\mathsf{Self}(n) & \mapsto & \mathtt{self}(n,R) & & & \\
\exists R.A \sqsubseteq C & \mapsto & \mathtt{subSomeValues}(R,\hat{A},\hat{C}) & \exists R.\mathsf{Self} \sqsubseteq C & \mapsto & \mathtt{selfImplies}(R,\hat{C}) \\
A \sqsubseteq \exists R.B & \mapsto & \mathtt{someValues}(\hat{A},R,\hat{B},d_{\exists R.B}) & A \sqsubseteq \exists R.\mathsf{Self} & \mapsto & \mathtt{impliesSelf}(\hat{A},R) \\
A \sqsubseteq \forall R.B & \mapsto & \mathtt{allValuesFrom}(\hat{A},R,\hat{B}) & A \sqsubseteq\, \leqslant 1\, R.B & \mapsto & \mathtt{atMostOne}(\hat{A},R,\hat{B}) \\
R \sqsubseteq T & \mapsto & \mathtt{subProperty}(R,T) & \mathsf{Disj}(R,S) & \mapsto & \mathtt{disjoint}(R,S) \\
R \circ S \sqsubseteq T & \mapsto & \mathtt{subPropertyChain}(R,S,T) & R^- \sqsubseteq S & \mapsto & \mathtt{subInverseOf}(R,S)
\end{array}
$$

For each individual name n in the ontology, add the fact $\mathtt{nom}(n)$ to the transformation.

For each class name or nominal A in the ontology, add the fact $\mathtt{subClass}(A,\top)$.

Fig. 2. Creating an initial fact base from DL axioms in Orel; for a class C define $\hat{C} := n$ if $C = \{n\}$ is a nominal class, and $\hat{C} := C$ if C is a class name, $\top$, or $\bot$

better performance when more facts and less rules are given. Similarly, DBMS can handle large amounts of data while implications in the above formulation work on this data and would thus correspond to database operations. The next section therefore introduces a modified approach that is taken in Orel. This observation calls for a different encoding strategy, where ontological information (such as subclass relationships) is stored as facts, while logical ramifications are governed by "meta-rules" that resemble the rules of a deduction calculus. Thereby, classes and properties have to be treated as datalog individuals. The above example might then be encoded by the following facts:

$$
\begin{aligned}
&\mathtt{subClass}(PhD, AcademicDegree) \\
&\mathtt{someValues}(PostDoc, Has, PhD, d_{\exists has.PhD}) \\
&\mathtt{someValues}(Graduate, Has, AcademicDegree, d_{\exists has.AD}) \\
&\mathtt{subSomeValues}(Has, AcademicDegree, Graduate)
\end{aligned}
$$

The predicate names used here hint at the intended interpretation but are not formally related to the OWL 2 vocabulary. Note that the auxiliary constants $d_{\exists has.PhD}$ and $d_{\exists has.AD}$ are already included in the above facts. Since we are interested in a rule set without function symbols (datalog), all required constant symbols must be explicitly created beforehand. We now can encode the intended semantics in derivation rules such as the following:

$$
\begin{aligned}
\mathtt{subClass}(a,b) \wedge \mathtt{inst}(x,a) &\rightarrow \mathtt{inst}(x,b) \\
\mathtt{someValues}(a,r,b,d) \wedge \mathtt{inst}(x,a) &\rightarrow \mathtt{triple}(x,r,d) \\
\mathtt{someValues}(a,r,b,d) \wedge \mathtt{inst}(x,a) &\rightarrow \mathtt{inst}(d,b) \\
\mathtt{subSomeValues}(r,a,b) \wedge \mathtt{triple}(x,r,y) \wedge \mathtt{inst}(y,a) &\rightarrow \mathtt{inst}(x,b),
\end{aligned}
$$

Here we encode assertions about instances in the obvious way with the additional meta-predicates $\mathtt{inst}$ for class instances, and $\mathtt{triple}$ for role assertions. All terms in the above rules are variables, but here and below we use different letters for capturing the underlying intuition: a, b, c for class names, r, s, t for role names, x, y, z for individual names, and d for auxiliary constants.

As in the above example, most features of OWL EL and RL can be supported by suitable meta-rules based on the datalog translation in [4]. For the most prominent features of the two profiles, the translation of axioms to meta-facts is specified in Fig. 2, and the according materialisation rules are presented in Fig. 3. The translation assumes

$$(1) \qquad \text{nom}(x) \rightarrow \text{inst}(x,x)$$
$$(2) \qquad \text{nom}(x) \wedge \text{triple}(x,r,x) \rightarrow \text{self}(x,r)$$
$$(3) \qquad \text{subClass}(a,b) \wedge \text{inst}(x,a) \rightarrow \text{inst}(x,b)$$
$$(4) \qquad \text{subIntersect}(a,b,c) \wedge \text{inst}(x,a) \wedge \text{inst}(x,b) \rightarrow \text{inst}(x,c)$$
$$(5) \qquad \text{subSomeValues}(r,a,c) \wedge \text{triple}(x,r,y) \wedge \text{inst}(y,a) \rightarrow \text{inst}(x,c)$$
$$(6) \qquad \text{someValues}(a,p,b,d) \wedge \text{inst}(x,a) \rightarrow \text{triple}(x,p,d)$$
$$(7) \qquad \text{someValues}(a,p,b,d) \wedge \text{inst}(x,a) \rightarrow \text{inst}(d,b)$$
$$(8) \qquad \text{selfImplies}(r,c) \wedge \text{self}(x,r) \rightarrow \text{inst}(x,c)$$
$$(9) \qquad \text{impliesSelf}(a,r) \wedge \text{inst}(x,a) \rightarrow \text{self}(x,r)$$
$$(10) \qquad \text{impliesSelf}(a,r) \wedge \text{inst}(x,a) \rightarrow \text{triple}(x,r,x)$$
$$(11) \qquad \text{subProperty}(r,t) \wedge \text{triple}(x,r,y) \rightarrow \text{triple}(x,t,y)$$
$$(12) \qquad \text{subProperty}(r,t) \wedge \text{self}(x,r) \rightarrow \text{self}(x,t)$$
$$(13) \quad \text{subPropertyChain}(r,s,t) \wedge \text{triple}(x,r,y) \wedge \text{triple}(y,s,z) \rightarrow \text{triple}(x,t,z)$$
$$(14) \qquad \text{disjoint}(r,s) \wedge \text{triple}(x,r,y) \wedge \text{triple}(x,s,y) \rightarrow \text{inst}(x,\bot)$$
$$(15) \qquad \text{inst}(x,y) \wedge \text{nom}(y) \rightarrow \text{inst}(y,x)$$
$$(16) \qquad \text{inst}(x,y) \wedge \text{nom}(y) \rightarrow \text{nom}(x)$$
$$(17) \qquad \text{triple}(x1,r,y) \wedge \text{inst}(x2,y) \wedge \text{nom}(y) \rightarrow \text{triple}(x1,r,x2)$$
$$(18) \qquad \text{allValuesFrom}(a,r,b) \wedge \text{nom}(x) \wedge \text{nom}(y) \wedge$$
$$\text{triple}(x,r,y) \wedge \text{inst}(x,a) \rightarrow \text{inst}(y,b)$$
$$(19) \; \text{atMostOne}(a,r,b) \wedge \text{nom}(x) \wedge \text{nom}(y_1) \wedge \text{nom}(y_2) \wedge \text{inst}(x,a) \wedge$$
$$\text{triple}(x,r,y_1) \wedge \text{inst}(y_1,b) \wedge \text{triple}(x,r,y_2) \wedge \text{inst}(y_2,b) \rightarrow \text{inst}(y1,y2)$$
$$(20) \qquad \text{subInverseOf}(r,s) \wedge \text{nom}(x) \wedge \text{nom}(y) \wedge \text{triple}(x,r,y) \rightarrow \text{triple}(y,s,x)$$

Fig. 3. Inference rules for deriving entailments in Orel

that all axioms have first been decomposed into a simplified normal form that does not use more than one concept operator per concept expression. To simplify the presentation, we use the names of classes, roles, and individuals, as well as $\top$ and $\bot$ as constant symbols in the database instead of assigning numerical identifiers to such names as done in the actual implementation.

Regarding the rules of Fig. 3, we can observe that the rules only derive new facts for the predicates `inst`, `triple`, and `self` that correspond to assertional axioms, as well as for the auxiliary predicate `nom`. To see the purpose of the latter, first note that a special simplification of the rule set is achieved by using the same identifiers for individual names and for nominal classes containing only this individual. Constants that can be considered as nominal classes are marked with `nom`, so that the rule (1) of Fig. 3 generates tautological assertions of the form $\{n\}(n)$. It is not hard to see that all equality statements that can be derived in OWL EL must involve at least one individual name, and can thus be expressed by a class assertion axiom for a nominal class. These observations allow us to simplify the equality theory of [4] to rules (15)–(17) of Fig. 3.

All rules that relate to features that are specific to OWL RL are restricted to individuals in `nom`. This corresponds to the restriction of DL-safety that has been also used in [4]. As noted there, the relevant entailments of an OWL RL ontology can be obtained when restricting reasoning to *named* individuals. *Anonymous* individuals, in contrast, cannot be inferred to exist in OWL RL and are only relevant for the EL part of a knowledge base. As discussed in [4], the DL-safe combination of EL and RL features not only captures all entailments that would be expected from either language in isolation,

but also allows some semantic interactions between the two languages. In this case, however, the above inferencing algorithm is not guaranteed to produce all entailments – indeed, a polynomial time algorithm cannot achieve this goal.

Features that are missing in Fig. 2 and 3 are only OWL EL's restricted form of property ranges, the universal role, and concrete domains (data ranges). Orel interprets all property ranges as OWL RL axioms of the form $\top \sqsubseteq \forall R.C$, and does not currently support the universal role. Concrete domains, however, are supported and the according rules are omitted here for reasons of space. Various other features, such as assymmetry of roles, that have been omitted above can readily be expressed in terms of the given features.

Finally, it should be observed that the given inference rules do not materialise facts that can be concluded if the knowledge base is inconsistent. However, it is ensured that inconsistencies lead to derivations of the form $\mathtt{inst}(n, \bot)$ for some constant n. Orel checks for this condition for being able to return correct answers without explicitly materialising all possible inferences in the database.

3   Schema Reasoning with Orel

The calculus that has been introduced above is able to derive assertional axioms such as the instances of an atomic concept. For complex concept expressions, it might be required to first extend the knowledge base with auxiliary axioms and to (re)complete the materialisation process thereafter. Such auxiliary axioms, however, are hardly affecting the semantics of the knowledge base since they conservatively extend it, and hence many such checks can safely be performed without resetting the database.

The matter is different when checking for the entailment of schema axioms such as concept subsumption. Indeed, there are practically important ontologies such as SNOMED CT which do not contain any individual names, and for which concept subsumption is the chief inferencing problem. It is well known that this problem can be reduced to instance retrieval: for checking if an axiom $A \sqsubseteq B$ is entailed, a new "test" individual c is introduced into the knowledge base together with the assertion $A(c)$. If this implies $B(c)$ then the subsumption is concluded.

Unfortunately, this approach to testing does not preserve the semantics of the knowledge base. Indeed, asserting $A(c)$ may even lead to a global inconsistency (in which case $B(c)$ and thus $A \sqsubseteq B$ is also entailed). Thus, test assertions disallow the naive parallel execution of many queries that could be considered typical for a database system, and they require possibly expensive deletion operations after the test is finished. While it is of course possible to execute each test on a separate copy of the database – possibly realised by marking facts in the database as belonging to a particular test instead of separating databases on the DBMS layer – this approach multiplies the data that has to be stored at each time, and reduces the performance gains due to the re-use of persistently stored previous computations.

The problem is less severe when restricting to smaller languages than OWL EL. For example, the algorithm described in [10] computes all concept subsumptions of an $\mathcal{ELH}$ knowledge base in parallel without executing separate tests for each. While $\mathcal{ELH}$ allows for this mode of reasoning, it is not clear how to establish such an algorithm for

$\mathcal{EL}^{++}$. In particular, the original algorithm as proposed in [7] is incomplete. The glitch can be fixed, but only at the price of specifying the subsumption axiom the entailment of which is to be tested before running the algorithm, thus requiring many runs instead of one. We conjecture that this is unavoidable.

Due to these difficulties, Orel uses a mixed approach for finding concept subsumptions. The calculus uses the simple rules that have been introduced above when this is guaranteed to yield correct results, but it creates additional copies of axioms when the computation results in derivations that cannot be handled in this manner. The goal of our approach is to avoid the significant overhead that is required in the general case whenever possible, but tuning the calculus for this purpose is subject of ongoing work. Currently, Orel's schema inferencing is most efficient when ontologies do not contain nominal classes (in certain problematic contexts), and it decreases in performance when combinations of nominals, existential quantifiers, and OWL RL features occur.

4   Applying Derivation Rules on RDBMS

Relational database management systems (RDBMS) are tailored toward the processing of large amounts of data, and the efficient manipulation of such data. As such they appear to be well-suited for implementing materialisation on a persistent storage system. However, inferencing operations are often still rather atypical for RDBMS since they involve large inner joins over all entries in a table. Moreover, RDBMS provide elaborate functions such as transaction management that are not required by typical inferencing scenarios but that can significantly slow down operations. For this reason, various optimisations are needed for using RDBMS as a basis for implementing the outlined inferencing procedure.

It is well-known that datalog rules are closely related to operations in relational algebra [3]. The correspondence is achieved by storing the extension of each datalog predicate in a database table, the columns of which correspond to the arguments taken by the predicate. Rule (3) of Fig. 3 could therefore be realised by the following SQL operation:

```
INSERT INTO inst (x,y) SELECT t1.x AS x, t2.y AS y
FROM subClass AS t1 INNER JOIN inst AS t2 ON t1.x=t2.y
```

Executing this SQL statement extends the `inst` table with all facts that can be derived in one application of rule (3) of Fig. 3. We provide this statement for illustrating the mapping to SQL commands – using it iteratively in an implementation would lead to prohibitively large amounts of unnecessary computations. Indeed, the operation derives the same conclusions in each iteration, just like the original rule does when processed operationally.

Various optimisations have been proposed and thoroughly investigated to overcome this problem [3]. One way to increase efficiency is to keep track of the iteration step in which a fact was derived, and to make sure that rule applications require new facts to be involved in the derivation. This leads to the so-called *semi-naive* bottom-up evaluation which is largely used in Orel. Writing inst^i for the predicate that corresponds to the extension of `inst` as derived in step *i*, this strategy boils down to evaluating the following rule:

$$\text{subClass}(x, y) \land \text{inst}^i(y, z) \rightarrow \text{inst}^{i+1}(x, z)$$

Unfortunately, semi-naive evaluation can still derive large numbers of redundant facts during inferencing. More efficient general purpose optimisations like *magic sets* are available when only certain entailments are of interest (typically at query time) but are not useful for full materialisation. But more efficient forward chaining algorithms do exist as well, and have been studied in the area of databases, and in particular in relation with transitive closure computations [11]. Since these approaches often assume very simple rule sets, they can not be directly adopted to the inference rules of Orel, and part of the ongoing development effort around the tool is to suitably adapt techniques from this area.

5   Implementation and Initial Results

Orel is implemented in Java, using the OWL API [12] for accessing OWL documents. The current default RDBMS that is used in Orel is MySQL although only minor adjustments would be needed to move to another RDBMS. Orel is free software and can be obtained (including its source code) from `http://code.google.com/p/orel/`.

The current implementation of Orel is still not fully optimised in various respects. On the one hand, we are exploring heuristics for improving the inferencing control flow. On the other hand, optimising database queries for a particular RDBMS is a tedious process with many dependencies on the technical infrastructure used in testing. We have found that different server setups and machine configurations can lead to a 50% reduction in ontology loading times while incurring a slow-down of several orders of magnitude for materialisation. Thus, while we cannot give reproducible evaluation figures, we can provide some first insights into general runtime behaviour.

The OWL 2 test cases[2] have been used to test the correctness of the implementation. For performance testing, we specifically focussed on the well-known SNOMED CT ontology, a medical terminology of about 425,000 axioms with a strong focus on subclass subsumptions. We considered loading and inference materialisation for this ontology. Load times have shown to be rather similar across very systems of diverse performance, typically ranging between 9min and 20min. These times reflect the slow inserting behaviour of relational databases – the given times are already based on an optimised loading phase that controls transaction management and indexing, and that exploits client-side caching and rewritten bulk updates. Yet, the writing speed is a strong limiting factor (computing the data for writing takes but a few seconds). Application areas for DBMS-based systems of course assume axioms to change at a slow rate, thus reducing the relevance of initial loading times.

Loading does not involve reasoning, i.e. materialisation. At the current stage of implementation, Orel is able to successfully classify SNOMED CT but it cannot compete with highly optimised in-memory systems like Condor [13]: almost 2 hours are needed on a fast database server. This reflects some of the limitations of using an off-the-shelf RDBMS, and we expect significant potential for speed-up by using alternative backends. Similar results have been reported for the SAOR inference engine for OWL Horst

---

[2] `http://owl.semanticweb.org/`

[14], and we are not aware of any system that uses MySQL as a reasoning backend. In spite of the comparatively low performance of the current implementation, we were still able accomplish major speed improvements for the classification by improving control structure and inference rules. Most of these optimisations are directly applicable to other backends as well.

6 Related Work

The objective or Orel is to provide a stable framework for OWL ontology management and inferencing based on persistent storage. Approaches of rule-based bottom-up materialisation of consequences have a long history, and Orel therefore can build on a significant amount of prior work, both practical and theoretical in nature.

On the theoretical side, there is a large body of well-established research to be found in the area of (relational) databases, especially related to the optimisation of recursive queries [3] and the construction of materialised views [15]. We have discussed herein only briefly the basic use of a semi-naive evaluation strategy, but other approaches are applicable in a similar fashion when optimising for further use cases. Typical examples for such techniques are magic sets (used for optimising complex bottom-up computations needed at query time) and incremental materialisation (used for efficiently recreating inferences when new data is added).

More recently, much work has been conducted on "no-SQL" approaches to persistent storage, leading to a number of database-like systems that are tailored toward improved efficiency for non-relational data schemes such as sets of RDF triples, JSON documents, or simple key value pairs. These developments can be beneficial for selecting more suitable storage backends for Orel in the future, but they are not directly related to the work on the current system. Indeed, Orel's architecture abstracts all storage operations so that inference and control structures do not refer to SQL or any other concrete DBMS feature in any way.

On the more practical side, there are a number of past and current systems that support rule-based inferencing on relational databases. We are not aware of any tool that supports more than a single OWL 2 profile based on such an approach, making Orel's multi-profile integration novel. Also, the overall architectures of systems differ significantly, even if rule-based inferences are used at the core. The main relationship to Orel therefore is in the actual reasoning module that saturates a knowledge base for a given set of rules, whereas functions such as checking ontology entailment are often highly specific to a given tool. In fact, we do not know of any freely available database-driven reasoner that can check ontology entailment for any OWL profile, the implementation of which was not a minor part of the current Orel system.

The system whose inferencing is most closely related to Orel is the DB reasoner for $\mathcal{ELH}$ [10]. This system supports only a small fragment of the OWL EL profile, but the rules applied for this part are closely related to those used in Orel for the respective features. The only inference problem that DB supports is classification, but it shows some very good performance characteristics for this task, especially regarding memory usage.

The only other database-driven inference engine for $\mathcal{EL}$ that we are aware of is a prototype system that was presented in [16]. In this case, the focus is on conjunctive query answering, with the main contribution being to show that such queries can be answered rather directly on databases with a certain state of materialisation. Loading performance and memory consumption have not been optimised in this work, and are not as good as for the DB reasoner, but outstanding query performance could be obtained. The existence of prototype systems like the above add to our motivation for developing a stable, free platform that can be used to integrate and refine the underlying approaches and algorithms.

Most other database systems that support OWL reasoning focus on OWL RL or on a subset thereof. The most current such implementation that was reported is the OWL reasoner of Oracle 11g.[3] Many systems focus on DLP [17] or pD* [18], thus providing only incomplete coverage of OWL RL inferencing. Prominent examples include OWLIM [19], DLDB2 [20], and Minerva [21].

Further systems provide yet more restricted amounts of OWL or RDFS inferences mostly for augmenting RDF-based instance data. An interesting example is SAOR for which a non-standard storage implementation has lead to significant performance increases as compared to MySQL [14]. Even though SAOR does not support many OWL features yet, this hints at the potential that non-SQL databases may have for improving the efficiency of systems like Orel.

Finally, rule-based inferencing on top of RDF data has been supported by some tools, the most prominent among which is probably Jena which features a proprietary inference rule implementation.[4] In this case, rules are rather understood in the sense of production rule systems where they form a configurable part of an application that is used to perform relevant computations.

A rather different class of database-driven ontology reasoners are systems that allow for OWL QL querying, such as the QuOnto system.[5] The nature of the problems involved here are somewhat different, and query rewriting often plays a central role. However, recent works in this field have also suggested the use of partial materialisation for improved query performance [22].

7   Conclusion and Future Work

We have presented the new ontology inference and management engine Orel, and its underlying approach based on rule-based bottom-up materialisation of consequences in a database. Similar materialisation approaches have been explored for (sometimes incomplete) OWL Full/RDF(S) inferencing, most notably in SAOR [14] and OWLIM [19]. Conversely, there are also a number of fast in-memory implementations available for handling (parts of) OWL EL. Orel is different from both classes of systems as it provides RDBMS based inferencing for OWL EL, and is using a new algorithmic basis that allows for a unified treatment of OWL EL and RL.

---

[3] `http://www.oracle.com/technology/tech/semantic_technologies/`

[4] `http://jena.sourceforge.net/inference/`

[5] `http://www.dis.uniroma1.it/~quonto/`

The ongoing work on Orel pursues a number of independent goals. Of course, performance is considered as a core challenge, and both the deduction calculus and the storage backend can be improved to address it. For improving the calculus, we develop rule sets that avoid redundant conclusions, and experiment with optimisation methods for efficiently computing closures of datalog programs. Regarding the storage backend, we consider other database paradigms related to recent no-SQL approaches. Another vital feature for a database-driven system are efficient update methods for adding and deleting axioms without recomputing all derivations. Methods for maintenance of materialised views are well known [15], but strongly depend on the details of the implemented calculus.

Besides these obvious goals, there are a number of interesting directions to further develop the core system. Relevant additional features include (conjunctive) query answering, explanation, and extensions with non-standard expressive features such as nonmonotonic inferencing. Other important fields of research concern distribution and parallelisation. At the same time, we seek concrete application scenarios that can be used to explore the practical utility of a robust and scalable OWL inferencing system.

References

1. Hitzler, P., Krötzsch, M., Parsia, B., Patel-Schneider, P.F., Rudolph, S., eds.: OWL 2 Web Ontology Language: Primer. W3C Recommendation (27 October 2009) Available at `http://www.w3.org/TR/owl2-primer/`.

2. Motik, B., Cuenca Grau, B., Horrocks, I., Wu, Z., Fokoue, A., Lutz, C., eds.: OWL 2 Web Ontology Language: Profiles. W3C Recommendation (27 October 2009) Available at `http://www.w3.org/TR/owl2-profiles/`.

3. Abiteboul, S., Hull, R., Vianu, V.: Foundations of Databases. Addison Wesley (1994)

4. Krötzsch, M., Rudolph, S., Hitzler, P.: ELP: Tractable rules for OWL 2. In Sheth, A., Staab, S., Dean, M., Paolucci, M., Maynard, D., Finin, T., Thirunarayan, K., eds.: Proc. 7th Int. Semantic Web Conf. (ISWC'08). Volume 5318 of LNCS., Springer (2008) 649–664

5. Horrocks, I., Patel-Schneider, P.F., Boley, H., Tabet, S., Grosof, B.N., Dean, M.: SWRL: A Semantic Web Rule Language. W3C Member Submission (21 May 2004) Available at `http://www.w3.org/Submission/SWRL/`.

6. Boley, H., Kifer, M., eds.: RIF Basic Logic Dialect. W3C Candidate Recommendation (1 October 2009) Available at `http://www.w3.org/TR/rif-bld/`.

7. Baader, F., Brandt, S., Lutz, C.: Pushing the $\mathcal{EL}$ envelope. In Kaelbling, L., Saffiotti, A., eds.: Proc. 19th Int. Joint Conf. on Artificial Intelligence (IJCAI'05), Professional Book Center (2005) 364–369

8. Krötzsch, M., Rudolph, S., Hitzler, P.: Description logic rules. In Ghallab, M., Spyropoulos, C.D., Fakotakis, N., Avouris, N., eds.: Proceedings of the 18th European Conference on Artificial Intelligence (ECAI'08), IOS Press (2008) 80–84

9. Motik, B., Sattler, U., Studer, R.: Query answering for OWL DL with rules. Journal of Web Semantics **3**(1) (2005) 41–60

10. Delaitre, V., Kazakov, Y.: Classifying $\mathcal{ELH}$ ontologies in SQL databases. In Patel-Schneider, P.F., Hoekstra, R., eds.: Proceedings of the OWLED 2009 Workshop on OWL: Experiences and Directions. Volume 529 of CEUR Workshop Proceedings., CEUR-WS.org (2009)

11. Ioannidis, Y.E., Ramakrishnan, R.: Efficient transitive closure algorithms. In Bancilhon, F., DeWitt, D.J., eds.: Proceedings of the 14th International Conference on Very Large Data Bases (VLDB'88), Morgan Kaufmann (1988) 382–394

12. Horridge, M., Bechhofer, S., Noppens, O.: Igniting the OWL 1.1 touch paper: The OWL API. In Golbreich, C., Kalyanpur, A., Parsia, B., eds.: Proceedings of the OWLED 2007 Workshop on OWL: Experiences and Directions. Volume 258 of CEUR Workshop Proceedings., CEUR-WS.org (2007)

13. Kazakov, Y.: Consequence-driven reasoning for horn $\mathcal{SHIQ}$ ontologies. [23] 2040–2045

14. Hogan, A., Harth, A., Polleres, A.: SAOR: authoritative reasoning for the web. International Journal on Semantic Web and Information Systems (IJSWIS) **2** (2009)

15. Gupta, A., Mumick, I.S., eds.: Materialized Views: Techniques, Implementations, and Applications. MIT Press (1999)

16. Lutz, C., Toman, D., Wolter, F.: Conjunctive query answering in the description logic $\mathcal{EL}$ using a relational database system. [23] 2070–2075

17. Grosof, B.N., Horrocks, I., Volz, R., Decker, S.: Description logic programs: combining logic programs with description logic. In: Proceedings of the 12th International Conference on World Wide Web (WWW'03), ACM (2003) 48–57

18. ter Horst, H.: Completeness, decidability and complexity of entailment for RDF Schema and a semantic extension involving the OWL vocabulary. Journal of Web Semantics **3** (2005)

19. Kiryakov, A., Ognyanov, D., Manov, D.: OWLIM – a pragmatic semantic repository for OWL. In: In Proc. Conf. on Web Information Systems Engineering (WISE) Workshops. (2005) 182–192

20. Pan, Z., Zhang, X., Heflin, J.: Dldb2: A scalable multi-perspective semantic web repository. In: Proc. 2008 IEEE/WIC/ACM Int. Conf. on Web Intelligence (WI'08), IEEE (2008) 489–495

21. Zhou, J., Ma, L., Liu, Q., Zhang, L., Yu, Y., Pan, Y.: Minerva: A scalable OWL ontology storage and inference system. In Mizoguchi, R., Shi, Z., Giunchiglia, F., eds.: Proc. 1st Asian Semantic Web Conf. (ASWC'08). Volume 4185 of LNCS., Springer (2006) 429–443

22. Kontchakov, R., Lutz, C., Toman, D., Wolter, F., Zakharyaschev, M.: Combined fo rewritability for conjunctive query answering in dl-lite. In Cuenca Grau, B., Horrocks, I., Motik, B., Sattler, U., eds.: Proceedings of the 22nd International Workshop on Description Logics (DL'09). Volume 477 of CEUR Workshop Proceedings., CEUR-WS.org (2009)

23. Boutilier, C., ed.: Proceedings of the 21st International Conference on Artificial Intelligence (IJCAI'09), IJCAI (2009)

Optimal Rewritings in Definitorially Complete Description Logics⋆

İnanç Seylan, Enrico Franconi, and Jos de Bruijn

Free University of Bozen-Bolzano, Italy
{seylan,franconi,debruijn}@inf.unibz.it

Abstract. In this paper, we revisit the problem of *definitorial completeness*, i.e., whether a given general TBox $\mathcal{T}$ in a description logic (DL) $\mathscr{L}$ can be rewritten to an acyclic TBox $\mathcal{T}'$ in $\mathscr{L}$. This is an important problem because crucial optimisations in DL reasoners rely on acyclic parts in TBoxes. It is known that such rewritings are possible for *definitorial* TBoxes in $\mathcal{ALC}$ and in logics $\mathcal{ALC}X$ for $X \subseteq \{\mathcal{S}, \mathcal{H}, \mathcal{I}\}$. Here we establish optimal bounds on the sizes of the resulting acyclic TBoxes. In particular, we reduce the known triple exponential upper bound on $\mathcal{ALC}$-TBoxes to single exponential. Additionally, we prove the same upper bound for those extensions with $X \subseteq \{\mathcal{S}, \mathcal{H}, \mathcal{I}\}$ for which there was no established result before. This means, together with the already known exponential lower bound for $\mathcal{ALC}$, that our bounds are tight.

1 Introduction

Description logic (DL) TBoxes enable one to introduce names for complex concepts using *concept definitions*. For example, the definition $Parent \equiv Mother \sqcup Father$ classifies all individuals that are either mothers or fathers as parents. Here, $Parent$ is called a *defined* concept, and $Mother$ and $Father$ are *primitive* concepts. In some sense, instances of primitive concepts come directly from the application domain whereas defined concepts help us to define views or constraints. Baader and Nutt [1] call a finite set of concept definitions a *terminology* if no concept name is defined more than once.

Terminologies can be cyclic, i.e., a defined concept may refer to itself directly in its definition or indirectly through some other defined concept. Cyclicity is a syntactic condition and for certain cyclic terminologies there may be equivalent acyclic ones. For example, the definition

$$Parent \equiv (Parent \sqcup \neg Parent) \sqcap (Mother \sqcup Father)$$

contains the tautological expression $(Parent \sqcup \neg Parent)$. By removing this expression we obtain an equivalent acyclic definition.

Acyclic TBoxes are of particular interest because reasoning with them is "easier" than with general TBoxes. For example, satisfiability of an acyclic $\mathcal{ALC}$-TBox is a PSPACE-complete problem whereas the variant of the problem for general $\mathcal{ALC}$-TBoxes is EXPTIME-complete [2]. On the practical side of things, absorption is an

---

indispensable optimisation technique in DL reasoners which makes use of the acyclic part of a TBox [3]. Therefore a natural question arises: from which cyclic terminologies can we obtain equivalent acyclic ones? Baader and Nutt answer this question by identifying a semantic condition on terminologies called *definitoriality* [1]. Intuitively, if a terminology is definitorial and the instances of primitive concepts are known then the instances of defined concepts are completely determined. In particular, Baader and Nutt show that $\mathcal{ALC}$ is *definitorially complete*, i.e., for every definitorial $\mathcal{ALC}$-terminology there is an equivalent acyclic $\mathcal{ALC}$-terminology. As also noted by the authors, definitorial completeness is a form of Beth Definability [4] – a property of first-order logic – for DLs.

Another relevant question which is of practical interest is how an equivalent acyclic terminology can be obtained from a definitional one. Ten Cate et al. [5] give a constructive method for calculating an acyclic $\mathcal{ALC}$-terminology from a definitorial one and prove definitorial completeness for some extensions of $\mathcal{ALC}$. To be more precise, Ten Cate et al. consider the same problem for general TBoxes instead of terminologies. In general TBoxes, it is not clear from the syntactic shape of the TBox anymore which predicate is primitive and which is defined. In this setting, primitive predicates are assumed as given. Moreover, Ten Cate et al. establish a single exponential lower and a triple exponential upper bound on the size of the generated TBoxes in $\mathcal{ALC}$. However, the exact characterisation of the succinctness of general TBoxes over acyclic ones was left as an open problem.

In this paper, we reduce the upper bound on the size of the equivalent acyclic terminologies obtained from definitorial $\mathcal{ALC}$-TBoxes to single exponential, which is tight. We then extend this result to all logics $\mathcal{ALC}X$ for $X \subseteq \{\mathcal{S}, \mathcal{H}, \mathcal{I}\}$, for which there were no earlier established results. In previous work [6], we used Beth Definability (adapted to DLs) to rewrite a given concept into an equivalent one for efficient instance retrieval using databases. Our results in this paper extend to that scenario as well. More precisely, here we give an optimal version of the algorithm that computes rewritings.

We start by giving a brief introduction to standard notions we will use from DLs in Section 2. In Section 3 we give our main result for $\mathcal{ALC}$ after introducing relevant terminology. These results are based on the algorithm we present in Section 4. Our results for extensions of $\mathcal{ALC}$ are presented in Section 5, after which we conclude.

2   Preliminaries

Let N_C and N_R be countably infinite and disjoint sets of *concept* and *role* names, respectively. With N_P we denote the set of *predicates* $N_C \cup N_R$.

The set of $\mathcal{SHI}$-*roles* is defined as $N_R \cup \{R^- \mid R \in N_R\}$. A *role inclusion axiom* is of the form $R \sqsubseteq S$, with R and S $\mathcal{SHI}$-roles. A *transitivity axiom* is of the form $\mathsf{Trans}(R)$, for R a $\mathcal{SHI}$-role. A *role hierarchy* $\mathcal{H}$ is a finite set of role inclusion and transitivity axioms.

For a role hierarchy $\mathcal{H}$, we define the function Inv over roles as $\mathsf{Inv}(R) := R^-$ if $R \in N_R$ and $\mathsf{Inv}(R) := S$ if $R = S^-$, for some $S \in N_R$. Further we define $\sqsubseteq_{\mathcal{H}}$ as the smallest transitive reflexive relation on $\mathcal{SHI}$-roles in $\mathcal{H}$ such that $R \sqsubseteq S \in \mathcal{H}$ implies $R \sqsubseteq_{\mathcal{H}} S$ and $\mathsf{Inv}(R) \sqsubseteq_{\mathcal{H}} \mathsf{Inv}(S)$.

The set of $\mathcal{SHI}$-*concepts* and their semantics are defined in the standard way [7]. A $\mathcal{SHI}$-*TBox* $\mathcal{T}$ is a finite set of *concept inclusion axioms* $C \sqsubseteq D$ and/or *concept definitions* $A \equiv C$, where A is a concept name, and C and D are $\mathcal{SHI}$-concepts. A $\mathcal{SHI}$ *knowledge base* (KB) $\mathcal{K}$ is a pair $(\mathcal{T}, \mathcal{H})$, where $\mathcal{T}$ is a $\mathcal{SHI}$-TBox and $\mathcal{H}$ is a role hierarchy. For a $\mathcal{SHI}$-concept C and a $\mathcal{SHI}$-KB $\mathcal{K} = (\mathcal{T}, \mathcal{H})$, $\mathsf{rol}(C, \mathcal{K})$ and $\mathsf{sig}(C, \mathcal{K})$ denote, respectively, the sets of role and predicate (i.e., concept or role) names occurring in C or $\mathcal{K}$. We are interested in special acyclic TBoxes.

Definition 1 ([5]). *Let $\mathcal{T}$ be a TBox. A concept name A* directly uses *a concept name B in $\mathcal{T}$ if there is some $A \equiv C \in \mathcal{T}$ and $B \in \mathsf{sig}(C)$;* uses *is the transitive closure of the relation* directly uses.

Let $\Sigma \subseteq \mathsf{sig}(\mathcal{T})$. $\mathcal{T}$ is Σ-acyclic if it satisfies the following two properties:

1. *$\mathcal{T}$ consists of exactly one concept definition $A \equiv C$ for each concept name $A \in (\mathsf{sig}(\mathcal{T}) \setminus \Sigma)$, plus a number of concept inclusion axioms $C \sqsubseteq D$, where $\mathsf{sig}(C) \subseteq \Sigma$ and $\mathsf{sig}(D) \subseteq \Sigma$.*
2. *There is no concept name A that uses itself in $\mathcal{T}$.*

The notion of an interpretation satisfying a role hierarchy or TBox is defined in the usual way (cf. [7]). An interpretation $\mathcal{I}$ satisfies $\mathcal{K} = (\mathcal{T}, \mathcal{H})$ if and only if $\mathcal{I}$ satisfies $\mathcal{T}$ and $\mathcal{H}$. In this case, we say that $\mathcal{I}$ is a model of $\mathcal{K}$. $\mathcal{K}$ is *satisfiable* if $\mathcal{K}$ has a model. Two KBs are *equivalent* if they have the same models. A concept C is *satisfiable w.r.t.* $\mathcal{K}$ if and only if there is some model $\mathcal{I}$ of $\mathcal{K}$ such that $C^{\mathcal{I}} \neq \emptyset$. The concept subsumption and equivalence problems, i.e., checking whether $\mathcal{K} \models C \sqsubseteq D$ (respectively, ($\mathcal{K} \models C \equiv D$), are defined in the usual way.

A concept C is in *negation normal form* (NNF) if and only if the negation sign appears only in front of concept names in C. A concept can be transformed into an equivalent one in NNF in linear time and thus, we assume all concepts to be in NNF. For a concept C, we denote its negation in NNF by $\dot{\neg}C$. Moreover, we will sometimes consider only concept inclusions of the form $\top \sqsubseteq C$ to which every concept inclusion and definition can be rewritten again in linear time.

The *concept closure* $\mathsf{cl}(C_0, \mathcal{K})$ of C_0 and $\mathcal{K}$ is the smallest set of concepts satisfying the following conditions:

- $C_0 \in \mathsf{cl}(C_0, \mathcal{K})$;
- if $\top \sqsubseteq C \in \mathcal{T}$ then $C \in \mathsf{cl}(C_0, \mathcal{K})$;
- if $C \in \mathsf{cl}(C_0, \mathcal{K})$ and D is a subconcept of C then $D \in \mathsf{cl}(C_0, \mathcal{K})$;
- if $\forall R.C \in \mathsf{cl}(C_0, \mathcal{K})$, $S \sqsubseteq_{\mathcal{H}} R$, and $\mathsf{Trans}(S) \in \mathcal{H}$ then $\forall S.C \in \mathsf{cl}(C_0, \mathcal{K})$.

We define the notions of closure $\mathsf{f}(e)$, for $\mathsf{f} \in \{\mathsf{sig}, \mathsf{cl}, \mathsf{rol}\}$ and $e \in \{C, \mathcal{T}, \mathcal{H}, \mathcal{K}\}$, analogously. The *size* of a concept C (written $|C|$) is the number of elements in $\mathsf{cl}(C)$. For a TBox $\mathcal{T}$, $|\mathcal{T}| := \sum_{\top \sqsubseteq C \in \mathcal{T}} |C|$.

3   Beth Definability

We introduce in this Section implicit and explicit *definability* for concepts. We used these notions in [6] to reduce the instance retrieval problem in DLs with DBoxes to

SQL query answering. In this section, we will use them again to rewrite definitorial TBoxes into acyclic ones in a more direct way than Ten Cate et al. did in [5]. We start by giving a semantic characterisation of implicit definability.

Definition 2 (Reduct). *Let $\mathcal{I} = \langle \Delta^{\mathcal{I}}, \cdot^{\mathcal{I}} \rangle$ be an interpretation and let $\Sigma \subseteq N_P$. An interpretation $\mathcal{J} = \langle \Delta^{\mathcal{J}}, \cdot^{\mathcal{J}} \rangle$ is the* reduct *of $\mathcal{I}$ to Σ (denoted by $\mathcal{I}|_\Sigma$) if and only if $\Delta^{\mathcal{I}} = \Delta^{\mathcal{J}}$ and $\cdot^{\mathcal{J}}$ is defined only on the symbols in Σ.*

Definition 3 (Implicit definability). *Let C be a concept, $\mathcal{K}$ a KB, and $\Sigma \subseteq \mathsf{sig}(C, \mathcal{K})$. C is* implicitly definable *from Σ under $\mathcal{K}$ if and only if for any two models $\mathcal{I}$ and $\mathcal{J}$ of $\mathcal{K}$, $\Delta^{\mathcal{I}} = \Delta^{\mathcal{J}}$ and $\mathcal{I}|_\Sigma = \mathcal{J}|_\Sigma$ implies $C^{\mathcal{I}} = C^{\mathcal{J}}$.*

In other words, given a TBox, a concept C is implicitly definable if the set of all its instances depends only on the extension of the predicates in Σ.

Example 1. Consider the KB $\mathcal{K} = (\mathcal{T}, \emptyset)$, where $\mathcal{T}$ is equal to:

$$Project \sqsubseteq Activity$$
$$Meeting \sqsubseteq Activity$$
$$Activity \sqsubseteq Project \sqcup Meeting$$
$$Project \sqsubseteq \neg Meeting$$

and let $\Sigma = \{Meeting, Activity\}$. *Project* is implicitly definable from Σ under $\mathcal{K}$ since its extension depends only on the (fixed) extension of *Meeting* and *Activity*.

The following proposition provides an alternative, syntactic definition of implicit definability. In particular, it reduces checking implicit definability to the entailment problem in the same logic. Let a concept $\widetilde{C}$ (resp., KB $\widetilde{\mathcal{K}}$) be like C (resp., $\mathcal{K}$) except that every occurrence of each predicate $P \in (\Sigma \setminus \mathsf{sig}(C))$ (resp. $P \in (\Sigma \setminus \mathsf{sig}(\mathcal{K}))$) is replaced with a new predicate $\widetilde{P}$.

Proposition 1. *A concept C is implicitly definable from Σ under $\mathcal{K}$ if and only if $\mathcal{K} \cup \widetilde{\mathcal{K}} \models C \equiv \widetilde{C}$.*

If a concept is implicitly definable from Σ, then it may be possible to find an expression using only predicates in Σ whose instances are the same as in the original concept: this would be its explicit definition.

Definition 4 (Explicit definability). *Let C be a concept, $\mathcal{K}$ a KB, and $\Sigma \subseteq \mathsf{sig}(C, \mathcal{K})$. C is* explicitly definable *from Σ under $\mathcal{K}$ if and only if there is some concept D such that $\mathcal{K} \models C \equiv D$ and $\mathsf{sig}(D) \subseteq \Sigma$. Such a D is called an* explicit definition *of C from Σ under $\mathcal{K}$.*

In Example 1, the explicit definition of *Project* is $Activity \sqcap \neg Meeting$. It is not hard to see that explicit definability implies implicit definability. Beth [4] shows that the converse holds for the case of first-order logic: if C is implicitly definable from Σ in $\mathcal{K}$, then it is explicitly definable. This property for $\mathcal{ALC}$ with general TBoxes is proved in [6] by exploiting interpolation. Here we state a stronger version of the theorem in [6] by putting an exponential bound on the size of the explicit definition and give the proof again to show how interpolation is used.

Definition 5. *Let $\mathcal{K} = (\mathcal{T}, \mathcal{H})$ be a KB. A labelling of $\mathcal{K}$ is an ordered pair $\langle \mathcal{K}_\mathbf{l}, \mathcal{K}_\mathbf{r} \rangle$ of KBs where $\mathcal{K}_\mathbf{l} = (\mathcal{T}_\mathbf{l}, \mathcal{H}_\mathbf{l})$, $\mathcal{K}_\mathbf{r} = (\mathcal{T}_\mathbf{r}, \mathcal{H}_\mathbf{r})$, $\mathcal{T} = \mathcal{T}_\mathbf{l} \cup \mathcal{T}_\mathbf{r}$, and $\mathcal{H} = \mathcal{H}_\mathbf{l} \cup \mathcal{H}_\mathbf{r}$; $\langle \mathcal{T}_\mathbf{l}, \mathcal{T}_\mathbf{r} \rangle$ is a labelling of the TBox $\mathcal{T}$.*

Definition 6 (Interpolant). *Let C, D be concepts and let $\mathcal{K}$ be a KB such that $\mathcal{K} \models C \sqsubseteq D$. A concept I is called an interpolant of C and D under a labelling $\langle \mathcal{K}_\mathbf{l}, \mathcal{K}_\mathbf{r} \rangle$ of $\mathcal{K}$ if $\mathsf{sig}(I) \subseteq \mathsf{sig}(C, \mathcal{K}_\mathbf{l}) \cap \mathsf{sig}(D, \mathcal{K}_\mathbf{r})$, $\mathcal{K} \models C \sqsubseteq I$, and $\mathcal{K} \models I \sqsubseteq D$.*

Section 4 is devoted to a constructive proof for the following lemma by using an optimal tableau calculus for $\mathcal{ALC}$.

Lemma 1. *Let C and D be $\mathcal{ALC}$-concepts and let $\mathcal{K} = \mathcal{T}$ be an $\mathcal{ALC}$-KB such that $\mathcal{K} \models C \sqsubseteq D$. If $\langle \mathcal{K}_\mathbf{l}, \mathcal{K}_\mathbf{r} \rangle$ is a labelling of $\mathcal{K}$ then there exists an interpolant of C and D under $\langle \mathcal{K}_\mathbf{l}, \mathcal{K}_\mathbf{r} \rangle$ whose size is at most exponential in $|\mathcal{T}| + |C| + |D|$.*

Theorem 1 (Beth Definability). *Let C be an $\mathcal{ALC}$-concept, let $\mathcal{K} = \mathcal{T}$ be an $\mathcal{ALC}$-KB, and let $\Sigma \subseteq \mathsf{sig}(C, \mathcal{K})$. If C is implicitly definable from Σ under $\mathcal{K}$ then C is explicitly definable from Σ under $\mathcal{K}$, and the size of the explicit definition is at most exponential in $|\mathcal{T}| + |C|$.*

Proof. We have that $\mathcal{K} \cup \widetilde{\mathcal{K}} \models C \equiv \widetilde{C}$ by implicit definability of C. Moreover, $\langle \mathcal{K}, \widetilde{\mathcal{K}} \rangle$ is a labelling of $\mathcal{K} \cup \widetilde{\mathcal{K}}$. Now, by Lemma 1 and $|C| = |\widetilde{C}|$, there is an interpolant I of C and $\widetilde{C}$ under $\langle \mathcal{K}, \widetilde{\mathcal{K}} \rangle$ and the size of I is at most exponential in $|\mathcal{T}| + |C|$. Since it is an interpolant, $\mathsf{sig}(I) \subseteq \mathsf{sig}(C, \mathcal{K}) \cap \mathsf{sig}(\widetilde{C}, \widetilde{\mathcal{K}}) = \Sigma$, and both (a) $\mathcal{K} \cup \widetilde{\mathcal{K}} \models C \sqsubseteq I$ and (b) $\mathcal{K} \cup \widetilde{\mathcal{K}} \models I \sqsubseteq \widetilde{C}$. By (b) and $\mathcal{K} \cup \widetilde{\mathcal{K}} \models \widetilde{C} \sqsubseteq C$, we have $\mathcal{K} \cup \widetilde{\mathcal{K}} \models I \sqsubseteq C$, from which $\mathcal{K} \cup \widetilde{\mathcal{K}} \models C \equiv I$ follows by (a). From the structure of $\widetilde{\mathcal{K}}$ and the fact that $\mathsf{sig}(C), \mathsf{sig}(I) \subseteq \mathsf{sig}(\mathcal{K})$ straightforwardly follows that $\mathcal{K} \models C \equiv I$. □

This proof of Beth definability for $\mathcal{ALC}$ with general TBoxes is constructive, provided we have a constructive method of finding interpolants as defined in Definition 6. As we will see in Section 4, this constructive method is based on tableau. To be more precise, the tableau algorithm will allow us to check whether a concept is implicitly definable and if this is the case, we will use the same tableau proof to construct an explicit definition. Note that Theorem 1 also establishes a single exponential upper bound on the size of explicit definitions we calculate. Together with the following theorem which establishes the lower bound, we can conclude that our procedure for calculating explicit definitions is worst-case optimal.

Theorem 2 ([5]). *There are an $\mathcal{ALC}$-concept C, $\mathcal{ALC}$-KB $\mathcal{K} = \mathcal{T}$, and $\Sigma \subseteq \mathsf{sig}(C, \mathcal{K})$ such that C is implicitly definable from Σ under $\mathcal{K}$ and the smallest explicit definition of C is exponential in $|C| + |\mathcal{K}|$.*

We now formally define the notions we discussed in the introduction. However, unlike Baader and Nutt [1], we consider general TBoxes instead of terminologies. In general TBoxes, it is not clear from the syntactic shape of the TBox which predicate is primitive and which is defined. Therefore, we assume that primitive predicates, i.e., Σ, are specified beforehand. This is similar to the approach by Ten Cate et al. [5].

Definition 7. *Let $\mathcal{T}$ be a TBox and let $\Sigma \subseteq \mathsf{sig}(\mathcal{T})$. $\mathcal{T}$ is Σ-definitorial if and only if for every interpretation $\mathcal{I}$ that interprets only the predicates in Σ there is at most one interpretation $\mathcal{J}$ such that $\Delta^{\mathcal{I}} = \Delta^{\mathcal{J}}$, $P^{\mathcal{I}} = P^{\mathcal{J}}$ for every predicate $P \in \Sigma$, and $\mathcal{J}$ is a model of $\mathcal{T}$.*

It is not hard to see the connection between definitoriality and implicit definability.

Theorem 3. *Let $\mathcal{T}$ be a TBox and let $\Sigma \subseteq \mathsf{sig}(\mathcal{T})$. $\mathcal{T}$ is Σ-definitorial if and only if every concept name $A \in \mathsf{sig}(\mathcal{T}) \setminus \Sigma$ is implicitly definable from Σ under $\mathcal{T}$.*

We are interested in rewriting general TBoxes to Σ-acyclic ones. It is clear from the definition of Σ-acyclic TBoxes that they may contain general concept inclusions involving only predicates from Σ. This restriction is needed because unlike in [1] we may be given a TBox that is not a terminology. A Σ-acyclic TBox is also Σ-definitorial, but the converse may not always be true. DLs possessing this property are called definitorially complete.

Definition 8. *A description logic $\mathcal{L}$ is called definitorially complete if each Σ-definitorial $\mathcal{L}$-TBox $\mathcal{T}$ is equivalent to a Σ-acyclic $\mathcal{L}$-TBox $\mathcal{T}'$.*

Baader and Nutt show that $\mathcal{ALC}$ is definitorially complete [1]. Ten Cate et al. give a concrete algorithm for computing acyclic TBoxes from definitorial ones in $\mathcal{ALC}$ [5]. The algorithm is based on a special normal form for concepts and uniform interpolation. This involves at most a triple exponential blowup. Here we take a more direct approach using interpolation and improve this upper bound to a single exponential one, which is the main result of this section.

Theorem 4. *Let $\mathcal{T}$ be an $\mathcal{ALC}$-TBox and let $\Sigma \subseteq \mathsf{sig}(\mathcal{T})$. If $\mathcal{T}$ is Σ-definitorial, then there exists an equivalent Σ-acyclic $\mathcal{ALC}$-TBox $\mathcal{T}^*$, which is at most exponential in the size of $\mathcal{T}$.*

Proof. Let $\mathcal{T}$ be a Σ-definitorial $\mathcal{ALC}$-TBox. By Theorem 3 and Theorem 1, for every $A \in (\mathsf{sig}(\mathcal{T}) \setminus \Sigma)$, there is some concept C_A such that $\mathcal{T} \models A \equiv C_A$, $\mathsf{sig}(C_A) \subseteq \Sigma$, and $|C_A|$ is at most exponential in $|A| + |\mathcal{T}|$. However, since $A \in \mathsf{sig}(\mathcal{T})$, we can conclude that $|C_A|$ is at most exponential only in $|\mathcal{T}|$.

Let $\mathcal{T}^*$ be the TBox obtained from $\mathcal{T}$ by systematically replacing each occurrence of all A by C_A, and adding the relevant concept definitions $A \equiv C_A$. Then $\mathcal{T}^*$ is Σ-acyclic and equivalent to $\mathcal{T}$. Finally, the length of $\mathcal{T}^*$ is easily seen to be at most exponential in the length of $\mathcal{T}$.   □

4   Optimally Constructing Interpolants

In this section, we give a constructive proof of Lemma 1. In other words, we present a method for constructing an interpolant using a tableau proof. We have presented a constructive method in [6]. However, the algorithm there is based on standard $\mathcal{ALC}$ tableau techniques, which do not guarantee termination in ExpTime, and are not worst-case optimal, since checking satisfiability in $\mathcal{ALC}$ is known to be in ExpTime. Here we aim at obtaining exponential size interpolants by using a worst-case optimal tableau algorithm in the style of Goré and Nyugen [8].

The $R_\perp$ rule

Condition: $\{C^\lambda, (\dot{\neg}C)^\kappa\} \subseteq g.\mathsf{content}.$

Action: $g.\mathsf{status} := \mathtt{unsat}.$

The $R_\sqcap$ rule

Condition: $(C_1 \sqcap C_2)^\lambda \in g.\mathsf{content}, \{C_1^\lambda, C_2^\lambda\} \not\subseteq g.\mathsf{content}.$

Action: $g'.\mathsf{content} := g.\mathsf{content} \cup \{C_1^\lambda, C_2^\lambda\};$

The $R_\sqcup$ rule

Condition: $(C_1 \sqcup C_2)^\lambda \in g.\mathsf{content}, \{C_1^\lambda, C_2^\lambda\} \cap g.\mathsf{content} = \emptyset.$

Action: $g'.\mathsf{content} := g.\mathsf{content} \cup \{C_1^\lambda\};$
$g''.\mathsf{content} := g.\mathsf{content} \cup \{C_2^\lambda\};$

The $R_\exists$ rule

Condition: $\{(\exists R_1.C_1)^{\lambda_1}, \ldots, (\exists R_n.C_n)^{\lambda_n}\} \subseteq g.\mathsf{content};$
$(\exists R.C)^\lambda \in g.\mathsf{content}$ implies $\exists i \in \{1, \ldots, n\}$ s.t. $(\exists R.C)^\lambda = (\exists R_i.C_i)^{\lambda_i}.$

Action: $g_i.\mathsf{content} := \{C_i^{\lambda_i}\} \cup \{D^\lambda \mid (\forall R.D)^\lambda \in g.\mathsf{content}$ and $R_i = R\},$
$g_i.\mathsf{content} := g_i.\mathsf{content} \cup \{E^\mathbf{l} \mid \top \sqsubseteq E \in \mathcal{T}_\mathbf{l}\} \cup \{E^\mathbf{r} \mid \top \sqsubseteq E \in \mathcal{T}_\mathbf{r}\}$
for $1 \leq i \leq n.$

Fig. 1. Tableau expansion rules for $\mathcal{ALC}$.

4.1 An Optimal Tableau Algorithm for Satisfiability

We start by presenting a tableau algorithm for deciding concept subsumption. To this aim, we fix two $\mathcal{ALC}$-concepts C and D, and an $\mathcal{ALC}$-TBox $\mathcal{T}$ with the labelling $\langle \mathcal{T}_\mathbf{l}, \mathcal{T}_\mathbf{r} \rangle$. A *biased tableau* (tableau for short) for $\langle C, D, \mathcal{T}_\mathbf{l}, \mathcal{T}_\mathbf{r} \rangle$ is a directed graph $\langle \mathcal{V}, \mathcal{E} \rangle$, where $\mathcal{V}$ is the set of nodes and $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ is the set of edges.

In the following, we will be using *biased concepts* which are expressions of the form C^λ, where C is an $\mathcal{ALC}$-concept and $\lambda \in \{\mathbf{l}, \mathbf{r}\}$ is a bias. Let $\mathsf{cl_l} := \{E^\mathbf{l} \mid E \in \mathsf{cl}(C, \mathcal{T}_\mathbf{l})\}$ and $\mathsf{cl_r} := \{E^\mathbf{r} \mid E \in \mathsf{cl}(\dot{\neg}D, \mathcal{T}_\mathbf{r})\}$. We associate four different labels to nodes in $\mathcal{V}$: $\mathsf{content} : V \to 2^{\mathsf{cl_l} \cup \mathsf{cl_r}}$, $\mathsf{type} : V \to \{\mathtt{and-node}, \mathtt{or-node}\}$, $\mathsf{status} : V \to \{\mathtt{sat}, \mathtt{unsat}\}$, and $\mathsf{availability} : V \to \{\mathtt{expanded}, \mathtt{unexpanded}\}$. The function of these labels are explained when they are used.

The *tableau expansion rules* given in Figure 1 expand a tableau by making use of the semantics of concepts, and thus make implicit information explicit. We assume that a rule can be applied to a node g if $g.\mathsf{availability} = \mathtt{unexpanded}$ and if a rule is applied to g then $g.\mathsf{availability} := \mathtt{expanded}$ without writing it explicitly in rule definitions. In order to guarantee a finite expansion, we use *proxies* in the following way. Whenever a rule creates a new node g' from g, before attaching the edge $\langle g, g' \rangle$ to $\mathcal{E}$, the tableau is searched for a node $g'' \in \mathcal{V}$ such that $g'.\mathsf{content} = g''.\mathsf{content}$. If such a g'' is found then the edge $\langle g, g'' \rangle$ is added to $\mathcal{E}$ and g' is discarded.

We are interested in deciding $\mathcal{T} \models C \sqsubseteq D$. The tableau algorithm starts with the *initial tableau* $\mathbf{T} = \langle \{g_0\}, \emptyset \rangle$ for $\langle C, D, \mathcal{T}_\mathbf{l}, \mathcal{T}_\mathbf{r} \rangle$, where $g_0.\mathsf{content} = \{C^\mathbf{l}, (\dot{\neg}D)^\mathbf{r}\} \cup \{E^\mathbf{l} \mid \top \sqsubseteq E \in \mathcal{T}_\mathbf{l}\} \cup \{E^\mathbf{r} \mid \top \sqsubseteq E \in \mathcal{T}_\mathbf{r}\}$ and $g_0.\mathsf{availability} = \mathtt{unexpanded}$. $\mathbf{T}$ is then expanded by repeatedly applying the tableau expansion rules in such a way that if more than one rule is applicable at the same time then the first applicable rule in the list $[R_\perp, R_\sqcap, R_\sqcup, R_\exists]$ is chosen. The expansion continues until none of the rules is applicable to $\mathbf{T}$. Such a tableau is called *complete*.

Let **T** be a complete tableau for $\langle C, D, \mathcal{T}_\mathbf{l}, \mathcal{T}_\mathbf{r} \rangle$. The type of a node g is determined as follows: g.type $=$ or-node if $\mathsf{R}_\sqcup$ has been applied to g, and g.type $=$ and-node otherwise. Until it is no more possible to assign a *status* to a node in $\mathcal{V}$, we run the following algorithm.

- Pick a node $g \in V$.
- If g is a sink node[1] with g.status $\neq$ unsat then g.status $:=$ sat.
- If g.type $=$ and-node and
  - all g's direct successors have status sat then g.status $:=$ sat;
  - one of g's direct successors has status unsat then g.status $:=$ unsat.
- If g.type $=$ or-node and
  - all g's direct successors have status unsat then g.status $:=$ unsat;
  - one of g's direct successors has status sat then g.status $:=$ sat.

If g_0.status is still undefined then for every $g \in \mathcal{V}$ with g.status $\neq$ unsat, set g.status $:=$ sat.

A complete tableau for $\langle C, D, \mathcal{T}_\mathbf{l}, \mathcal{T}_\mathbf{r} \rangle$ is *closed* if g_0 has status unsat and it is *open*, otherwise. If the tableau algorithm constructs an open tableau for $\langle C, D, \mathcal{T}_\mathbf{l}, \mathcal{T}_\mathbf{r} \rangle$ then it returns "$\mathcal{T} \not\models C \sqsubseteq D$", and "$\mathcal{T} \models C \sqsubseteq D$" otherwise.

Termination is a consequence of using proxies and $\mathsf{cll} \cup \mathsf{clr}$ being finite. In the worst case, there are $2^{O(\sharp(\mathsf{cll} \cup \mathsf{clr}))}$ nodes in a complete tableau **T**. Checking for proxies and determining the status of g_0 both take polynomial number of steps in the size of **T**. As it is apparent, we use a refutation proof for $\mathcal{T} \models C \sqsubseteq D$, i.e., we check the unsatisfiability of $C \sqcap \dot{\neg} D$ w.r.t. $\mathcal{T}$. For soundness, given a model $\mathcal{I}$ of $\mathcal{T}$ such that $(C \sqcap \dot{\neg} D)^\mathcal{I} \neq \emptyset$, we can guide the tableau algorithm to construct an open tableau for $\langle C, D, \mathcal{T}_\mathbf{l}, \mathcal{T}_\mathbf{r} \rangle$ by making use of the information in $\mathcal{I}$. As for completeness, we can construct a model $\mathcal{I}$ of $\mathcal{T}$ such that $(C \sqcap \dot{\neg} D)^\mathcal{I} \neq \emptyset$ from an open tableau for $\langle C, D, \mathcal{T}_\mathbf{l}, \mathcal{T}_\mathbf{r} \rangle$. Combining all these, we get the following theorem.

Theorem 5 ([8]). *Let C, D be $\mathcal{ALC}$-concepts and $\mathcal{T}$ be an $\mathcal{ALC}$-TBox. The tableau algorithm decides $\mathcal{T} \models C \sqsubseteq D$ in time exponential in $|C| + |D| + |\mathcal{T}|$.*

4.2 An Algorithm for Calculating Interpolants

As is clear from the definition of the tableau expansion rules, we use some additional bookkeeping (compared with the algorithm of [8]) for calculating interpolants. In particular, it is necessary to identify from which TBox ($\mathcal{T}_\mathbf{l}$ or $\mathcal{T}_\mathbf{r}$) or concept (C or D) a concept in the content of a node is derived. After we have that information, we can extract an interpolant from a closed tableau.

The interpolant calculation rules are presented in Figure 2. Given a closed tableau **T** for $\langle C, D, \mathcal{T}_\mathbf{l}, \mathcal{T}_\mathbf{r} \rangle$, the interpolant calculation algorithm starts by calculating a concept $\mathsf{int}(g)$ for every sink node g in **T** with g.status $=$ unsat[2] using $\mathsf{C}_\bot$. While g_0 is not assigned a concept $\mathsf{int}(g_0)$, it repeatedly applies the following steps.

1. Pick a node g such that $\mathsf{int}(g)$ is undefined and g.status $=$ unsat.

---

[1] a node with no outgoing edges

[2] Note that a node g in **T** with g.status $=$ unsat is a sink if and only if $\mathsf{R}_\bot$ is applied to g.

The $\mathsf{C}_\perp$ **rule**	The $\mathsf{C}_\sqcup$ **rule**
$\mathsf{int}(g) := \perp$, if $\lambda = \kappa = \mathbf{l}$;	$\mathsf{int}(g) := \mathsf{int}(g') \sqcup \mathsf{int}(g'')$, if $\lambda = \mathbf{l}$;
$\mathsf{int}(g) := C$, if $\lambda = \mathbf{l}$ and $\kappa = \mathbf{r}$;	$\mathsf{int}(g) := \mathsf{int}(g') \sqcap \mathsf{int}(g'')$, if $\lambda = \mathbf{r}$.
$\mathsf{int}(g) := \top$, if $\lambda = \kappa = \mathbf{r}$;	
$\mathsf{int}(g) := \dot{\neg} C$, if $\lambda = \mathbf{r}$ and $\kappa = \mathbf{l}$.	
The $\mathsf{C}_\sqcap$ rule	**The $\mathsf{C}_\exists$ rule**
$\mathsf{int}(g) := \mathsf{int}(g')$.	$\mathsf{int}(g) := \exists R_i.\mathsf{int}(g_i)$, if $\lambda_i = \mathbf{l}$;
	$\mathsf{int}(g) := \forall R_i.\mathsf{int}(g_i)$, if $\lambda_i = \mathbf{r}$ for some $i \in \{1, \dots, n\}$.

Fig. 2. Interpolant calculation rules for $\mathcal{ALC}$.

2. If $g.\mathsf{type} = \mathtt{and\text{-}node}$, and g has a direct successor g' (g_i for some $i \in \{1, \dots, n\}$) with $\mathsf{int}(g')$ (resp. $\mathsf{int}(g_i)$) defined then apply $\mathsf{C}_\sqcap$ (resp. $\mathsf{C}_\exists$) .
3. If $g.\mathsf{type} = \mathtt{or\text{-}node}$, and for all direct successors g', g'' of g we have that $\mathsf{int}(g')$ and $\mathsf{int}(g'')$ defined then apply $\mathsf{C}_\sqcup$.

This algorithm terminates in time polynomial in the size of $\mathbf{T}$ because all sink nodes with status $\mathtt{unsat}$ are reachable from g_0, and it is guaranteed to have nodes satisfying conditions 2 and 3 by the virtue of $g_0.\mathsf{status} = \mathtt{unsat}$. The correctness of the algorithm is shown in two steps: first we show that our interpolant calculation rules are sound, i.e., they compute interpolants; second we show completeness, i.e., that we always find an interpolant. The proofs of these theorems are very similar to the corresponding ones we presented in [6]. Lemma 1 now follows straightforwardly from the termination and correctness of the interpolant calculation algorithm, and Theorem 5.

5   Beth Definability in Extensions of $\mathcal{ALC}$

In this section, we present a polynomial reduction from $\mathcal{SHI}$ KB satisfiability to $\mathcal{ALC}$ KB satisfiability. This reduction allows us to prove Beth definability and definitorial completeness properties for any extension of $\mathcal{ALC}$ with constructors from $\{\mathcal{S}, \mathcal{H}, \mathcal{I}\}$. Ten Cate et al. [5] showed that these logics are definitorially complete but because of their model theoretic argument, they provide no information on the size of the resulting TBoxes. In this section, we establish a tight upper bound for explicit definitions in these logics.

We will proceed in three steps: first reduce $\mathcal{SHI}$-concept satisfiability w.r.t. a KB to the same problem in $\mathcal{ALCHI}$, then $\mathcal{ALCHI}$ to $\mathcal{ALCI}$, and finally $\mathcal{ALCI}$ to $\mathcal{ALC}$. All these reductions use the axiom schema instantiation technique [9] which is based on the idea of removing the constructor at hand by instantiating its corresponding (modal) axiom schema [10] for each concept in cl or a relevant concept closure, and adding these instances to the TBox to obtain an equi-satisfiable KB. The first and third of these reductions are given in [11] and [12], respectively. To the best of our knowledge, the second one has not beed used before.

Definition 9. *Let C_0 be a $\mathcal{SHI}$-concept and $\mathcal{K} = (\mathcal{T}, \mathcal{H})$ be an $\mathcal{SHI}$-KB. Then $\tau_\mathcal{S}(C_0, \mathcal{K})$ is defined as the $\mathcal{ALCHI}$ KB $(\mathcal{T} \cup \mathcal{T}', \mathcal{H}')$, where*

- $\mathcal{T}' = \{\forall R.C \sqsubseteq \forall S.\forall S.C \mid \forall R.C \in \mathsf{cl}(C_0, \mathcal{K}),\ S \sqsubseteq_{\mathcal{H}} R \text{ and } \mathsf{Trans}(R) \in \mathcal{H}\}$.
- $\mathcal{H}'$ *is obtained from* $\mathcal{H}$ *by removing all transitivity axioms.*

Theorem 6 ([11]). *A* $\mathcal{SHI}$-*concept* C_0 *is satisfiable w.r.t. a* $\mathcal{SHI}$-*KB* $\mathcal{K} = (\mathcal{T}, \mathcal{H})$ *if and only if* C_0 *is satisfiable w.r.t. the* $\mathcal{ALCHI}$-*KB* $\tau_{\mathcal{S}}(C_0, \mathcal{K})$.

Definition 10. *Let* C_0 *be an* $\mathcal{ALCHI}$-*concept and let* $\mathcal{K} = (\mathcal{T}, \mathcal{H})$ *be an* $\mathcal{ALCHI}$-*KB. Then* $\tau_{\mathcal{H}}(C_0, \mathcal{K})$ *is defined as the* $\mathcal{ALCI}$-*KB* $(\mathcal{T} \cup \mathcal{T}', \emptyset)$, *where* $\mathcal{T}' = \{\forall S.C \sqsubseteq \forall R.C \mid \forall S.C \in \mathsf{cl}(C_0, \mathcal{K}),\ R \sqsubseteq_{\mathcal{H}} S,\ \text{and}\ R \neq S\}$.

Theorem 7. *An* $\mathcal{ALCHI}$-*concept* C_0 *is satisfiable w.r.t. an* $\mathcal{ALCHI}$-*KB* $\mathcal{K} = (\mathcal{T}, \mathcal{H})$ *if and only if* C_0 *is satisfiable w.r.t. the* $\mathcal{ALCI}$ *KB* $\tau_{\mathcal{H}}(C_0, \mathcal{K})$.

Dealing with inverse roles is a bit more intricate because the signature of the original KB needs to be changed. Let the $\mathcal{ALC}$-concept $\zeta(C)$ be like the $\mathcal{ALCI}$-concept C except that every occurrence of each inverse role R^- in C is replaced with a new role name R^c; the renaming transformation $\zeta(\cdot)$ extends to TBoxes in the natural way. Moreover, let $\iota(\cdot)$ be the inverse of $\zeta(\cdot)$, i.e., $\iota(\zeta(C)) = C$.

Definition 11. *Let* C_0 *be an* $\mathcal{ALCI}$-*concept and let* $\mathcal{T}$ *be an* $\mathcal{ALCI}$-*TBox. Then* $\tau_{\mathcal{I}}(C_0, \mathcal{T})$ *is defined as the* $\mathcal{ALC}$-*TBox* $\mathcal{T}_1 \cup \mathcal{T}_2$, *where:*

1. $\mathcal{T}_1 = \zeta(\mathcal{T})$.
2. $\mathcal{T}_2$ *is the set of all concept inclusion axioms of the forms* $C \sqsubseteq (\forall R.\exists R^c.C)$ *and* $C \sqsubseteq (\forall R^c.\exists R.C)$ *such that* C *is in* $\mathsf{cl}(\zeta(C_0), \mathcal{T}_1)$ *and* R *is a role name in* $\mathsf{rol}(C_0, \mathcal{T})$.

Theorem 8 ([12]). *An* $\mathcal{ALCI}$-*concept* C_0 *is satisfiable w.r.t. an* $\mathcal{ALCI}$-*TBox* $\mathcal{T}$ *if and only if the* $\mathcal{ALC}$-*concept* $\zeta(C_0)$ *is satisfiable w.r.t. the* $\mathcal{ALC}$-*TBox* $\tau_{\mathcal{I}}(C_0, \mathcal{T})$.

The following theorem follows directly from Theorems 6, 7, and 8.

Theorem 9. *A* $\mathcal{SHI}$-*concept* C_0 *is satisfiable w.r.t. a* $\mathcal{SHI}$-*KB* $\mathcal{K} = (\mathcal{T}, \mathcal{H})$ *if and only if the* $\mathcal{ALC}$-*concept* $\zeta(C_0)$ *is satisfiable w.r.t. the* $\mathcal{ALC}$-*KB* $\tau_{\mathcal{I}}(C_0, \tau_{\mathcal{H}}(C_0, \tau_{\mathcal{S}}(C_0, (\mathcal{T}, \mathcal{H}))))$.

Except for the last one, all the reductions presented in this section preserve the signature of the given KB. Therefore, an explicit definition in the less expressive logic is also an explicit definition in the more expressive one. As for the last reduction, it is possible to reconstruct an explicit definition in $\mathcal{ALCI}$ from the one in $\mathcal{ALC}$ by replacing role names corresponding to inverse roles, as demonstrated by the following theorem.

Theorem 10. *Let* $X \subseteq \{\mathcal{S}, \mathcal{H}, \mathcal{I}\}$ *and let* $\mathcal{K} = (\mathcal{T}, \mathcal{H})$ *be an* $\mathcal{ALCX}$-*KB. If an* $\mathcal{ALCX}$-*concept* C *is implicitly definable from* $\Sigma \subseteq \mathsf{sig}(C, \mathcal{K})$ *under* $\mathcal{K}$ *then* C *is explicitly definable from* Σ *under* $\mathcal{K}$, *and the size of the explicit definition of* C *from* Σ *under* $\mathcal{K}$ *is at most exponential in* $|\mathcal{T}| + |\mathcal{H}| + |C|$.

Proof. Let $\mathcal{K}$ be a $\mathcal{SHI}$-KB and let C and D be $\mathcal{SHI}$-concepts. Furthermore, let $\mathcal{K}_\star = (\tau_{\mathcal{I}}(E, \tau_{\mathcal{H}}(E, \tau_{\mathcal{S}}(E, (\mathcal{T}, \mathcal{H})))))$, where $E = (C \sqcap \neg D) \sqcup (D \sqcap \neg C)$. We have the following chain of equivalences: (†) $\mathcal{K} \models C \equiv D \Leftrightarrow E$ is unsatisfiable w.r.t. $\mathcal{K}$ $\Leftrightarrow^{[\text{Theorem 9}]} \zeta(E)$ is unsatisfiable w.r.t. $\mathcal{K}_\star \Leftrightarrow \mathcal{K}_\star \models \zeta(C) \equiv \zeta(D)$.

Let $\mathcal{K} = (\mathcal{T}, \mathcal{H})$ be an $\mathcal{ALCX}$-KB and let C be an $\mathcal{ALCX}$-concept such that C is implicitly definable from Σ under $\mathcal{K}$. Then $\mathcal{K} \cup \widetilde{\mathcal{K}} \models C \equiv \widetilde{C}$. Every $\mathcal{ALCX}$-concept and every $\mathcal{ALCX}$-KB is trivially a $\mathcal{SHI}$-concept and a $\mathcal{SHI}$-KB, respectively, and therefore, by (†), we have that $\mathcal{K}_\star \cup \widehat{\mathcal{K}_\star} \models \zeta(C) \equiv \widehat{\zeta(C)}$, where $\widehat{\cdot}$ is exactly like $\widetilde{\cdot}$ except that Σ is substituted by $\Sigma_\star = \Sigma \cup \{R^c \mid R^c \in \mathsf{rol}(\zeta(C), \mathcal{K}_\star) \text{ and } R \in \Sigma\}$. $\zeta(C)$ and $\mathcal{K}_\star$ are an $\mathcal{ALC}$-concept and $\mathcal{ALC}$-KB, respectively. By Theorem 1 we have that there is an explicit definition D of $\zeta(C)$ from $\Sigma_\star$ under $\mathcal{K}_\star$ the size of which is at most exponential in $|\tau_{\mathcal{I}}(C, \tau_{\mathcal{H}}(C, \tau_{\mathcal{S}}(C, (\mathcal{T}, \mathcal{H}))))| + |\zeta(C)|$. In other words, $\mathcal{K}_\star \models \zeta(C) \equiv D$ and the size of D is at most exponential in $|\mathcal{T}| + |\mathcal{H}| + |C|$. By (†), $\mathcal{K} \models C \equiv \iota(D)$, where $\iota(D)$ is an $\mathcal{ALCX}$-concept. But by Definition 4, $\iota(D)$ is an explicit definition of C from Σ under $\mathcal{K}$. $\qquad\square$

6 Conclusion

In this paper, we revisited the problem of definitorial completeness, i.e., whether a given general TBox $\mathcal{T}$ in a DL $\mathscr{L}$ can be rewritten to an acyclic TBox $\mathcal{T}'$ in $\mathscr{L}$. $\mathcal{ALC}$ and every $\mathcal{ALCX}$ for $X \subseteq \{\mathcal{S}, \mathcal{H}, \mathcal{I}\}$ were already known to be definitorially complete [5]. Our main contribution in this paper was to establish a tight exponential bound on the size of the resulting acyclic TBoxes.

Our results show that concept definitions can be written exponentially more succinctly in general TBoxes than in acyclic TBoxes for the logics we considered. Therefore, general concept inclusions may help increase the readability of a TBox/ontology. However, general concept inclusions introduce a lot of non-determinism to reasoning algorithms. If a general TBox is used for concept definitions and our DL is definitorially complete then this is not much of a restriction since an equivalent acyclic TBox can be obtained. This suggests the use of the general TBox for user friendliness and a precomputed acyclic counterpart for reasoning.

Definitorial completeness is a fragile property and not every DL enjoys it. For example, $\mathcal{ALCO}$ does not have this property and requires the @-operator from hybrid logics to become definitorially complete [5]. It is worthwhile to note that the algorithm for computing acyclic TBoxes is based on tableau, and thus it is suitable for optimised implementations. We leave as an open problem whether a tight upper bound for $\mathcal{SHIQ}$ can be obtained in a similar way.

References

1. Baader, F., Nutt, W.: Basic description logics. [13] 43–95
2. Donini, F.M.: Complexity of reasoning. [13] 96–136
3. Horrocks, I.: Implementation and optimization techniques. [13] 306–346
4. Beth, E.W.: On Padoa's methods in the theory of definitions. Koninklijke Nederlandse Akademie van Wetenschappen, Proceedings **56** (1953) 330–339
5. Ten Cate, B., Conradie, W., Marx, M., Venema, Y.: Definitorially complete description logics. In Doherty, P., Mylopoulos, J., Welty, C.A., eds.: KR, AAAI Press (2006) 79–89
6. Seylan, I., Franconi, E., de Bruijn, J.: Effective query rewriting with ontologies over dboxes. In Boutilier, C., ed.: IJCAI. (2009) 923–925

7. Baader, F.: Description logic terminology. [13] 485–495
8. Goré, R., Nguyen, L.A.: Exptime tableaux for alc using sound global caching. In Calvanese, D. et al., eds.: Description Logics(2007)
9. Calvanese, D., Giacomo, G.D., Lenzerini, M., Nardi, D.: Reasoning in expressive description logics. In Handbook of Automated Reasoning, Elsevier and MIT Press (2001) 1581–1634
10. Blackburn, P., de Rijke, M., Venema, Y.: Modal logic. Cambridge University Press, New York, NY, USA (2001)
11. Motik, B.: Reasoning in Description Logics using Resolution and Deductive Databases. PhD thesis, Univesität Karlsruhe (TH), Karlsruhe, Germany (January 2006)
12. Calvanese, D., Giacomo, G.D., Rosati, R.: A note on encoding inverse roles and functional restrictions in alc knowledge bases. In Description Logics(1998)
13. Baader, F. et al., eds.: The Description Logic Handbook: Theory, Implementation, and Applications. Cambridge University Press (2003)

Extending OWL with Integrity Constraints

Jiao Tao[1], Evren Sirin[2], Jie Bao[1], and Deborah L. McGuinness[1]

[1] Department of Computer Science, Rensselaer Polytechnic Institute, Troy, NY, USA
[2] Clark&Parsia, LLC, Washington, DC, USA

1 Introduction

The Web Ontology Language (OWL) [1] is an expressive ontology language based on Description Logics (DL)[1]. The semantics of OWL addresses distributed knowledge representation scenarios where complete knowledge about the domain cannot be assumed. Further, the semantics has the following characteristics:

- Open World Assumption (OWA): i.e., a statement cannot be inferred to be false on the basis of failures to prove it.
- Absence of the Unique Name Assumption (UNA): i.e., two different names may refer to the same object.

However, these characteristics can make it difficult to use OWL for data validation purposes in real-world applications where complete knowledge can be assumed for some or all parts of the domain.

Example 1 *Suppose we have the following inventory KB $\mathcal{K}$. One might add the following axiom α to express the constraint "a product is produced by a producer".*
$$\mathcal{K} = \{\texttt{Product}(p)\}, \quad \alpha : \texttt{Product} \sqsubseteq \exists\texttt{hasProducer.Producer}$$

In this example, due to the OWA, *not* having a known producer for p does not cause a logical inconsistency. Therefore, we cannot use α to detect (or prevent) that a product is added to the KB without the producer information.

Example 2 *Suppose we have the following inventory KB $\mathcal{K}$. One might add the following axiom α to express the constraint "a product has at most one producer".*
$$\mathcal{K} = \{\texttt{Product}(p), \texttt{hasProducer}(p, m_1), \texttt{hasProducer}(p, m_2)\},$$
$$\alpha : \texttt{Product} \sqsubseteq \ \leq 1\texttt{hasProducer.}\top$$

Since m_1 and m_2 are not explicitly defined to be different from each other, they will be inferred to be same due to the cardinality restriction. However, in many cases, the reason to use functional properties is not to draw this inference, but to detect an inconsistency. When the information about instances are coming from multiple sources we cannot always assume explicit inequalities will be present.

In these scenarios, there is a strong need to use OWL as an Integrity Constraint (IC) language with closed world semantics. That is, we would like to adopt the OWA without the UNA for parts of the domain where we have incomplete

---

[1] Throughout the paper we use the terms OWL and DL interchangeably.

knowledge, and the Closed World Assumption (CWA)[2] with UNA otherwise. This calls for the ability to combine the open world reasoning of OWL with closed world constraint validation.

In this paper, we describe an alternative IC semantics for OWL, which enables developers to augment OWL ontologies with IC axioms. Standard OWL axioms in the ontologies are used to compute inferences with open world semantics and ICs are used to validate instance data using closed world semantics. Our goal is to enable OWL as an IC language, especially in settings where OWL KBs are integrated with relational databases and ICs are needed to enforce the *named* individuals to have some *known* values. We show that IC validation can be reduced to query answering when the KB expressivity is $\mathcal{SRI}$ or the constraint expressivity is $\mathcal{SROI}$. The queries generated from ICs can be expressed in the SPARQL query language allowing existing OWL reasoners to be used for IC validation easily.

2 IC Use Cases

There are several common use cases for closed world constraint checking that have been identified in the relational and deductive databases literature [2, Chap. 11]. We prepared a user survey to gather use cases and requirements for ICs from the OWL community. These use cases are similar to what we consider to be the canonical IC use cases and can be summarized as follows:

Typing constraints Typing constraints require that individuals that participate in a relation should be instances of certain types. For example, closed world interpretation of domain and range axioms in OWL would fit into this category. Given the following ICs

$\exists$hasProducer.$\top \sqsubseteq$ Product, $\top \sqsubseteq \forall$hasProducer.Producer

The following role assertion

hasProducer(product1, producer1)

would violate these ICs since product1 and producer1 are not explicitly known to be instances of Product and Producer respectively. The data would be valid with the addition of the following assertions:

Product(product1), Producer(producer1).

Domain and range axioms can be seen as global typing constraints; that is they affect instances of every class that participates in a property assertion. OWL also allows finer-grained typing constraints using universal restrictions.

Participation constraints Participation constraints require that instances of the constrained class should have a role assertion. Given an IC semantics, the existential restrictions in OWL can be used for this purpose. For instance, in Example 1, α is a participation constraint. With IC semantics, we expect $\mathcal{K}$ to be invalid w.r.t. this constraint since the producer of p is not known. $\mathcal{K}$ would be valid only when additional axioms in the following form are added:

hasProducer(p, producer), Producer(producer).

---

[2] With CWA, a statement is inferred to be false if it is not known to be true, which is the opposite of OWA.

Uniqueness constraints Uniqueness constraints require that an individual cannot participate in multiple role assertions with the same role. The keys in relational databases enforce such constraints. A similar restriction can be expressed in OWL with a `FunctionalProperty` declaration. For instance, in Example 2, α is an uniqueness constraint. With IC semantics, $\mathcal{K}$ is invalid w.r.t. this constraint since p has two producers $m1$ and $m2$ which are not known to be same. $\mathcal{K}$ would be valid after adding the assertion $m1 = m2$.

3 Related Work

The research on integrating ICs with OWL has been conducted in multiple directions. One approach to achieve this combination is to couple OWL with rule-based formalisms and express ICs as rules without heads as in [3, 4]. For example, according to the proposal in [3], the constraint axiom α in Example 1 is expressed with rules as follows:

$\perp \leftarrow DL[\texttt{Product}](x), \textbf{not}\, P(x, y)$

$P(x, y) \leftarrow DL[\texttt{hasProducer}](x, y), DL[\texttt{Producer}](y)$

where atoms with prefix DL are *DL atoms* which are evaluated as queries to the OWL KB, **not** is the *Negation As Failure (NAF)* operator [3], and $\perp$ is a special predicate representing the empty rule head. The addition of constraints (rules) to a DL KB constitutes a hybird KB, and the detection of a constraint violation is reduced to checking if the special predicate $\perp$ is entailed by the hybrid KB. With this approach, ontology developers have to deal with one more additional formalism, i.e., rules, besides the ontology language OWL to model the domain.

ICs can also be expressed with the epistemic query language EQL-Lite [5] where EQL-Lite allows one to pose epistemic FOL queries that contain the $\mathcal{K}$ operator used against standard FOL KBs. Since every OWL axiom can be represented as an FOL formula we can translate the constraint axiom in Example 1 to the following EQL-Lite query:

$\mathcal{K}\texttt{Product}(x) \rightarrow \exists y.(\mathcal{K}\texttt{hasProducer}(x, y) \wedge \mathcal{K}\texttt{Producer}(y))$

where the answers of this query return the individuals in the KB that satisfy the constraint, and the answers of the negated query will return the individuals that violate the IC. Although the data complexity of answering domain independent EQL-Lite queries in DL-Lite is LOGSPACE, it would require substantially more effort to support EQL-Lite in DL KBs with full expressivity and the complexity results are still unknown.

Another line of approach is based on the epistemic extension of DLs [6, 7] where modal operators $\mathcal{K}$ and $\mathcal{A}$ can be used in concept and role expressions of the given DL KB. Intuitively, $\mathcal{K}C$ represents the set of individuals that are known to be instances of C and $\mathcal{K}R$ represents the pair of individuals that are known to be related with the role R. Operator $\mathcal{A}$ is interpreted in terms of autoepistemic assumptions. Then the ICs are represented as epistemic DL axioms, and the satisfaction of ICs is defined as the entailment of the epistemic IC axioms by the

---

[3] NAF is widely used in logic programming systems. With NAF, axioms that cannot be proven to be true are assumed to be false

standard DL KB. For example, the constraint α in Example 1 can be translated into the following epistemic DL axiom:

$\mathcal{K}\texttt{Product} \sqsubseteq \exists\mathcal{K}\texttt{hasProducer}.\mathcal{K}\texttt{Producer}$.

One important feature of [6, 7] is that all interpretation domains are same, and an individual name always refers to the same object in every interpretation. Due to this feature, strict UNA is enforced. That is, two different names always denote different resources. However, this is not compatible with OWL since it is possible that standard OWL axioms infer that two different names identify the same individual. While existing research has focused on epistemic extensions for relatively inexpressive $\mathcal{ALC}$ there has not been much research for combining epistemic logics with more expressive DLs.

Besides the above work, there are some other proposals concerning on integration of ICs with OWL. In this paper, we focus on approaches that reuse OWL as an IC language. Our closest related work is a proposal by Motik et al. [8] based on a minimal Herbrand model semantics of OWL: here, a constraint axiom is satisfied if all minimal Herbrand models satisfy it. This approach may result in counterintuitive results or a significant modeling burden in the following cases.

First, unnamed individuals can satisfy constraints, which is not desirable for closed world data validation.

Example 3 *Consider the KB $\mathcal{K}$ that contains a product instance and its unknown producer, and the constraint α that every product has a known producer:*

$\mathcal{K} = \{\texttt{Product}(p), \exists\texttt{hasProducer}.\texttt{Producer}(p)\}$

$\alpha : \texttt{Product} \sqsubseteq \exists\texttt{hasProducer}.\texttt{Producer}$

Since p has a producer in every minimal Herbrand model of $\mathcal{K}$, α is satisfied, even though the producer is unknown.

Second, if a constraint needs to be satisfied only by named individuals, then a special concept O has to be added into the original IC axiom, and every named individual should be asserted as an instance of O. This adds a significant maintenance burden on ontology developers, but still doesn't capture the intuition behind the constraint;

Example 4 *Suppose we have a KB $\mathcal{K}$ where there are two possible producers for a product and a constraint α:*

$K = \{\texttt{Product}(p), (\exists\texttt{hasProducer}.\{m_1, m_2\})(p), O(p),$
$\quad\quad \texttt{Producer}(m_1), \texttt{Producer}(m_2), O(m_1), O(m_2)\}$

$\alpha : \texttt{Product} \sqsubseteq \exists\texttt{hasProducer}.(\texttt{Producer} \sqcap O)$

The intuition behind constraint α is that the producer of every product should be known. Even though we do not know the producer of p is m_1 or m_2 for sure, α is still satisfied by the semantics of [8] because in every minimal Herbrand model p has a producer that is also an instance of $\texttt{Producer}$ and O.

Third, the disjunctions and ICs may also interact in unexpected ways.

Example 5 *Consider the following KB $\mathcal{K}$ where there are two categories for products and a constraint α defined on one of the categories:*

$\mathcal{K} = \{\texttt{Product} \sqsubseteq \texttt{Category1} \sqcup \texttt{Category2}, \texttt{Product}(p)\}$
$\alpha : \texttt{Category1} \sqsubseteq \exists\texttt{categoryType}.\top$

Since we do not know for sure that p belongs to `Category1`, it is reasonable to assume that the constraint α will not apply to p and α will not be violated. However, with [8] semantics, α is violated because there is a minimal model where p belongs to `Category1` but it does not have a `categoryType` value.

In this paper, we present a new IC semantics for OWL that overcomes the above issues and enables efficient IC validation for OWL.

4   Preliminaries

4.1   Description Logics $\mathcal{SROIQ}$

In this section, we give a brief description about the syntax and semantics of the Description Logic $\mathcal{SROIQ}$ [9], which is the logical underpinning of OWL 2 [10]. More details can be found in [9].

Let N_C, N_R, N_I be non-empty and pair-wise disjoint sets of *atomic concepts*, *atomic roles* and *named individuals* respectively. The $\mathcal{SROIQ}$ role R is an atomic role or its inverse R^-. Concepts are defined inductively as follows:

$$C \leftarrow A \mid \neg C \mid C_1 \sqcap C_2 \mid \geq nR.C \mid \exists R.\text{Self} \mid \{a\}$$

where $A \in N_C$, $a \in N_I$, $C_{(i)}$ a concept, R a role.

We use the following standard abbreviations for concept descriptions: $\bot = C \sqcap \neg C$, $\top = \neg\bot$, $C \sqcup D = \neg(\neg C \sqcap \neg D)$, $\leq nR.C = \neg(\geq n+1\,R.C)$, $\exists R.C = (\geq 1\,R.C)$, $\forall R.C = \neg(\exists R.\neg C)$, $\{a_1, \ldots, a_n\} = \{a_1\} \sqcup \cdots \sqcup \{a_n\}$.

A $\mathcal{SROIQ}$-*interpretation* $\mathcal{I} = (\Delta, \cdot^{\mathcal{I}})$, where Δ is the *domain*, and $\cdot^{\mathcal{I}}$ is the *interpretation function* which maps $A \in N_C$ to a subset of Δ, $R \in N_R$ to a subset of $\Delta \times \Delta$, $a \in N_I$ to an element of Δ. The interpretation can be extended to inverse roles and complex concepts as follows:

$$(R^-)^{\mathcal{I}} = \{\langle y, x \rangle \mid \langle x, y \rangle \in R^{\mathcal{I}}\}, (\neg C)^{\mathcal{I}} = \Delta \setminus C^{\mathcal{I}}, (C \sqcap D)^{\mathcal{I}} = C^{\mathcal{I}} \cap D^{\mathcal{I}},$$

$$(\geq nR.C)^{\mathcal{I}} = \{x \mid \#\{y.\langle x, y \rangle \in R^{\mathcal{I}} \text{ and } y \in C^{\mathcal{I}}\} \geq n\}$$

$$(\exists R.\text{Self})^{\mathcal{I}} = \{x \mid \langle x, x \rangle \in R^{\mathcal{I}}\}, \{a\}^{\mathcal{I}} = \{a^{\mathcal{I}}\}.$$

where $\#$ denotes the cardinality of a set.

A $\mathcal{SROIQ}$ knowledge base $\mathcal{K}$ is a collection of $\mathcal{SROIQ}$ axioms, including TBox, RBox, and ABox axioms. A $\mathcal{SROIQ}$-interpretation $\mathcal{I}$ satisfies an axiom α, denoted $\mathcal{I} \models \alpha$, if $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ ($R_1^{\mathcal{I}} \subseteq R_2^{\mathcal{I}}$, $R_1^{\mathcal{I}} \circ \ldots \circ R_n^{\mathcal{I}} \subseteq R^{\mathcal{I}}$, $\forall x \in \Delta : \langle x, x \rangle \in R^{\mathcal{I}}$, $\forall x \in \Delta : \langle x, x \rangle \notin R^{\mathcal{I}}$, $R_1^{\mathcal{I}} \cap R_2^{\mathcal{I}} = \emptyset$ resp.) holds when $\alpha = C \sqsubseteq D$ ($R_1 \sqsubseteq R_2$, $R_1 \ldots R_n \sqsubseteq R$, $\texttt{Ref}(R)$, $\texttt{Irr}(R)$, $\texttt{Dis}(R_1, R_2)$ resp.). Note that, there are also four kinds of ABox axioms ($C(a)$, $R(a,b)$, $a = b$, $a \neq b$). Their semantics is given by encoding them as TBox axioms ($\{a\} \sqsubseteq C$, $\{a\} \sqsubseteq \exists R.\{b\}$, $\{a\} \sqsubseteq \{b\}$, $\{a\} \sqsubseteq \neg\{b\}$, resp.). $\mathcal{I}$ is a *model* of $\mathcal{K}$ if it satisfies all the axioms in $\mathcal{K}$. We define $Mod(\mathcal{K})$ to be the set of all interpretations that are models of $\mathcal{K}$. We say $\mathcal{K}$ *entails* α, written as $\mathcal{K} \models \alpha$, if $\mathcal{I} \models \alpha$ for all models $\mathcal{I} \in Mod(\mathcal{K})$.

4.2   Distinguished Conjunctive Queries (DCQs)

We now describe the syntax and semantics of *distinguished conjunctive queries* (DCQs). Let N_V be a non-empty set of variable names disjoint from N_I, N_C, and N_R. A *query atom* is an ABox axiom where variables can be used in place of individuals. Formally, it is defined as follows:

$$q \leftarrow C(x) \mid R(x,y) \mid \neg R(x,y) \mid x = y \mid x \neq y$$

where $x, y \in N_I \cup N_V$, C is a concept, and R is a role. A *conjunctive query (CQ)* is the conjunction of query atoms:

$$Q \leftarrow q \mid Q_1 \wedge Q_2$$

A DCQ is a CQ containing only distinguished variables.[4]

The semantics of DCQs are given in terms of interpretations defined in Section 4.1. We define an *assignment* $\sigma : N_V \rightarrow N_I$ to be a mapping from the variables used in the query to named individuals in the KB. We define $\sigma(Q)$ to denote the application of an assignment σ to a query Q such that the variables in the query are replaced with individuals according to the mapping. We say a KB $\mathcal{K}$ entails a *DCQ Q* with an assignment σ, written as $\mathcal{K} \models^\sigma Q$, if:

$$
\begin{array}{lll}
\mathcal{K} \models^\sigma q & \text{iff} & \mathcal{K} \models \sigma(q) \\
\mathcal{K} \models^\sigma Q_1 \wedge Q_2 & \text{iff} & \mathcal{K} \models^\sigma Q_1 \text{ and } \mathcal{K} \models^\sigma Q_2
\end{array}
$$

We define the *answers to a query*, $\mathbf{A}(Q, \mathcal{K})$, to be the set of all assignments for which the KB entails the query. That is, $\mathbf{A}(Q, \mathcal{K}) = \{\sigma \mid \mathcal{K} \models^\sigma Q\}$. We say that a query is true w.r.t. a KB, denoted $\mathcal{K} \models Q$, if there is at least one answer for the query, and false otherwise.

5   IC Semantics for OWL

There has been a significant amount of research to define the semantics of ICs for relational databases, deductive databases, and knowledge representation systems in general. There are several proposals based on KB consistency or KB entailment. Against both of these approaches, Reiter argued that ICs are epistemic in nature and are about "what the knowledge base knows" in [11]. He proposed that ICs should be epistemic first-order queries that will be asked to a standard KB that does not contain epistemic axioms.

We agree with Reiter about the epistemic nature of ICs and believe this is the most appropriate semantics for ICs. In the following section, we describe an alternative IC semantics for OWL axioms, which is similar to how the semantics of epistemic DL $\mathcal{ALCK}$ [6] and MKNF DL $\mathcal{ALCK}_{\mathcal{NF}}$ [7] are defined. Then, in Section 5.2, we discuss how the IC semantics addresses the issues explained in Section 1 and Section 3, and enables OWL to be an IC language.

---

[4] A distinguished variable can be mapped to only known individuals, i.e., an element from N_I

5.1 Formalization

We define *IC-interpretation* as a pair $\mathcal{I}, \mathcal{U}$ where $\mathcal{I}$ is a $\mathcal{SROIQ}$ interpretation defined over the domain $\Delta^{\mathcal{I}}$ and $\mathcal{U}$ is a set of $\mathcal{SROIQ}$ interpretations. The IC-interpretation function $.^{\mathcal{I},\mathcal{U}}$ maps concepts to a subset of Δ, roles to a subset of $\Delta \times \Delta$ and individuals to an element of Δ as follows:

$$C^{\mathcal{I},\mathcal{U}} = \{x^{\mathcal{I}} \mid x \in N_I \text{ s.t. } \forall \mathcal{J} \in \mathcal{U}, x^{\mathcal{J}} \in C^{\mathcal{J}}\}$$
$$R^{\mathcal{I},\mathcal{U}} = \{\langle x^{\mathcal{I}}, y^{\mathcal{I}} \rangle \mid x, y \in N_I \text{ s.t. } \forall \mathcal{J} \in \mathcal{U}, \langle x^{\mathcal{J}}, y^{\mathcal{J}} \rangle \in R^{\mathcal{J}}\}$$

where C is an atomic concept and R is a role. According to this definition, $C^{\mathcal{I},\mathcal{U}}$ is the interpretation of named individuals that are instances of C in every (conventional) interpretation from $\mathcal{U}$. $R^{\mathcal{I},\mathcal{U}}$ can be understood similarly.

IC-interpretation is extended to inverse roles and complex concepts as follows:

$$(R^-)^{\mathcal{I},\mathcal{U}} = \{\langle x^{\mathcal{I}}, y^{\mathcal{I}} \rangle \mid \langle y^{\mathcal{I}}, x^{\mathcal{I}} \rangle \in R^{\mathcal{I},\mathcal{U}}\},$$
$$(C \sqcap D)^{\mathcal{I},\mathcal{U}} = C^{\mathcal{I},\mathcal{U}} \cap D^{\mathcal{I},\mathcal{U}}, \quad (\neg C)^{\mathcal{I},\mathcal{U}} = N_I \setminus C^{\mathcal{I},\mathcal{U}},$$
$$(\geq nR.C)^{\mathcal{I},\mathcal{U}} = \{x^{\mathcal{I}} \mid x \in N_I \text{ s.t. } \#\{y^{\mathcal{I}} \mid \langle x^{\mathcal{I}}, y^{\mathcal{I}} \rangle \in R^{\mathcal{I},\mathcal{U}}, y^{\mathcal{I}} \in C^{\mathcal{I},\mathcal{U}}\} \geq n\},$$
$$(\exists R.\mathrm{Self})^{\mathcal{I},\mathcal{U}} = \{x^{\mathcal{I}} \mid x \in N_I \text{ s.t. } \langle x^{\mathcal{I}}, x^{\mathcal{I}} \rangle \in R^{\mathcal{I},\mathcal{U}}\}, \{a\}^{\mathcal{I},\mathcal{U}} = \{a^{\mathcal{I}}\}.$$

We can see that the IC-interpretation $\mathcal{I}, \mathcal{U}$ is using the closed-world assumption. For example, the elements of $C^{\mathcal{I},\mathcal{U}}$ are the interpretation of named individuals that should be in the interpretation set of $C^{\mathcal{I}}$ for all $\mathcal{I} \in \mathcal{U}$. Any named individual that can not be proven to be an instance of C is assumed to be an instance of $\neg C$ since $(\neg C)^{\mathcal{I},\mathcal{U}}$ is the complement of $C^{\mathcal{I},\mathcal{U}}$ w.r.t. N_I.

Note that, although the IC interpretations have some similarities to the epistemic interpretations of $\mathcal{ALCK}$ and $\mathcal{ALCK_{NF}}$ [6, 7], there are some important differences. First, the IC interpretation in our approach is applicable to any $\mathcal{SROIQ}$ DL KB while the expressivity of DLs in [6, 7] is limited to $\mathcal{ALC}$. Second, in $\mathcal{ALCK}$ and $\mathcal{ALCK_{NF}}$ [6, 7], strict UNA is used by the interpretations which is not the case in IC interpretations.

In our IC semantics, we want to adopt a weak form of UNA; that is, two named individuals with different identifiers are assumed to be different by default unless their equality is required to satisfy the axioms in the KB. This idea is similar to minimal model semantics where equality relation is treated as a congruence relation and minimized.

We formalize this notion of weak UNA by defining Minimal Equality (ME) models. We start by defining the $\prec_=$ relation. Given two models $\mathcal{I}$ and $\mathcal{J}$, we say $\mathcal{J} \prec_= \mathcal{I}$ if all of the following conditions hold:

- For every concept C, $\mathcal{J} \models C(a)$ implies $\mathcal{I} \models C(a)$;
- For every role R, $\mathcal{J} \models R(a, b)$ implies $\mathcal{I} \models R(a, b)$;
- $E_{\mathcal{J}} \subset E_{\mathcal{I}}$

where $E_{\mathcal{I}}$ is the set of equality relations between named individuals (equality relations, for short) satisfied by $\mathcal{I}$:

$$E_{\mathcal{I}} = \{\langle a, b \rangle \mid a, b \in N_I \text{ s.t. } \mathcal{I} \models a = b\}$$

$Mod_{ME}(\mathcal{K})$ is the models of $\mathcal{K}$ with minimal equality (ME) between named individuals. Formally, we define

$$Mod_{ME}(\mathcal{K}) = \{\mathcal{I} \in Mod(\mathcal{K}) \mid \nexists \mathcal{J}, \mathcal{J} \in Mod(\mathcal{K}), \mathcal{J} \prec_{=} \mathcal{I}\}$$

It is easy to see that for every ME model $\mathcal{I}$ in $Mod_{ME}(\mathcal{K})$, there is no model $\mathcal{J}$ of $\mathcal{K}$ where $E_{\mathcal{J}} \subset E_{\mathcal{I}}$. Two different named individuals are interpreted as equivalent in $\mathcal{I} \in Mod_{ME}(\mathcal{K})$ only if this equality is necessary to make $\mathcal{I}$ being a model of $\mathcal{K}$. For example, suppose we have the axiom $a = \{b\} \sqcup \{c\}$ in $\mathcal{K}$. Then, $\forall \mathcal{I} \in Mod(\mathcal{K})$, one of the following three conditions hold: (1) $a^{\mathcal{I}} = b^{\mathcal{I}}, a^{\mathcal{I}} \neq c^{\mathcal{I}}$; (2) $a^{\mathcal{I}} = c^{\mathcal{I}}, a^{\mathcal{I}} \neq b^{\mathcal{I}}$; (3) $a^{\mathcal{I}} = b^{\mathcal{I}} = c^{\mathcal{I}}$. If (1) or (2) holds, then $\mathcal{I} \in Mod_{ME}(\mathcal{K})$ because a has to be interpreted to be equivalent to at least one of b and c to make $\mathcal{I}$ being a model of $\mathcal{K}$. Whereas for case (3), $\mathcal{I} \notin Mod_{ME}(\mathcal{K})$ since the equality relations in $\mathcal{I}$ are not minimal.

An IC-interpretation $\mathcal{I}, \mathcal{U}$ satisfies an axiom α, denoted as $\mathcal{I}, \mathcal{U} \models \alpha$, if $C^{\mathcal{I}, \mathcal{U}} \subseteq D^{\mathcal{I}, \mathcal{U}}$ ($R_1^{\mathcal{I}, \mathcal{U}} \subseteq R_2^{\mathcal{I}, \mathcal{U}}$, $R_1 \sqsubseteq R_2$, $R_1^{\mathcal{I}, \mathcal{U}} \subseteq R_2^{\mathcal{I}, \mathcal{U}}$, $\forall x \in N_I : \langle x^{\mathcal{I}, \mathcal{U}}, x^{\mathcal{I}, \mathcal{U}} \rangle \in R^{\mathcal{I}, \mathcal{U}}$, $\forall x \in N_I : \langle x^{\mathcal{I}, \mathcal{U}}, x^{\mathcal{I}, \mathcal{U}} \rangle \notin R^{\mathcal{I}, \mathcal{U}}$, $R_1^{\mathcal{I}, \mathcal{U}} \cap R_2^{\mathcal{I}, \mathcal{U}} = \emptyset$ resp.) holds when $\alpha = C \sqsubseteq D$ ($R_1 \sqsubseteq R_2$, $R_1 \ldots R_n \sqsubseteq R$, $\mathtt{Ref}(R)$, $\mathtt{Irr}(R)$, $\mathtt{Dis}(R_1, R_2)$ resp.).

Given a $\mathcal{SROIQ}$ KB $\mathcal{K}$ and a $\mathcal{SROIQ}$ constraint α, the IC-satisfaction of α by $\mathcal{K}$, i.e., $\mathcal{K} \models_{IC} \alpha$, is defined as:

$$\mathcal{K} \models_{IC} \alpha \text{ iff } \forall \mathcal{I} \in \mathcal{U}, \mathcal{I}, \mathcal{U} \models \alpha, \text{where } \mathcal{U} = Mod_{ME}(\mathcal{K})$$

We define an *extended KB* as a pair $\langle \mathcal{K}, \mathcal{C} \rangle$ where $\mathcal{K}$ is a $\mathcal{SROIQ}$ KB as before and $\mathcal{C}$ is a set of $\mathcal{SROIQ}$ axioms interpreted with IC semantics. We say that $\langle \mathcal{K}, \mathcal{C} \rangle$ is valid if $\forall \alpha \in \mathcal{C}, \mathcal{K} \models_{IC} \alpha$, otherwise there is an IC violation.

5.2   Discussion

It is easy to verify that the IC semantics provides expected results for the examples presented in Section 1 and Section 3. For instance, we get an IC violation in Example 1 since the IC interpretation of $\mathtt{Product}$ contains p but the IC interpretation of $(\exists \mathtt{hasProducer}.\mathtt{Producer})$ is empty.

The following example shows how weak UNA allows the individuals that are not asserted to be equal to be treated different for constraint validation purposes.

Example 6 *Consider the KB $\mathcal{K}$ and the constraint α:*

$$\mathcal{K} = \{C(c), R(c, d_1), R(c, d_2), D(d_1), D(d_2)\}, \quad \alpha : C \sqsubseteq \geq 2R.D$$

With the weak UNA, d_1 and d_2 are interpreted to be different in every ME model. Therefore, the IC-interpretation of $(\geq 2R.D)$ includes c, and α is satisfied by $\mathcal{K}$.

Now we illustrate another point regarding disjunctions in constraints.

Example 7 *Suppose we have the KB $\mathcal{K}$ and constraint α:*

$$\mathcal{K} = \{C(a), (C_1 \sqcup C_2)(a)\}, \quad \alpha : C \sqsubseteq C_1 \sqcup C_2$$

Constraint α should be read as "every instance of C should be either a known instance of C_1 or a known instance of C_2". Since we do not know *for sure* whether a belongs to C_1 or C_2, α is expected to be violated by $\mathcal{K}$. Indeed, according to our semantics we get $C^{\mathcal{I},\mathcal{U}} = \{a^{\mathcal{I}}\}$ and $(C_1 \sqcup C_2)^{\mathcal{I},\mathcal{U}} = \emptyset$. Therefore $C^{\mathcal{I},\mathcal{U}} \nsubseteq (C_1 \sqcup C_2)^{\mathcal{I},\mathcal{U}}$ and we conclude there is an IC violation.

If we want to represent the alternative constraint: "every instance of C should be an instance of C_1 or C_2", we can define a new name C' in the KB to substitute $C_1 \sqcup C_2$, thus having the new KB $\mathcal{K}'$ and constraint α' as follows:

$$\mathcal{K}' = \{C(a), (C_1 \sqcup C_2)(a), C' \equiv C_1 \sqcup C_2\}, \quad \alpha' : C \sqsubseteq C'$$

There is no IC violation in this version because now the disjunction is interpreted as standard OWL axioms. As these examples show, we can model the constraints to express different disjunctions in a flexible way.

6 IC Validation

We have defined in Section 5.1 that, the extended KB $\langle \mathcal{K}, \mathcal{C} \rangle$ is valid if every IC axiom in $\mathcal{C}$ is IC-satisfied by $\mathcal{K}$. In this section, we describe how to do IC validation, i.e., check IC-satisfaction by translating constraint axioms to queries with the NAF operator **not** . We start by giving the formal semantics for **not** in DCQs, then describe the translation rules from IC axioms to DCQ$^{\mathbf{not}}$ and finally provide a theorem showing that IC validation can be reduced to answering DCQ$^{\mathbf{not}}$ under certain conditions.

6.1 DCQ$^{\mathbf{not}}$

In Section 4.2 , we introduced standard DCQs. However, the expressivity of standard DCQs is not enough to capture the closed world nature of IC semantics. For this reason, we add the **not** operator to DCQs to get DCQ$^{\mathbf{not}}$ queries. The syntax of DCQ$^{\mathbf{not}}$ is defined as follows:

$$Q \leftarrow q \mid Q_1 \wedge Q_2 \mid \mathbf{not}\, Q$$

The semantics of **not** is defined as:

$$\mathcal{K} \models^{\sigma} \mathbf{not}\, Q \qquad \text{iff} \qquad \nexists \sigma' \text{ s.t. } \mathcal{K} \models^{\sigma'} \sigma(Q)$$

And we use the abbreviation $Q_1 \vee Q_2$ for $\mathbf{not}\,(\mathbf{not}\, Q_1 \wedge \mathbf{not}\, Q_2)$. We can see

$$\mathcal{K} \models^{\sigma} Q_1 \vee Q_2 \qquad \text{iff} \qquad \mathcal{K} \models^{\sigma} Q_1 \text{ or } \mathcal{K} \models^{\sigma} Q_2$$

6.2 Translation Rules: from ICs to DCQ$^{\mathbf{not}}$

We now present the translation rules from IC axioms to DCQ$^{\mathbf{not}}$ queries. The translation rules are similar in the spirit to the Lloyd-Topor transformation [12]

but instead of rules we generate DCQ**not** queries. The idea behind the translation is to translate a constraint axiom into a query such that when the constraint is violated the KB entails the query. In other words, whenever the answer to the query is not empty, we can conclude that the constraint is violated.

The translation contains two operators: $\mathcal{T}_c$ for translating concepts and $\mathcal{T}$ for translating axioms. $\mathcal{T}_c$ is a function that takes a concept expression and a variable as input and returns a DCQ**not** query as the result:

$$\mathcal{T}_c(C_a, x) := C_a(x)$$
$$\mathcal{T}_c(\neg C, x) := \textbf{not } \mathcal{T}_c(C, x)$$
$$\mathcal{T}_c(C_1 \sqcap C_2, x) := \mathcal{T}_c(C_1, x) \wedge \mathcal{T}_c(C_2, x)$$
$$\mathcal{T}_c(\geq nR.C, x) := \bigwedge_{1 \leq i \leq n} (R(x, y_i) \wedge \mathcal{T}_c(C, y_i)) \bigwedge_{1 \leq i < j \leq n} \textbf{not } (y_i = y_j)$$
$$\mathcal{T}_c(\exists R.\text{Self}, x) := R(x, x)$$
$$\mathcal{T}_c(\{a\}, x) := (x = a)$$

where C_a is an atomic concept, $C_{(i)}$ is a concept, R is a role, a is an individual, x is an input variable, and $y_{(i)}$ is a fresh variable.

$\mathcal{T}$ is a function that maps a $\mathcal{SROIQ}$ axiom to a DCQ**not** query as follows:

$$\mathcal{T}(C_1 \sqsubseteq C_2) := \mathcal{T}_c(C_1, x) \wedge \textbf{not } \mathcal{T}_c(C_2, x)$$
$$\mathcal{T}(R_1 \sqsubseteq R_2) := R_1(x, y) \wedge \textbf{not } R_2(x, y)$$
$$\mathcal{T}(R_1 \ldots R_n \sqsubseteq R) := R_1(x, y_1) \wedge \ldots R_n(y_{n-1}, y_n) \wedge \textbf{not } R(x, y_n)$$
$$\mathcal{T}(\texttt{Ref}(R)) := \textbf{not } R(x, x)$$
$$\mathcal{T}(\texttt{Irr}(R)) := R(x, x)$$
$$\mathcal{T}(\texttt{Dis}(R_1, R_2)) := R_1(x, y) \wedge R_2(x, y)$$

where $C_{(i)}$ is a concept, $R_{(i)}$ is a role, x and $y_{(i)}$ is variable.

6.3   Reducing IC Validation to Answering DCQ**not**

In Theorem 1, we show that IC validation via query answering is sound and complete when the expressivity of the extended KB is either $\langle \mathcal{SRI}, \mathcal{SROIQ} \rangle$ or $\langle \mathcal{SROIQ}, \mathcal{SROI} \rangle$. Note that, when the expressivity is $\langle \mathcal{SROIQ}, \mathcal{SROIQ} \rangle$, we can not reduce IC validation to query answering in a straightforward way due to the interaction between the disjunctive (in)equality axioms in $\mathcal{K}$ and the cardinality constraints in $\mathcal{C}$. We limit this interaction by either excluding nominals and cardinality restrictions in $\mathcal{K}$ thus prohibiting disjunctive (in)equality to appear in $\mathcal{K}$, or by prohibiting cardinality restrictions in $\mathcal{C}$. Due to space limitations we only present the main theorem here. The complete proofs are presented in the technical report [13].

Theorem 1 *Given an extended KB $\langle \mathcal{K}, \mathcal{C} \rangle$ with expressivity $\langle \mathcal{SRI}, \mathcal{SROIQ} \rangle$ ($\langle \mathcal{SROIQ}, \mathcal{SROI} \rangle$ resp.), we have that $\mathcal{K} \models_{IC} \alpha$ iff $\mathcal{K} \not\models \mathcal{T}(\alpha)$ where $\alpha \in \mathcal{C}$.*

7 Implementation

The emerging best practice query language for OWL ontologies is SPARQL
[14] which is known to have the same expressive power as nonrecursive Datalog
programs [15] and can express DCQ**not** queries. Therefore, based on the results
from Section 6.3, we can reduce IC validation to SPARQL query answering if
the KB is $\mathcal{SRI}$ or the ICs do not contain cardinality restrictions.

We have built a prototype IC validator[5] by extending the OWL 2 DL reasoner
Pellet[6]. The prototype reads ICs expressed as OWL axioms and translates each
IC first to a DCQ**not** query and then to a SPARQL query. The resulting query
is executed by the SPARQL engine in Pellet where a non-empty result indicates
a constraint violation. Since the translation algorithm is reasoner independent
this prototype can be used in conjunction with any OWL reasoner that supports
SPARQL query answering.

We have used this proof-of-concept prototype to validate ICs with several
large ontologies such as the LUBM dataset.[7] For testing, we removed several
axioms from the LUBM ontology and declared them as ICs instead. The dataset
is logically consistent but turning axioms into ICs caused some violations to
be detected. Since each constraint is turned into a separate query there is no
dependence between the validation time of different constraints. We have not
performed extensive performance analysis for IC validation but as a simple com-
parison we looked at logical consistency checking time vs. IC validation time. For
LUBM(5), which has 100K individuals and 800K ABox axioms, logical consis-
tency checking was in average 10s whereas validating a single IC took in average
2s. The naive approach in our prototype to execute each query separately would
not scale well as the number of ICs increase. However, there are many improve-
ment possibilities ranging from combining similar queries into a single query to
running multiple queries in parallel.

8 Conclusions and Future Work

In this paper, we described how to provide an IC semantics for OWL axioms
that can be used for data validation purposes. Our IC semantics provide intuitive
results for various different use cases we examined. We presented translation
rules from IC axioms to DCQ**not** queries, showing that IC validation can be
reduced to query answering when the KB expressivity is $\mathcal{SRI}$ or constraint
expressivity is $\mathcal{SROI}$. Our preliminary results with a prototype IC validator
implementation show that existing OWL reasoners can be used for IC validation
efficiently with little effort. Using SPARQL queries for IC validation makes our
approach applicable to a wide range of reasoners. In the future, we will be looking
at the performance of IC validation in realistic datasets and will be exploring
the IC validation algorithms for the full expressivity of $\mathcal{SROIQ}$.

---

[5] http://clarkparsia.com/pellet/oicv-0.1.2.zip

[6] http://clarkparsia.com/pellet

[7] http://swat.cse.lehigh.edu/projects/lubm/

References

1. Smith, M.K., Welty, C., McGuiness, D.: OWL web ontology language guide (2004)
2. Colomb, R.M.: Deductive Databases and Their Applications. CRC Press (1998)
3. Eiter, T., Ianni, G., Lukasiewicz, T., Schindlauer, R., Tompits, H.: Combining answer set programming with description logics for the semantic web. AI **172**(12-13) (2008) 1495–1539
4. Motik, B.: A faithful integration of description logics with logic programming. In: IJCAI2007. (2007) 477–482
5. Calvanese, D., Giacomo, G.D., Lembo, D., Lenzerini, M., Rosati, R.: EQL-Lite: Effective first-order query processing in description logics. In: IJCAI2007. (2007) 274–279
6. Donini, F.M., Lenzerini, M., Nardi, D., Nutt, W., Schaerf, A.: An epistemic operator for description logics. AI **100**(1–2) (1998) 225–274
7. Donini, F.M., Nardi, D., Rosati, R.: Description logics of minimal knowledge and negation as failure. ACM Transactions on Computational Logic **3** (2002) 177–225
8. Motik, B., Horrocks, I., Sattler, U.: Bridging the gap between OWL and relational databases. In: WWW2007. (2007) 807–816
9. Horrocks, I., Kutz, O., Sattler, U.: The even more irresistible SROIQ. In: KR2006. (2006) 57–67
10. Motik, B., Patel-Schneider, P.F., Grau, B.C.: OWL 2 web ontology language direct semantics (2009)
11. Reiter, R.: On integrity constraints. In: TARK1988. (1988) 97–111
12. Lloyd, J.W.: Foundations of logic programming. (1987)
13. Tao, J., Sirin, E., Bao, J., McGuinness, D.: Integrity constraints in OWL. Technical report, Rensselaer Polytechnic Institute, Troy, NY, USA (2010)
14. Prud'hommeaux, E., Seaborne, A.: SPARQL query language for RDF (2008)
15. Angles, R., Gutierrez, C.: The expressive power of SPARQL. In: ISWC2008. (2008) 114–129

Query Answering in the Description Logic $\mathcal{S}^\star$

Meghyn Bienvenu[1], Thomas Eiter[2], Carsten Lutz[1],
Magdalena Ortiz[2], Mantas Šimkus[2]

[1] Fachbereich Mathematik und Informatik
Universität Bremen, Germany
(meghyn|clu)@informatik.uni-bremen.de

[2] Institute of Information Systems
TU Vienna, Austria
(eiter|ortiz|simkus)@kr.tuwien.ac.at

Abstract. We consider the complexity of answering conjunctive queries in the description logic $\mathcal{S}$, i.e., in $\mathcal{ALC}$ extended with transitive roles. While a co-NExpTime lower bound was recently established in [5], the best known upper bound was 2-ExpTime. In this paper, we concentrate on the case where only a single transitive role (and no other role) is present and establish a tight co-NExpTime upper bound.

1 Introduction

Formal ontologies have gained significant importance in the last decade and play an increasing role in a growing number of application areas including the semantic web, ontology-based information integration, and peer-to-peer data management. As a result, ontology formalisms such as description logics (DLs) are nowadays required to offer support for query answering that goes beyond simple taxonomic questions and membership queries. In particular, conjunctive queries (CQs) over instance data play a central role in many applications and have consequently received considerable attention, cf. [11, 6, 9] and references therein and below.

A main aim of recent research has been to identify the potential and limitations of CQ answering in various DLs by mapping out the complexity landscape of this reasoning problem. When concerned with inexpressive DLs such as DL-Lite and $\mathcal{EL}$, one is typically interested in data complexity and efficient implementations based on relational database systems [3, 8]. In expressive DLs, the data complexity is almost always coNP-complete and it is more interesting to study combined complexity. While 2-ExpTime upper bounds for expressive DLs of the $\mathcal{ALC}$ family are known since 1998 [4], lower bounds except ExpTime-hardness (which is trivially inherited from satisfiability) have long been elusive. A first step was made in [7], where *inverse roles* were identified as a source of complexity: CQ answering in plain $\mathcal{ALC}$ remains ExpTime-complete, but goes up to 2-ExpTime-completeness in $\mathcal{ALCI}$. When further extending $\mathcal{ALCI}$ to the popular DL $\mathcal{SHIQ}$, CQ answering remains 2-ExpTime-complete [6].

Interestingly, inverse roles turn out not to be the only source of complexity in $\mathcal{SHIQ}$. In [5], we have shown that transitive roles, which play a central role in many ontologies and are used to represent fundamental relations such as "part of" [10], also increase the complexity of CQ answering. More specifically, CQ answering is CO-NExpTime-hard in the DL $\mathcal{S}$, which is $\mathcal{ALC}$ extended with transitive roles and the basic logic of the $\mathcal{SHIQ}$ family, even with only a single transitive role and no other roles (and when the TBox is empty). We have also shown in [5] that if we further add role hierarchies and thus extend $\mathcal{S}$ to $\mathcal{SH}$, CQ answering even becomes 2-ExpTime-complete.

However, the precise complexity of CQ answering in $\mathcal{S}$ has remained open between CO-NExpTime and 2-ExpTime. The only existing tight bound (also from [5]) concerns tree-shaped ABoxes, for which CQ answering in $\mathcal{S}$ is only ExpTime-complete (which is remarkable because previously known lower bounds for CQ answering in DLs did not rely on the ABox structure). In this paper, we present ongoing work on CQ answering in $\mathcal{S}$ and show that, in the presence of only a single transitive role and no other role, CQ answering in $\mathcal{S}$ is in CO-NExpTime, thus CO-NExpTime-complete. This result is interesting for two reasons. First, CO-NExpTime is an unusual complexity class for CQ answering in expressive DLs as all previous extensions of $\mathcal{ALC}$ have turned out to be complete for a deterministic time complexity class; the only exception is a CO-NExpTime result for $\mathcal{ALCI}$ in [7] which is, however, entirely unsurprising because it concerns a syntactically and semantically restricted case ("rooted CQ answering") where a CO-NExpTime bound comes naturally. And second, we believe that the presented upper bound can be extended to the general case where an arbitrary number of roles is allowed, though at the expense of making it considerably more technical.

As usual, we consider conjunctive query entailment instead of CQ answering, i.e., we replace the search problem by its decision problem counterpart. We use the following strategy to obtain a CO-NExpTime upper bound for CQ entailment. First, we use a standard technique to show that CQ entailment over unrestricted ABoxes can be reduced to entailment of UCQs (unions of conjunctive queries) over ABoxes that contain only a single individual and no role assertions. More precisely, we use a Turing reduction that requires an exponential number of UCQ entailment checks, where each UCQ contains exponentially many disjuncts in the worst case. Thus, it suffices to establish a co-NExpTime upper bound for each of the required UCQ entailments. Second, we show that if one of the UCQ entailments does not hold, then there is a tree-shaped counter-model with only polynomially many types on each path. Third, we characterize counter-models in terms of tree-interpretations that are annotated in a certain way with sub-queries of the original CQ (so-called Q-markings). Thus, we can decide UCQ-(non)-entailment by deciding the existence of a Q-marked tree-interpretation. Fourth, we show that, additionally to the restriction on the number of types, it suffices to consider Q-marked tree-interpretations in which there are only polynomially many different annotations on each path. Finally, we prove that the existence of a Q-marked tree-interpretation with the mentioned restrictions on

the number of types and annotations can be checked by guessing an initial part of the annotated tree-interpretation that has only polynomial depth and thus exponential size, which gives the desired co-NExpTime bound.

2 Preliminaries

We briefly introduce the description logic $\mathcal{S}$, conjunctive queries, and conjunctive query entailment.

Knowledge Bases. We assume standard notation for the syntax and semantics of $\mathcal{S}$ knowledge bases [6]. In particular, $\mathsf{N_C}$ and $\mathsf{N_I}$ are countably infinite and disjoint sets of *concept names* and *individual names*. For the purpose of this paper, we consider a *single transitive role*, denoted throughout by r. *Concepts* are defined inductively: (a) each $A \in \mathsf{N_C}$ is a concept, and (b) if C, D are concepts, then $C \sqcap D$, $\neg C$, and $\exists r.C$ are concepts.[1] A *TBox* is a set of concept inclusions $C \sqsubseteq D$. An *ABox* is a set of *assertions* $C(a)$ and $r(a, b)$. A *knowledge base (KB)* is a pair $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ consisting of a TBox $\mathcal{T}$ and an ABox $\mathcal{A}$. We use $\mathcal{I}$ to denote an interpretation, $\Delta^{\mathcal{I}}$ for its domain, and $C^{\mathcal{I}}$ and $r^{\mathcal{I}}$ for the interpretation of a concept C and the role r, respectively. We denote by $\mathsf{Ind}(\mathcal{A})$ the set of all individual names in an ABox $\mathcal{A}$.

Conjunctive Query Entailment. Let $\mathsf{N_V}$ be a countably infinite set of *variables*. A *conjunctive query (CQ or query)* over a KB $\mathcal{K}$ is a finite set of atoms of the form $A(x)$ or $r(x, y)$, where $x, y \in \mathsf{N_V}$, and A is a concept name.[2] For a CQ q over $\mathcal{K}$, let $\mathsf{Var}(q)$ denote the variables occurring in q. A *match for q in an interpretation $\mathcal{I}$* is a mapping $\pi : \mathsf{Var}(q) \rightarrow \Delta^{\mathcal{I}}$ such that (i) $\pi(x) \in A^{\mathcal{I}}$ for each $A(x) \in q$, and (ii) $(\pi(x), \pi(y)) \in r^{\mathcal{I}}$ for each $r(x, y) \in q$. We write $\mathcal{I} \models q$ if there is a match for q in $\mathcal{I}$. If $\mathcal{I} \models q$ for every model $\mathcal{I}$ of $\mathcal{K}$, then $\mathcal{K}$ *entails* q, written $\mathcal{K} \models q$. The *query entailment problem* is to decide, given $\mathcal{K}$ and q, whether $\mathcal{K} \models q$. We sometimes also consider *unions of conjunctive queries (UCQs)*, which take the form $\bigcup_i q_i$, where each q_i is a conjunctive query. The notions $\mathcal{I} \models q$ and $\mathcal{K} \models q$ are lifted from CQs to UCQs in the obvious way.

The directed graph G_q associated with a query q is defined as (V, E), where $V = \mathsf{Var}(q)$ and $E = \{(x, y) \mid r(x, y) \in q\}$. When deciding CQ entailment, we assume without loss of generality that the input query q (i.e., the graph G_q) is connected. For $V \subseteq \mathsf{Var}(q)$, we use $q|_{V^{\downarrow}}$ to denote the restriction of q to the set of variables that are reachable in G_q starting from some element in V. We call $q|_{V^{\downarrow}}$ a *proper subquery* of q if it is connected, and use $\mathsf{sub}(q)$ to denote the set of all proper subqueries of q. Obviously, $q \in \mathsf{sub}(q)$.

---

[1] Concepts of the form $C \sqcup D$ and $\forall r.C$ are viewed as abbreviations.

[2] As usual, individuals in q can be simulated, and queries with answer variables can be reduced to the Boolean CQs considered here.

3   Reduction to Unary ABoxes

The objective of this section is to reduce CQ entailment over arbitrary knowledge bases to UCQ entailment over knowledge bases whose ABoxes contain only a single concept assertion and no role assertions.

Let $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ be a knowledge base and q a CQ for which we want to decide whether $\mathcal{K} \models q$. We assume without loss of generality that $\mathcal{T} = \{\top \sqsubseteq C_{\mathcal{T}}\}$. The announced reduction, which is similar to one used in [5], makes use of the fact that if there is an interpretation $\mathcal{I}$ of $\mathcal{K}$ with $\mathcal{I} \not\models q$, then there is a forest-shaped such model, i.e., a model that consists of an ABox part of unrestricted relational structure and a tree-shaped part rooted at each ABox individual. To check for the existence of a countermodel of this form, we consider all ways in which the query variables can be distributed among the different parts of the model. The query has no match if for each possible distribution, we can select an ABox individual a such that some subquery assigned to the tree model below a is *not* matched in that tree model. This leaves us with the problem of determining the existence of certain tree models (one for each ABox individual) that spoil a (worst-case exponential) set of subqueries.

To formally implement this idea, we require a few preliminary definitions. We use $\mathsf{cl}(\mathcal{K})$ to denote the smallest set that contains $C_{\mathcal{T}}$, each concept C with $C(a) \in \mathcal{A}$, and is closed under single negation and subconcepts. A *type* is a subset $t \subseteq \mathsf{cl}(\mathcal{K})$ that satisfies the following conditions:

1. $\neg C \in t$ iff $t \notin C$, for all $\neg C \in \mathsf{cl}(\mathcal{T})$;
2. $C \sqcap D \in t$ iff $C \in t$ and $D \in t$, for all $C \sqcap D \in \mathsf{cl}(\mathcal{T})$;
3. $C_{\mathcal{T}} \in t$.

We use $\mathsf{tp}(\mathcal{K})$ to denote the set of all types for $\mathcal{K}$. A *completion* of $\mathcal{A}$ is an ABox $\mathcal{A}'$ such that

- $\mathcal{A} \subseteq \mathcal{A}'$ with $\mathsf{Ind}(\mathcal{A}) = \mathsf{Ind}(\mathcal{A}')$;
- for each $a \in \mathsf{Ind}(\mathcal{A})$, we have $\{C \mid C(a) \in \mathcal{A}'\} \in \mathsf{tp}(\mathcal{K})$;
- $r(a, b), r(b, c) \in \mathcal{A}'$ implies $r(a, c) \in \mathcal{A}'$;
- $\exists r.C \in \mathsf{cl}(\mathcal{K})$, $r(a, b) \in \mathcal{A}$, and $C(b) \in \mathcal{A}'$ implies $(\exists r.C)(a) \in \mathcal{A}'$.

We use $\mathsf{cpl}(\mathcal{A})$ to denote the set of all completions for $\mathcal{A}$. A *match candidate* for a completion $\mathcal{A}' \in \mathsf{cpl}(\mathcal{A})$ describes a way of distributing the query variables among the different parts of the model. Formally, it is a mapping $\zeta : \mathsf{Var}(q) \rightarrow \{a, a^{\downarrow} \mid a \in \mathsf{Ind}(\mathcal{A})\}$ such that

- if $A(x) \in q$ and $\zeta(x) = a$, then $A(a) \in \mathcal{A}'$;
- if $r(x, y) \in q$, $\zeta(x) = a$, and $\zeta(y) = b$, then $r(a, b) \in \mathcal{A}'$;
- if $r(x, y) \in q$, $\zeta(x) = a$, $\zeta(y) = b^{\downarrow}$, and $a \neq b$, then $r(a, b) \in \mathcal{A}'$;
- $r(x, y) \in q$ and $\zeta(x) = a^{\downarrow}$ implies $\zeta(y) = a^{\downarrow}$.

For every $r(x, y) \in q$ with $\zeta(x) = a$ and $\zeta(y) = b^{\downarrow}$ (where potentially $a = b$), define a subset $V \subseteq \mathsf{Var}(q)$ as the smallest set such that

- $y \in V$;

- if $r(x', y') \in q$ with $x' \in V$, then $y' \in V$;
- if $r(x', y') \in q$ with $y' \in V$ and $\zeta(x') = b^{\downarrow}$, then $x' \in V$.

We use $q|_{r(x,y)}$ to denote the restriction of q to the variables in V. Let Q_{ζ} denote the set of all queries $q|_{r(x,y)}$ obtained in this way. It is straightforward to verify that all these queries are proper subqueries, i.e., $Q_{\zeta} \subseteq \mathsf{sub}(q)$.

A *query annotation* for $\mathcal{A}'$ identifies the subqueries that do not have a match in the counter-model that we construct. Formally, it is a map $\alpha : \mathsf{Ind}(\mathcal{A}) \to 2^{\mathsf{sub}(q)}$ that satisfies the following conditions:

1. for every match candidate ζ for $\mathcal{A}'$, there is a query $q|_{r(x,y)} \in Q_{\zeta}$ such that $q|_{r(x,y)} \in \alpha(a)$ where $\zeta(y) = a^{\downarrow}$;
2. $q \in \alpha(a)$ for all $a \in \mathsf{Ind}(\mathcal{A})$.

For each $a \in \mathsf{Ind}(\mathcal{A})$, we use $\mathcal{A}'|_a$ to denote the restriction of $\mathcal{A}'$ to assertions of the form $C(a)$. The proof of the following lemma is similar to that of a closely related result in [6].

Lemma 1. $\mathcal{K} \not\models q$ *iff there is a completion* $\mathcal{A}'$ *of* $\mathcal{A}$ *and a query annotation* α *for* $\mathcal{A}'$ *such that for all* $a \in \mathsf{Ind}(\mathcal{A})$, *we have* $\mathcal{K}_a \not\models \bigcup \alpha(a)$, *where* $\mathcal{K}_a = (\mathcal{T}, \mathcal{A}'|_a)$.

Lemma 1 constitutes the announced reduction: to decide whether $\mathcal{K} \models q$, we can enumerate all completions $\mathcal{A}'$ of $\mathcal{A}$ and query annotations α for $\mathcal{A}'$, and then perform the required UCQ entailment checks.

4 Characterization of Counter-models

It remains to decide whether $\mathcal{K}_a \models \bigcup \alpha(a)$ holds for each $a \in \mathsf{Ind}(\mathcal{A})$. Since $\alpha(a)$ may contain exponentially many different subqueries of q (this is what actually happens in the lower bound proved in [5]), it is challenging to do this in CO-NEXPTIME. We start with a characterization of counter-models. In the remainder of the section, for readability, we fix some $a \in \mathsf{Ind}(\mathcal{A})$, and we use Q to denote $\alpha(a)$ and C_a to denote $\bigsqcap \{C \mid C(a) \in \mathcal{A}'\}$.

Many of the subsequent techniques and results will be concerned with trees and tree interpretations, which we introduce next. Let Σ be an arbitrary set. Then a *tree (over Σ with root p)* is a set $T = \{p \cdot w \mid w \in S\}$ where $p \in \Sigma^*$ and $S \subseteq \Sigma^*$ is a prefix-closed set of words. Each node $w \cdot c \in T$, where $w \in T$ and $c \in \Sigma$, is a *child* of w. For a node $w \in T$, $|w|$ denotes the length of w, disregarding the prefix p (so that the root of T has length 0). We say *the branching degree of T is bounded by k* if $|\{c \in \Sigma \mid w \cdot c \in T\}| \leq k$ for all $w \in T$. A *path in T*, is a (potentially infinite) sequence $w_0, w_1, \ldots$ of elements from T such that (i) w_0 is the root of T, and (ii) for each $i > 0$, w_i is a child of w_{i-1}. If T is a tree and $f : T \to S$ is a function with S finite, then we use $\mathsf{max}(T, f)$ to denote the maximal number of distinct values that f can take on an arbitrary path in T.

An interpretation $\mathcal{I}$ is a *tree interpretation* if $\Delta^{\mathcal{I}}$ is a tree. We introduce the notation $\mathsf{root}(\mathcal{I})$ to denote the root of the tree $\Delta^{\mathcal{I}}$. A tree interpretation $\mathcal{I}$ is a *tree model* of $\mathcal{K}_a$ if

– $\mathcal{I}$ is a model of $\mathcal{T}$, and $\mathsf{root}(\mathcal{I}) \in C_a^{\mathcal{I}}$,
– $r^{\mathcal{I}} = \{(w, w \cdot c) \mid w, w \cdot c \in \Delta^{\mathcal{I}} \wedge c \in \Sigma\}^+$, and
– for all $\exists r.C \in \mathsf{cl}(\mathcal{K})$ and $w \in (\exists r.C)^{\mathcal{I}}$, there is $c \in \Sigma$ such that $w \cdot c \in C^{\mathcal{I}}$, i.e., all relevant existential restrictions are satisfied in one step.

Given a tree interpretation $\mathcal{I}$ and $w \in \Delta^{\mathcal{I}}$, we use $\mathcal{I}|_w$ to denote the restriction of $\mathcal{I}$ to the subtree rooted at w.

The following lemma shows that we can restrict our attention to tree-shaped interpretations in which only polynomially many types appear on any given path. As the proof of the lemma is surprisingly subtle, we defer it to the appendix of a longer version of this submission [1]. Given an interpretation $\mathcal{I}$, we use $t_{\mathcal{I}}(w)$ to refer to the type of $w \in \Delta^{\mathcal{I}}$ in $\mathcal{I}$, i.e. $\{C \in \mathsf{cl}(\mathcal{K}) \mid w \in C^{\mathcal{I}}\}$.

Lemma 2. *If $\mathcal{K}_a \not\models \bigcup Q$, then there is an interpretation $\mathcal{I}$ such that:*

1. *$\mathcal{I}$ is a tree model of $\mathcal{K}_a$, and $\mathcal{I} \not\models \bigcup Q$, and*
2. *$\mathsf{max}(\Delta^{\mathcal{I}}, t_{\mathcal{I}}) \leq |\mathsf{cl}(\mathcal{K})|$.*

To characterize counter-models, we employ *marking* of interpretations, similar to that in [5]. A marking simulates a top-down walk through a tree interpretation $\mathcal{I}$ greedily matching the variables of the queries in Q. The marking fails if we arrive at a subquery that is fully matched along this walk. As we show next, the existence of a marking for a tree interpretation $\mathcal{I}$ is a necessary and sufficient condition for $\mathcal{I} \not\models \bigcup Q$.

For a query p and a variable $x \in \mathsf{Var}(p)$, we say that x is *consumed (in p)* by a type t if $\{A \mid A(x) \in p\} \subseteq t$ and $\{y \mid r(y, x) \in p\} = \emptyset$. Given a type $t \in \mathsf{tp}(\mathcal{K})$ and a query $p \in \mathsf{sub}(q)$, we denote by $\mathsf{sub}^t(p)$ the set of all proper subqueries of p^t, where p^t is obtained from p by removing all atoms involving a variable that is consumed by t. In other words, $\mathsf{sub}^t(p)$ is the set of connected components in the reduced query p^t. Trivially, $\mathsf{sub}^t(p) = \{p\}$ if t does not consume any variable in p.

The following lemma describes a single step of the top-down walk through a tree interpretation.

Lemma 3. *Assume a tree interpretation $\mathcal{I}$, $w \in \Delta^{\mathcal{I}}$ and any set P of queries. Then $\mathcal{I}|_w \not\models \bigcup P$ iff there is a set P' such that:*

(i) *P' contains some non-empty $p' \in \mathsf{sub}^{t_{\mathcal{I}}(w)}(p)$ for each $p \in P$;*
(ii) *$\mathcal{I}|_{w'} \not\models \bigcup P'$ for each child w' of w in $\Delta^{\mathcal{I}}$.*

Proof. For the if direction, we show that if $\mathcal{I}|_w \models \bigcup P$, then there is no set P' satisfying (i) and (ii). If $\mathcal{I}|_w \models \bigcup P$, then there is a match π in $\mathcal{I}|_w$ for some $p \in P$. We show that then, for each $p' \in \mathsf{sub}^{t_{\mathcal{I}}(w)}(p)$, there exists a child w' of w such that $\mathcal{I}|_{w'}$ admits a match for p'. This implies that there is no set P', since there is no possible choice of a subquery in $\mathsf{sub}^{t_{\mathcal{I}}(w)}(p)$ to be included.

Let π be a match for p in $\mathcal{I}|_w$, and let $\mathsf{sub}^{\pi(w)}(p)$ denote the set of all proper subqueries of the query $p^{\pi(w)}$ that results from p by dropping each atom involving a variable x with $\pi(x) = w$. By definition of a match, each $x \in \mathsf{Var}(p)$ with

$\pi(x) = w$ is consumed by $t_{\mathcal{I}}(w)$. This implies that all atoms removed from p to obtain $p^{\pi(w)}$ are also removed to obtain $p^{t_{\mathcal{I}}(w)}$, and thus each $p' \in \mathsf{sub}^{t_{\mathcal{I}}(w)}(p)$ is contained in some $p'' \in \mathsf{sub}^{\pi(w)}(p)$. Since π is a match for p, each $p'' \in \mathsf{sub}^{\pi(w)}(p)$ has a match in $\mathcal{I}|_{w'}$ for some child w' of w (in particular, π restricted to the domain of $\mathcal{I}|_{w'}$ is such a match), and so does each $p' \subseteq p''$. This shows that, for each $p' \in \mathsf{sub}^{t_{\mathcal{I}}(w)}(p)$, there exists a child w' of w such that $\mathcal{I}|_{w'} \models p'$.

For the other direction we show that if there does not exist a set P' as above, then $\mathcal{I}|_w \models \bigcup P$. Assume that there is no P' satisfying (i) and (ii). Then we can select some $p \in P$ such that for each non-empty $p' \in \mathsf{sub}^{t_{\mathcal{I}}(w)}(p)$, there is a child w' of w with $\mathcal{I}|_{w'} \models p'$, and we can select a match $\pi_{p'}$ in $\mathcal{I}|_{w'}$ for each p'. Observe that each $x \in \mathsf{Var}(p)$ that is not consumed by $t_{\mathcal{I}}(w)$ occurs in some p' and is in the scope of some $\pi_{p'}$. It can be easily verified that a match π for p can be composed by taking the union of all π'_p, and setting $\pi(x) = w$ for all remaining variables x. This shows $\mathcal{I}|_w \models p$ and $\mathcal{I}|_w \models \bigcup P$. ❏

We can now formally define the notion of a marking, which describes a top-down walk through a whole tree interepretation.

Definition 1. *Let $\mathcal{I}$ be a tree interpretation. A Q-marking for $\mathcal{I}$ is a mapping $\mu : \Delta^{\mathcal{I}} \to 2^{\mathsf{sub}(q)}$ such that:*

1. *$\mu(\mathsf{root}(\mathcal{I})) = Q$,*
2. *for each $w \in \Delta^{\mathcal{I}}$ and each pair $w \cdot i, w \cdot j \in \Delta^{\mathcal{I}}$, $\mu(w \cdot i) = \mu(w \cdot j)$,*
3. *for each $w \cdot i \in \Delta^{\mathcal{I}}$, $\mu(w \cdot i)$ is a set containing a non-empty $p' \in \mathsf{sub}^{t_{\mathcal{I}}(w)}(p)$ for each $p \in \mu(w)$.*

Using Lemma 3, we can characterize query non-entailment as follows:

Lemma 4. *There is a Q-marking for a tree interpretation $\mathcal{I}$ iff $\mathcal{I} \not\models \bigcup Q$.*

Proof. For the if direction, assume $\mathcal{I} \not\models \bigcup Q$. We define a Q-marking μ for $\mathcal{I}$ inductively:

- $\mu(\mathsf{root}(\mathcal{I})) = Q$,
- $\mu(w \cdot c) = \mu(w)'$ for all $w \cdot c \in \Delta^{\mathcal{I}}$, where $\mu(w)'$ is a $\subseteq$-minimal set of subqueries satisfying conditions (i) and (ii) of Lemma 3 (where we take $P = \mu(w)$ and $P' = \mu(w)'$).

Note that a suitable set $\mu(\mathsf{root}(\mathcal{I}))'$ exists for the children of the root because $\mathcal{I} \not\models \bigcup Q$. Then at each step $w \cdot c$, condition (ii) in Lemma 3 ensures that $\mathcal{I}|_{w \cdot c} \not\models \bigcup \mu(w \cdot c)$. Applying the lemma again we ensure the existence of a suitable set $\mu(w \cdot c)'$ for the children of $w \cdot c$. It is trivial to verify that μ satisfies the conditions in the definition of Q-marking (in particular, for condition 3 we use condition (i) in Lemma 3).

The other direction follows easily from the first condition in Definition 1, which ensures that the root is always marked with Q, and the following claim:

(∗) If μ is a Q-marking for $\mathcal{I}$, then $\mathcal{I}|_w \not\models \bigcup \mu(w)$ for every $w \in \Delta^{\mathcal{I}}$.

To show $(*)$, we assume for a contradiction that μ is a Q-marking and that $\mathcal{I}|_w \models \bigcup \mu(w)$ for some $w \in \Delta^{\mathcal{I}}$. That is, $\mathcal{I}|_w \models p$ for some $p \in \mu(w)$. Among all such pairs (w, p), we select one with minimal $|\mathsf{Var}(p)|$, i.e., such that $|\mathsf{Var}(p)| \leq |\mathsf{Var}(p')|$ for every $w' \in \Delta^{\mathcal{I}}$ and every $p' \in \mu(w')$ such that $\mathcal{I}|_{w'} \models p'$. In the case where $t_{\mathcal{I}}(w)$ consumes no variable in p, we have that for every child w' of w, $\mu(w) = \mu(w')$ and $\mathcal{I}|_w \models p$ iff $\mathcal{I}|_{w'} \models p$. We can iteratively apply this argument to choose a $w^* \in \Delta^{\mathcal{I}|_w}$ (either w itself or a first descendant where some variable is consumed) such that $t_{\mathcal{I}}(w^*)$ consumes some $x \in \mathsf{Var}(p)$, $\mathcal{I}|_{w^*} \models p$, and $\mu(w^*) = \mu(w)$. The fact that $t_{\mathcal{I}}(w^*)$ consumes some $x \in \mathsf{Var}(p)$ ensures $|\mathsf{Var}(p')| < |\mathsf{Var}(p)|$ for every $p' \in \mathsf{sub}^{t_{\mathcal{I}}(w^*)}(p)$. Since μ is a Q-marking for $\mathcal{I}$ and $p \in \mu(w^*)$, by conditions 2 and 3 in Definition 1, there must be some non-empty $p' \in \mathsf{sub}^{t_{\mathcal{I}}(w^*)}(p)$ such that $p' \in \mu(w')$ for all children w' of w^*. We know from Lemma 3 that $\mathcal{I}|_{w^*} \models \{p\}$ implies that $\mathcal{I}|_{w'} \models \{p'\}$ for some child w' of w^*. But as $|\mathsf{Var}(p')| < |\mathsf{Var}(p)|$, this is a contradiction.   ❑

We have shown that UCQ non-entailment reduces to deciding the existence of a marking. The following lemma will help us to show that the latter problem can be decided in NExpTime. It shows that, even though there can be exponentially many queries in Q, the query set changes only a few times on each path of a marked interpretation. More precisely:

Lemma 5. *If $\mathcal{I} \not\models \bigcup Q$, then $\mathcal{I}$ admits a Q-marking μ with $\mathsf{max}(\Delta^{\mathcal{I}}, \mu) \leq |\mathsf{Var}(q)|^2 + 1$.*

Proof. Let μ be the Q-marking defined in the proof of Lemma 4. We consider an arbitrary path $w_1, w_2, \ldots$ in $\mathcal{I}$, and show that $l = |\{\mu(w_1), \mu(w_2), \ldots\}| \leq |\mathsf{Var}(q)|^2 + 1$. We let $J = \{i \mid \mu(w_i) \neq \mu(w_{i+1})\}$. We will show that $|J| \leq |q|^2$. The desired bound will follow from this and the fact that $l \leq |J| + 1$. Let $t_i = t_{\mathcal{I}}(w_i)$ for all $i \geq 0$. We say a query q' is *i-matched* if q' has a match in $\mathcal{I}_i$ but not on $\mathcal{I}_{i-1}$, where $\mathcal{I}_k$ is defined by setting (i) $\Delta^{\mathcal{I}_k} = \{(1, t_1), \ldots, (k, t_k)\}$; (ii) $r^{\mathcal{I}_k} = \{((i, t_i), (j, t_j)) \mid j > i\}$; (iii) $A^{\mathcal{I}_k} = \{(i, t_i) \mid A \in t_i\}$ for all $A \in \mathsf{N_C}$. Note that, for any query q', there is at most one index i such that q' is i-matched. For each pair $x, y \in \mathsf{Var}(q)$, let $q|^{x,y}$ be the query that is obtained by restricting $q|_{\{x\}\downarrow}$ to the variable y and the variables that reach y in the graph G_q. Let $X = \{q|^{x,y} \mid x, y \in \mathsf{Var}(q)\}$. Note that $|X| \leq |\mathsf{Var}(q)|^2$. We now show that for each $i \in J$, there exists some $q' \in X$ such that q' is i-matched. Since there is at most one i for each q', this implies $|J| \leq |X| \leq |q|^2$ and the bound follows.

Consider an arbitrary $i \in J$. Then $\mu(w_i) \neq \mu(w_{i+1})$ implies that for some $p' \in \mu(w_i)$, $\mu(w_{i+1})$ contains some $p'' \neq p'$ from $\mathsf{sub}^{t_{\mathcal{I}}(w)}(p')$, and some $x \in \mathsf{Var}(p')$ is consumed by $t_{\mathcal{I}}(w_i)$. By definition, the query p' is a proper subquery of some $p \in Q$. Observe that, if we restrict our attention to p and its subqueries, the marking μ 'moves' to a strictly smaller subquery at every type that consumes some variable. Let M be the set of source variables in the query graph G_p of this p, i.e. $M = \{y \in \mathsf{Var}(p) \mid \{y' \mid r(y', y) \in p\} = \emptyset\}$. It is not hard to see that, if $x \in \mathsf{Var}(p')$ is consumed by $t_{\mathcal{I}}(w_i)$, each $q|^{y,x}$ with $y \in M$ has a match in $\mathcal{I}_i$. To see that there exists at least one $y \in M$ such that $q|^{y,x}$ is i-matched, assume towards a contradiction that there is some $j < i$ such that each $q|^{y,x}$ has a match

in $\mathcal{I}_j$, and take the smallest such j. Then all variables that reach x in G_q are consumed by some type on the path to w_j, and w_j is marked with some $p'' \subseteq p$ where $\{y \mid r(y,x) \in p''\} = \emptyset$. As x is consumed by $t_{\mathcal{I}}(w_j)$, then the markings of all descendants of w_j contain some subquery of p'' where x does not occur. This contradicts the fact that $p' \in \mu(w_i)$ and $x \in \mathsf{Var}(p')$. ❏

As a direct consequence of Lemmas 2, 4 and 5, we obtain the following characterization of counter-models; this is the basis of our UCQ entailment algorithm.

Theorem 1. $\mathcal{K}_a \not\models \bigcup Q$ *iff there is a tree interpretation $\mathcal{I}$ such that:*

(A) $\mathcal{I}$ is a model of $\mathcal{K}_a$ with $\mathsf{max}(\Delta^{\mathcal{I}}, t_{\mathcal{I}}) \leq |\mathsf{cl}(\mathcal{K})|$;
(B) $\mathcal{I}$ admits some Q-marking μ and $\mathsf{max}(\Delta^{\mathcal{I}}, \mu) \leq |\mathsf{Var}(q)|^2 + 1$.

By removing domain elements not needed to satisfy existential restrictions from $\mathsf{cl}(\mathcal{K})$, it is standard to show that we can assume the interpretation $\mathcal{I}$ from Theorem 1 to have branching degree at most $|\mathsf{cl}(\mathcal{K})|$.

5 Witnesses of Counter-models

By Theorem 1, $\mathcal{K}_a \not\models \bigcup Q$ can be decided by checking whether there is a tree interpretation that satisfies conditions (A) and (B). As we show next, the existence of such an interpretation $\mathcal{I}$ is guaranteed if we can find an initial part of $\mathcal{I}$ whose depth is bounded by $d_{\mathcal{K},q} := |\mathsf{cl}(\mathcal{K})| \times (|\mathsf{Var}(q)|^2 + 1)$. Since the branching degree of $\mathcal{I}$ is linear in the size of $\mathcal{K}$, this initial part is of at most exponential size. A nondeterministic exponential time procedure for checking $\mathcal{K}_a \not\models \bigcup Q$ is then almost immediate. We represent initial parts of countermodels as follows.

Definition 2. *A witness for "$\mathcal{K}_a \not\models \bigcup Q$" is a node-labeled tree $W = (T, \tau, \rho)$ where $\tau : T \to \mathsf{tp}(\mathcal{K})$ and $\rho : T \to 2^{\mathsf{sub}(q)}$, such that:*

1. *The branching degree of T is bounded by $|\mathsf{cl}(\mathcal{K})|$.*
2. *For each $w \in T$, $|w| \leq d_{\mathcal{K},q}$.*
3. *$\mathsf{max}(T, \tau) \leq |\mathsf{cl}(\mathcal{K})|$ and $\mathsf{max}(T, \rho) \leq |\mathsf{Var}(q)|^2 + 1$;*
4. *$\{C \mid C(a) \in \mathcal{A}'\} \subseteq \tau(e)$ and $\rho(e) = Q$ for the root e of T.*
5. *For all $w \in T$ with $|w| < d_{\mathcal{K},q}$ and $\exists r.C \in \tau(w)$, there is a child w' of w with $C \in \tau(w')$.*
6. *For each $w \in T$ and each child w' of w, $\neg\exists r.D \in \tau(w)$ implies $\{\neg D, \neg\exists r.D\} \subseteq \tau(w')$.*
7. *For each pair w_1, w_2 of children of w, $\rho(w_1) = \rho(w_2)$ is a set containing some nonempty $p' \in \mathsf{sub}^t(p)$ for each $p \in \rho(w)$.*

An initial part of a tree interpretation represented by a witness can be unravelled into a tree interpretation that satisfies (A) and (B) of Theorem 1, thus witnessing $\mathcal{K}_a \not\models \bigcup Q$.

Theorem 2. $\mathcal{K}_a \not\models \bigcup Q$ *iff there exists a witness W for "$\mathcal{K}_a \not\models \bigcup Q$".*

Proof. For the 'only if' direction, by Theorem 1 there exists a tree-model $\mathcal{I}$ of $\mathcal{K}_a$ and a Q-marking μ for $\mathcal{I}$ such that $\max(\Delta^\mathcal{I}, t_\mathcal{I}) \leq |\mathsf{cl}(\mathcal{K})|$, $\max(\Delta^\mathcal{I}, \mu) \leq |\mathsf{Var}(q)|^2 + 1$, and the branching degree of $\mathcal{I}$ is at most $|\mathsf{cl}(\mathcal{K})|$. We can obtain a witness by restricting $\mathcal{I}$ and μ to the first $d_{\mathcal{K},q}$ levels. More precisely, $W = (T, \tau, \rho)$ is obtained by setting:

- $T = \{w \in \Delta^\mathcal{I} \mid |w| \leq d_{\mathcal{K},q}\}$;
- $\tau(w) = t_\mathcal{I}(w)$ and $\rho(w) = \mu(w)$ for all $w \in T$.

For the other direction, observe that a witness $W = (T, \tau, \rho)$ is almost a Q-marked model of $\mathcal{K}_a$, except a node $w \in T$ with $|w| = d_{\mathcal{K},q}$ may not have the children it needs to satisfy the existential restrictions. However, since the path from the root to w has $d_{\mathcal{K},q} + 1$ nodes and due to (3) in Definition 2, there exists a pair of nodes on this path that share the same type and query set. This allows us to obtain a tree-model and a Q-marking by unraveling W as follows.

For each node $w \in T$, let $s(w)$ be the shortest prefix of w such that $\tau(s(w)) = \tau(w)$ and $\rho(s(w)) = \rho(w)$. Let $D \subseteq T^*$ be the smallest set of such that:

- the root of T belongs to D, and
- if $w_0 \cdots w_n \in D$, then $w_0 \cdots w_n w \in D$ for all children w of $s(w_n)$.

Consider the following interpretation $\mathcal{I}$ and marking μ:

- $\Delta^\mathcal{I} = D$;
- $A^\mathcal{I} = \{w_0 \cdots w_n \in \Delta^\mathcal{I} \mid A \in \tau(v_n)\}$ for all concept names A;
- $r^\mathcal{I} = \{(w_0 \cdots w_{n-1}, w_0 \cdots w_n) \mid w_0 \cdots w_n \in \Delta^\mathcal{I}\}$;
- $\mu(w_0 \cdots w_n) = \rho(w_n)$ for all $w_0 \cdots w_n \in \Delta^\mathcal{I}$.

It is easy to check that μ is a Q-marking for $\mathcal{I}$. To see that $\mathcal{I}$ is model of $\mathcal{K}_a$, observe that for each node $w \in T$ with $|w| = d_{\mathcal{K},q}$, there is a proper prefix w' of w such that $s(w') \neq w'$. This means that such a w will never be added to a path in $\Delta^\mathcal{I}$. This implies that each $w_0 \cdots w_n \in \Delta^\mathcal{I}$ has $|w_n| < d_{\mathcal{K},q}$ and hence satisfies all the existential restrictions. ❏

We can check for the existence of a witness by nondeterministically guessing an (exponential size) candidate structure $W = (T, \tau, \rho)$ and then verifying conditions (1-7) in Definition 2. The latter is feasible in time exponential in $|\mathcal{K}|$ and $|q|$. Hence, $\mathcal{K}_a \not\models \bigcup Q$ can be decided nondeterministically in time exponential in $|\mathcal{K}|$ and $|q|$.

For the overall algorithm, observe that each completion $\mathcal{A}'$ of $\mathcal{A}$ is of size polynomial in $|\mathcal{K}|$ and $|q|$, while the size of $\alpha(a)$ is at most exponential in $|\mathcal{K}|$ and $|q|$ for each $a \in \mathsf{Ind}(\mathcal{A})$. Thus, using Lemma 1, checking $\mathcal{K} \not\models q$ is trivially in NExpTime provided that checking $\mathcal{K}_a \not\models \bigcup \alpha(a)$ is NExpTime. By combining this with the matching lower bound in [5], we get:

Theorem 3. *CQ entailment over $\mathcal{S}$ KBs with one transitive role, and no other roles, is* co-NExpTime-*complete.*

6 Conclusion

We believe that Theorem 3 can be extended to the case where there is an arbitrary number of roles, both transitive and unrestricted ones. This requires the combination of the techniques presented in this paper with the ones developed in [5]. In particular, different roles used in a query $p \in Q$ induce a partitioning of p into different "clusters", and each cluster can be treated in a similar way as an entire, unpartitioned query $p \in Q$ in the current paper. Since the technical details, which we are currently working out, can be expected to become somewhat cumbersome, we believe that it is instructive to first concentrate on the case of a single transitive role as we have done in this paper.

It is interesting to note that the techniques from this paper can be used to reprove in a transparent way the ExpTime upper bound for CQ answering over $\mathcal{S}$ knowledge bases that contain only a single concept assertion and no role assertions from [5]—restricted to a single transitive role, of course. In the case of such ABoxes, we do not need the machinery from Sections 3 and 5, nor the (subtle to prove) Lemma 2. The essential technique is Q-markings, which can be simplified to maps from $\Delta^{\mathcal{I}}$ to $\mathsf{sub}(q)$ instead of to $2^{\mathsf{sub}(q)}$ because Q is a singleton that consists only of the input query. By Lemma 4, it suffices to check for the existence of a tree-shaped interpretation $\mathcal{I}$ along with a Q-marking for $\mathcal{I}$. This can be done by a standard type-elimination procedure.

References

1. Meghyn Bienvenu, Thomas Eiter, Magdalena Ortiz, and Mantas Šimkus. Query answering in the description logic $\mathcal{S}$. Technical Report INFSYS RR-1843-10-01 (available at `http://www.kr.tuwien.ac.at/research/reports/`), 2010.
2. Craig Boutilier, editor. *IJCAI 2009, Proceedings of the 21st International Joint Conference on Artificial Intelligence, Pasadena, California, USA, July 11-17, 2009*, 2009.
3. Diego Calvanese, Giuseppe De Giacomo, Domenico Lembo, Maurizio Lenzerini, and Riccardo Rosati. Tractable reasoning and efficient query answering in description logics: The DL-Lite family. *J. of Automated Reasoning*, 39(3):385–429, 2007.
4. Diego Calvanese, Giuseppe De Giacomo, and Maurizio 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.
5. Thomas Eiter, Carsten Lutz, Magdalena Ortiz, and Mantas Šimkus. Query answering in description logics with transitive roles. In Boutilier [2], pages 759–764.
6. Birte Glimm, Carsten Lutz, Ian Horrocks, and Ulrike Sattler. Answering conjunctive queries in the $\mathcal{SHIQ}$ description logic. *Journal of Artificial Intelligence Research*, 31:150–197, 2008.
7. Carsten Lutz. The complexity of conjunctive query answering in expressive description logics. In Alessandro Armando, Peter Baumgartner, and Gilles Dowek, editors, *Proceedings of the 4th International Joint Conference on Automated Reasoning (IJCAR2008)*, number 5195 in LNAI, pages 179–193. Springer, 2008.

8. Carsten Lutz, David Toman, and Frank Wolter. Conjunctive query answering in the description logic EL using a relational database system. In Boutilier [2], pages 2070–2075.

9. Magdalena Ortiz, Diego Calvanese, and Thomas Eiter. Data complexity of query answering in expressive description logics via tableaux. *J. of Automated Reasoning*, 41(1):61–98, 2008. `doi:10.1007/s10817-008-9102-9`. Preliminary version available as Tech.Rep. INFSYS RR-1843-07-07, Institute of Information Systems, TU Vienna, Nov. 2007.

10. Ulrike Sattler. Description logics for the representation of aggregated objects. In W. Horn, editor, *Proceedings of the Fourteenth European Conference on Artificial Intelligence (ECAI'00)*. IOS Press, Amsterdam, 2000.

11. Sergio Tessaris. *Questions and Answers: Reasoning and Querying in Description Logic*. PhD thesis, University of Manchester, Department of Computer Science, April 2001.

Optimizing Algebraic Tableau Reasoning for $\mathcal{SHOQ}$: First Experimental Results

Jocelyne Faddoul and Volker Haarslev

Concordia University, Montreal, Canada
{j_faddou,haarslev}@cse.concordia.ca

Abstract. In this paper we outline an algebraic tableau algorithm for the DL $\mathcal{SHOQ}$, which supports more informed reasoning due to the use of semantic partitioning and integer programming. We introduce novel and adapt known optimization techniques and show their effectiveness on the basis of a prototype reasoner implementing the optimization techniques for the algebraic approach. Our first set of benchmarks clearly indicates the effectiveness of our approach and a comparison with the DL reasoners Pellet and HermiT demonstrates a runtime improvement of several orders of magnitude.

1 Motivation

Nominals play an important role in Description Logics (DLs) as they allow one to express the notion of identity and enumeration; nominals must be interpreted as singleton sets. An example for the use of nominals in $\mathcal{SHOQ}$ would be *Eye_Color* $\equiv$ *Green* $\sqcup$ *Blue* $\sqcup$ *Brown* $\sqcup$ *Black* $\sqcup$ *Hazel* where each color is represented as a nominal. The cardinality of *Eye_Color* is restricted to have at most 5 instances, i.e., the above-mentioned nominals. Qualified cardinality restrictions (QCRs) allow one to specify lower ($\geq n R.C$) and upper ($\leq n R.C$) bounds on the number of elements related via a certain role with additionally specifying qualities on the related elements. Due to the interaction between nominals and QCRs the $\mathcal{SHOQ}$ concept $\geq 6\, has\_color.Eye\_Color$ is unsatisfiable. Each *nominal* must be interpreted as a set with the cardinality 1 (and thus can be used to enumerate domain elements), whereas an atomic concept is interpreted as a set with an unbounded cardinality. Moreover, the quasi-tree model property, which has always been advantageous for DL tableau methods, does not hold for $\mathcal{SHOQ}$.

Resolution-based reasoning procedures were proposed in [8] and were proven to be weak in dealing with QCRs containing large numbers. Hypertableaux [9] were recently studied to minimize non-determinism in DL reasoning with no special treatment for QCRs. These approaches and standard tableau techniques suffer from the low level of information about the cardinalities of concepts and the number of role successors implied by nominals and QCRs (e.g., see the example above) because these algorithms treat these cardinalities in a blind and uninformed way.

Our early work on performance improvements for reasoning with QCRs for the DL $\mathcal{SHQ}$ was based on a so-called signature calculus [5] and, alternatively, on algebraic reasoning [6] (not applicable to Aboxes). Our algebraic approach represents the knowledge about implied cardinalities as linear inequations. The advantages of such

an approach have been demonstrated in [4] where an Abox calculus combining tableau and algebraic reasoning for $\mathcal{SHQ}$ is presented that dramatically improves the runtime performance for reasoning with QCRs. This paper extends this line of research [1, 3, 4] to $\mathcal{SHOQ}$. This calculus [2] is by no means a simple extension because (i) the quasi-tree model property is lost, (ii) QCRs cannot be dealt with locally anymore, and (iii) possible interactions between QCRs and nominals need to be considered globally.

2   The Description Logic $\mathcal{SHOQ}$

Let N_C, N_R be non-empty and disjoint sets of concept and role names respectively. Let $N_o \subseteq N_C$ be the set of nominals, and $N_{R^+} \subseteq N_R$ the set of transitive role names. An RBox $\mathcal{R}$ is a finite set of *role inclusion axioms* (RIAs) of the form $R \sqsubseteq S$, where R, S are role names in N_R. With $\sqsubseteq_*$ we denote the reflexive transitive closure of $\sqsubseteq$ on $\mathcal{R}$. A role name R is called *simple* if it is neither transitive nor has a transitive subrole. A TBox $\mathcal{T}$ is a finite set of *general concept inclusion axioms* (GCIs) of the form $C \sqsubseteq D$, where C, D are concepts, and $C \equiv D$ abbreviates $\{C \sqsubseteq D,\ D \sqsubseteq C\}$. The set of $\mathcal{SHOQ}$ concepts is the smallest set such that: (i) $A \in N_C$ is a concept, and (ii) if C, D are concepts, $R \in N_R$, and $S \in N_R$ is a simple role then $\neg C$, $(C \sqcup D)$, $(C \sqcap D)$, $(\exists R.C)$, $(\forall R.C)$, $(\geq nS.C)$, $(\leq nS.C)$ with $n \in \mathbb{N}$ are also concepts. We use $\top$ $(\bot)$ as an abbreviation for $A \sqcup \neg A$ $(A \sqcap \neg A)$ and $\geq nS$ $(\leq nS)$ for $\geq nS.\top$ $(\leq nS.\top)$. We do not consider descriptions of the form $\exists R.C$ as they can be converted to $\geq 1 R.C$, without imposing the simple role restriction.

We assume a standard Tarski-style interpretation $\mathcal{I} = (\Delta^\mathcal{I}, \cdot^\mathcal{I})$ such that $A^\mathcal{I} \subseteq \Delta^\mathcal{I}$ for $A \in N_C$, $R^\mathcal{I} \subseteq \Delta^\mathcal{I} \times \Delta^\mathcal{I}$ for $R \in N_R$. Using # to denote the cardinality of a set, we define the set of *R-fillers* for a given role name R and an individual s as $FIL(R, s) = \{t \in \Delta^\mathcal{I} \mid \langle s, t \rangle \in R^\mathcal{I}\}$ and the set of all *R-fillers* as: $FIL(R) = \bigcup_{s \in \Delta^\mathcal{I}} FIL(R, s)$. The semantics of $\mathcal{SHOQ}$ concept descriptions is such that $(C \sqcap D)^\mathcal{I} = C^\mathcal{I} \cap D^\mathcal{I}$, $(C \sqcup D)^\mathcal{I} = C^\mathcal{I} \cup D^\mathcal{I}$, $(\neg C)^\mathcal{I} = \Delta^\mathcal{I} \setminus C^\mathcal{I}$, $\#o^\mathcal{I} = 1$ for all $o \in N_o$, $(\forall R.C)^\mathcal{I} = \{s \in \Delta^\mathcal{I} \mid \langle s, t \rangle \in R^\mathcal{I} \Rightarrow t \in C^\mathcal{I}\}$, $(\exists R.C)^\mathcal{I} = \{s \in \Delta^\mathcal{I} \mid \exists t : \langle s, t \rangle \in R^\mathcal{I} \wedge t \in C^\mathcal{I}\}$, $(\geq n S.C)^\mathcal{I} = \{s \in \Delta^\mathcal{I} \mid \#(FIL(S, s) \cap C^\mathcal{I}) \geq n\}$, $(\leq n S.C)^\mathcal{I} = \{s \in \Delta^\mathcal{I} \mid \#(FIL(S, s) \cap C^\mathcal{I}) \leq n\}$.

Let KB$(\mathcal{T}, \mathcal{R})$ denote a $\mathcal{SHOQ}$ knowledge base consisting of a TBox $\mathcal{T}$ and an RBox $\mathcal{R}$. The KB$(\mathcal{T}, \mathcal{R})$ is said to be consistent iff there exists an interpretation $\mathcal{I}$ satisfying $C^\mathcal{I} \subseteq D^\mathcal{I}$ for each $C \sqsubseteq D \in \mathcal{T}$ and $R^\mathcal{I} \subseteq S^\mathcal{I}$ for each $R \sqsubseteq S \in \mathcal{R}$. In this case, $\mathcal{I}$ is said to be a model of KB$(\mathcal{T}, \mathcal{R})$. A concept C is said to be satisfiable w.r.t. KB$(\mathcal{T}, \mathcal{R})$ iff $C^\mathcal{I} \neq \emptyset$. $\mathcal{I}$ is called a model of C w.r.t. $\mathcal{R}$ and $\mathcal{T}$. A $\mathcal{SHOQ}$ ABox $\mathcal{A}$ is a finite set of concept membership assertions of the form $a : C$ or role membership assertions of the form $(a, b) : R$ with a, b two individual names. An Abox $\mathcal{A}$ is said to be consistent w.r.t. KB$(\mathcal{T}, \mathcal{R})$ if there exists a model $\mathcal{I}$ of $\mathcal{T}$ and $\mathcal{R}$ such that $a^\mathcal{I} \in C^\mathcal{I}$ is satisfied for each $a : C$ in $\mathcal{A}$ and $(a^\mathcal{I}, b^\mathcal{I}) \in R^\mathcal{I}$ for each $(a, b) : R$ in $\mathcal{A}$. Using nominals, concept satisfiability and ABox consistency can be reduced to KB consistency. Hence, without loss of generality we restrict our attention to KB consistency in the following.

We assume all concepts to be in their *negation normal form* (NNF). We use $\dot\neg C$ to denote the NNF of $\neg C$ and $nnf(C)$ to denote the NNF of C. When checking KB$(\mathcal{T}, \mathcal{R})$ consistency, the concept axioms in $\mathcal{T}$ can be reduced to a single axiom $\top \sqsubseteq C_\mathcal{T}$ such that $C_\mathcal{T}$ abbreviates $\bigsqcap_{C \sqsubseteq D \in \mathcal{T}} nnf(\neg C \sqcup D)$. A TBox consistency test can be checked by

testing the consistency of $o \sqsubseteq C_{\mathcal{T}}$ with $o \in N_o$ new in $\mathcal{T}$, which means that at least $o^{\mathcal{I}} \in C_{\mathcal{T}}{}^{\mathcal{I}}$ and $C_{\mathcal{T}}{}^{\mathcal{I}} \neq \emptyset$. Moreover, since $\top^{\mathcal{I}} = \varDelta^{\mathcal{I}}$ then every domain element must also satisfy $C_{\mathcal{T}}$ (every domain element is a member of $C_{\mathcal{T}}$).

3   Algebraic Tableau for $\mathcal{SHOQ}$

Given KB $(\mathcal{T}, \mathcal{R})$, such that we have $\top \sqsubseteq C_{\mathcal{T}}$, we apply a rewriting algorithm (see [2] for details) to $C_{\mathcal{T}}$ which returns $C'_{\mathcal{T}}$ and extends $\mathcal{R}$ with role inclusion axioms. This rewriting transforms all QCRs of the form $\geq nR.C$ or $\leq nR.C$, where C can be also equal to $\top$, into unqualified cardinality restrictions of the form $\geq nR'$ ($\leq nR'$) by using a new role-set difference operator ($\forall_{\backslash}$) and adding universal restrictions using newly introduced subroles ($R' \sqsubseteq R$). Roughly speaking, $\geq n R.C$ is transformed into $\geq n R' \sqcap \forall R'.C$ with adding $R' \sqsubseteq R$ to $\mathcal{R}$, and $\leq n R.C$ into $\leq n R' \sqcap \forall R'.C \sqcap \forall (R \backslash R').\dot{\neg}C$ with adding $R' \sqsubseteq R$ to $\mathcal{R}$. In both cases R' is always fresh in $\mathcal{R}$, and the transformation is satisfiability-preserving (see [2] for a proof and more details). The semantics of the role-set operator is defined such that $(\forall (R \backslash S).D)^{\mathcal{I}} = \{s \in \varDelta^{\mathcal{I}} \mid \langle s, t \rangle \in R^{\mathcal{I}} \wedge \langle s, t \rangle \notin S^{\mathcal{I}} \Rightarrow t \in D^{\mathcal{I}}\}$.

3.1   Partitioning domain elements

The key technique and major difference between algebraic and standard tableau reasoning for $\mathcal{SHOQ}$ is the *atomic decomposition technique* [10] which is used to compute a partitioning of domain elements into disjoint subsets allowing numerical restrictions implied by QCRs and nominals to be encoded into sets of inequations.

Let $H(R)$ denote the set of role names for all subroles of $R \in N_R$: $H(R) = \{R' \mid R' \sqsubseteq_* R\}$. For technical reasons we do not add R to $H(R)$ since R is a superrole for elements in $H(R)$ and R does not occur in number restrictions anymore after preprocessing. For every role $R' \in H(R)$, the set of R'-*fillers* forms a subset of the set of R-*fillers* ($FIL(R') \subseteq FIL(R)$). We define $\overline{R'}$ to be the complement of R' w.r.t. $H(R)$, the set of $\overline{R'}$-*fillers* is then defined as $\overline{R'}$-*fillers* $= (FIL(R) \setminus FIL(R'))$. Since we do not have $\geq nR$ or $\leq nR$ concept expressions using role complements, no role complement will be explicitly used. For ease of presentation, we do not list role complements.

Qualifications on Role fillers: The atomic decomposition must also consider when $FIL(R)$ intersects with the interpretation of a *qualifying concept*. A qualifying concept D is a concept used to impose a qualification, D, on the set of R-fillers for some role $R \in N_R$. Let $Q_C(R) = \{D \mid \forall S.D$ occurs in $C_{\mathcal{T}}$ with $R \sqsubseteq_* S \in \mathcal{R}\}$ be the set of qualifying concepts for $R \in N_R$. Since $D \in Q_C(R)$ could be a complex expression or a nominal, and for ease of presentation, we assign a unique qualification name q for each $D \in Q_C(R)$. Let Q_N be the set of all qualification names assigned, and $Q_C = \bigcup_{R \in N_R} Q_C(R)$ be the set of qualifying concepts in $C_{\mathcal{T}}$. We maintain a mapping between qualification names and their corresponding concept expressions using a bijection $\theta : Q_N \to Q_C$; in case a nominal $o \in N_o$ has been used as a qualifying concept expression then o is also used as the qualification name and $\theta(o) = o$. Let $Q_N(R)$ denote the set of qualification names for a role ($R \in N_R$) then $Q_N(R)$ is defined as $Q_N(R) = \{q \in Q_N \mid \theta(q) \in Q_C(R)\}$.

We define $Q_C^{\neg} = \{\dot{\neg} D \,|\, D \in Q_C\}$ as the set of negated qualifying concepts in their NNF. A mapping $\dot{\neg}_Q$ is maintained between Q_C and $Q_C^{\neg}$ such that given a qualifying concept $D \in Q_C$, $\dot{\neg}_Q(D) = \dot{\neg} D$ with $\dot{\neg} D \in Q_C^{\neg}$.

Interaction with Nominals: For each nominal $o \in N_o$, $o^{\mathcal{I}}$ can interact with *R-fillers* for some R in N_R such that $(o^{\mathcal{I}} \subseteq FIL(R))$. Also the same nominal o can interact with *R-fillers* and *S-fillers* for $R, S \in N_R$ such that R, S do not necessarily share subroles or superroles in $\mathcal{R}$. This means that *R-fillers* and *S-fillers* could interact with each other due to their common interaction with the same nominal o. These interactions lead to the following definitions.

Definition 1 (Decomposition Set). Given a role R we define the decomposition set for R-fillers as $\mathcal{D}_R = H(R) \cup Q_N(R) \cup N_o$. $\mathcal{D}_R$ is a decomposition set since each subset P of $\mathcal{D}_R$ defines a unique set of nominals, roles, and/or qualification names that admits an interpretation $P^{\mathcal{I}} = \bigcap_{o \in P \cap N_o} o^{\mathcal{I}} \cap \bigcap_{i \in N_o \setminus P} \neg i^{\mathcal{I}} \cap \bigcap_{R' \in P \cap H(R)} FIL(R') \cap \bigcap_{R'' \in (H(R) \setminus P)} FIL(\overline{R''}) \cap \bigcap_{p \in P \cap Q_N(\mathcal{R})} \theta(p)^{\mathcal{I}} \cap \bigcap_{q \in (Q_N(R) \setminus P)} (\dot{\neg}\theta(q))^{\mathcal{I}}$. For all sets $P, Q \subseteq \mathcal{D}_R$ with $P \neq Q$, it holds by definition that $P^{\mathcal{I}} \neq Q^{\mathcal{I}}$. This makes all $P^{\mathcal{I}}$ with $P \subseteq \mathcal{D}_R$ disjoint with one another and the set of all P with $P \subseteq \mathcal{D}_R$ defines a partitioning of $\mathcal{D}_R$.

Definition 2 (Global Partitioning). Let $\mathcal{DS} = (\bigcup_{R \in N_R} \mathcal{D}_R \cup N_o) \setminus \{\neg C \,|\, \{C, \neg C\} \subseteq Q_C\}$[1]. The set $\mathcal{P} = \{P \,|\, P \subseteq \mathcal{DS}\}$ defines a *global partitioning* of $\mathcal{DS}$ and $\mathcal{P}^{\mathcal{I}} = \Delta^{\mathcal{I}}$ because it includes all possible domain elements which correspond to a nominal and/or a role filler: $\mathcal{P}^{\mathcal{I}} = \bigcup_{P \subseteq \mathcal{DS}} P^{\mathcal{I}}$.

3.2 Encoding Numerical Restrictions into Inequations

Given $\mathcal{T}$ and a partitioning $\mathcal{P}$ for $\mathcal{DS}$, one can reduce the satisfiability of expressions of the form $(\geq nR)$ and $(\leq mR)$ and the satisfiability of the nominals semantics into inequation solving based on the following principles.

Mapping Cardinalities to Variables A variable name v is assigned for each partition name P such that v can be mapped to a non-negative integer value n using $\sigma : \mathcal{V} \to \mathbb{N}$ with $\sigma(v)$ denoting the cardinality of $P^{\mathcal{I}}$. Let $\mathcal{V}$ be the set of all variable names and $\alpha : \mathcal{V} \to \mathcal{P}$ be a one-to-one mapping between each partition name $P \in \mathcal{P}$ and a variable $v \in \mathcal{V}$ such that $\alpha(v) = P$, and if a non-negative integer n is assigned to v using σ then $\sigma(v) = n = \#P^{\mathcal{I}}$. Given $L \subseteq \mathcal{DS}$, let V_L denote the set of variable names mapped to partitions satisfying $L^{\mathcal{I}}$, V_L is defined as

$$V_L = \begin{pmatrix} \{v \in \mathcal{V} \,|\, p \in \alpha(v) \text{ for each } p \in (L \cap N_R)\} \cap \\ \{v \in \mathcal{V} \,|\, oq \in \alpha(v) \text{ for each } oq \in (L \cap (N_o \cup Q_N))\} \cap \\ \{v \in \mathcal{V} \,|\, oq \notin \alpha(v) \text{ for each } \neg oq \in L, oq \in (N_o \cup Q_N)\} \end{pmatrix}$$

Encoding Inequations Since the partitions in $\mathcal{P}$ are mutually disjoint the cardinality of a union of partitions is equal to the sum of the cardinalities of the partitions (e.g., if $P_1, P_2 \in \mathcal{P}$, then $\#(P_1 \cup P_1) = \#P_1 + \#P_2$) and one can encode a cardinality restriction on a partition's elements into an inequation using ξ such that $\xi(L, \geq, n) = \sum_{v \in V_L} \sigma(v) \geq n$, and $\xi(L, \leq, m) = \sum_{v \in V_L} \sigma(v) \leq m$ where $L \subseteq \mathcal{DS}$. With $\mathcal{SHOQ}$ we distinguish and encode the following cardinalities: (i) Concepts of the form $(\geq nR)$ and $(\leq mR)$ in the label of a node x express cardinality bounds n and m, respectively, on the set $FIL(R, x)$

---

[1] When C and $\dot{\neg} C$ are both used as qualifying concepts, we only include C in $\mathcal{DS}$.

for some $R \in N_R$. These bounds can be reduced into inequations using $\xi(L, \geq, n)$ and $\xi(L, \leq, m)$ for $L = \{R\}$ or $L = \{R, q\}$, if additionally, we have $\forall S.C$ such that $(R \sqsubseteq_* S)$ with $C \in DS$ and $\theta(q) = C$. (ii) Nominals represent singleton sets. This cardinality bound can be encoded into inequations using $\xi(\{o\}, \geq, 1)$ and $\xi(\{o\}, \leq, 1)$ for each nominal $o \in N_o$. When cardinalities (i) and (ii) are both encoded into inequations, the interaction between nominals and role fillers is handled while preserving the semantics of nominals.

Getting a Solution Given a set ξ of inequations, an integer solution defines the mapping σ for each variable v occurring in ξ to a non-negative integer n denoting the cardinality of the corresponding partition. For example, assuming $\sigma(v_a) = 1$ and $\alpha(v_a) = \{R_1, R_2\}$, this means that the corresponding partition $(\alpha(v_a))^I$ must have 1 element; $\#(FIL(R_1) \cap FIL(R_2)) = 1$. Additionally, by setting the objective function to minimize the sum of all variables, a minimum number of role fillers is ensured at each level. A solution σ then defines a distribution of individuals that is consistent with the numerical restrictions encoded in ξ.

3.3   Tableau Algorithm

The tableau algorithm described in this section relies on an inequation solver working together with tableau expansion rules to construct a representation of a tableau model using a *compressed completion graph*.

Definition 3. [Compressed Completion Graph] A (CCG) is a directed graph $G = (V, E, \mathcal{L}, \mathcal{L}_E, \mathcal{L}_P)$, where nodes represent domain elements and the edges between the nodes represent role relations. Each node $x \in V$ is labeled with three labels: $\mathcal{L}(x)$, $\mathcal{L}_E(x)$ and $\mathcal{L}_P(x)$, and each edge $\langle x, y \rangle \in E$ is labeled with a set, $\mathcal{L}(\langle x, y \rangle) \subseteq N_R$, of role names. $\mathcal{L}(x)$ denotes a set of concept expressions, $\mathcal{L}(x) \subseteq clos(\mathcal{T})$, that the domain element, i_x, represented by x must satisfy. $\mathcal{L}_P(x)$ denotes a non-atomic partition name (i.e., we consider the set $\mathcal{L}_P(x)$ as a name) and is used as a tag for x based on the partition that i_x belongs to. A partition name $\mathcal{L}_P(x) \subseteq \mathcal{DS}$ can include roles, nominals, or qualification names.

When a role $R \in N_R$ appears in $\mathcal{L}_P(x)$ this means that i_x belongs to the partition for *R-fillers* and can therefore be used as an *R-filler*. When a nominal $o \in N_o$ appears in $\mathcal{L}_P(x)$ this means that $i_x \in o^I$, and o is added to $\mathcal{L}(x)$ when x is created. On the other hand if a nominal $i \in N_o$ does not appear in $\mathcal{L}_P(x)$ this means that i_x satisfies the complement of i, $i_x \in (\neg i)^I$ and $(\neg i)$ is added to $\mathcal{L}(x)$ when x is created (see *fil*-Rule). When a qualification name $q \in Q_N$ appears in $\mathcal{L}_P(x)$ this means that i_x satisfies the qualifying concept mapped to q, $i_x \in \theta(q)^I$ and $\theta(q)$ is added to $\mathcal{L}(x)$ when x is created. As with the nominals case, if a qualification name $p \in Q_N$ does not appear in $\mathcal{L}_P(x)$ this means that i_x satisfies the complement of the qualifying concept mapped to p, $i_x \in \dot{\neg}(\theta(p))^I$ and $\dot{\neg}\theta(p)$ is added to $\mathcal{L}(x)$ when x is created (see *fil*-Rule). Using $\mathcal{L}_P(x)$ as a tagging allows for the re-use of nodes instead of creating new ones.

$\mathcal{L}_E(x)$ denotes a set ξ_x of inequations that must have a non-negative integer solution. The set ξ_x is the encoding of number restrictions and qualifications that must be satisfied for x. In order to make sure that numerical restrictions local for a node x are

satisfied while the global restrictions carried with nominals are not violated, the in-equation solver collects all inequations and variable assignment in $\mathcal{L}_E$ before returning a distribution. This makes sure that an initial distribution of nominals and/or role fillers is globally preserved while still satisfying the numerical restrictions (a distribution of role fillers) at each level.

Definition 4. [Proxy node] A proxy node is a representative for the elements of each partition. Proxy nodes can be used since partitions are disjoint and all elements within a partition P satisfy common restrictions (see [2] for proofs).

Let us assume that KB($\mathcal{T}, \mathcal{R}$) such that $\mathcal{T}$ has been preprocessed and rewritten into $C'_{\mathcal{T}}$. To check KB consistency, the algorithm starts with the completion graph $G = (\{r_0\}, \emptyset, \mathcal{L}, \mathcal{L}_E)$. With $\mathcal{L}_E(r_o) = \bigcup_{o \in N_o}\{\xi(o, \leq, 1), \xi(o, \geq, 1)\}$ which is an encoding of the nominal semantics into inequations. The node r_0 is artificial and is not considered as part of the tableau model, it is only used to process the numerical restrictions on nominals using the inequation solver which returns a distribution for them.

The distribution of nominals is processed by the ***fil*-Rule** which is used to generate individual nodes depending on the solution (σ) returned by the inequation solver. The *fil*-Rule rule is fired for every non-empty partition P using $\sigma(v)$. It generates one proxy node y as the representative for the m elements assigned to $P^{\mathcal{I}}$ by the inequation solver. In the case of nominals, m is always equal to 1. The node y is tagged with its partition name using $\alpha(v)$ in $\mathcal{L}_P(y)$. The set of inequations is accumulated in $\mathcal{L}_E(y)$. Nominals and qualifications satisfied by the partition elements are extracted from the partition name and added to $\mathcal{L}(y)$. $C'_{\mathcal{T}}$ is also added to $\mathcal{L}(y)$ to make sure that every node created by the *fil*-Rule also satisfies $C'_{\mathcal{T}}$.

After at least one nominal is created, G is expanded by applying the expansion rules given in Fig. 1 until no rules are applicable or a clash occurs. The $\sqcap$-Rule, $\sqcup$-Rule, $\forall$-Rule and the $\forall_+$-Rule are similar to the ones in [1, 7]. The $\forall_\backslash$-**Rule** is used to enforce the semantics of the role set difference operator $\forall_\backslash$ introduced at preprocessing by making sure that all *R-fillers* are labelled. The $\bowtie$-**Rule** encodes the numerical restrictions in the label $\mathcal{L}$ of a node x, for some role $R \in N_R$, into a set of inequations maintained in $\mathcal{L}_E(x)$. The inequation solver is always active and responsible for finding a non-negative integer solution σ or triggering a clash if no solution is possible. If the inequations added by this rule do not trigger a clash, then the encoded at-least/at-most restriction can be satisfied by a possible distribution of role fillers. We distinguish two cases.

Case (i): *R*-fillers of x must also satisfy a qualifying concept C due to a $\forall S.C$ restriction on a role S such that $R \sqsubseteq_* S$ and C is either a nominal or a qualifying concept such that $\theta^-(C)$ in $\mathcal{DS}$. Then the numerical restriction is encoded on partitions $P \in \mathcal{P}$ with $P^{\mathcal{I}} \subseteq (C^{\mathcal{I}} \cap FIL(R))$ which means $\{R, \theta^-(C)\} \subseteq P$.

Case (ii): There exist no qualified restrictions on *R*-fillers of x due to a $\forall$ restriction on a role S such that $R \sqsubseteq_* S$. In this case the numerical restriction is encoded on partitions $P \in \mathcal{P}$ with $P^{\mathcal{I}} \subseteq FIL(R)$ which means $\{R\} \subseteq P$.

ch-**Rule.** This rule checks for empty partitions while ensuring completeness of the algorithm. Given a set of inequations in the label $\mathcal{L}_E(x)$ of a node x and a variable v such that $\alpha(v) = P$ and $P \in \mathcal{P}$ we distinguish between two cases.

(i) $P^{\mathcal{I}}$ must be empty ($v \leq 0$); this happens when restrictions on elements of this partition trigger a clash because the signature of P cannot be satisfied. For instance,

⊓-Rule	**If** $C \sqcap D \in \mathcal{L}(x)$, and $\{C, D\} \nsubseteq \mathcal{L}(x)$
	Then set $\mathcal{L}(x) = \mathcal{L}(x) \cup \{C, D\}$.
⊔-Rule	**If** $C \sqcup D \in \mathcal{L}(x)$, and $\{C, D\} \cap \mathcal{L}(x) = \emptyset$
	Then set $\mathcal{L}(x) = \mathcal{L}(x) \cup \{E\}$ with $E \in \{C, D\}$.
∀-Rule	**If** $\forall R.C \in \mathcal{L}(x)$ and there exists y such that $\mathcal{L}(\langle x, y \rangle) \cap (H(R) \cup \{R\}) \neq \emptyset$, and $C \notin \mathcal{L}(y)$
	Then set $\mathcal{L}(y) = \mathcal{L}(y) \cup \{C\}$.
∀₊-Rule	**If** $\forall R.C \in \mathcal{L}(x)$ and there exists y such that $\mathcal{L}(\langle x, y \rangle) \cap (H(S) \cup \{S\}) \neq \emptyset$, $S \in N_{R^+}$ with $S \sqsubseteq_* R$, and $\forall S.C \notin \mathcal{L}(y)$
	Then set $\mathcal{L}(y) = \mathcal{L}(y) \cup \{\forall S.C\}$.
⋈-Rule	**If** $(\bowtie n R) \in \mathcal{L}(x)$ for $\bowtie \in \{\leq, \geq\}$,
	Then If $\forall S.C \in \mathcal{L}(x)$ with $R \sqsubseteq_* S$ and $\xi(\{R, \theta^-(C)\}, \bowtie, n) \notin \mathcal{L}_E(x)$
	Then set $\mathcal{L}_E(x) = \mathcal{L}_E(x) \cup \{\xi(\{R, \theta^-(C)\}, \bowtie, n)\}$.
	Else If $\xi(\{R\}, \bowtie, n) \notin \mathcal{L}_E(x)$
	Then set $\mathcal{L}_E(x) = \mathcal{L}_E(x) \cup \{\xi(\{R\}, \bowtie, n)\}$.
ch-Rule	**If** there exists v occurring in $\mathcal{L}_E(x)$ such that $\{v \geq 1, v \leq 0\} \cap \mathcal{L}_E(x) = \emptyset$
	Then set $\mathcal{L}_E(x) = \mathcal{L}_E(x) \cup \{V\}$, $V \in \{v \geq 1, v \leq 0\}$.
e-Rule	**If** $(\bowtie n R) \in \mathcal{L}(x)$, and there exists y such that $R \in \mathcal{L}_P(y)$ and $R \notin \mathcal{L}(\langle x, y \rangle)$
	Then If $\forall S.C \in \mathcal{L}(x)$ with $R \sqsubseteq_* S$ and $\theta^-(C) \in \mathcal{L}_P(y)$, OR $\forall S.C \notin \mathcal{L}(x)$ with $R \sqsubseteq_* S$
	Then set $\mathcal{L}(\langle x, y \rangle) = \mathcal{L}(\langle x, y \rangle) \cup \{R\}$, and
	If $\mathcal{L}_E(x) \nsubseteq \mathcal{L}_E(y)$ **Then** set $\mathcal{L}_E(y) = \mathcal{L}_E(y) \cup \mathcal{L}_E(x)$.
fil-Rule	**If** there exists v occurring in $\mathcal{L}_E(x)$ with $\sigma(v) = m$ and $m > 0$, and there exists no y with $\mathcal{L}_P(y) = \alpha(v)$
	Then 1. create a new node y, 2. set $\mathcal{L}_P(y) = \alpha(v)$, 3. set $\mathcal{L}_E(y) = \mathcal{L}_E(x)$, 4. set $\mathcal{L}(y) = \bigcup_{o \in (\alpha(v) \cap N_o)} o \cup \bigcup_{i \in (N_o \setminus \alpha(v))} \neg i \cup \bigcup_{q \in (Q_N \cap \alpha(v))} \theta(q) \cup \bigcup_{p \in (Q_N \setminus (Q_N \cap \alpha(v)))} \dot{\neg}_Q \theta(p) \cup \{C'_\mathcal{T}\}$
∀\-Rule	**If** $\forall (R \setminus S).C \in \mathcal{L}(x)$, and there exists y such that $\mathcal{L}(\langle x, y \rangle) \cap (H(R) \cup \{R\}) \neq \emptyset$, $\mathcal{L}(\langle x, y \rangle) \cap (H(S) \cup \{S\}) = \emptyset$, and $C \notin \mathcal{L}(y)$
	Then set $\mathcal{L}(y) = \mathcal{L}(y) \cup \{C\}$.

Fig. 1. Completion rules for $\mathcal{SHOQ}$ (in groups of decreasing priority from top to bottom)

if $\{\forall R_1.A, \forall R_2.\neg A\} \subseteq \mathcal{L}(x)$, $v_{R_1 R_2} \geq 1 \in \mathcal{L}_E(x)$ and there exists a node y with $\mathcal{L}_P(y) = \{R_1, R_2\}$ and $\{R_1, R_2\} \subseteq \mathcal{L}(\langle x, y \rangle)$, the qualifications on R_1 and R_2-fillers trigger a clash $\{A, \neg A\} \subseteq \mathcal{L}(y)$ and $v_{R_1 R_2} \leq 0$ is enforced.

(ii) $P^{\mathcal{I}}$ must have at least one element ($1 \leq m \leq \sigma(v)$); if $P^{\mathcal{I}}$ can have at least one element without causing any logical clash, this means that the signature of P is satisfiable and we can also have m elements in $P^{\mathcal{I}}$ without a clash.

***e*-Rule.** This rule creates the edges between the proxy nodes created by the *fil*-Rule. If $\geq n R \in \mathcal{L}(x)$ for some R, this means that x must be connected to a number r of R-fillers such that $n \leq r$. If $\leq m R \in \mathcal{L}(x)$ then x could be connected to a maximum number r' of R-fillers such that $r' \leq m$. If there exists a node y such that $R \in \mathcal{L}_P(y)$, this means that a distribution of R-fillers has been assigned by the inequation solver such that the numbers n and m are satisfied and y is a representative for a number p of R-fillers such that $r \leq p \leq r'$. We distinguish between two cases.

(i): R-fillers of x must also satisfy a qualifying concept C due to a $\forall S.C$ restriction on a role S such that $R \sqsubseteq_* S$. In this case, if $\theta^-(C)$ is also in $\mathcal{L}_P(y)$ then the partition represented by y intersects with $C^{\mathcal{I}}$ and y is a member of C.

(ii): There exists no qualified restrictions on R-fillers. In this case there is no restriction on the partitions intersecting with R-fillers.

In both cases, an edge can safely be created between x and y such that $R \in \mathcal{L}(\langle x, y \rangle)$ and this edge is also a representative for the number p of edges between x and the p elements represented by y. If S is also in $\mathcal{L}_P(y)$ this means that the p R-fillers represented by y are also S-fillers and y is a representative for a partition $p \in \mathcal{P}$ such that $p^I \subseteq FIL(R) \cap FIL(S)$. Therefore y can be re-used to connect x or another node y having $\geq n'S$ or $\leq m'S$, $n' \leq n$ and $m' \geq m$, in their label.

Definition 5. [Strategy of Rule Application] Given a node x in the completion graph, the rules are triggered when applicable based on the following order (listed with decreasing priority) in order to ensure completeness of the algorithm (see [2] for details): 1. $\sqcap$-*Rule*, $\sqcup$-*Rule*, $\forall$-*Rule*, $\forall_+$-*Rule*, *ch-Rule*, $\bowtie$-*Rule*, *e-Rule*. These rules can be fired in arbitrary order. 2. *fil-Rule*. 3. $\forall_{\backslash}$-*Rule*.

Definition 6. [Clash] A node x in $(V \setminus \{r_0\})$ is said to contain a *clash* if: (i) $\{C, \neg C\} \subseteq \mathcal{L}(x)$, or (ii) a subset of inequations $\xi_x \subseteq \mathcal{L}_E(x)$ does not admit a non-negative integer solution, this case is decided by the inequation solver.

When no rules are applicable or there is a clash, a *completion graph* is said to be *complete*. When G is complete and clash free it means that a model exists for $KB(\mathcal{T}, \mathcal{R})$ satisfying the numerical and the logical restrictions; the algorithm returns that $KB(\mathcal{T}, \mathcal{R})$ is consistent, otherwise it returns that $KB(\mathcal{T}, \mathcal{R})$ is inconsistent.

4 Optimizing Algebraic Tableau Reasoning

The main goal for introducing algebraic reasoning to DL is to efficiently handle reasoning with QCRs and/or nominals. Although global partitioning of domain elements gives a worst-case double exponential algorithm (see [2] for proofs), one can exploit its high level of information to adapt well known and devise new optimization techniques for improving reasoning with nominals and QCRs. The atomic decomposition technique allows a more semantically structured model construction algorithm which exhibits a high level of information on cardinalities implied by QCRs and nominals.

The next two optimization techniques exploit simple interactions between so-called "told nominals" and QCRs to discard unnecessary partitions and impose some ordering on applying the ch-Rule for nominal variables.

Discarding Partitions This optimization aims at reducing the number of partitions and their variables. It does this at the preprocessing level by collecting and analyzing the following interactions between nominals and QCRs.

(i) We have $\geq nR.C$ with $C \equiv o_1 \sqcup \cdots \sqcup o_n$ or $C \sqsubseteq o_1 \sqcup \cdots \sqcup o_n$. For example, $\geq 1R.o$ is rewritten into $\geq 1R' \sqcap \forall R'.o$ and this means that the partition for R'-fillers must intersect with the partition for the nominal o and, therefore, the partitions for R'-fillers that do not intersect with o can be safely discarded when computing the global partitioning.

(ii) We have $\leq nR.C$ with $C \equiv o_1 \sqcup \cdots \sqcup o_n$ or $C \sqsubseteq o_1 \sqcup \cdots \sqcup o_n$. For example, $\leq 1R.o$ is rewritten into $\leq 1R' \sqcap \forall R'.o \sqcap \forall R \backslash R'.\neg o$ and similar to the case with $\geq nR.C$ the partitions for R'-fillers that do not intersect with o can be discarded. Additionally,

the partitions for R-fillers that do not intersect with R'-fillers and intersect with o can also be discarded.

Variable Preference For each nominal o, only one variable $v \in V_o$ can be assigned ≥ 1 by the ch-Rule. This heuristic aims at selecting nominal variables that are more likely to succeed. It does this similarly to the case of discarding partitions and allows the ch-Rule to branch on a partition, where nominals intersect with their interacting roles, before branching on a variable, where these nominals do not intersect with the role fillers. For example, we have two variables v_1 and v_2 for a nominal o with $\alpha(v_1) = \{o\}$ and $\alpha(v_2) = \{o, R'\}$ and R' is mapped to $\{o\}$. The variable-preference heuristic then directs the ch-Rule to branch on $v_2 \geq 1$ before branching on $v_1 \geq 1$.

Skip UnSat ch-Rule This optimization affects the ch-Rule and aims at bypassing choice points that are known to lead to a clash. For example, if the ch-Rule is applied to a variable v_a with $o \in \alpha(v_a)$ for $o \in N_o$ and $v \leq 0$ for all $v \in V_o$, this means that branching on $v_a \leq 0$ will result in a clash because the encoded inequation $\xi(o, \geq, 1)$ for o becomes infeasible. The branch for $v_a \leq 0$ can be safely bypassed. If $R \in \alpha(v_a)$ for some $R \in N_R$ and we have $v \leq 0$ for all $v \in V_R$, then the branch for $v_a \leq 0$ can therefore be safely bypassed if v_a occurs in an inequation encoding an at-least restriction. Similarly, the branch for $v_a \geq 1$ is discarded if assigning v_a a value ≥ 1 renders the inequation where v_a occurs obviously infeasible.

Using noGood Variables A variable v is assigned to be a noGood if v must have the value zero. This can happen for a partition P where $\alpha(v) = P$ must be empty because no domain element can be distributed over P without causing a clash. Using the *ch*-Rule a semantic split is performed over each partition's elements; $v \geq 1$ is the case when the restrictions on the partition's elements can be satisfied, and $v \leq 0$ means the restrictions on the partition's elements cannot be satisfied.

Skip UnSat OR-Rule This optimization affects the $\sqcup$-Rule and aims at bypassing choice points that are known to lead to a clash. When the $\sqcup$-Rule is applied to a node y, the branch adding C to $\mathcal{L}(y)$ can be discarded for the following cases: (i) C is a restriction $\geq nR$ and all variables mapped to R are noGood variables, then choosing this disjunct will result in an arithmetic clash. (ii) C is a nominal o and y is assigned to a partition P intersecting with $\neg o$ ($\{o\} \notin P$). (iii) C is the complement of a nominal, $\neg o$, and y is assigned to a partition P intersecting with o ($\{o\} \in P$).

Dependency Directed Backtracking This is a well known optimization technique which allows a search algorithm to bypass choice points. We identify three types of clashes: the logical, OR, and arithmetic clash, and for each type a clash handler is responsible for setting the next choice point to explore.

Logical Clash Handler If a node y has $\{C, \neg C\} \subseteq \mathcal{L}(y)$, y is said to contain a logical clash. The logical clash handler analyzes the clash sources looking for alternative choice points where the algorithm can backjump to. If no such alternative choice is found, then y cannot survive without causing a clash. One can safely assume that the corresponding partition represented in $\mathcal{L}_p(y)$ must be empty and the variable v with $\mathcal{L}_P(y) = \alpha(v)$ must be zero. The algorithm can backjump to the ch-Rule choice point where $v \leq 0$ and safely bypass the choice points with $v \geq 1$. Additionally, if the noGood variable optimization is turned on, then v is also assigned to be a noGood variable.

OR Clash Handler If we have $\neg o \sqcup \neg C \in \mathcal{L}(y)$ and we have $o, C \in \mathcal{L}_P(y)$ then the node y will not survive because all choice points generated by the $\sqcup$-Rule will result in a clash ($\{o, \neg o\} \subseteq \mathcal{L}(y)$ or $\{C, \neg C\} \subseteq \mathcal{L}(y)$). The node y is said to contain an OR-clash and the variable v with $\mathcal{L}_P(y) = \alpha(v)$ must be zero. The algorithm can backjump to the ch-Rule choice point where $v \leq 0$ and safely bypass the choice points with $v \geq 1$. An OR-clash can only be detected if the "Skip UnSat" optimization is turned on and the OR clash handler cannot find alternative choice points because the applicability of the OR-Rule returns an empty list of choice points, i.e., all choices would clash.

Arithmetic Clash Handler An arithmetic clash is detected when the system of inequations cannot have a solution. The following arithmetic clashes can be detected and handled even before running the Simplex procedure (i.e., as soon as inequations are added by the $\bowtie$-Rule). *Clash A*: If there exists a node $y \in G$ such that $\xi(L, \geq, m) \in \mathcal{L}_E(y)$ and $v \leq 0 \in \mathcal{L}_E(y)$ for all $v \in V_L$ (due to the *ch*-Rule), then $\xi(L, \geq, m)$ is infeasible and renders ξ_y infeasible. *Clash B*: If there exists a node $y \in G$ such that $\xi(L, \bowtie, m) \in \mathcal{L}_E(y)$ and for all $v \in V_L$, v has been assigned a value $\sigma(v)$ (due to a previous distribution σ) such that $\sum_{v \in V_L} \sigma(v)$ does not satisfy m. *Clash C*: If there exists a node $y \in G$ such that $\xi(L, \bowtie, m) \in \mathcal{L}_E(y)$ and for some $v_n \in V_L$, the ch-Rule must skip the branch where $v_n \geq 1$ because v_n is a noGood and branching on $v_n \leq 0$ triggers a clash of type A. In all three cases the algorithm can backjump to a branching point for some $v \in V_L$ where $v \geq 1$ and v has not been assigned to be a noGood.

5 Evaluation: First Experimental Results

Our prototype reasoner HARD (Hybrid Algebraic Reasoner for DL) is implemented in Java and uses the OWL-API. We integrated the reasoner interfaces of Pellet v.2.0.0 [11] and HermiT v.1.1 [9] into our implementation and run KB consistency tests using HARD, HermiT, or Pellet. This first evaluation was targeted to test how the algebraic tableau in combination with the proposed optimizations scales for KBs exhibiting interactions between nominals and QCRs. Unfortunately, there are not many suitable ontologies available because QCRs were only recently added to OWL 2 and the ones that are available do not serve well as benchmarks for HARD because their potential difficulty is not caused by interactions between nominals and QCRs. Furthermore, HARD was designed as a research prototype to demonstrate the effectiveness of our algebraic tableau approach and intentionally does not implement most of the optimization techniques implemented by other DL reasoners. It is therefore not the focus of this paper to evaluate HARD's performance for real world ontologies due to the overhead necessary to implement other optimization techniques not related to this line of research.

A typical nominal-QCR interaction occurs when a KB includes axioms of the form $C \equiv o_1 \sqcup \cdots \sqcup o_n$ with $o_1, \ldots, o_n$ nominals, and $D \sqsubseteq \geq mR.C$ or $D \sqsubseteq \leq mR.C$ with $n, m \geq 0$ in $\mathcal{T}$. Our claim is that these patterns are more likely to occur in real world ontologies. For example, in a KB used to classify countries based on their spoken languages one could find axioms of the form *SSC* ≡ *Argentina* ⊔ *Belize* ⊔ *Bolivia* ⊔ $\cdots$ ⊔ *Venezuela* (*SSC* stands for *Spanish_Speaking_Countries*) and *South_America* $\sqsubseteq$ ≥ 11 *Includes.SSC* $\sqcap$ ≤ 11 *Includes.SSC*, and *Caribbean* $\sqsubseteq$ ≥ 3 *Includes.SSC* where *Argentina*, . . . , *Venezuela* are all distinct nominals representing unique countries.

(a) Increasing n in $\mathcal{T}_A$ ($n = m$) (b) Increasing n in $\mathcal{T}_B$ ($n = m$) (c) Increasing m in $\mathcal{T}_B$ (log-linear scale)
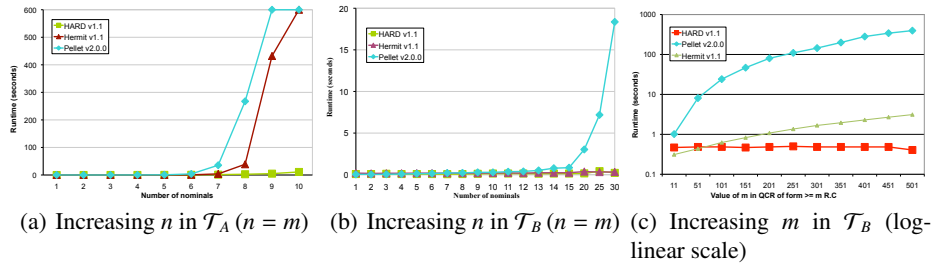
Fig. 2. Evaluation of HARD with HermiT and Pellet (all runtimes in seconds)

We developed two sets of benchmarks consisting of the simple TBoxes $\mathcal{T}_A$ and $\mathcal{T}_B$ defined below where $o_1, \ldots, o_n$ are all disjoint nominals and n and m positive numbers:
$$\mathcal{T}_A = \{C \equiv o_1 \sqcup \ldots \sqcup o_n, \ D \sqsubseteq \, \geq (m+1)\,R.C\}, \mathcal{T}_B = \{C \equiv o_1 \sqcup \ldots \sqcup o_n, \ D \sqsubseteq \, \geq m\,R.C\}$$

In a first set of benchmarks we set $n = m$ and increment n by 1. Notice that due to the nominals semantics and their interaction with $FIL(R)$, $\mathcal{T}_A$ is inconsistent because the cardinality of $FIL(R)$ can be at most n while $\mathcal{T}_B$ is consistent. The results of the tests are shown in Fig. 2 (the runtimes were computed as the average of 10 independent runs). For HARD all optimization techniques described above were switched on. In the case of inconsistent KBs (Fig. 2(a)) one can easily see that HARD outperforms the other reasoners whose performance quickly degrades even with small values of n. In the case of consistent KBs (Fig. 2(b)) HARD performs similar to HermiT while Pellet's performance degrades. In a second set of tests for consistent KBs the size of m in $\mathcal{T}_B$ increases but the number of nominals remains constant; we set $n = 5$ and increment m by 50. Fig. 2(c) clearly demonstrates that HARD's performance remains constant while the performance of the other reasoners severely degrades as m grows (observe the logarithmic scale for the runtime).

6 Conclusion and Future Work

We exploited the high level of information of the algebraic method and presented optimization techniques related to nominals, QCRs and their interactions. Our first experimental results show that algebraic reasoning outperforms existing DL reasoning methods by several orders of magnitude, although we used small examples. One might argue that these results are based on special case patterns, however, it is clear that such patterns are inevitable for designing some real world ontologies. It is part of ongoing work to report on performance improvements in more general cases. We are also working on extending our calculus to $\mathcal{SHOIQ}$ by additionally allowing inverse roles. Our conjecture is that the worst-case complexity of our calculus might remain unchanged and, thus, would become worst-case optimal for $\mathcal{SHOIQ}$.

References

1. J. Faddoul, N. Farsinia, V. Haarslev, and R. Möller. A hybrid tableau algorithm for $\mathcal{ALCQ}$. In *Proc. of the 2008 Int. Workshop on Description Logics, also in 18th European Conference on Artificial Intelligence (ECAI 2008)*, pages 725–726, 2008.

2. J. Faddoul and V. Haarslev. Algebraic tableau reasoning for the description logic $\mathcal{SHOQ}$. *Logic Journal of the IGPL, Special Issue on Hybrid Logics*, 2010. To appear.

3. J. Faddoul, V. Haarslev, and R. Möller. Algebraic tableau algorithm for $\mathcal{ALCOQ}$. In *Proc. of the 2009 Int. Workshop on Description Logics (DL 2009)*, 2009.

4. N. Farsiniamarj and V. Haarslev. Practical reasoning with qualified number restrictions: A hybrid Abox calculus for the description logic $\mathcal{SHQ}$. *AI Communications*, Special Issue on Practical Aspects of Automated Reasoning, 2009. To appear.

5. V. Haarslev and R. Möller. Optimizing reasoning in description logics with qualified number restrictions. In *Description Logics*, pages 142–151, 2001.

6. V. Haarslev, M. Timmann, and R. Möller. Combining tableaux and algebraic methods for reasoning with qualified number restrictions. In *Description Logics*, pages 152–161, 2001.

7. I. Horrocks and U. Sattler. Ontology reasoning in the $\mathcal{SHOQ}$(D) description logic. In *Proc. of the 17th Int. Joint Conf. on Artificial Intelligence (IJCAI 2001)*, pages 199–204. Morgan Kaufmann, Los Altos, 2001.

8. Y. Kazakov and B. Motik. A resolution-based decision procedure for $\mathcal{SHOIQ}$. *Journal of Automated Reasoning*, 40(2-3):89–116, 2008.

9. B. Motik, R. Shearer, and I. Horrocks. Optimized reasoning in description logics using hypertableaux. In *(CADE-21)*, volume 4603 of *Lecture Notes in Artificial Intelligence*, pages 67–83. Springer, 2007.

10. H. J. Ohlbach and J. Koehler. Modal logics, description logics and arithmetic reasoning. *Artificial Intelligence*, 109(1-2):1–31, 1999.

11. B. Parsia, B. Cuenca Grau, and E. Sirin. From wine to water: Optimizing description logic reasoning for nominals. In *Proc. of the 10th Int. Conference on Principles of Knowledge Representation and Reasoning (KR)*, pages 90–99, 2006.

Complexity of Axiom Pinpointing
in the DL-Lite Family

Rafael Peñaloza[1] and Barış Sertkaya[2]

[1] Theoretical Computer Science TU Dresden, Germany
`penaloza@tcs.inf.tu-dresden.de`
[2] SAP Research Center Dresden, Germany
`baris.sertkaya@sap.com`

1   Introduction

In real world applications where ontologies are employed, often the knowledge engineer not only wants to know whether her ontology has a certain (unwanted) consequence or not, but also wants to know *why* it has this consequence. Even for ontologies of moderate size, finding explanations for a given consequence is not an easy task without getting support from an automated tool. The task of finding explanations for a given consequence, i.e., minimal subsets of the original ontology that have the given consequence is called *axiom pinpointing* in the literature.

Existing work on axiom pinpointing in DLs can be classified under two main categories, namely the glass-box approach, and the black-box approach. The idea underlying the *glass-box approach* is to extend the existing reasoning algorithms such that while doing reasoning, at the same time they can keep track of the axioms used, and detect which of the axioms in the TBox are responsible for a given consequence. In [24] a pinpointing extension of the tableau-based satisfiability algorithm for the DL $\mathcal{ALC}$ has been introduced. Later in [19], this approach has been further extended to DLs that are more expressive than $\mathcal{ALC}$. In [17] a pinpointing algorithm for $\mathcal{ALC}$ with general concept inclusions (GCIs) has been presented by following the approach in [2]. In order to overcome the problem of developing a pinpointing extension for every particular tableau-based algorithm, a general pinpointing extension for tableau algorithms has been developed in [3, 6]. Similarly, an automata-based general approach for obtaining glass-box pinpointing algorithms has been introduced in [4, 5].

In contrast to the glass-box approach, the idea underlying the *black-box approach* is to make use of the existing highly optimized reasoning algorithms wihout having to modify them. The most naïve black-box approach would of course be to generate every subset of the original TBox, and ask a DL reasoner whether this subset has the given consequence or not, which obviously is very inefficient. In [16] more efficient approaches based on Reiter's hitting set tree algorithm [23] have been presented. The experimental resuts in [16] demonstrate that this approach behaves quite well in practice on realistic TBoxes written in expressive DLs. A similar approach has successfully been used in [14] for explaining inconsistencies in OWL ontologies. The main advantages of the black-box

approach are that one can use existing DL reasoners, and that it is independent of the DL reasoner being used. In [13] the black-box approach has been used for computing more fine grained explanations, i.e., not just the set of relevant axioms in the TBox but parts of these axioms that actually lead to the given consequence.

Although various methods and aspects of axiom pinpointing have been considered in the literature, its computational complexity has not been investigated in detail yet. Obviously, axiom pinpointing is at least as hard as standard reasoning. Nevertheless, especially for tractable DLs it makes sense to investigate whether explanations for a consequence can efficiently be enumerated or not. In [7] it has been shown that a given consequence can have exponentially many explanations (there called *MinAs*, which stands for *minimal axiom sets*), and checking the existence of a MinA within a cardinality bound is NP-hard even for a fragment of $\mathcal{EL}$ that only allows for conjunction on both sides of a GCI. In [20–22] we have investigated the complexity of axiom pinpointing in the propositional Horn fragment, and in the tractable DL $\mathcal{EL}$. We have given a polynomial delay algorithm for enumerating MinAs in the propositional Horn setting that works even if the MinAs are required to be enumerated in reverse lexicographic order. We have also shown that for the dual-Horn setting, where the axioms have at most one negative literal, this problem is at least as hard as the hypergraph transversal enumeration problem, whose exact complexity is a prominent open problem [12]. Moreover, we have shown that for $\mathcal{EL}$ TBoxes MinAs cannot be enumerated in output-polynomial time unless P = NP.

In the present work we investigate the complexity of axiom pinpointing in the other family of tractable DLs, namely the DL-Lite family, which has been very popular due to its success in efficiently accessing large data and answering complex queries on this data [10, 1]. For this family various aspects of finding explanations have already been considered in [9, 8]. There the main focus is on the problem of explaining query answering and ABox reasoning, which are the most standard types of reasoning problems in the DL-Lite family. In particular the authors investigate in detail the problem of determining why a value is returned as an answer to a conjunctive query posed to a DL-Lite ABox, why a conjunctive query is unsatifiable, and why a particular value is not returned as answer to a conjunctive query. Complementary to the work in [9, 8] here we consider the problem of explaining TBox reasoning. We investigate in detail the complexity of enumerating MinAs in a DL-Lite TBox for a given consequence of this TBox. We show that for $DL-Lite_{core}^{\mathcal{H}}$, $DL-Lite_{krom}^{\mathcal{H}}$ and $DL-Lite_{horn}^{\mathcal{N}}$ TBoxes MinAs are efficiently enumerable with polynomial delay, but for $DL-Lite_{bool}$ they cannot be enumerated in output-polynomial time unless P = NP.

2　Preliminaries

We briefly introduce the syntax of the DL-Lite family following the notation in [1]. DL-Lite concepts and roles are constructed as follows:

$$r := p \mid p^-, \qquad B := \bot \mid A \mid \geq q\ r, \qquad C := B \mid \neg C \mid C_1 \sqcap C_2,$$

where A is a concept name, p is a role name, and q is a natural number. Concepts of the form B are called *basic*, and those of form C are called *general* concepts.

A $DL-Lite_{bool}^{\mathcal{N}}$ TBox is a set of axioms of the form $C_1 \sqsubseteq C_2$, where C_1, C_2 are general concepts. A TBox is called *core*, denoted as $DL-Lite_{core}^{\mathcal{N}}$, if its axioms are of the form $B_1 \sqsubseteq B_2$, or $B_1 \sqsubseteq \neg B_2$, where B_1, B_2 are basic concepts. *Krom* TBoxes generalize core ones by allowing also axioms of the form $\neg B_1 \sqsubseteq B_2$. These TBoxes are denoted as $DL-Lite_{krom}^{\mathcal{N}}$. Finally, a *Horn* TBox $DL-Lite_{horn}^{\mathcal{N}}$ is composed only of axioms of the form $\bigsqcap_k B_k \sqsubseteq B$. We can drop the superscript $\mathcal{N}$ from the knowledge bases by allowing only number restrictions of the form $\geq 1\ r$ for constructing basic concepts. We will sometimes use the expression $\exists r$ to represent $\geq 1\ r$. To any of the previously defined TBoxes, we can add *role inclusion axioms* of the form $r_1 \sqsubseteq r_2$. This will be denoted using the superscript $\mathcal{H}$ in the name; e.g. $DL-Lite_{bool}^{\mathcal{H}\mathcal{N}}$. Since we are not dealing with individuals in the present work, role inclusion axioms do not add any expressivity to $DL-Lite_{\alpha}^{\mathcal{H}}$ TBoxes for $\alpha \in \{core, horn, krom\}$. Indeed, a basic concept B will only make use of a role r if B is an existential restriction $\exists r$. As we are only interested in concept subsumption, we can represent the role inclusion axiom $r_1 \sqsubseteq r_2$ by the concept inclusion $\exists r_1 \sqsubseteq \exists r_2$. Thus, the complexity results we present here for for $DL-Lite_{\alpha}$ TBoxes also hold for $DL-Lite_{\alpha}^{\mathcal{H}}$ TBoxes.[3] For sake of simplicity, in the present work we do not consider inverse roles.

Finally we recall basic notions from complexity of enumeration algorithms. For analyzing the performance of algorithms where the size of the output can be exponential in the size of the input, we consider other measures of efficiency. We say that an algorithm runs with *polynomial delay* [15] if the time until the first output is generated, and thereafter the time between any two consecutive outputs is bounded by a polynomial in the size of the input. We say that it runs in *output polynomial time* [15] if it outputs all solutions in time polynomial in the size of the input *and the output*.

3 Complexity of Enumerating all MinAs

The main problem we consider in the present work is, given a DL-Lite TBox and a consequence of it, compute all MinAs for this consequence in the given TBox. We start with defining a MinA.

Definition 1. *Let $\mathcal{T}$ be a DL-Lite TBox and φ a DL-Lite axiom that follows from it, i.e., $\mathcal{T} \models \varphi$. We call a set $\mathcal{M} \subseteq \mathcal{T}$ a* minimal axiom set *or* MinA *for φ in $\mathcal{T}$ if $\mathcal{M} \models \varphi$ and it is minimal w.r.t. set inclusion.*

We define our problem without mentioning a particular DL-Lite fragment but investigate its computational complexity for different fragments in the coming sections separately. In the following, the only consequences we consider are subsumption relations that can be expressed by axioms in the corresponding DL-Lite fragment.

---

[3] Notice that this may not be true if number restrictions are allowed; that is, the complexity results for $DL-Lite_{\alpha}^{\mathcal{N}}$ may not transfer to $DL-Lite_{\alpha}^{\mathcal{H}\mathcal{N}}$.

Problem: MINA-ENUM
Input: A DL-Lite TBox $\mathcal{T}$ and a DL-Lite axiom φ such that $\mathcal{T} \models \varphi$.
Output: The set of all MinAs for φ in $\mathcal{T}$.

3.1 Enumerating MinAs in $DL\text{--}Lite_{core}$ and $DL\text{--}Lite_{krom}$ TBoxes

We start with a basic observation. In the simplest setting where we can consider MINA-ENUM, $\mathcal{T}$ is a $DL\text{--}Lite_{core}$ TBox whose concept inclusion axioms are all of the form $A_1 \sqsubseteq A_2$ for atomic concepts A_1, A_2. Note that in his setting $\mathcal{T}$ becomes just a directed graph, and a MinA for $A_n \sqsubseteq A_m$ is just a simple path between the nodes A_n and A_m.[4] That is, MINA-ENUM boils down to enumerating the simple paths between two vertices in a given directed graph. This problem is well-known, and can be solved with polynomial delay, even if the simple paths are required to be output in the increasing order of their lengths [25]. This observation has already been briefly mentioned in the works [9, 8], which mainly concentrate on explaining query answering.

In $DL\text{--}Lite_{core}$ TBoxes, additionally we need to deal with unqualified existential restriction, and also with inclusion axioms that have negated basic concepts in the right hand side. Since unqualified existential restrictions cannot interact and give rise to additional MinAs in a $DL\text{--}Lite_{core}$ TBox, we can treat them as atomic concepts. We need to deal with the axioms with a negated basic concept in the right hand side separately since they can lead to additional MinAs due to contraposition. We demonstrate this with an example.

Example 1. Consider the $DL\text{--}Lite_{core}$ TBox $\mathcal{T} = \{A \sqsubseteq \neg \exists r_1,\ \exists r_2 \sqsubseteq \exists r_1,\ D \sqsubseteq \exists r_2,\ D \sqsubseteq \exists r_1,\ A \sqsubseteq D\}$ and the axiom $\varphi : A \sqsubseteq \neg D$ which follows from $\mathcal{T}$. We can treat $\exists r_1$ and $\exists r_2$ just like atomic concepts since without role inclusion axioms they cannot interact and lead to additional MinAs. That is we have the MinAs $M_1 = \{A \sqsubseteq \neg \exists r_1,\ \exists r_2 \sqsubseteq \exists r_1,\ D \sqsubseteq \exists r_2\}$, and $M_2 = \{A \sqsubseteq \neg \exists r_1,\ D \sqsubseteq \exists r_1\}$.

Note that A is actually unsatisfiable, i.e., it is subsumed by any other concept. This might also be the reason why φ follows from $\mathcal{T}$. This means that we also need to find out the reasons why A is unsatisfiable. The only MinA for $A \sqsubseteq \neg A$ in $\mathcal{T}$ is $M = \{A \sqsubseteq \neg \exists r_1,\ D \sqsubseteq \exists r_1,\ A \sqsubseteq D\}$. However, it contains M_2, which is a MinA for φ, thus M is not a *minimal* axiom set, i.e., a MinA for φ. It means that when we are looking for MinAs for an axiom $B_1 \sqsubseteq B_2$ s.t. B_1 is unsatisfiable, we also need to find MinAs for $B_1 \sqsubseteq \neg B_1$ that do *not* contain any of the MinAs for the original axiom.

Our algorithm that takes all these cases into account is described in detail in Algorithm 1 where $t(\varphi)$ stands for the tail (i.e. the left hand side), and $h(\varphi)$ stands for the head (i.e. the right hand side) of axiom φ.

Theorem 1. *Algorithm 1 solves* MINA-ENUM *for $DL\text{--}Lite_{krom}$ TBoxes with polynomial delay.*

---

[4] A simple path is a path with no repeated vertices.

Algorithm 1 Enumerating all MinAs for $DL-Lite_{krom}$ TBoxes

---

Procedure: ALL-MINAS($\mathcal{T}, \varphi$)   ($\mathcal{T}$ a $DL-Lite_{krom}$ TBox, φ an axiom s.t. $\mathcal{T} \models \varphi$)

 1: ALL-MINAS-AUX($\mathcal{T}, \varphi$)
 2: **if** $\mathcal{T} \models t(\varphi) \sqsubseteq \neg t(\varphi)$ **then**
 3:     $\mathcal{T}' := \{\psi \in \mathcal{T} \mid h(\psi) \neq h(\varphi) \text{ and } t(\psi) \neq \neg h(\varphi)\}$
 4:     ALL-MINAS-AUX($\mathcal{T}', t(\varphi) \sqsubseteq \neg t(\varphi)$)   (MinAs for unsatisfiability of $t(\varphi)$)
 5: **end if**

Procedure: ALL-MINAS-AUX($\mathcal{T}, \varphi$)   ($\mathcal{T}$ a $DL-Lite_{krom}$ TBox, φ an axiom, $\mathcal{T} \models \varphi$)

 1: **if** $t(\varphi) = h(\varphi)$ **then return** $\emptyset$
 2: **end if**
 3: **for all** $\psi \in \mathcal{T}$ **do**
 4:     **if** $t(\varphi) = t(\psi)$ and $\mathcal{T} \setminus \{\psi\} \models h(\psi) \sqsubseteq h(\varphi)$ **then**
 5:         **print**$\{\psi\} \cup$ ALL-MINAS($\mathcal{T} \setminus \{\psi\}, h(\psi) \sqsubseteq h(\varphi)$)
 6:     **end if**
 7:     **if** $t(\varphi) = \neg h(\psi)$ and $\mathcal{T} \setminus \{\psi\} \models \neg t(\psi) \sqsubseteq h(\varphi)$ **then**
 8:         **print**$\{\psi\} \cup$ ALL-MINAS($\mathcal{T} \setminus \{\psi\}, \neg t(\psi) \sqsubseteq h(\varphi)$)
 9:     **end if**
10: **end for**

Proof. It is not difficult to see that the algorithm terminates. Termination of the procedure ALL-MINAS depends on the termination of the procedure ALL-MINAS-AUX. ALL-MINAS-AUX terminates since the base case of the recursion is well established, and there are finitely many ψ in $\mathcal{T}$.

The algorithm is sound. ALL-MINAS-AUX outputs an axiom ψ, only if using it φ can be derived. Moreover, as soon as the head and the tail of φ become equal, it terminates in line 1. That is it does not allow 'cycles', or redundant axioms in the output. Hence, the outputs of ALL-MINAS-AUX are indeed MinAs for φ in $\mathcal{T}$. ALL-MINAS additionally checks if the tail of φ is unsatisfiable, and if this is the case also outputs the MinAs for $t(\varphi) \sqsubseteq \neg t(\varphi)$ that do not contain any of the previously output MinAs.

The algorithm is complete. ALL-MINAS-AUX iterates over the axioms in $\mathcal{T}$ and searches for the MinAs for φ in a depth-first manner. If $\mathcal{T} \models t(\varphi) \sqsubseteq \neg t(\varphi)$, then ALL-MINAS additionally searches for MinAs for $t(\varphi) \sqsubseteq \neg t(\varphi)$, in the same manner. These are all MinAs for φ in $\mathcal{T}$.

Note that in lines 4 and 7 of the procedure ALL-MINAS-AUX the algorithm checks whether the selected axiom ψ will lead to a MinA. Clearly, for $DL-Lite_{core}$ and $DL-Lite_{krom}$ this check is polynomial. Moreover, this check avoids the algorithm picking a 'wrong' axiom that will result in an exponential number of recursive calls that do not lead to a MinA. That is, it guarantees that the algorithm outputs the next MinA, or stops, after at most a polynomial number of steps, i.e., it is polynomial delay.   □

3.2   MinAs in $DL-Lite_{horn}^{\mathcal{N}}$ TBoxes

Next we show that for $DL-Lite_{horn}^{\mathcal{N}}$ TBoxes, MinAs can be enumerated with polynomial delay as well. Furthermore, we show that this is true even if the

MinAs are required to be output in a given reverse lexicographic order. To do this, we construct, for every $DL-Lite_{horn}^{\mathcal{N}}$ TBox $\mathcal{T}$ a propositional Horn TBox $\mathcal{G}_{\mathcal{T}}$ as follows: for every basic concept B create a propositional variable v_B; for every axiom $\prod_{i=1}^{n} B_i \sqsubseteq B$ add the Horn clause $\bigwedge_{i=1}^{n} v_{B_i} \to v_B$; and for each pair of number restrictions $\geq q_1 r, \geq q_2 r$ with $q_1 > q_2$ appearing in $\mathcal{T}$, add the Horn clause $v_{\geq q_1 r} \to v_{\geq q_2 r}$. We will call the latter ones *implicit axioms*. It is not difficult to see that $\mathcal{T} \models \prod_{i=1}^{n} A_i \sqsubseteq C$ iff $\mathcal{G}_{\mathcal{T}} \models \bigwedge_{i=1}^{n} v_{A_i} \to v_C$. Furthermore, MinA $\mathcal{M}$ in $\mathcal{G}_{\mathcal{T}}$ gives rise to a MinA in $\mathcal{T}$ consisting of all axioms representing non implicit axioms in $\mathcal{M}$. However, different MinAs in $\mathcal{G}_{\mathcal{T}}$ can give rise to the same MinA in $\mathcal{T}$. For instance let $\mathcal{T} = \{A \sqsubseteq\, \geq 2r, A \sqsubseteq\, \geq 3r, \geq 1r \sqsubseteq B\}$. Clearly $\mathcal{G}_{\mathcal{T}}$ constructed from $\mathcal{T}$ as described has three MinAs for $v_A \to v_B$, but there are only two MinAs for $A \sqsubseteq B$ in $\mathcal{T}$. The reason is that the implicit subsumption $\geq 3r \sqsubseteq\, \geq 1r$ is represented twice in $\mathcal{G}_{\mathcal{T}}$: one through the direct edge, and another with a path travelling along $v_{\geq 2r}$. We solve this problem by using *immediate* MinAs.

Definition 2. *Let $\mathcal{T}$ be a $DL-Lite_{horn}^{\mathcal{N}}$ TBox. A MinA $\mathcal{M}$ in $\mathcal{G}_{\mathcal{T}}$ is called* immediate *if for every implicit axiom $\tau \in \mathcal{G}_{\mathcal{T}}, \mathcal{M} \models \tau$ implies $\tau \in \mathcal{M}$.*

Note that there is a one-to-one correspondence between MinAs for $\prod_{i=1}^{n} A_i \sqsubseteq C$ in $\mathcal{T}$ and immediate MinAs for $\bigwedge_{i=1}^{n} v_{A_i} \to v_C$ in $\mathcal{G}_{\mathcal{T}}$. Thus, if we can enumerate all immediate MinAs in $\mathcal{G}_{\mathcal{T}}$ in output polynomial time, we will be able to enumerate also all MinAs in $\mathcal{T}$ within the same complexity bound. We now show how all immediate paths can be computed. For this, we first need to introduce the notion of a valid ordering on the axioms in a TBox.

Definition 3. *Let $\mathcal{T}$ be a propositional Horn TBox, and $\phi = \bigwedge_{i=1}^{n} a_i \to b$ be an axiom in $\mathcal{T}$. We denote the left-handside (lhs) of ϕ with $\mathsf{T}(\phi)$, and its right-handside (rhs) with $\mathsf{h}(\phi)$, i.e., $\mathsf{T}(\phi) := \{a_1, \ldots, a_n\}$ and $\mathsf{h}(\phi) := b$. With $\mathsf{h}^{-1}(b)$ we denote the set of axioms in $\mathcal{T}$ whose rhs are b. Let $\mathcal{M} = \{t_1, \ldots, t_m\}$ be a MinA for $\bigwedge_{a \in A} a \to c$. We call an ordering $t_1 < \ldots < t_m$ a* valid ordering *on $\mathcal{M}$ if for every $1 \leq i \leq m$, $\mathsf{T}(t_i) \subseteq A \cup \{\mathsf{h}(t_1), \ldots, \mathsf{h}(t_{i-1})\}$ holds.[5]*

It is easy to see that for every immediate MinA there is always at least one such valid ordering. In the following, we use this fact to construct a set of sub-TBoxes that contain all and only the remaining immediate MinAs, following the ideas in [18].

Definition 4. *Let $\mathcal{M}$ be an immediate MinA in $\mathcal{G}_{\mathcal{T}}$ with $|\mathcal{M}| = m$, and $<$ be a valid ordering on $\mathcal{M}$. For each $1 \leq i \leq m$ we obtain a TBox $\mathcal{T}_i$ from $\mathcal{G}_{\mathcal{T}}$ as follows: if t_i is an implicit axiom, then $\mathcal{T}_i = \emptyset$; otherwise, (i) for each j s.t. $i < j \leq m$ remove all axioms in $\mathsf{h}^{-1}(\mathsf{h}(t_j))$ except for t_j, i.e., remove all axioms with the same rhs as t_j except for t_j itself, (ii) remove t_i, and (iii) add all implicit axioms.*

The naïve method for computing one MinA can be easily adapted to the computation of an immediate MinA in polynomial time by simply considering

---

[5] That is, each variable on the lhs of t_i is in A, or it is the rhs of a previous axiom.

Algorithm 2 Enumerating all MinAs for $DL\text{--}Lite_{horn}^{\mathcal{N}}$ TBoxes

---

Procedure ALL-MINAS($\mathcal{T},\phi$)   ($\mathcal{T}$ a $DL\text{--}Lite_{horn}^{\mathcal{N}}$ TBox, ϕ an axiom s.t. $\mathcal{T} \models \phi$)

1: **if** $\mathcal{T} \not\models \phi$ **then return**
2: **else**
3:      $\mathcal{M} :=$ an immediate MinA in $\mathcal{G}_{\mathcal{T}}$
4:      $\mathcal{I} := \{t \mid t$ is an implicit axiom$\}$
5:      output $\mathcal{M} \setminus \mathcal{I}$
6:      **for** $1 \leq i \leq |\mathcal{M}|$ **do**
7:          compute $\mathcal{T}_i$ from $\mathcal{M}$ as in Definition 4
8:          ALL-MINAS($\mathcal{T}_i \setminus \mathcal{I},\phi$)
9:      **end for**
10: **end if**

---

first all non-implicit axioms for removal, and ordering the implicit ones as follows: if $t_1 := (\geq q_1 r) \sqsubseteq (\geq q_2 r)$, and $t_2 := (\geq q_1' r) \sqsubseteq (\geq q_2' r)$ are two implicit axioms and $q_1 - q_2 < q_1' - q_2'$, then t_1 appears before t_2.

Lemma 1. *Let $\mathcal{M}$ be an immediate MinA for ϕ in $\mathcal{T}$, and let $\mathcal{T}_1, \ldots, \mathcal{T}_m$ be constructed from $\mathcal{T}$ and $\mathcal{M}$ as in Definition 4. Then, for every immediate MinA $\mathcal{N}$ for ϕ in $\mathcal{T}$ that is different from $\mathcal{M}$, there exists **exactly one** i, where $1 \leq i \leq m$, such that $\mathcal{N}$ is a MinA for ϕ in $\mathcal{T}_i$.*

Proof. Let $t_1 < \ldots < t_m$ be a valid ordering on $\mathcal{M}$, and $\mathcal{N}$ an immediate MinA for ϕ in $\mathcal{T}$ such that $\mathcal{N} \neq \mathcal{M}$. Then, $\mathcal{M} \setminus \mathcal{N} \neq \emptyset$. Let t_k be the largest non-implicit axiom in $\mathcal{M} \setminus \mathcal{N}$ w.r.t. the ordering $<$. We show that $\mathcal{N} \subseteq \mathcal{T}_k$ and $\mathcal{N} \not\subseteq \mathcal{T}_i$ for all $i \neq k$, $1 \leq i \leq m$.

Assume there is an axiom $t \in \mathcal{N}$ s.t. $t \notin \mathcal{T}_k$. Since $\mathcal{T}_k$ contains all implicit axioms, t should be one of the non-implicit axioms removed from $\mathcal{T}$ either in step (i) or in step (ii) of Definition 4. It cannot be step (ii) because $t_k \notin \mathcal{N}$ since $t_k \in \mathcal{M} \setminus \mathcal{N}$. Thus, it should be step (i). This implies that there exists a j, $k < j \leq m$, such that t_j satisfies $\mathsf{h}(t) = \mathsf{h}(t_j)$. Recall that we chose k to be the largest axiom in $\mathcal{M} \setminus \mathcal{N}$ w.r.t. the valid ordering $<$ on $\mathcal{M}$. Then this t_j should be in $\mathcal{N}$. But then $\mathcal{N}$ contains two axioms with the rhs $\mathsf{h}(t)$, which contradicts with the fact that $\mathcal{N}$ is a MinA, and thus it is minimal. Hence, $\mathcal{N} \subseteq \mathcal{T}_k$.

Now take an i s.t. $i \neq k$. If $i > k$, then $t_i \in \mathcal{N}$ but $t_i \notin \mathcal{T}_i$, and hence $\mathcal{N} \not\subseteq \mathcal{T}_i$. If $i < k$, then there is an axiom $t \in \mathcal{N}$ such that $\mathsf{h}(t) = \mathsf{h}(t_k)$ since otherwise $\mathcal{M}$ and $\mathcal{N}$ would not be MinAs. By construction, $t \notin \mathcal{T}_i$, hence $\mathcal{N} \not\subseteq \mathcal{T}_i$. □

Lemma 1 gives an idea of how to compute the remaining MinAs from a given one in the $DL\text{--}Lite_{horn}^{\mathcal{N}}$ setting. Algorithm 2 describes how we can use this lemma to enumerate all MinAs in a $DL\text{--}Lite_{horn}^{\mathcal{N}}$ TBox $\mathcal{T}$ by enumerating all immediate MinAs in $\mathcal{G}_{\mathcal{T}}$.

Theorem 2. *Algorithm 2 solves* MINA-ENUM *for $DL\text{--}Lite_{horn}^{\mathcal{N}}$ TBoxes with polynomial delay.*

Proof. The algorithm terminates since $\mathcal{T}$ is finite. It is sound since its outputs are MinAs for ϕ in $\mathcal{T}$. Completeness follows from Lemma 1.

In each recursive call of the algorithm there is one consequence check (line 1), and one MinA computation (line 3). The consequence check can be done in polynomial time [1]. One MinA is computed in polynomial time by iterating over the axioms in $\mathcal{T}$ and removing the redundant ones. Thus the algorithm spends at most polynomial time between each output, i.e., it is polynomial delay.   □

We now modify Algorithm 2 and show that it can also enumerate MinAs in reverse lexicographic order with polynomial delay. The lexicographic order we use is defined as follows:

Definition 5. *Let the elements of a set S be linearly ordered. This order induces a linear strict order on $\mathscr{P}(S)$, which is called the* lexicographic order*. We say that a set $R \subseteq S$ is* lexicographically smaller *than a set $T \subseteq S$ where $R \neq T$ if the first element at which they disagree is in R.*

The modified algorithm keeps a set of TBoxes in a priority queue $\mathcal{Q}$. These TBoxes are the "candidates" from which the MinAs are going to be computed. Each TBox can contain zero or more MinAs. They are inserted into $\mathcal{Q}$ by the algorithm at a cost of $O(|\mathcal{T}| \cdot \log(M))$ per insertion, where $\mathcal{T}$ is the original TBox and M is the total number of TBoxes inserted. Note that M can be exponentially bigger than $|\mathcal{T}|$ since there can be exponentially many MinAs. That is the algorithm uses potentially exponential space. The other operation that the algorithm performs on $\mathcal{Q}$ is to find and delete the maximum element of $\mathcal{Q}$. The maximum element of $\mathcal{Q}$ is the TBox in $\mathcal{Q}$ that contains the lexicographically largest MinA among the MinAs contained in all other TBoxes in $\mathcal{Q}$. This operation can also be performed within $O(|\mathcal{T}| \cdot \log(M))$ time bound. Note that given a $\mathcal{T}$, the lexicographically largest MinA in $\mathcal{T}$ can be computed by starting with the axiom that is the smallest one w.r.t. the linear oder on $\mathcal{T}$, iterating over the axioms and removing an axiom if the resulting TBox still has the required consequence. Obviously this operation is in $O(|\mathcal{T}|)$. This is why the time bounds for insertion and deletion depend also on $|\mathcal{T}|$ and not only on M.

Theorem 3. *Algorithm 3 enumerates all MinAs for a $DL-Lite_{horn}^{\mathcal{N}}$ TBox in reverse lexicographic order with polynomial delay.*

Proof. The algorithm terminates since $\mathcal{T}$ is finite. Soundness is shown as follows: $\mathcal{Q}$ contains initially only the original TBox $\mathcal{T}$. Thus the first output is lexicographically the last MinA in $\mathcal{T}$. By Lemma 1 the MinA that comes just before the last one is contained in exactly one of the $\mathcal{T}_i$s that are computed and inserted into $\mathcal{Q}$ in lines 8 and 9. In line 3 $\mathcal{J}$ is assigned the TBox that contains this MinA. Thus the next output will be the MinA that comes just before the lexicographically last one. It is not difficult to see that in this way the MinAs will be enumerated in reverse lexicographic order. By Lemma 1 it is guaranteed that the algorithm enumerates all MinAs.

In one iteration, the algorithm performs one find operation and one delete operation on $\mathcal{Q}$, each of which takes time $O(n \cdot \log(M))$, and a MinA computation

---

Algorithm 3 Enumerating all MinAs in reverse lexicographical order

---

Procedure ALL-MINAS-REV-ORD($\mathcal{T}$,ϕ)   ($\mathcal{T}$ a $DL-Lite_{horn}^{\mathcal{N}}$ TBox, ϕ an ax., $\mathcal{T} \models \phi$)

 1: $\mathcal{Q} := \{\mathcal{T}\}$
 2: **while** $\mathcal{Q} \neq \emptyset$ **do**
 3:     $\mathcal{J} :=$ maximum element of $\mathcal{Q}$
 4:     remove $\mathcal{J}$ from $\mathcal{Q}$
 5:     $\mathcal{M} :=$ the lexicographical largest MinA in $\mathcal{J}$
 6:     output $\mathcal{M}$
 7:     **for** $1 \leq i \leq |\mathcal{M}|$ **do**
 8:         compute $\mathcal{T}_i$ from $\mathcal{M}$ as in Definition 4
 9:         insert $\mathcal{T}_i$ into $\mathcal{Q}$ if $\mathcal{T}_i \models \phi$
10:     **end for**
11: **end while**

---

that takes $O(n)$ time, where $n = |\mathcal{T}|$. In addition it performs at most n $\mathcal{T}_i$ computations, and at most n insertions into $\mathcal{Q}$. Each $\mathcal{T}_i$ requires $O(n^2)$ time to be constructed, and each insertion into $\mathcal{Q}$ takes $O(n \cdot \log(M))$ time. The total delay is thus $O(2 \cdot (n \cdot \log(M)) + n + n \cdot (n^2 + n \cdot \log(M))) = O(n^3)$.   □

3.3   MinAs in $DL-Lite_{bool}$ TBoxes

The axioms that we have used so far allowed for only basic concepts and their negations, and we were able to show that in this restricted setting, MinAs are enumerable with polynomial delay. However, we have not yet explored the complexity of these problems if general concepts are allowed. As shown in [1], deciding whether an axiom follows from a $DL-Lite_{bool}$ TBox is already NP-hard. Since computing a MinA is at least as hard as doing a consequence check, we cannot expect to find a single MinA in polynomial time. This in particular implies that MinAs cannot be enumerated with polynomial delay in the $DL-Lite_{bool}$ setting. What we can ask next is whether all MinAs are computable in output polynomial time. In order to answer this, we investigate the decision version of this problem:

Problem: ALL-MINAS
Input: A DL-Lite TBox $\mathcal{T}$ and an axiom φ such that $\mathcal{T} \models \varphi$, and a set of TBoxes $\mathscr{T} \subseteq \mathscr{P}(\mathcal{T})$.
Question: Is $\mathscr{T}$ precisely the set of all MinAs for φ in $\mathcal{T}$?

Because if this problem is not solvable in polynomial time, then all MinAs cannot be computed in output-polynomial time. Due to lack of space, we cannot include the proof of this claim here. The proof is based on a general argument and can be found in [21] (Proposition 6). Next we show that ALL-MINAS is coNP-hard for $DL-Lite_{bool}$ TBoxes.

Lemma 2. ALL-MINAS *is coNP-hard for* $DL-Lite_{bool}$ *TBoxes. This already holds if the axioms in* $\mathcal{T}$ *are of the form* $A \sqsubseteq C$ *where* A *is a concept name and* C *a general concept.*

Proof. We present a reduction from the following coNP-hard problem [11, 7].

Problem: ALL-MV
Input: A monotone Boolean formula ϕ and a set $\mathscr{V}$ of minimal valuations satisfying ϕ.
Question: Is $\mathscr{V}$ precisely the set of all minimal valuations satisfying ϕ?

Let $\phi, \mathscr{V}$ be an instance of ALL-MV. We introduce a concept name A_p for each propositional variable p appearing in ϕ and two additional concept names A_0, A_1. From ϕ we construct the general concep C_ϕ by changing each conjunction $\wedge$ to $\sqcap$, each disjunction $\vee$ to $\sqcup$ and each propositional variable p to $\neg B_p$.[6] Using these we construct the TBox $\mathcal{T} := \{A_1 \sqsubseteq \neg C_\phi\} \cup \{B_p \sqsubseteq \neg A_0 \mid p \in \mathsf{var}(\phi)\}$ and the set of MinAs $\mathscr{T} := \{\{A_1 \sqsubseteq C_\phi\} \cup \{B_p \sqsubseteq \neg A_0 \mid p \in \mathcal{V}\} \mid \mathcal{V} \in \mathscr{V}\}$. It is easy to see that $\mathcal{T}$ and $\mathscr{T}$ indeed form an instance of ALL-MINAS for the axiom $A_0 \sqsubseteq \neg A_1$. Furthermore, $\mathscr{T}$ is the set of all MinAs for $A_0 \sqsubseteq \neg A_1$ iff $\mathscr{V}$ is the set of all minimal valuations satisfying ϕ. $\square$

The following is an immediate consequence of Lemma 2.

Corollary 1. *For $DL-Lite_{bool}$ TBoxes all MinAs cannot be computed in output-polynomial time unless* P = NP.

4 Concluding Remarks and Future Work

We have investigated the complexity of axiom pinpointing in the DL-Lite family. We have shown that for $DL-Lite_{core}^{\mathcal{H}}$, $DL-Lite_{krom}^{\mathcal{H}}$ and $DL-Lite_{horn}^{\mathcal{N}}$ TBoxes MinAs are efficiently enumerable with polynomial delay, but for $DL-Lite_{bool}$ they cannot be enumerated in output-polynomial time unless P = NP. For simplicity we did not consider inverse roles here, although we believe our results will hold in presence of inverse roles. As future work we are going to investigate whether this is the case.

Finding explanations for query answering and ABox reasoning has already been considered in [9, 8]. However, these works investigate computing only one explanation. As future work we are going to work on the problem of computing all MinAs for explaining the reasoning problems considered there.

References

1. A. Artale, D. Calvanese, R. Kontchakov, and M. Zakharyaschev. The DL-Lite family and relations. *Journal of Artificial Intelligence Research*, 36:1–69, 2009.

---

[6] We use the abbreviation $X \sqcup Y$ for $\neg(\neg X \sqcap \neg Y)$.

2. F. Baader and B. Hollunder. Embedding defaults into terminological representation systems. *Journal of Automated Reasoning*, 14:149–180, 1995.
3. F. Baader and R. Peñaloza. Axiom pinpointing in general tableaux. In *Proc. of TABLEAUX 2007*, volume 4548 of *LNCS*, pages 11–27. Springer-Verlag, 2007.
4. F. Baader and R. Peñaloza. Automata-based axiom pinpointing. In *Proc. of the 4th Int. Joint Conf. on Automated Reasoning, (IJCAR 2008)*, volume 5195 of *LNCS*, pages 226–241. Springer-Verlag, 2008.
5. F. Baader and R. Peñaloza. Automata-based axiom pinpointing. *Journal of Automated Reasoning*, 2009. To appear.
6. F. Baader and R. Peñaloza. Axiom pinpointing in general tableaux. *Journal of Logic and Computation*, 2010. To appear.
7. F. Baader, R. Peñaloza, and B. Suntisrivaraporn. Pinpointing in the description logic $\mathcal{EL}^+$. In *Proc. of the 30th German Conf. on Artificial Intelligence (KI2007)*, volume 4667 of *LNAI*, pages 52–67. Springer-Verlag, 2007.
8. A. Borgida, D. Calvanese, and M. Rodriguez-Muro. Explanation in DL-Lite. In *Proc. of the 2008 Int. Workshop on Description Logics (DL 2008)*, volume 353 of *CEUR-WS*, 2008.
9. A. Borgida, D. Calvanese, and M. Rodriguez-Muro. Explanation in the DL-Lite family of Description Logics. In *Proc. of the 7th Int. Conf. on Ontologies, DataBases, and Applications of Semantics (ODBASE 2008)*, volume 5332 of *LNCS*, pages 1440–1457. Springer-Verlag, 2008.
10. 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.
11. T. Eiter and G. Gottlob. Identifying the minimal transversals of a hypergraph and related problems. Technical Report CD-TR 91/16, TU Vienna, 1991.
12. T. Eiter, K. Makino, and G. Gottlob. Computational aspects of monotone dualization: A brief survey. *Discrete Applied Mathematics*, 156(11):2035–2049, 2008.
13. M. Horridge, B. Parsia, and U. Sattler. Laconic and precise justifications in OWL. In *Proc. of the 7th Int. Semantic Web Conf., (ISWC 2008)*, volume 5318 of *LNCS*, pages 323–338. Springer-Verlag, 2008.
14. M. Horridge, B. Parsia, and U. Sattler. Explaining inconsistencies in OWL ontologies. In *Proc. of the Third Int. Conf. on Scalable Uncertainty Management, (SUM 2009)*, volume 5785 of *LNCS*, pages 124–137. Springer-Verlag, 2009.
15. D. S. Johnson, M. Yannakakis, and C. H. Papadimitriou. On generating all maximal independent sets. *Information Processing Letters*, 27(3):119–123, 1988.
16. A. Kalyanpur, B. Parsia, M. Horridge, and E. Sirin. Finding all justifications of OWL DL entailments. In *Proc. of the 6th Int. Semantic Web Conf., 2nd Asian Semantic Web Conf., (ISWC 2007 + ASWC 2007)*, volume 4825 of *LNCS*, pages 267–280. Springer-Verlag, 2007.
17. T. Meyer, K. Lee, R. Booth, and J. Z. Pan. Finding maximally satisfiable terminologies for the description logic $\mathcal{ALC}$. In *Proc. of the 21st Natonal Conf. on Artificial Intelligence (AAAI 2006)*, pages 269–274. AAAI Press/The MIT Press, 2006.
18. L. A. Nielsen, K. A. Andersen, and D. Pretolani. Finding the K shortest hyperpaths. *Computers and Operations Research*, 32(6):1477-1497, 2005.
19. B. Parsia, E. Sirin, and A. Kalyanpur. Debugging OWL ontologies. In *Proc. of the 14th international conference on World Wide Web (WWW 2005)*, pages 633–640. ACM, 2005.
20. R. Peñaloza and B. Sertkaya. Axiom pinpointing is hard. In *Proc. of the 2009 Int. Workshop on Description Logics (DL2009)*, volume 477 of *CEUR-WS*, 2009.

21. R. Peñaloza and B. Sertkaya. On the complexity of axiom pinpointing in Description Logics. LTCS-Report LTCS-09-04, Chair for Automata Theory, Institute for Theoretical Computer Science, Dresden University of Technology, Germany, 2009. See http://lat.inf.tu-dresden.de/research/reports.html.

22. R. Peñaloza and B. Sertkaya. On the complexity of axiom pinpointing in the $\mathcal{EL}$ Family of Description Logics. In *Proc. of the Twelfth Int. Conf. on Principles and Knowledge Representation and Reasoning (KR-10)*. Morgan Kaufmann, 2010. To appear.

23. R. Reiter. A theory of diagnosis from first principles. *Artificial Intelligence*, 32(1):57–95, 1987.

24. S. Schlobach and R. Cornet. Non-standard reasoning services for the debugging of description logic terminologies. In *Proc. of the Eighteenth Int. Joint Conf. on Artificial Intelligence (IJCAI'03)*, pages 355–362. Morgan Kaufmann, 2003.

25. J. Y. Yen. Finding K shortest loopless paths in a network. *Management Science*, 17(11):712–716, 1971.

Distributed Island-based Query Answering for Expressive Ontologies

Sebastian Wandelt and Ralf Moeller

Hamburg University of Technology, Hamburg, Germany,
`wandelt@tuhh.de`, `r.f.moeller@tuhh.de`

Abstract. Scalability of reasoning systems is one of the main criteria which will determine the success of Semantic Web systems in the future. The focus of recent work is either on (a) expressive description logic systems which rely on in-memory structures or (b) not-so-expressive ontology languages, which can be dealt with by using database technologies. In this paper we introduce a method to perform query answering for semi-expressive ontologies without the limit of in-memory structures. Our main idea is to compute small and characteristic representations of the assertional part of the input ontology. Query answering is then more efficiently performed over a reduced set of these small representations. We show that query answering can be distributed in a network of description logic reasoning systems in order to support scalable reasoning. Our initial results are encouraging.

1 Introduction

As the Semantic Web evolves, scalability of inference techniques becomes increasingly important. While in recent years the focus was on pure terminological reasoning, the interest shifts now more to reasoning with respect to large assertional parts, e.g. in the order of millions or billions of triples. Research on ontologies with medium-sized assertional information has already been conducted on less expressive description logics, e.g. in [CDGL$^+$05]. Further techniques were investigated in [FKM$^+$06]. The authors propose to extract a condensed summary graph out of the assertional part of an ontology, and then perform reasoning on that summary. [FKM$^+$06] reports encouraging performance results. However, for avoiding inconsistencies due to merging, the summaries have to be rewritten in expensive *query-dependent* refinement steps. With increasing numbers of refinement steps necessary, the performance of the approach degrades [DFK$^+$09]. Moreover, the technical criteria for summarization (creating representative nodes by grouping concept sets), seems arbitrary. In [WM08], a method is proposed to identify the relevant *islands*, i.e. set of assertions/information, required to reason about a given individual. The main motivation is to enable in-memory reasoning over ontologies with a large ABox, for traditional tableau-based reasoning systems.

Given the island of an individual, we will make the idea of summarization more formal. In this paper we present an approach to execute efficient instance
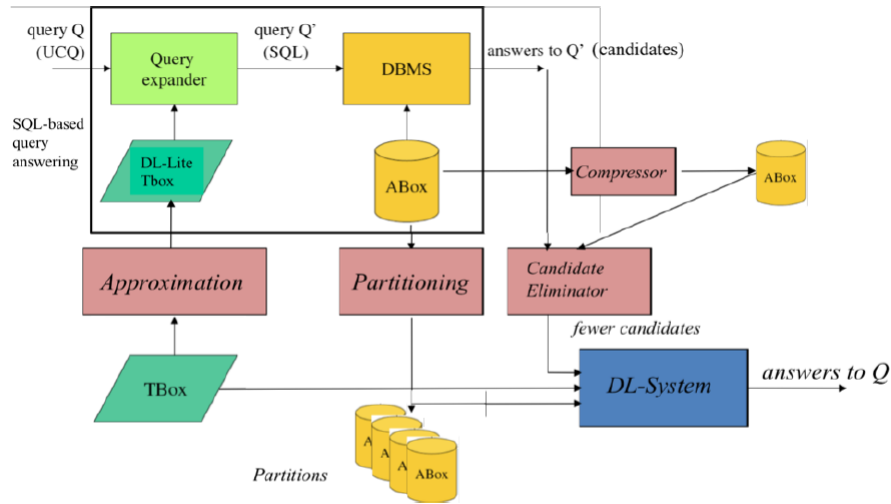
Fig. 1. Efficient query answering for expressive description logics ([KMWW08])

retrieval tests on database-oriented ontologies. The main insight of our work is that the islands computed in [WM08] can be checked for similarity and instance retrieval can then be performed over equivalence classes of similar islands. The query answering algorithm for instance retrieval over similar islands is implemented in a distributed manner. We report interesting scalability results with respect to our test ontology: increasing the number of nodes in the network by the factor of n almost reduces the query answering time to $\frac{1}{n}$. Moreover, we implemented our algorithm in such a way that the input ontology can be loaded in an offline phase and changed afterwards incrementally online.

Figure 1 is taken from [KMWW08] and shows the general structure of an optimized query answering system for expressive description logics. Let us assumed that a query Q is to be answered w.r.t. a Tbox (lower-left corner). The Tbox is approximated into a DL-Lite Tbox (a complete but unsound approximation is use). Exploiting DL-Lite query answering by transforming queries w.r.t. ontologies (Tboxes) into SQL queries (see the rectangle in the left-upper corner) [CDGL$^+$05], candidates are generated for the query w.r.t. the original Tbox. Afterwards, candidates can possibly be eliminated in beforehand. The remaining candidates must be investigated using a reasoner for the expressive Tbox. In order to be able to do this, huge Aboxes are partitioned (as explained above). The approach described in this paper is situated in the modules *Partitioning* and *Candidate Eliminator*.

The remaining parts of the paper are structured as follows. Section 2 introduces necessary formal notions and gives an overview over related work. In Section 3 we adopt the underlying island computation process from [WM08]. The main theoretical contribution of our work is in Section 4, the isomorphism

criteria for islands. We show our implementation in Section 5 and provide initial evaluation results in Section 6. The paper is concluded in Section 7.

There is an extended version of this paper available with proofs and further comments on the implementation and evaluation [WM10].

2 Preliminaries

For details about syntax and semantics of the description logic $\mathcal{ALCHI}$ we refer to [BCM$^+$07]. Some definitions are appropriate to explain our nomenclature, however. We assume a collection of disjoint sets: a set of *concept names* N_{CN}, a set of *role names* N_{RN} and a set of *individual names* N_I. The *set of roles* N_R is $N_{RN} \cup \{R^- | R \in N_{RN}\}$. We say that a concept description is *atomic*, if it is a concept name or its negation. With $\mathcal{S}_{AC}$ we denote all atomic concepts.

Furthermore we assume the notions of TBoxes ($\mathcal{T}$), RBoxes ($\mathcal{R}$) and ABoxes ($\mathcal{A}$) as in [BCM$^+$07]. An *ontology* $\mathcal{O}$ is a tuple $\langle \mathcal{T}, \mathcal{R}, \mathcal{A} \rangle$, where $\mathcal{T}$ is a TBox, $\mathcal{R}$ is a RBox and $\mathcal{A}$ is a ABox. We restrict the concept assertions in $\mathcal{A}$ to only use atomic concepts. This is a common assumption, e.g. in [GH06], when dealing with large assertional datasets stemming from databases. With $Ind(\mathcal{A})$ we denote the set of individuals occurring in $\mathcal{A}$. Throughout the remaining part of the paper we assume the Unique Name Assumption (UNA), i.e. two distinct individual names denote distinct domain objects.

In Example 1 we define an example ontology, used throughout the remaining part of the paper to explain definitions. The example ontology is in the setting of universities. We evaluate our ideas w.r.t. to "full" LUBM [GPH05] (in fact LUBM without the transitive role $subOrganzationOf$, because we handle $\mathcal{ALCHI}$) in Section 6. Although this is a synthetic benchmark, several (if not most) papers on scalability of ontological reasoning consider it as a base reference.

Example 1. Let $\mathcal{O}_{EX1} = \langle \mathcal{T}_{EX1}, \mathcal{R}_{EX1}, \mathcal{A}_{EX1} \rangle$, s.t.

$$
\begin{aligned}
\mathcal{T}_{EX1} =& \{Chair \equiv (\exists headOf.Department) \sqcap Person, Prof \sqsubseteq Person, \\
& GraduateCourseTeacher \equiv Prof \sqcap \exists teaches.GraduateCourse\} \\
\mathcal{R}_{EX1} =& \{headOf \sqsubseteq worksFor\} \\
\mathcal{A}_{EX1} =& \texttt{see Figure 2}
\end{aligned}
$$

Next we discuss related work relevant to our contribution. In [SP08], the authors discuss a general approach to partition OWL knowledge bases and distribute reasoning over partitions to different processors/nodes. The idea is that the input for their partitioning algorithm is a fixed number of desired partitions, which can be calculated by different means (weighted graphs, hash-based distribution or domain specific partitions). The partitions are not independent from each other. Moreover, in some cases, the data is just arbitrarily spread over the different nodes in the networks. This leads to a noticeable amount of communication overhead between the nodes because partial results have to be passed in between the nodes. The authors discuss rather small data sets,
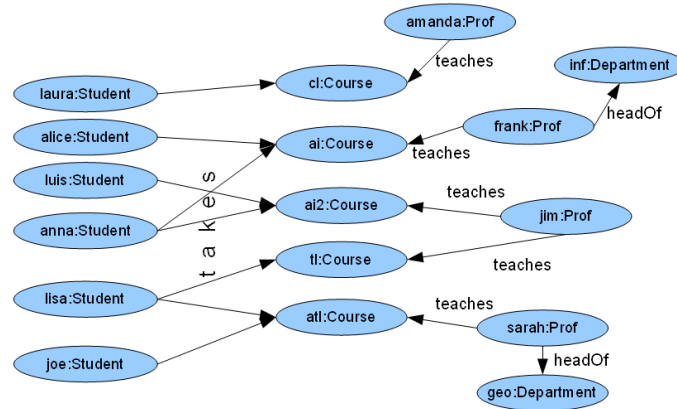
Fig. 2. Example ABox $\mathcal{A}_{EX1}$

e.g., 1 million triples. These problems can already be solved with state-of-the-art tableau-based reasoning systems. Furthermore, their evaluation only talks about speed-up, without mentioning the actual run-time, or referring to some open/state-of-the art implementation.

The work in [UKOvH09] proposes a technique based on MapReduce [DG04] to compute the closure (set of all implications) over ontologies (in the spirit of Abox realization). Given the underlying MapReduce framework, their approach could scale in theory. The major difference to our work is that we focus on query answering instead of brute force (bottom-up) generation of all possible implications of a given knowledge base. Moreover we focus on more expressive description logics and it is at least doubtful, whether their approach will work for expressive logics, i.e., logics allowing for disjunctions or the specification of ontologies which have only infinite models.

The authors of [BS03] discuss an approach to integrate ontologies from different sources in a distributed setting. They introduce so-called bridge-rules to identify, which parts of the ontologies overlap (and thus need to be communicated between the different reasoning nodes). The main focus of their work is rather on the integration of distributed ontologies, but not on scalable query answering over large ontologies. There is additional work on distributed Datalog implementations (see, e.g., [ZWC95] and [GST90]) and on non-distributed reasoning optimization techniques for description logics [GH06].

3   Island calculation

In [WM08], a method is proposed to identify the relevant information (assertions) to reason about an individual. The main motivation is to enable in-memory reasoning over large ontologies, i.e. ontologies with a large ABox, for traditional

tableau-based reasoning systems. More formally, given an input individual a, the proposal is to compute a set of ABox assertions $\mathcal{A}_{isl}$ (a subset of the source ABox $\mathcal{A}$), such that for all concept descriptions C, we have $\langle \mathcal{T}, \mathcal{R}, \mathcal{A} \rangle \vDash C(a)$ iff $\langle \mathcal{T}, \mathcal{R}, \mathcal{A}_{isl} \rangle \vDash C(a)$. We call these sets of assertions *islands*. Despite the fact that query answering is , in order to support more complex reasoning tasks, e.g., answering conjunctive queries, island computation as described in [WM08] is not enough. Given an instance retrieval task for concept C with respect to ontology $\mathcal{O} = \langle \mathcal{T}, \mathcal{R}, \mathcal{A} \rangle$, a naive approach will iterate over all individuals $a \in Ind(\mathcal{A})$ of the input ABox in order to determine whether $\langle \mathcal{T}, \mathcal{R}, \mathcal{A} \rangle \vDash C(a)$. If yes, then a is included in the result set for the instance retrieval query.

The performance of instance retrieval queries in [WM08] depends highly on the number of individuals in the ABox. For 100 universities, we have around 300.000 individuals, i.e., 300.000 islands. If we assumed that one instance retrieval check takes around 1 ms, we would need already 1 hour to answer one instance retrieval query on a single machine. If one intended to improve query answering times using a MapReduce approach, one could parallelize the island checks. For example, in the best case, one would need to have 3600 machines at ones disposal to obtain all answers within one second. If the average instance checking time is higher, or the number of individuals is larger (= more universities), then the situation becomes even worse. Thus, our motivation is to further improve instance retrieval time, while still supporting expressive logics.

4 Similarity of Islands

In the following, we discuss how islands can be used for optimized instance retrieval tests. The main insight is that many of the computed islands are similar to each other. Especially in database-oriented scenarios, ontologies contain a lot of individuals following patterns defined by a schema (the terminology of the ontology). If it is possible to define a formal notion of similarity for islands, and to show that it is sufficient to perform reasoning over one representative island instead, then query answering can potentially be increased by several orders of magnitude (depending on the number of dissimilar island classes). We consider an example to demonstrate the idea of island similarities.

In Figure 3 we show the extracted islands of all professors in our example ontology $\mathcal{O}_{EX1}$. While all four graphs are different, they have some similarities in common, and this can be exploited to optimize reasoning over these islands. To define similarities over islands, we formally introduce the notion of an island and define the similarity criterion.

Definition 1. *A individual-island-graph IIG is a tuple $\langle N, \phi_n, \phi_e, root \rangle$, such that*
- *N is a set of nodes,*
- *$\phi_n : N \rightarrow 2^{\mathcal{S}_{AC}}$ is a node-labeling function ($\mathcal{S}_{AC}$ is the set of atomic concepts),*
- *$\phi_e : N \times N \rightarrow 2^{L_e}$ is a edge-labeling function*
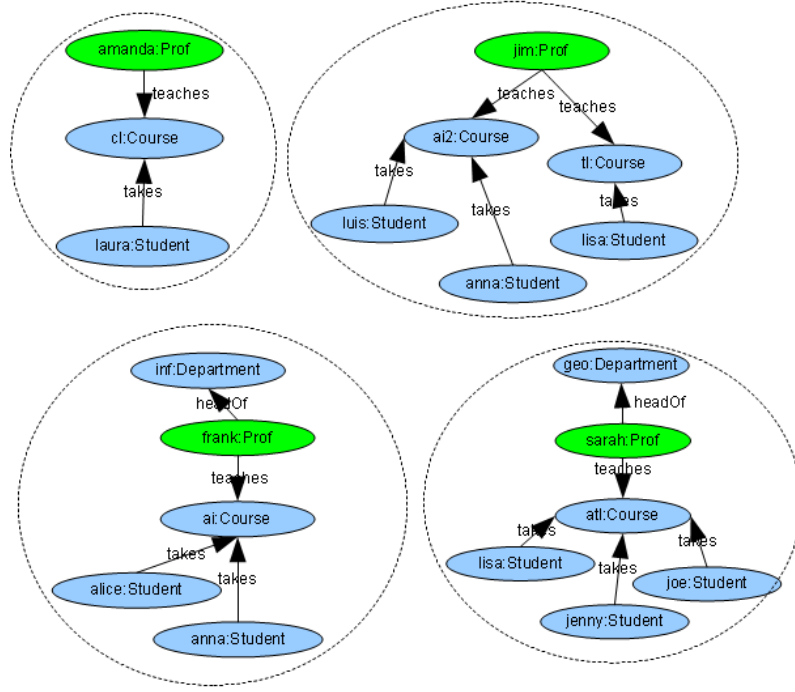
Fig. 3. Example: Islands of the four Professors in $\mathcal{O}_{EX1}$

- *root $\in N$ is a distinguished root node.*

If we have $\phi_e(a,b) = \rho$ and $\rho \neq \emptyset$, then we write $a \xrightarrow{\rho}_{IIG} b$. The definition of individual-island-graphs is quite straight-forward. In the following we define a similarity relation over two individual-island-graphs, based on graph bisimulations. Although the term *bisimulation* is usually used in process algebra to define similar processes, we use it here in the context of graphs.

Definition 2. *A* bisimulation *over $IIG_1 = \langle N_{IIG_1}, \phi_{n\,IIG_1}, \phi_{e\,IIG_1}, root_{IIG_1}\rangle$ and $IIG_2 = \langle N_{IIG_2}, \phi_{n\,IIG_2}, \phi_{e\,IIG_2}, root_{IIG_2}\rangle$ is a binary relation $R_{IIG_1,IIG_2} \subseteq N_{IIG_1} \times N_{IIG_2}$, such that*
- *$R_{IIG_1,IIG_2}(root_{IIG_1}, root_{IIG_2})$*
- *if $R_{IIG_1,IIG_2}(a,b)$ then $\phi_{n\,IIG_1}(a) = \phi_{n\,IIG_2}(b)$*
- *if $R_{IIG_1,IIG_2}(a,b)$ and $a \xrightarrow{\rho}_{IIG_1} a'$ then there exists a $b' \in N_{IIG_2}$ with $b \xrightarrow{\rho}_{IIG_2} b'$ and $R_{IIG_1,IIG_2}(a',b')$*
- *if $R_{IIG_1,IIG_2}(a,b)$ and $b \xrightarrow{\rho}_{IIG_2} b'$ then there exists a $a' \in N_{IIG_1}$ with $a \xrightarrow{\rho}_{IIG_2} a'$ and $R_{IIG_1,IIG_2}(a',b')$*

Definition 3. *Two individual-island-graphs IIG_1 and IIG_2 are called* bisimilar, *if there exists a bisimulation R for them.*

Example 2. To illustrate these definitions we show individual-island-graphs for $amanda$, jim, and $frank$, together with a possible bisimulation between $amanda$ and jim:

- $IIG_{amanda} = \langle N_{amanda}, \phi_{n\,amanda}, \phi_{e\,amanda}, root_{amanda} \rangle$, s.t.

$$N_{amanda} = \{x_{amanda}, x_{cl}, x_{laura}\}$$
$$\phi_{n\,amanda} = \{x_{amanda} \to \{Prof\}, x_{cl} \to \{Course\}, x_{laura} \to \{Student\}\}$$
$$\phi_{e\,amanda} = \{(x_{amanda}, x_{cl}) \to \{teaches\}, (x_{laura}, x_{cl}) \to \{takes\}\}$$
$$root_{amanda} = \{x_{amanda}\}$$

- $IIG_{jim} = \langle N_{jim}, \phi_{n\,jim}, \phi_{e\,jim}, root_{jim} \rangle$, s.t.

$$N_{jim} = \{y_{jim}, y_{ai2}, y_{tl}, y_{luis}, y_{anna}, y_{lisa}\}$$
$$\phi_{n\,jim} = \{y_{jim} \to \{Prof\}, y_{ai2} \to \{Course\}, y_{tl} \to \{Course\}, y_{luis} \to \{Student\}, ...\}$$
$$\phi_{e\,jim} = \{(y_{jim}, y_{ai2}) \to \{teaches\}, (y_{jim}, y_{tl}) \to \{teaches\}, (y_{luis},$$
$$x_{ai2}) \to \{takes\}, ...\}$$
$$root_{jim} = \{y_{jim}\}$$

- $IIG_{frank} = \langle N_{frank}, \phi_{n\,frank}, \phi_{e\,frank}, root_{frank} \rangle$, s.t.

$$N_{frank} = \{z_{frank}, z_{ai}, z_{inf}, z_{alice}, z_{anna}\}$$
$$\phi_{n\,frank} = \{z_{frank} \to \{Prof\}, z_{ai} \to \{Course\}, z_{inf} \to \{Department\},$$
$$z_{alice} \to \{Student\}, z_{anna} \to \{Student\}\}$$
$$\phi_{e\,frank} = \{(z_{frank}, z_{ai}) \to \{teaches\}, (z_{frank}, z_{inf}) \to \{headOf\},$$
$$(z_{alice}, z_{ai}) \to \{takes\}, (z_{anna}, z_{ai}) \to \{takes\}\}$$
$$root_{jim} = \{z_{jim}\}$$

- $R_{jim,amanda} =$

$$\{(x_{amanda}, y_{jim}), (x_{cl}, y_{ai2}), (x_{cl}, y_{tl}), (x_{laura}, y_{luis}),$$
$$(x_{anna}, y_{lisa})\}$$

It is easy to see that $R_{jim,amanda}$ is a bisimulation for the islands (graphs) of the individuals jim and $amanda$. Furthermore, it is easy to see that there cannot be a bisimulation, for instance, between jim and $frank$.

The important insight is that bisimilar islands entail the same concept sets for their root individual if the underlying description logic is restricted to $ALCHI$. This is shown in the following theorem.

Theorem 1. *Given two individuals a and b and any concept description C, it holds that $\langle \mathcal{T}, \mathcal{R}, ISLAND(a) \rangle \vDash C(a) \iff \langle \mathcal{T}, \mathcal{R}, ISLAND(b) \rangle \vDash C(b)$ if there exists a bisimulation $R_{a,b}$, for $ISLAND(a)$ and $ISLAND(b)$.*

For the proof see [WM10]. The above theorem can be easily lifted to the case of more than two individuals, i.e. if we have n individuals, and for all of their islands one can find a bisimilarity relation, it is sufficient to perform instance checking on one island. In practice, especially in database-oriented ontologies, this can dramatically speed up the time for instance retrieval. To show this, we need to further introduce some mathematical notions.

Definition 4. *An* individual-island-equivalence $\sim_{ISL}$ *is an equivalence relation over individual islands, such that we have* $\sim_{ISL} (ISL_1, ISL_2)$ *if we can find a bisimulation* R_{ISL_1, ISL_2} *between the two islands* ISL_1 *and* ISL_2. *With* $[\sim_{ISL}]$ *we denote the set of equivalence classes of* $\sim_{ISL}$.

The main theoretical result of our work is summarized in the following theorem.

Theorem 2. *Given an ontology* $\langle \mathcal{T}, \mathcal{R}, \mathcal{A} \rangle$, *one can perform grounded instance retrieval for the atomic concept* C *over* $[\sim_{ISL}]$.

For details see [WM10]. Please note that our approach does not work directly for more expressive description logics, e.g. $\mathcal{SHOIQ}$. In the presence of cardinality restrictions we will need more sophisticated bisimulation criteria to identify similar nodes, since the number of related similar individuals matters. Nominals can further complicate the bisimulation criteria, since individuals can be forced by the terminological axioms to refer to the same domain object, i.e. one might need to compute all TBox consequences in the worst case.

5   Distributed Implementation

We have implemented our proposal for Island Simulations in Java. For ontology access we use the OWLAPI 2.2.0 [BVL03]. The general structure of our implementation, a description of each component, and additional performance optimization insights can be found in [WM10]. Here we only give a short overview on the modules.

- (Server) OWL-Converter: converts OWL data to an internal representation
- (Server) Update Handler: determines changed islands in case of ontology updates
- (Server) Island Computer: computes the island for a given individual and performs similarity computation
- *(Server) Node Scheduler*: determines the responsible node for the island: Round-Robin / capability-based
- (Server) TBox/ABox Storage: terminological/assertional part of the ontology.
- (Client) Query Manager: determines all active islands and uses the DL Reasoner module to find out which islands match the input query.
- (Client) DL Reasoner: implements an interface to a general description logic reasoner (in our case we used RacerPro [HM03]).

One more remark on our implementation should be pointed out here: While loading an ontology we built a dependency tree for storing which impact updates on particular islands have, e.g., we store that if we add a *teaches*-relation to a particular individual (island), we obtain another fixed island. This kind of lookup-table greatly improved the performance in our tests because we do not have to recompute complete islands in case there were similar updates before. As usual, in order to obtain optimal performances, many details have to be appropriately handled.
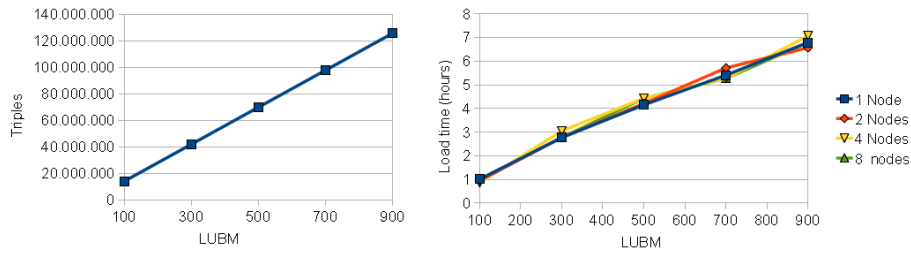
Fig. 4. Input size and load time (number of universities on the x-axis)

6 Evaluation

Our tests were run with respect to the synthetic benchmark ontology LUBM [GPH05]. Although some people claim that LUBM does not fully represent all the capabilities provided by the complete OWL specification, we think that it fits our constraint of database-oriented ontologies: rather small and simple TBox, but a bulk of assertional information with a realistic distribution with respect to numbers of professors, students, departments, etc. In our evaluation we compare three different measures to determine the performance of our implementation:

- *Load time*: In Figure 4 we show the size of the assertional part in triples and compare the load time with different number of nodes in our network (1, 2 and 4 nodes). The load time only represents the time spent to traverse the input ontology once in order to compute the bisimilarity relation over all islands of all individuals. It can be seen that the load time increases linearly with the number of triples in the assertional part. Please note that our loading algorithm is designed and implemented as an incremental algorithm. Thus, if we add a new assertion, we do not have to recompute all internal structures, but only update relevant structures.
- *Preparation time*: This measure indicates an initial preparation time after the ontology is fully loaded. The time is spent to prepare the internal structures of the DL reasoner for incoming queries. Please note that this preprocessing step is independent of the query and only performed once after the ontology was updated. The idea is that we can perform incremental bulk loading (measured in *load time*) without updating the (expensive) internal structures of the DL reasoner all the time.

  In the left part of Figure 5, we show the query preparation time for different numbers of universities and different numbers of nodes in the network. The number of nodes indeed affects the query preparation time. If we use 8 nodes, the preparation time is almost $\frac{1}{8}$ of the time needed for one node. Thus, the distribution of computational power works for query preparation.

  In the right part of Figure 5 we indicate the necessary number of islands to perform instance retrieval with the original work in [WM08]. The number
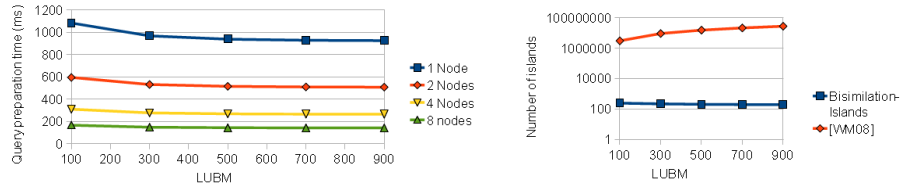
Fig. 5. Query preparation time and island count

of islands increases linearly with the size of the input ontology for [WM08] (please note the logarithmic scale). Using bisimulation, the number of islands is almost constant for all input ontologies, since most of the newly introduced individual-islands are bisimilar to each other, e.g., professors who teach particular students in particular kinds of courses.

– *Query answering time*: The third measure indicates how long the actual query answering process takes. In Figure 6, the query answering time (for instance retrieval) for the concepts *Chair* (small number of answers, linearly growing with the number of universities) are shown. Please note that query answering times are rather independent from the chosen concept description for instance retrieval. We only focus on *Chair*, since it is also commonly used in the literature to perform benchmarks on LUBM because instances of *Chair* are not syntactically identifiable. In Figure 6 we show the time needed to identify the islands which entail instances of the concept *Chair*. This is the actual description-logic-hard task. In addition one needs to lookup all individuals for the given islands, which is a database-dominated task and usually takes linear time.
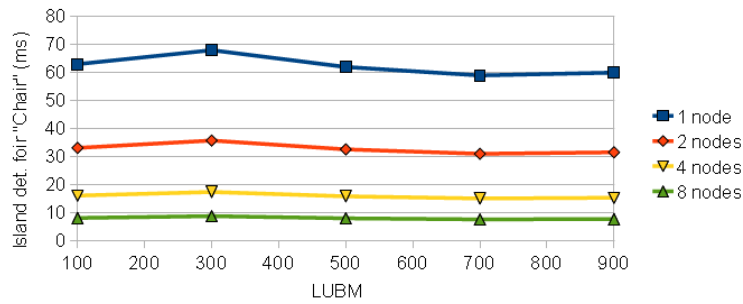
Fig. 6. Query answering time

7 Conclusions

We have proposed a method for instance retrieval over ontologies in a distributed system of DL reasoners. To the best of our knowledge, we are the first to propose instance retrieval reasoning based on similarity of individual-islands. The results are encouraging so far. We emphasize that our approach especially works for ontologies with a rather simple or average size terminological part. For future work, it will be important to investigate more ontologies and check the performance of our proposal. Furthermore, we want to extend our proposal to more expressive description logics, e.g. SHIQ or even SHOIQ.

References

[BCM+07] Franz Baader, Diego Calvanese, Deborah L. McGuinness, Daniele Nardi, and Peter F. Patel-Schneider. *The Description Logic Handbook*. Cambridge University Press, New York, NY, USA, 2007.

[BS03] Alex Borgida and Luciano Serafini. Distributed description logics: Assimilating information from peer sources. *J. of Data Semantics*, 1:153–184, 2003.

[BVL03] S. Bechhofer, R. Volz, and P. Lord. Cooking the Semantic Web with the OWL API, 2003.

[CDGL+05] Diego Calvanese, Giuseppe De Giacomo, Domenico Lembo, Maurizio Lenzerini, and Riccardo Rosati. DL-Lite: Tractable description logics for ontologies. In *Proc. of the 20th Nat. Conf. on Artificial Intelligence (AAAI 2005)*, pages 602–607, 2005.

[DFK+09] Julian Dolby, Achille Fokoue, Aditya Kalyanpur, Edith Schonberg, and Kavitha Srinivas. Scalable highly expressive reasoner (SHER). *Web Semant.*, 7(4):357–361, 2009.

[DG04] Jeffrey Dean and Sanjay Ghemawat. MapReduce: Simplified data processing on large clusters. In *OSDI 2004*, pages 137–150, 2004.

[FKM+06] Achille Fokoue, Aaron Kershenbaum, Li Ma, Edith Schonberg, and Kavitha Srinivas. The summary abox: Cutting ontologies down to size. *The Semantic Web - ISWC 2006*, pages 343–356, 2006.

[GH06] Yuanbo Guo and Jeff Heflin. A scalable approach for partitioning OWL knowledge bases. In *SSWS 2006*, Athens, GA, USA, November 2006.

[GPH05] Yuanbo Guo, Zhengxiang Pan, and Jeff Heflin. LUBM: A benchmark for OWL knowledge base systems. *J. Web Sem.*, 3(2-3):158–182, 2005.

[GST90] Sumit Ganguly, Avi Silberschatz, and Shalom Tsur. A framework for the parallel processing of datalog queries. *SIGMOD Rec.*, 19(2):143–152, 1990.

[HM03] V. Haarslev and R. Möller. Racer: A core inference engine for the semantic web. In *Proceedings of the 2nd International Workshop on Evaluation of Ontology-based Tools (EON2003), located at the 2nd International Semantic Web Conference ISWC 2003, Sanibel Island, Florida, USA, October 20*, pages 27–36, 2003.

[KMWW08] Alissa Kaplunova, Ralf Möller, Sebastian Wandelt, and Michael Wessel. Approximation and ABox Segmentation. Technical report, Institute for Software Systems (STS), Hamburg University of Technology, Germany, 2008. See http://www.sts.tu-harburg.de/tech-reports/papers.html.

[SP08]   Ramakrishna Soma and V. K. Prasanna. Parallel inferencing for OWL knowledge bases. In *ICPP '08: Proceedings of the 2008 37th International Conference on Parallel Processing*, pages 75–82, Washington, DC, USA, 2008. IEEE Computer Society.

[UKOvH09] Jacopo Urbani, Spyros Kotoulas, Eyal Oren, and Frank van Harmelen. Scalable distributed reasoning using MapReduce. In *8th International Semantic Web Conference (ISWC2009)*, October 2009.

[WM08]   Sebastian Wandelt and Ralf Möller. Island reasoning for ALCHI ontologies. In *Proceedings of the 2008 conference on Formal Ontology in Information Systems*, pages 164–177, Amsterdam, The Netherlands, 2008. IOS Press.

[WM10]   Sebastian Wandelt and Ralf Möller. Distributed island-based query answering for expressive ontologies. Technical report, Institute for Software Systems (STS), Hamburg University of Technology, Germany, 2010. See http://www.sts.tu-harburg.de/tech-reports/papers.html.

[ZWC95]   Weining Zhang, Ke Wang, and Siu-Cheung Chau. Data partition and parallel evaluation of datalog programs. *IEEE Transactions on Knowledge and Data Engineering*, 7(1):163–176, 1995.

Paraconsistent Description Logics Revisited

Norihiro Kamide

Waseda Institute for Advanced Study, Waseda University,
1-6-1 Nishi Waseda, Shinjuku-ku, Tokyo 169-8050, JAPAN.
logician-kamide@aoni.waseda.jp

Abstract. Inconsistency handling is of growing importance in Knowledge Representation since inconsistencies may frequently occur in an open world. Paraconsistent (or inconsistency-tolerant) description logics have been studied by several researchers to cope with such inconsistencies. In this paper, a new paraconsistent description logic, $\mathcal{PALC}$, is obtained from the description logic $\mathcal{ALC}$ by adding a paraconsistent negation. Some theorems for embedding $\mathcal{PALC}$ into $\mathcal{ALC}$ are proved, and $\mathcal{PALC}$ is shown to be decidable. A tableau calculus for $\mathcal{PALC}$ is introduced, and the completeness theorem for this calculus is proved.

1 Introduction

Inconsistency handling is of growing importance in Knowledge Representation since inconsistencies may frequently occur in an open world. *Paraconsistent (or inconsistency-tolerant) description logics* have been studied by several researchers [5–8, 11–13, 16, 18, 19] to cope with such inconsistencies.

However, the existing paraconsistent description logics have no good compatibility with the standard description logics such as $\mathcal{ALC}$ [15] etc. in the following sense:

1. these paraconsistent description logics are not a straightforward extension of the standard ones,
2. some paraconsistent description logics have no translation into a standard description logic.

Such compatibility is important to adopt and re-use the existing applications and algorithms for the standard description logics. A translation or reduction of a paraconsistent description logic into a standard description logic is especially important for such a compatibility issue [5, 6].

The aim of this paper is thus to introduce a compatible paraconsistent description logic which is a straightforward extension of $\mathcal{ALC}$ and is also embeddable into $\mathcal{ALC}$. To construct such a compatible paraconsistent description logic, some merits of some existing paraconsistent description logics are adopted and combined.

Some examples of studies of paraconsistent description logics are presented as follows. An *inconsistency-tolerant four-valued terminological logic* was originally

introduced by Patel-Schneider [13], three *inconsistency-tolerant constructive description logics*, which are based on intuitionistic logic, were studied by Odintsov and Wansing [11, 12], some *paraconsistent four-valued description logics* including $\mathcal{ALC}4$ were studied by Ma et al. [5, 6], some *quasi-classical description logics* were developed by Zhang et al. [18, 19], a sequent calculus for reasoning in four-valued description logics was introduced by Straccia [16], and an application of four-valued description logic to information retrieval was studied by Meghini et al. [7, 8].

The logic $\mathcal{ALC}4$ [5] has a good translation into $\mathcal{ALC}$, and using this translation, the satisfiability problem for $\mathcal{ALC}4$ is shown to be decidable. However, $\mathcal{ALC}4$ and its variations have no classical negation (or complement), i.e., these logics are not an extension of the standard description logics. The quasi-classical description logics [18, 19] have the classical negation, i.e., these logics are regarded as extensions of the standard description logics. However, translations of quasi-classical description logics into the corresponding standard description logics have not been proposed yet.

The paraconsistent description logic proposed in this paper supports both the merits of $\mathcal{ALC}4$ and the quasi-classical description logics, i.e., it has the translation and the classical negation. Moreover, a simple dual-interpretation semantics is used in the proposed logic. Such a dual-interpretation semantics is taken over from the dual-consequence Kripke-style semantics for *Nelson's paraconsistent four-valued logic with strong negation* N4 [1, 9].

A description logic (called $\mathcal{ALC}_{\sim}^{n}$) with such a dual (or multiple)-interpretation semantics was introduced and studied by Kaneiwa [4] to deal with a negation issue, but not to deal with an issue of inconsistency handling. The logic $\mathcal{ALC}_{\sim}^{n}$ is a natural extension of $\mathcal{ALC}$, and $\mathcal{ALC}_{\sim}^{n}$ is shown to be decidable (w.r.t. the concept satisfiability problem) and complete (w.r.t. a tableau calculus). But, $\mathcal{ALC}_{\sim}^{n}$ is not paraconsistent, and a translation into $\mathcal{ALC}$ has not been proposed yet. The present paper is based on the spirit of $\mathcal{ALC}_{\sim}^{n}$ for dual (or multiple)-interpretation semantics.

The contents of this paper are then summarized as follows. A new paraconsistent description logic, $\mathcal{PALC}$, is obtained from $\mathcal{ALC}$ by adding a paraconsistent negation similar to the strong negation in Nelson's N4. A *semantical embedding theorem* of $\mathcal{PALC}$ into $\mathcal{ALC}$ is shown by constructing a standard single-interpretation of $\mathcal{ALC}$ from a paraconsistent dual-interpretation of $\mathcal{PALC}$, and vice versa. By using this embedding theorem, the concept satisfiability problem for $\mathcal{PALC}$ is shown to be decidable. The complexity of the decision procedure for $\mathcal{PALC}$ is also shown to be the same complexity as that of $\mathcal{ALC}$. Next, a tableau calculus, $\mathcal{TPALC}$ (for $\mathcal{PALC}$), is introduced, and a *syntactical embedding theorem* of this calculus into a tableau calculus, $\mathcal{TALC}$ (for $\mathcal{ALC}$), is proved. The completeness theorem for $\mathcal{TPALC}$ is proved by combining both the semantical and syntactical embedding theorems. A comparision of $\mathcal{PALC}$ and other paraconsistent description logics is explained.

2 Paraconsistent Description Logic

In this section, firstly, we present a semantical definition of $\mathcal{ALC}$, and secondly, we introduce $\mathcal{PALC}$ by extending $\mathcal{ALC}$ with a paraconsistent negation.

2.1 $\mathcal{ALC}$

The $\mathcal{ALC}$-language is constructed from atomic concepts, atomic roles, $\sqcap$ (intersection), $\sqcup$ (union), $\neg$ (classical negation or complement), $\forall R$ (universal concept quantification) and $\exists R$ (existential concept quantification). We use the letters A and A_i for atomic concepts, the letter R for atomic roles, and the letters C and D for concepts.

Definition 1 *Concepts C are defined by the following grammar:*

$$C ::= A \mid \neg C \mid C \sqcap C \mid C \sqcup C \mid \forall R.C \mid \exists R.C$$

Definition 2 *An* interpretation $\mathcal{I}$ *is a pair* $\langle \Delta^{\mathcal{I}}, \cdot^{\mathcal{I}} \rangle$ *where*

1. $\Delta^{\mathcal{I}}$ *is a non-empty set,*
2. $\cdot^{\mathcal{I}}$ *is an interpretation function which assigns to every atomic concept A a set $A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$ and to every atomic role R a binary relation $R^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$.*

The interpretation function is extended to concepts by the following inductive definitions:

1. $(\neg C)^{\mathcal{I}} := \Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}$,
2. $(C \sqcap D)^{\mathcal{I}} := C^{\mathcal{I}} \cap D^{\mathcal{I}}$,
3. $(C \sqcup D)^{\mathcal{I}} := C^{\mathcal{I}} \cup D^{\mathcal{I}}$,
4. $(\forall R.C)^{\mathcal{I}} := \{a \in \Delta^{\mathcal{I}} \mid \forall b \ [(a,b) \in R^{\mathcal{I}} \Rightarrow b \in C^{\mathcal{I}}]\}$,
5. $(\exists R.C)^{\mathcal{I}} := \{a \in \Delta^{\mathcal{I}} \mid \exists b \ [(a,b) \in R^{\mathcal{I}} \wedge b \in C^{\mathcal{I}}]\}$.

An interpretation $\mathcal{I}$ is a model of a concept C (denoted as $\mathcal{I} \models C$) if $C^{\mathcal{I}} \neq \emptyset$. A concept C is said to be satisfiable *in $\mathcal{ALC}$ if there exists an interpretation $\mathcal{I}$ such that $\mathcal{I} \models C$.*

The syntax of $\mathcal{ALC}$ is extended by a non-empty set N_I of individual names. We denote individual names by o, o_1, o_2, x, y and z.

Definition 3 *An* ABox *is a finite set of expressions of the form: $C(o)$ or $R(o_1, o_2)$ where o, o_1 and o_2 are in N_I, C is a concept, and R is an atomic role. An expression $C(o)$ or $R(o_1, o_2)$ is called an* ABox statement. *An interpretation $\mathcal{I}$ in Definition 2 is extended to apply also to individual names o such that $o^{\mathcal{I}} \in \Delta^{\mathcal{I}}$. Such an interpretation is a model of an ABox $\mathcal{A}$ if for every $C(o) \in \mathcal{A}$, $o^{\mathcal{I}} \in C^{\mathcal{I}}$ and for every $R(o_1, o_2) \in \mathcal{A}$, $(o_1^{\mathcal{I}}, o_2^{\mathcal{I}}) \in R^{\mathcal{I}}$. An ABox $\mathcal{A}$ is called* satisfiable *in $\mathcal{ALC}$ if it has a model.*

We adopt the following *unique name assumption*: for any $o_1, o_2 \in N_I$, if $o_1 \neq o_2$, then $o_1^{\mathcal{I}} \neq o_2^{\mathcal{I}}$.

Definition 4 *A* TBox *is a finite set of expressions of the form: $C \sqsubseteq D$. The elements of a TBox are called* TBox statments. *An interpretation $\mathcal{I} := \langle \Delta^{\mathcal{I}}, \cdot^{\mathcal{I}} \rangle$ is called a* model *of $C \sqsubseteq D$ if $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$. An interpretation $\mathcal{I}$ is said to be a model of a TBox $\mathcal{T}$ if $\mathcal{I}$ is a model of every element of $\mathcal{T}$. A TBox $\mathcal{T}$ is called* satisfiable *in $\mathcal{ALC}$ if it has a model.*

Definition 5 *A knowledge base Σ is a pair $(\mathcal{T}, \mathcal{A})$ where $\mathcal{T}$ is a TBox and $\mathcal{A}$ is an ABox. An interpretation $\mathcal{I}$ is a model of Σ if $\mathcal{I}$ is a model of both $\mathcal{T}$ and $\mathcal{A}$. A knowledge base Σ is called* satisfiable *in $\mathcal{ALC}$ if it has a model.*

Since the satisfiability for an ABox, a TBox or a knowledge base can be reduced to the satisfiability for a concept [2], we focus on the concept satisfiability in the following discussion.

2.2 $\mathcal{PALC}$

Similar notions and terminologies for $\mathcal{ALC}$ are also used for $\mathcal{PALC}$. The $\mathcal{PALC}$-language is constructed from the $\mathcal{ALC}$-language by adding $\sim$ (paraconsistent negation).

Definition 6 *Concepts C are defined by the following grammar:*

$$C ::= A \mid \neg C \mid \sim C \mid C \sqcap C \mid C \sqcup C \mid \forall R.C \mid \exists R.C$$

Definition 7 *A* paraconsistent interpretation *$\mathcal{PI}$ is a structure $\langle \Delta^{\mathcal{PI}}, \cdot^{\mathcal{I}^+}, \cdot^{\mathcal{I}^-} \rangle$ where*

1. *$\Delta^{\mathcal{PI}}$ is a non-empty set,*
2. *$\cdot^{\mathcal{I}^+}$ is an interpretation function which assigns to every atomic concept A a set $A^{\mathcal{I}^+} \subseteq \Delta^{\mathcal{PI}}$ and to every atomic role R a binary relation $R^{\mathcal{I}^+} \subseteq \Delta^{\mathcal{PI}} \times \Delta^{\mathcal{PI}}$,*
3. *$\cdot^{\mathcal{I}^-}$ is an interpretation function which assigns to every atomic concept A a set $A^{\mathcal{I}^-} \subseteq \Delta^{\mathcal{PI}}$ and to every atomic role R a binary relation $R^{\mathcal{I}^-} \subseteq \Delta^{\mathcal{PI}} \times \Delta^{\mathcal{PI}}$,*
4. *for any atomic role R, $R^{\mathcal{I}^+} = R^{\mathcal{I}^-}$.*

The interpretation functions are extended to concepts by the following inductive definitions:

1. *$(\sim C)^{\mathcal{I}^+} := C^{\mathcal{I}^-}$,*
2. *$(\neg C)^{\mathcal{I}^+} := \Delta^{\mathcal{PI}} \setminus C^{\mathcal{I}^+}$,*
3. *$(C \sqcap D)^{\mathcal{I}^+} := C^{\mathcal{I}^+} \cap D^{\mathcal{I}^+}$,*
4. *$(C \sqcup D)^{\mathcal{I}^+} := C^{\mathcal{I}^+} \cup D^{\mathcal{I}^+}$,*
5. *$(\forall R.C)^{\mathcal{I}^+} := \{a \in \Delta^{\mathcal{PI}} \mid \forall b\ [(a,b) \in R^{\mathcal{I}^+} \Rightarrow b \in C^{\mathcal{I}^+}]\}$,*
6. *$(\exists R.C)^{\mathcal{I}^+} := \{a \in \Delta^{\mathcal{PI}} \mid \exists b\ [(a,b) \in R^{\mathcal{I}^+} \wedge b \in C^{\mathcal{I}^+}]\}$,*
7. *$(\sim C)^{\mathcal{I}^-} := C^{\mathcal{I}^+}$,*
8. *$(\neg C)^{\mathcal{I}^-} := \Delta^{\mathcal{PI}} \setminus C^{\mathcal{I}^-}$,*

9. $(C \sqcap D)^{\mathcal{I}^-} := C^{\mathcal{I}^-} \cup D^{\mathcal{I}^-}$,

10. $(C \sqcup D)^{\mathcal{I}^-} := C^{\mathcal{I}^-} \cap D^{\mathcal{I}^-}$,

11. $(\forall R.C)^{\mathcal{I}^-} := \{a \in \Delta^{\mathcal{PI}} \mid \exists b \ [(a,b) \in R^{\mathcal{I}^-} \wedge b \in C^{\mathcal{I}^-}]\}$,

12. $(\exists R.C)^{\mathcal{I}^-} := \{a \in \Delta^{\mathcal{PI}} \mid \forall b \ [(a,b) \in R^{\mathcal{I}^-} \Rightarrow b \in C^{\mathcal{I}^-}]\}$.

An expression $\mathcal{I}^* \models C$ $(* \in \{+, -\})$ *is defined as* $C^{\mathcal{I}^*} \neq \emptyset$. *A paraconsistent interpretation* $\mathcal{PI} := \langle \Delta^{\mathcal{PI}}, \cdot^{\mathcal{I}^+}, \cdot^{\mathcal{I}^-} \rangle$ *is a model of a concept* C *(denoted as* $\mathcal{PI} \models C$*)* if $\mathcal{I}^+ \models C$. *A concept* C *is said to be* satisfiable *in* $\mathcal{PALC}$ *if there exists a paraconsistent interpretation* $\mathcal{PI}$ *such that* $\mathcal{PI} \models C$.

The interpretation functions $\cdot^{\mathcal{I}^+}$ and $\cdot^{\mathcal{I}^-}$ are intended to represent "verification" and "falsification", respectively.

Definition 8 *A paraconsistent interpretation* $\mathcal{PI}$ *in Definition 7 is extended to apply also to individual names* o *such that* $o^{\mathcal{I}^+}, o^{\mathcal{I}^-} \in \Delta^{\mathcal{PI}}$ *and* $o^{I^+} = o^{I^-}$. *Such a paraconsistent interpretation is a model of an ABox* $\mathcal{A}$ *if for every* $C(o) \in \mathcal{A}$, $o^{\mathcal{I}^+} \in C^{\mathcal{I}^+}$ *and for every* $R(o_1, o_2) \in \mathcal{A}$, $(o_1^{\mathcal{I}^+}, o_2^{\mathcal{I}^+}) \in R^{\mathcal{I}^+}$. *Such a paraconsistent interpretation is called a* model *of* $C \sqsubseteq D$ *if* $C^{\mathcal{I}^+} \subseteq D^{\mathcal{I}^+}$. *The satisfiability of ABox, a TBox or a knowledge base in* $\mathcal{PALC}$ *is defined in the same way as in* $\mathcal{ALC}$.

3 Semantical Embedding and Decidability

In the following, we introduce a translation of $\mathcal{PALC}$ into $\mathcal{ALC}$, and by using this translation, we show a semantical embedding theorem of $\mathcal{PALC}$ into $\mathcal{ALC}$. The translation introduced is a slight modification of the translation introduced by Ma et al. [5] to embed $\mathcal{ALC}4$ into $\mathcal{ALC}$. A similar translation has been used by Gurevich [3] and Rautenberg [14] to embed Nelson's three-valued constructive logic [1, 9] into intuitionistic logic. The way of showing the semantical and syntactical embedding theorems of $\mathcal{PALC}$ into $\mathcal{ALC}$ is a new technical contribution developed in this paper. The semantical and syntactical embedding theorems are used to show the decidability and completeness theorems for $\mathcal{PALC}$.

Definition 9 *Let* N_C *be a non-empty set of atomic concepts and* N_C' *be the set* $\{A' \mid A \in N_C\}$ *of atomic concepts.* [1] *Let* N_R *be a non-empty set of atomic roles and* N_I *be a non-empty set of individual names. The language* $\mathcal{L}^\sim$ *of* $\mathcal{PALC}$ *is defined using* N_C, N_R, N_I, $\sim$, $\neg, \sqcap, \sqcup$, $\forall R$ *and* $\exists R$. *The language* $\mathcal{L}$ *of* $\mathcal{ALC}$ *is obtained from* $\mathcal{L}^\sim$ *by adding* N_C' *and deleting* $\sim$.

A mapping f from $\mathcal{L}^\sim$ to $\mathcal{L}$ is defined inductively by

1. *for any* $R \in N_R$ *and any* $o \in N_I$, $f(R) := R$ *and* $f(o) := o$,

2. *for any* $A \in N_C$, $f(A) := A$ *and* $f(\sim A) := A' \in N_C'$,

3. *For any* $A(o) \in N_C$, $f(A(o)) := A(f(o))$ *and* $f(\sim A(o)) := A'(f(o)) \in N_C'$,

4. $f(\neg C) := \neg f(C)$,

---

[1] A can include individual names, i.e., A can be $A(o)$ for any $o \in N_I$.

5. $f(C \sharp D) := f(C) \sharp f(D)$ *where* $\sharp \in \{\sqcap, \sqcup\}$,
6. $f(\forall R.C) := \forall f(R).f(C)$,
7. $f(\exists R.C) := \exists f(R).f(C)$,
8. $f(\sim\sim C) := f(C)$,
9. $f(\sim\neg C) := \neg f(\sim C)$,
10. $f(\sim(C \sqcap D)) := f(\sim C) \sqcup f(\sim D)$,
11. $f(\sim(C \sqcup D)) := f(\sim C) \sqcap f(\sim D)$,
12. $f(\sim\forall R.C) := \exists f(R).f(\sim C)$,
13. $f(\sim\exists R.C) := \forall f(R).f(\sim C)$.

Lemma 10 *Let f be the mapping defined in Definition 9. For any paraconsistent interpretation $\mathcal{PI} := \langle \Delta^{\mathcal{PI}}, \cdot^{\mathcal{I}^+}, \cdot^{\mathcal{I}^-} \rangle$ of $\mathcal{PALC}$, we can construct an interpretation $\mathcal{I} := \langle \Delta^{\mathcal{I}}, \cdot^{\mathcal{I}} \rangle$ of $\mathcal{ALC}$ such that for any concept C in $\mathcal{L}^\sim$,*

1. $C^{\mathcal{I}^+} = f(C)^{\mathcal{I}}$,
2. $C^{\mathcal{I}^-} = f(\sim C)^{\mathcal{I}}$.

Proof. Let N_C be a non-empty set of atomic concepts and N_C' be the set $\{A' \mid A \in N_C\}$ of atomic concepts. Let N_R and N_I be sets of atomic roles and individual names, respectively.

Suppose that $\mathcal{PI}$ is a paraconsistent interpretation $\langle \Delta^{\mathcal{PI}}, \cdot^{\mathcal{I}^+}, \cdot^{\mathcal{I}^-} \rangle$ where

1. $\Delta^{\mathcal{PI}}$ is a non-empty set,
2. $\cdot^{\mathcal{I}^+}$ is an interpretation function which assigns to every atomic concept $A \in N_C$ a set $A^{\mathcal{I}^+} \subseteq \Delta^{\mathcal{PI}}$, to every atomic role $R \in N_R$ a binary relation $R^{\mathcal{I}^+} \subseteq \Delta^{\mathcal{PI}} \times \Delta^{\mathcal{PI}}$ and to every individual name $o \in N_I$ an element $o^{\mathcal{I}^+} \in \Delta^{\mathcal{PI}}$,
3. $\cdot^{\mathcal{I}^-}$ is an interpretation function which assigns to every atomic concept $A \in N_C$ a set $A^{\mathcal{I}^-} \subseteq \Delta^{\mathcal{PI}}$, to every atomic role $R \in N_R$ a binary relation $R^{\mathcal{I}^-} \subseteq \Delta^{\mathcal{PI}} \times \Delta^{\mathcal{PI}}$ and to every individual name $o \in N_I$ an element $o^{\mathcal{I}^-} \in \Delta^{\mathcal{PI}}$,
4. for any $R \in N_R$ and any $o \in N_I$, $R^{\mathcal{I}^+} = R^{\mathcal{I}^-}$ and $o^{\mathcal{I}^+} = o^{\mathcal{I}^-}$.

Suppose that $\mathcal{I}$ is an interpretation $\langle \Delta^{\mathcal{I}}, \cdot^{\mathcal{I}} \rangle$ where

1. $\Delta^{\mathcal{I}}$ is a non-empty set such that $\Delta^{\mathcal{I}} = \Delta^{\mathcal{PI}}$,
2. $\cdot^{\mathcal{I}}$ is an interpretation function which assigns to every atomic concept $A \in N_C \cup N_C'$ a set $A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$, to every atomic role $R \in N_R$ a binary relation $R^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$ and to every individual name $o \in N_I$ an element $o^{\mathcal{I}} \in \Delta^{\mathcal{I}}$,
3. for any $R \in N_R$ and any $o \in N_I$, $R^{\mathcal{I}} = R^{\mathcal{I}^+} = R^{\mathcal{I}^-}$ and $o^{\mathcal{I}} = o^{\mathcal{I}^+} = o^{\mathcal{I}^-}$.

Suppose moreover that $\mathcal{PI}$ and $\mathcal{I}$ satisfy the following conditions: for any $A \in N_C$ and any $o \in N_I$,

1. $A^{\mathcal{I}^+} = A^{\mathcal{I}}$ and $(A(o))^{\mathcal{I}^+} = (A(o))^{\mathcal{I}}$,
2. $A^{\mathcal{I}^-} = (A')^{\mathcal{I}}$ and $(A(o))^{\mathcal{I}^-} = (A'(o))^{\mathcal{I}}$.

The lemma is then proved by (simultaneous) induction on the complexity of C. The base step is obvious. We show only some cases on the induction step below.

Case $C \equiv \neg D$: For (1), we obtain: $a \in (\neg D)^{\mathcal{I}^+}$ iff $a \in \Delta^{\mathcal{PI}} \setminus D^{\mathcal{I}^+}$ iff $a \in \Delta^{\mathcal{I}} \setminus D^{\mathcal{I}^+}$ (by the condition $\Delta^{\mathcal{PI}} = \Delta^{\mathcal{I}}$) iff $a \in \Delta^{\mathcal{I}} \setminus f(D)^{\mathcal{I}}$ (by induction hypothesis for 1) iff $a \in (\neg f(D))^{\mathcal{I}}$ iff $a \in f(\neg D)^{\mathcal{I}}$ (by the definition of f). For (2), we obtain: $a \in (\neg D)^{\mathcal{I}^-}$ iff $a \in \Delta^{\mathcal{PI}} \setminus D^{\mathcal{I}^-}$ iff $a \in \Delta^{\mathcal{I}} \setminus D^{\mathcal{I}^-}$ (by the condition $\Delta^{\mathcal{PI}} = \Delta^{\mathcal{I}}$) iff $a \in \Delta^{\mathcal{I}} \setminus f(\sim D)^{\mathcal{I}}$ (by induction hypothesis for 2) iff $a \in (\neg f(\sim D))^{\mathcal{I}}$ iff $a \in f(\sim \neg D)^{\mathcal{I}}$ (by the definition of f).

Case $C \equiv \sim D$: For (1), we obtain: $a \in (\sim D)^{\mathcal{I}^+}$ iff $a \in D^{\mathcal{I}^-}$ iff $a \in f(\sim D)^{\mathcal{I}}$ (by induction hypothesis for 2). For (2), we obtain: $a \in (\sim D)^{\mathcal{I}^-}$ iff $a \in D^{\mathcal{I}^+}$ iff $a \in f(D)^{\mathcal{I}}$ (by induction hypothesis for 1) iff $a \in f(\sim \sim D)^{\mathcal{I}}$ (by the definition of f).

Case $C \equiv \forall R.D$: We show only (2) below.

$d \in (\forall R.D)^{\mathcal{I}^-}$

iff $d \in \{a \in \Delta^{\mathcal{PI}} \mid \exists b \, [(a,b) \in R^{\mathcal{I}^-} \wedge b \in D^{\mathcal{I}^-}]\}$

iff $d \in \{a \in \Delta^{\mathcal{I}} \mid \exists b \, [(a,b) \in R^{\mathcal{I}} \wedge b \in D^{\mathcal{I}^-}]\}$ (by the conditions $\Delta^{\mathcal{PI}} = \Delta^{\mathcal{I}}$ and $R^{\mathcal{I}^-} = R^{\mathcal{I}}$)

iff $d \in \{a \in \Delta^{\mathcal{I}} \mid \exists b \, [(a,b) \in R^{\mathcal{I}} \wedge b \in f(\sim D)^{\mathcal{I}}]\}$ (by induction hypothesis for 2)

iff $d \in ((\exists R.f(\sim D))^{\mathcal{I}})$

iff $d \in ((\exists f(R).f(\sim D))^{\mathcal{I}})$ (by the definition of f)

iff $d \in ((f(\sim \forall R.D))^{\mathcal{I}})$ (by the definition of f).

∎

Lemma 11 *Let f be the mapping defined in Definition 9. For any paraconsistent interpretation $\mathcal{PI} := \langle \Delta^{\mathcal{PI}}, \cdot^{\mathcal{I}^+}, \cdot^{\mathcal{I}^-} \rangle$ of $\mathcal{PALC}$, we can construct an interpretation $\mathcal{I} := \langle \Delta^{\mathcal{I}}, \cdot^{\mathcal{I}} \rangle$ of $\mathcal{ALC}$ such that for any concept C in $\mathcal{L}^{\sim}$,*

1. $\mathcal{I}^+ \models C$ iff $\mathcal{I} \models f(C)$,
2. $\mathcal{I}^- \models C$ iff $\mathcal{I} \models f(\sim C)$.

Proof. By Lemma 10. ∎

Lemma 12 *Let f be the mapping defined in Definition 9. For any interpretation $\mathcal{I} := \langle \Delta^{\mathcal{I}}, \cdot^{\mathcal{I}} \rangle$ of $\mathcal{ALC}$, we can construct a paraconsistent interpretation $\mathcal{PI} := \langle \Delta^{\mathcal{PI}}, \cdot^{\mathcal{I}^+}, \cdot^{\mathcal{I}^-} \rangle$ of $\mathcal{PALC}$ such that for any concept C in $\mathcal{L}^{\sim}$,*

1. $\mathcal{I} \models f(C)$ iff $\mathcal{I}^+ \models C$,
2. $\mathcal{I} \models f(\sim C)$ iff $\mathcal{I}^- \models C$.

Proof. Similar to the proof of Lemma 11. ∎

Theorem 13 (Semantical embedding) *Let f be the mapping defined in Definition 9. For any concept C,*

C is satisfiable in $\mathcal{PALC}$ iff $f(C)$ is satisfiable in $\mathcal{ALC}$.

Proof. By Lemmas 11 and 12.   ▮

Theorem 14 (Decidability) *The concept satisfiability problem for $\mathcal{PALC}$ is decidable.*

Proof. By decidability of the satisfiability problem for $\mathcal{ALC}$, for each concept C of $\mathcal{PALC}$, it is possible to decide if $f(C)$ is satisfiable in $\mathcal{ALC}$. Then, by Theorem 13, the satisfiability problem for $\mathcal{PALC}$ is decidable.   ▮

The satisfiability problems of a TBox, an ABox and a knowledge base for $\mathcal{PALC}$ are also shown to be decidable.

Since f is a polynomial-time reduction, the complexities of the satisfiability problems of a TBox, an ABox and a knowledge base for $\mathcal{PALC}$ can be reduced to those for $\mathcal{ALC}$, i.e., the complexities of the problems for $\mathcal{PALC}$ are the same as those for $\mathcal{ALC}$. For example, the satisfiability problems of an acyclic TBox and a general TBox for $\mathcal{PALC}$ are PSPACE-complete and EXPTIME-complete, respectively.

For the concept satisfiability problem for $\mathcal{PALC}$, the existing tableau algorithms for $\mathcal{ALC}$ are applicable by using the translation f with Theorem 13.

4   Syntactical Embedding and Completeness

From a purely theoretical or logical point of view, a sound and complete axiomatization is required for the underlying semantics. In this section, we thus give a sound and complete tableau calculus $\mathcal{TAALC}$ for $\mathcal{PALC}$.

Definition 15 *A concept is called a* negation normal form *(NNF) if the classical negation connective $\neg$ occurs only in front of atomic concepts.*

Let $C(x)$ be a concept in NNF. In order to test satisfiability of $C(x)$, the tableau algorithm starts with the ABox $\mathcal{A} = \{C(x)\}$, and applies the inference rules of a tableau calculus to the ABox until no more rules apply.

Definition 16 ($\mathcal{TALC}$) *Let $\mathcal{A}$ be an ABox that consists only of NNF-concepts. The inference rules for the tableau calculus $\mathcal{TALC}$ for $\mathcal{ALC}$ are of the form:*

$$\frac{\mathcal{A}}{\mathcal{A} \cup \{C_1(x), C_2(x)\}} \ (\sqcap)$$

where $(C_1 \sqcap C_2)(x) \in \mathcal{A}$, $C_1(x) \notin \mathcal{A}$ or $C_2(x) \notin \mathcal{A}$,

$$\frac{\mathcal{A}}{\mathcal{A} \cup \{C_1(x)\} \quad | \quad \mathcal{A} \cup \{C_2(x)\}} \ (\sqcup)$$

where $(C_1 \sqcup C_2)(x) \in \mathcal{A}$ and $[C_1(x) \notin \mathcal{A}$ and $C_2(x) \notin \mathcal{A}]$,

$$\frac{\mathcal{A}}{\mathcal{A} \cup \{C(y)\}} \ (\forall R)$$

where $(\forall R.C)(x) \in \mathcal{A}$, $R(x, y) \in \mathcal{A}$ *and* $C(y) \notin \mathcal{A}$,

$$\frac{\mathcal{A}}{\mathcal{A} \cup \{C(y), R(x, y)\}} \ (\exists R)$$

where $(\exists R.C)(x) \in \mathcal{A}$, *there is no individual name* z *such that* $C(z) \in \mathcal{A}$ *and* $R(x, z) \in \mathcal{A}$, *and* y *is an individual name not occurring in* $\mathcal{A}$.

Definition 17 *Let* $\mathcal{A}$ *be an ABox that consists only of NNF-concepts. Then,* $\mathcal{A}$ *is called* complete *if there is no more rules apply to* $\mathcal{A}$. $\mathcal{A}$ *is called* clash *if* $\{A(x), \neg A(x)\} \subseteq \mathcal{A}$ *for some atomic concept* $A(x)$. *A tree produced by a tableau calculus from* $\mathcal{A}$ *is called* complete *if all the nodes in the tree are complete. A branch of a tree produced by a tableau calculus from* $\mathcal{A}$ *is called* clash-free *if all its nodes are not clash.*

The following theorem is known.

Theorem 18 (Completeness) *For any* $\mathcal{ALC}$-*concept* C *in NNF,* $\mathcal{T}\mathcal{ALC}$ *produces a complete tree with a clash-free branch from the Abox* $\{C\}$ *iff* C *is satisfiable in* $\mathcal{ALC}$.

For $\mathcal{PALC}$-concepts, we use the same definition of NNF as that of $\mathcal{ALC}$-concepts, i.e., "negation" in the term NNF means "classical negation." The way of obtaining NNFs for $\mathcal{PALC}$-concepts is almost the same as that for $\mathcal{ALC}$-concepts, except that we also use the law: $\neg{\sim}C \leftrightarrow {\sim}\neg C$, which is justified by the fact: $(\neg{\sim}C)^{\mathcal{I}^+} = ({\sim}\neg C)^{\mathcal{I}^+}$.

Definition 19 ($\mathcal{TPALC}$) *Let* $\mathcal{A}$ *be an ABox that consists only of NNF-concepts.*
   The inference rules for the tableau calculus $\mathcal{TPALC}$ *for* $\mathcal{PALC}$ *are obtained from* $\mathcal{T}\mathcal{ALC}$ *by adding the inference rules of the form:*

$$\frac{\mathcal{A}}{\mathcal{A} \cup \{C(x)\}} \ (\sim)$$

where ${\sim}{\sim}C(x) \in \mathcal{A}$, [2]

$$\frac{\mathcal{A}}{\mathcal{A} \cup \{{\sim}C_1(x)\} \quad | \quad \mathcal{A} \cup \{{\sim}C_2(x)\}} \ (\sim\sqcap)$$

where $({\sim}(C_1 \sqcap C_2))(x) \in \mathcal{A}$ *and* $[{\sim}C_1(x) \notin \mathcal{A}$ *and* ${\sim}C_2(x) \notin \mathcal{A}]$,

$$\frac{\mathcal{A}}{\mathcal{A} \cup \{{\sim}C_1(x), {\sim}C_2(x)\}} \ (\sim\sqcup)$$

where $({\sim}(C_1 \sqcup C_2))(x) \in \mathcal{A}$, ${\sim}C_1(x) \notin \mathcal{A}$ *or* ${\sim}C_2(x) \notin \mathcal{A}$,

$$\frac{\mathcal{A}}{\mathcal{A} \cup \{{\sim}C(y), R(x, y)\}} \ (\sim\forall R)$$

---

[2] We do not use the condition: $C(x) \notin \mathcal{A}$ in $(\sim)$. This is from a technical reason. See the proof of Theorem 20.

where $(\sim\forall R.C)(x) \in \mathcal{A}$, there is no individual name z such that $\sim C(z) \in \mathcal{A}$ and $R(x,z) \in \mathcal{A}$, and y is an individual name not occurring in $\mathcal{A}$,

$$\frac{\mathcal{A}}{\mathcal{A} \cup \{\sim C(y)\}} \ (\sim\exists R)$$

where $(\sim\exists R.C)(x) \in \mathcal{A}$, $R(x,y) \in \mathcal{A}$ and $\sim C(y) \notin \mathcal{A}$.

An expression $f(\mathcal{A})$ denotes the set $\{f(\alpha) \mid \alpha \in \mathcal{A}\}$.

Theorem 20 (Syntactical embedding) *Let $\mathcal{A}$ be an ABox that consists only of NNF-concepts in $\mathcal{L}^\sim$, and f be the mapping defined in Definition 9. Then:*

> *$\mathcal{TPALC}$ produces a complete tree with a clash-free branch from $\mathcal{A}$ iff $\mathcal{TALC}$ produces a complete tree with a clash-free branch from $f(\mathcal{A})$*

Proof. • ($\Longrightarrow$): By induction on the complete trees T with a clash-free branch from $\mathcal{A}$ in $\mathcal{TPALC}$. We distinguish the cases according to the first inference of T. The base step is obvious. The induction step is considered below. We show only the following case.

Case ($\sim$): The first inference of T is of the form:

$$\frac{\mathcal{A}}{\mathcal{A} \cup \{C(x)\}} \ (\sim)$$

where $\sim\sim C(x) \in \mathcal{A}$. By induction hypothesis, $\mathcal{TALC}$ produces a complete tree with a clash-free branch from $f(\mathcal{A}) \cup \{f(C(x))\}$ with $f(\sim\sim C(x)) \in f(\mathcal{A})$. By the definition of f, we have $f(\sim\sim C(x)) = f(C(x))$, and hence $f(\mathcal{A}) \cup \{f(C(x))\} = f(\mathcal{A}) \in f(\mathcal{A})$. Therefore, $\mathcal{TALC}$ provides a complete tree with a clash-free branch from $f(\mathcal{A})$.

• ($\Longleftarrow$): By induction on the complete trees T' with a clash-free branch from $f(\mathcal{A})$ in $\mathcal{TALC}$. We distinguish the cases according to the first inference of T'. We show only the following case.

Case ($\forall R$): The first inference of T' is of the form:

$$\frac{f(\mathcal{A})}{f(\mathcal{A}) \cup \{f(\sim C(y))\}} \ (\forall R)$$

where $\forall R.f(\sim C(x)) \in f(\mathcal{A})$, $f(R(x,y)) \in f(\mathcal{A})$ and $f(\sim C(y)) \notin f(\mathcal{A})$. By induction hypothesis, $\mathcal{TPALC}$ provides a complete tree with a clash-free branch from $\mathcal{A} \cup \{\sim C(y)\}$. By the definition of f, we have $\forall R.f(\sim C(x)) = \forall f(R).f(\sim C(x)) = f(\sim\exists R.C(x))$ and $f(R(x,y)) = R(x,y)$. Thus, we obtain:

$$\frac{\mathcal{A}}{\mathcal{A} \cup \{\sim C(y)\}} \ (\sim\exists R).$$

Therefore, $\mathcal{TPALC}$ provides a complete tree with a clash-free branch from $\mathcal{A}$. ∎

Theorem 21 (Completeness) *For any $\mathcal{PALC}$-concept C in NNF, $\mathcal{TPALC}$ produces a complete tree with a clash-free branch from the Abox $\{C\}$ iff C is satisfiable in $\mathcal{PALC}$.*

Proof. Let C be a $\mathcal{PALC}$-concept in NNF. Then, we obtain:

$\mathcal{TPALC}$ produces a complete tree with a clash-free branch from $\{C\}$

iff $\mathcal{TALC}$ produces a complete tree with a clash-free branch from $\{f(C)\}$ (by Theorem 20)

iff $f(C)$ is satisfiable in $\mathcal{ALC}$ (by Theorem 18)

iff C is satisfiable in $\mathcal{PALC}$ (by Theorem 13).

∎

5 Remarks

We now explain about some differences and similarities among $\mathcal{ALC}4$ [5], quasi-classical description logics [18, 19] and $\mathcal{PALC}$. In $\mathcal{ALC}4$, a four-valued interpretation $\mathcal{I} := (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ is defined using a pair $\langle P, N \rangle$ of subsets of $\Delta^{\mathcal{I}}$ and the projection functions $proj^+\langle P, N \rangle := P$ and $proj^-\langle P, N \rangle := N$. The interpretations of an atomic concept A and a conjunctive concept $C_1 \sqcap C_2$ are then defined as follows:

1. $A^{\mathcal{I}} := \langle P, N \rangle$ where $P, N \subseteq \Delta^{\mathcal{I}}$,
2. $(C_1 \sqcap C_2)^{\mathcal{I}} := \langle P_1 \cap P_2, N_1 \cup N_2 \rangle$ if $C_i^{\mathcal{I}} = \langle P_i, N_i \rangle$ for $i = 1, 2$.

In quasi-classical description logics, a reformulation or simplification of the four-valued interpretations of $\mathcal{ALC}4$ is used: An interpretation is defined using a pair $\langle +C, -C \rangle$ of subsets of $\Delta^{\mathcal{I}}$ without using projection functions. The interpretations of an atomic concept A and a conjunctive concept $C_1 \sqcap C_2$ are then defined as follows:

1. $A^{\mathcal{I}} := \langle +A, -A \rangle$ where $+A, -A \subseteq \Delta^{\mathcal{I}}$,
2. $(C_1 \sqcap C_2)^{\mathcal{I}} := \langle +C_1 \cap +C_2, -C_1 \cup -C_2 \rangle$.

The pairing functions used in the four-valued and quasi-classical semantics have been used in some algebraic semantics for Nelson's logics (see e.g. [10] and the references therein). On the other hand, the semantics of $\mathcal{PALC}$ is defined using two interpretation functions $\cdot^{\mathcal{I}^+}$ and $\cdot^{\mathcal{I}^+}$ instead of the pairing functions. These interpretation functions have been used in some Kripke-type semantics for Nelson's logics (see e.g. [17] and the references therein). The "horizontal" semantics using paring functions and the "vertical" semantics using two kinds of interpretation functions have thus essentially the same meaning.

References

1. A. Almukdad and D. Nelson, Constructible falsity and inexact predicates, *Journal of Symbolic Logic* 49, pp. 231–233, 1984.
2. F. Baader, D. Calvanese, D. McGuinness, D. Nardi and Peter F. Patel-Schneider (Eds.), The description logic handbook: Theory, implementation and applications, Cambridge University Press, 2003.
3. Y. Gurevich, Intuitionistic logic with strong negation, *Studia Logica* 36, pp. 49–59, 1977.
4. K. Kaneiwa, Description logics with contraries, contradictories, and subcontraries, *New Generation Computing* 25 (4), pp. 443–468, 2007.
5. Y. Ma, P. Hitzler and Z. Lin, Algorithms for paraconsistent reasoning with OWL, Proceedings of the 4th European Semantic Web Conference (ESWC 2007), LNCS 4519, pp. 399-413, 2007.
6. Y. Ma, P. Hitzler and Z. Lin, Paraconsistent reasoning for expressive and tractable description logics, Proceedings of the 21st International Workshop on Description Logic (DL 2008), CEUR Workshop Proceedings 353.
7. C. Meghini and U. Straccia, A relevance terminological logic for information retrieval, Proceedings of the 19th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 197–205, 1996.
8. C. Meghini, F. Sebastiani and U. Straccia, Mirlog: A logic for multimedia information retrieval, In: Uncertainty and Logics: Advanced Models for the Representation and Retrieval of Information, pp. 151–185, Kluwer Academic Publishing, 1998.
9. D. Nelson, Constructible falsity, *Journal of Symbolic Logic* 14, pp. 16–26, 1949.
10. S.P. Odintsov, Algebraic semantics for paraconsistent Nelson's logic, *Journal of Logic and Computation* 13 (4), pp. 453–468, 2003.
11. S.P. Odintsov and H. Wansing, Inconsistency-tolerant description logic: Motivation and basic systems, in: V.F. Hendricks and J. Malinowski, Editors, Trends in Logic: 50 Years of Studia Logica, Kluwer Academic Publishers, Dordrecht, pp. 301–335, 2003.
12. S.P. Odintsov and H. Wansing, Inconsistency-tolerant Description Logic. Part II: Tableau Algorithms, *Journal of Applied Logic* 6, pp. 343–360, 2008.
13. Peter F. Patel-Schneider, A four-valued semantics for terminological logics, *Artificial Intelligence* 38, pp. 319–351, 1989.
14. W. Rautenberg, Klassische und nicht-klassische Aussagenlogik, Vieweg, Braunschweig, 1979.
15. M. Schmidt-Schauss and G. Smolka, Attributive concept descriptions with complements, *Artificial Intelligence* 48, pp. 1–26, 1991.
16. U. Straccia, A sequent calculus for reasoning in four-valued description logics, Proceedings of International Conference on Automated Reasoning with Analytic Tableaux and Related Methods (TABLEAUX 1997), LNCS 1227, pp. 343–357, 1997.
17. H. Wansing, The logic of information structures, LNAI 681, 163 pages, 1993.
18. X. Zhang and Z. Lin, Paraconsistent reasoning with quasi-classical semantics in $\mathcal{ALC}$, Proceedings of the 2nd International Conference on Web Reasoning and Rule Systems (RR 2008), LNCS 5341, pp. 222–229, 2008.
19. X. Zhang, G. Qi, Y. Ma, Z. Lin, Quasi-classical semantics for expressive description logics, Proceedings of the 22nd International Workshop on Description Logic (DL 2009), CEUR Workshop Proceedings 477.

Optimized DL Reasoning via Core Blocking

Birte Glimm, Ian Horrocks, and Boris Motik

Oxford University Computing Laboratory, UK

1   Introduction

State of the art reasoners for expressive DLs are typically model building procedures that decide the (un)satisfiability of a knowledge base $\mathcal{K}$ via a constructive search for an abstraction of a model for $\mathcal{K}$. Despite numerous optimizations, certain existing and emerging knowledge bases still pose significant challenges to such reasoners mainly because these abstractions can be very large.

To ensure that only finite model abstractions are constructed (hyper)tableau reasoners use a cycle detection technique called *blocking*. It has already been demonstrated that using a more fine-grained blocking condition can make the constructed abstractions smaller, resulting in a significant speedup [1]. Even with such a blocking condition, however, the constructed model abstractions can be very large; furthermore, checking such fine-grained conditions can itself be costly.

To address these problems, we propose a new *core blocking* technique. Our technique first employs an easy-to-check and very "aggressive" blocking condition that can halt the model construction much earlier than existing techniques. This condition is so aggressive that, if used alone, it is not necessarily the case that the constructed abstraction can be expanded into a model. Therefore, after a model abstraction has been constructed, a detailed check is performed to ensure that all blocks are indeed valid, and the model construction terminates only if all blocks pass this check.

We further present an empirical evaluation using a prototypical implementation of our technique in the HermiT reasoner. The evaluation compares the performance of the hypertableau algorithm employing the original blocking condition and several core blocking variants on widely used ontologies. The evaluation shows that the model abstraction size can be reduced significantly. The effects of core blocking are most pronounced with large and complex ontologies such as DOLCE or GALEN. Furthermore, core blocking allows HermiT to classify an OWL version of the FMA ontology [2], whereas with standard blocking the reasoner runs out or memory.

Further details and evaluation results are available in a technical report [3].

2   Preliminaries

The formal definition of the hypertableau calculus is technically involved; therefore, we will introduce only those aspects needed to understand the idea behind core blocking. For further details and the precise definitions, we refer to [4].

The calculus is applicable to a knowledge base $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ expressed in $\mathcal{SROIQ}$ [5]. The calculus does not operate on $\mathcal{K}$ directly; rather, in order to reduce nondeterminism, it first translates $\mathcal{K}$ into a set of clauses $\mathcal{C}$ and an ABox $\mathcal{A}$. The class of clauses

on which the hypertableau calculus operates is called HT-clauses. An HT-clause is an implication of the form $\bigwedge_{i=1}^{m} U_i \rightarrow \bigvee_{j=1}^{n} V_j$, where U_i and V_j are called the *antecedent* and the *consequent* atoms, respectively. Most notably, for r an atomic role, s is a role, A is an atomic concept, and B, each *antecedent* atom is of the form $A(x)$, $r(x,x)$, $r(x,y_i)$, $r(y_i,x)$, $A(y_i)$, or $A(z_j)$. Each *consequent* atom is of the form $B(x)$, $\geqslant n\, s.B(x)$, $B(y_i)$, $r(x,x)$, $r(x,y_i)$, $r(y_i,x)$, $r(x,z_j)$, $r(z_j,x)$, $x \approx z_j$, or $y_i \approx y_j$. These syntactic restrictions reflect the structure of DL axioms and ultimately ensure termination of the calculus. HT-clauses are straightforwardly interpreted in first-order logic, and they intuitively state that at least one consequent atom must be true whenever all atoms in the antecedent are true. We next revise the derivation rules of the calculus.

The Hyp-rule is the main derivation rule. The rule is applicable to an HT-clause cl and an ABox $\mathcal{A}_\ell$ if a mapping σ from the variables in cl to the individuals in $\mathcal{A}_\ell$ exists such that $\sigma(U_i) \in \mathcal{A}_\ell$ for each $1 \le i \le m$, but $\sigma(V_j) \notin \mathcal{A}_\ell$ for each $1 \le j \le m$; if such σ exists, then a consequent atom V_j of cl is nondeterministically chosen and $\mathcal{A}_\ell$ is extended to $\mathcal{A}_{\ell+1} = \mathcal{A}_\ell \cup \{\sigma(V_j)\}$. For example, when applied to the HT-clause $r(x,y) \rightarrow (\geqslant 1\, r.A)(x) \vee D(y)$ and an ABox $\mathcal{A}_\ell$ containing $r(a,b)$, the Hyp-rule extends $\mathcal{A}_\ell$ either with $(\geqslant 1\, r.A)(a)$ or $D(b)$. The $\geqslant$-rule deals with existential quantifiers and number restrictions. Let $\mathsf{ar}(s,a,b) = s(a,b)$ if s is an atomic role and $\mathsf{ar}(s,a,b) = r(b,a)$ if s is an inverse role such that $s = r^-$. The rule is applicable to $(\geqslant n\, r.B)(a) \in \mathcal{A}_\ell$ if no individuals $b_1, \ldots, b_n$ exist such that $\mathsf{ar}(r,a,b_i) \in \mathcal{A}_\ell$ and $B(b_i) \in \mathcal{A}_\ell$ for each $1 \le i \le n$, and $b_i \not\approx b_j \in \mathcal{A}_\ell$ for each $1 \le i < j \le n$. If this is the case, then $\mathcal{A}_\ell$ is extended to $\mathcal{A}_{\ell+1}$ by introducing fresh individuals $c_1, \ldots, c_n$ and adding assertions $\mathsf{ar}(r,a,c_i)$ and $B(c_i)$ for $1 \le i \le n$, and $c_i \not\approx c_j$ for $1 \le i < j \le n$.

The $\approx$-rule deals with equality: given $a \approx b$, the rule replaces the individual a in all assertions with the individual b, and adds some bookkeeping information to keep track of the rule application. Finally, the $\perp$-rule detects contradictions—called *clashes*—such as $A(a)$ and $\neg A(a)$, or $a \not\approx a$. A clash-free ABox to which no derivation rule is applicable is called a *pre-model*.

2.1 Blocking

Unrestricted application of the $\geqslant$-rule could lead to nontermination of the HT calculus. To prevent that, the $\geqslant$-rule is applied to an assertion $(\geqslant n\, r.B)(a)$ only if the individual a is not *blocked*, as described next.

To apply blocking, the individuals are split into two sets. *Root* individuals (mainly individuals occurring in the input), which are never blocked and *blockable* individuals, which are introduced by the $\geqslant$-rule and they can be blocked. For A an atomic concept and r an atomic role, we define labels of an individual and an individual pair as follows:

$$\mathcal{L}_{\mathcal{A}}(s) = \{A \mid A(s) \in \mathcal{A}\} \qquad \mathcal{L}_{\mathcal{A}}(s,t) = \{r \mid r(s,t) \in \mathcal{A}\}$$

To prevent cyclic blocks, we use a strict order $\prec$ over all individuals, which coincides with the order in which individuals are inserted into the ABox.

Pairwise anywhere blocking is necessary for knowledge bases that use inverse roles and number restrictions. Each individual s in an ABox $\mathcal{A}$ is assigned by induction on $\prec$ a status as follows: s is *blocked* if it is directly or indirectly blocked; s is *indirectly*

blocked if it has a blocked ancestor; and s is *directly blocked* by an individual t if, for s' and t' the predecessors of s and t, respectively, s, t, s', and t' are all blockable, t is not blocked, $t \prec s$, and (1)–(4) hold.

$$\mathcal{L}_\mathcal{A}(s) = \mathcal{L}_\mathcal{A}(t) \qquad (1) \qquad\qquad \mathcal{L}_\mathcal{A}(s') = \mathcal{L}_\mathcal{A}(t') \qquad (2)$$
$$\mathcal{L}_\mathcal{A}(s,s') = \mathcal{L}_\mathcal{A}(t,t') \qquad (3) \qquad\qquad \mathcal{L}_\mathcal{A}(s',s) = \mathcal{L}_\mathcal{A}(t',t) \qquad (4)$$

For an efficient implementation, we build, for each individual, a blocking signature that consist of the four label sets. A hash table containing the blocking signatures for possible blockers can then be used to cheaply look-up a blocker for an unblocked individual before the $\geqslant$-rule is applied.

The simpler *single anywhere blocking* can be used on knowledge bases without inverse roles, and it differs from the above definition in that s is *directly blocked* by an individual t if s and t are blockable, t is not blocked, $t \prec s$, and (1) holds.

A pre-model $\mathcal{A}'$ can be extended to a model for $(\mathcal{A}, \mathcal{C})$ by unraveling. Roughly speaking, each individual s that is directly blocked in $\mathcal{A}'$ by t is replaced by a "copy" of t; a precise account of this process is given in [4].

3 Optimized Blocking Strategies

For tableau algorithms that normally require pairwise blocking, Horrocks and Sattler proposed a more precise blocking condition [1], which amounts to single subset blocking with additional constraints on the predecessor of the individual that is to be blocked and on the blocker itself. Although checking the blocking conditions is quite expensive, the optimization exhibits substantial improvements in reasoning performance due to the significantly smaller pre-models.

Related blocking optimizations were proposed in the context of first-order theorem proving [6]; however, these techniques do not guarantee termination for DLs such as $\mathcal{SROIQ}$ that provide for nominals, number restrictions, and inverse roles.

Caching [7] is an orthogonal approach for reducing the pre-model size by reusing already constructed pre-model fragments. In fact, caching techniques can be used to obtain a worst-case optimal algorithm for certain DLs [8, 9]; in contrast, standard (hyper)tableau algorithms are usually not worst-case optimal.

3.1 Core Blocking

Unlike existing blocking techniques, core blocking is approximate rather than exact: applying core blocking alone does not guarantee that a pre-model can indeed be unraveled into a model. To ensure the latter, a pre-model needs to be checked to discover invalid blocks; if such blocks are found, the derivation is continued until either a contradiction is derived or all blocks become valid.

To formalize the process of discovering approximate blocks, we assume that each assertion α in an ABox is associated with a Boolean flag that determines whether α is a *core assertion*. A *core blocking policy* will be used to determine which assertions are core. In Section 3.3 we present two policies that strike a balance between the potential for reduction in the pre-model size and the cost of validating blocks. Before that, however, we introduce a general notion of core blocking that is applicable to any policy.

Definition 1. *For an ABox $\mathcal{A}$ and a pair of individuals s and t, let*

$$\mathcal{L}_{\mathcal{A}}^{\mathsf{core}}(s) = \{A \mid A \in \mathcal{L}_{\mathcal{A}}(s) \text{ and } A(s) \text{ is a core assertion in } \mathcal{A}\} \text{ and}$$
$$\mathcal{L}_{\mathcal{A}}^{\mathsf{core}}(s,t) = \{r \mid r(s,t) \in \mathcal{L}_{\mathcal{A}}(s,t) \text{ and } r(s,t) \text{ is a core assertion in } \mathcal{A}\}.$$

Single *and* pairwise core blocking *are obtained from the respective definitions given in Section 2.1 by using $\mathcal{L}_{\mathcal{A}}^{\mathsf{core}}$ instead of $\mathcal{L}_{\mathcal{A}}$ in conditions (1)–(4); furthermore, in single core blocking, for s to be directly blocked by t we additionally require both s and t to be successors of blockable individuals.*

The requirement that s and t are successors of blockable individuals ensures that single core blocking can also be used with knowledge bases that contain inverse roles.

A blocking validation test checks whether any of the derivation rules would be applicable if we were to unravel a candidate pre-model $\mathcal{A}_\ell$ to a model. If no rule becomes applicable, then we can guarantee that the model construction succeeds, i.e., the block is indeed valid. To this end, we define an ABox $\mathsf{val}_{\mathcal{A}_\ell}(s)$ for a blockable individual s that, intuitively, contains the assertions from the unraveling of $\mathcal{A}_\ell$ that affect inferences involving s.

Definition 2. *Let $\mathcal{C}$ be a set of HT clauses, and let $\mathcal{A}_\ell$ be an ABox. For an individual w, let $|w| = w$ if w is not blocked in $\mathcal{A}_\ell$, and $|w| = w'$ if w is blocked in $\mathcal{A}_\ell$ by w'. For a blockable individual s, the ABox $\mathsf{val}_{\mathcal{A}_\ell}(s)$ is the union of the sets shown in the following table, where u denotes the predecessor of s, v denotes a successor of $|s|$, b denotes a root individual, C denotes a concept, and r denotes an atomic role.*

1	2	3
$\{C(u) \mid C(u) \in \mathcal{A}_\ell\}$	$\{r(u,s) \mid r(u,s) \in \mathcal{A}_\ell\}$	$\{r(s,u) \mid r(s,u) \in \mathcal{A}_\ell\}$
$\{C(s) \mid C(|s|) \in \mathcal{A}_\ell\}$		
$\{C(v) \mid C(|v|) \in \mathcal{A}_\ell\}$	$\{r(s,v) \mid r(|s|,v) \in \mathcal{A}_\ell\}$	$\{r(v,s) \mid r(v,|s|) \in \mathcal{A}_\ell\}$
$\{C(b) \mid C(b) \in \mathcal{A}_\ell\}$	$\{r(s,b) \mid r(|s|,b) \in \mathcal{A}_\ell\}$	$\{r(b,s) \mid r(b,|s|) \in \mathcal{A}_\ell\}$

A blockable individual s is safe for blocking *in an ABox $\mathcal{A}_\ell$ if the following conditions are satisfied:*

– *the* Hyp-*rule is not applicable to an HT-clause $\gamma \in \mathcal{C}$ and $\mathsf{val}_{\mathcal{A}_\ell}(s)$ with a mapping σ such that $\sigma(x) = s$, and*
– *the $\geqslant$-rule is not applicable to an assertion $(\geqslant n\, r.B)(s)$ in $\mathsf{val}_{\mathcal{A}_\ell}(s)$.*

A directly blocked individual s with predecessor s' is validly blocked *in $\mathcal{A}_\ell$ if both s and s' are safe for blocking.*

On knowledge bases that normally require single blocking (i.e., that do not contain inverse roles), Definitions 1 and 2 can be simplified. By the model construction from [4], $\mathsf{val}_{\mathcal{A}_\ell}(s)$ then needs to contain only sets from columns 1 and 2 in Definition 2; this, in turn, allows us to drop the extra requirement on the predecessors of s and t in Definition 1 in the case of single core blocking.

3.2 Applying Core Blocking in a Derivation

If an individual s is core-blocked by an individual t but the block is identified as invalid, one should reconsider t as a potential blocker for s only after $\mathsf{val}_{\mathcal{A}_\ell}(s)$ changes; otherwise, the calculus might get stuck in an endless loop trying to block s by t and subsequently discovering the block to be invalid. We deal with this problem by associating with each individual s in $\mathcal{A}_\ell$ a Boolean flag $\mathsf{mod}_{\mathcal{A}_\ell}(s)$ that is updated as the derivation progresses. Intuitively, $\mathsf{mod}_{\mathcal{A}_\ell}(s) = \mathsf{true}$ means that $\mathsf{val}_{\mathcal{A}_\ell}(s)$ has changed since the last time blocks were checked for validity. We also maintain a set S of pairs of validly blocked and blocking individuals, which we to ensure that the calculus terminates only when all blocks are valid.

Definition 3. *Let S be a set of pairs of individuals; let $\mathcal{A}_\ell$ be an ABox; and let s and t be individuals occurring in $\mathcal{A}_\ell$. Then, s is directly blocked by t in $\mathcal{A}_\ell$ for S-core blocking iff s is directly blocked by t in $\mathcal{A}_\ell$ for core blocking and*

$$\langle s, t \rangle \in S \quad \text{or} \quad \mathsf{mod}_{\mathcal{A}_\ell}(s) = \mathsf{true} \quad \text{or} \quad \mathsf{mod}_{\mathcal{A}_\ell}(t) = \mathsf{true}.$$

A derivation by the hypertableau calculus with core blocking *for a set of HT-clauses $\mathcal{C}$ and an ABox $\mathcal{A}$ is constructed by applying the following steps.*

1. *Set $S := \emptyset$, $\mathcal{A}^\mathsf{a} := \mathcal{A}$, and $\mathsf{mod}_{\mathcal{A}^\mathsf{a}}(s) := \mathsf{true}$ for each individual s in $\mathcal{A}^\mathsf{a}$.*
2. *Apply the hypertableau calculus exhaustively to $\mathcal{A}^\mathsf{a}$ and $\mathcal{C}$ while using S-core blocking in the $\geqslant$-rule; furthermore, whenever $\mathcal{A}_{\ell+1}$ is derived from $\mathcal{A}_\ell$, for each individual s in $\mathcal{A}_{\ell+1}$ set*
   (a) $\mathsf{mod}_{\mathcal{A}_{\ell+1}}(s) := \mathsf{true}$ *if $\mathsf{val}_{\mathcal{A}_{\ell+1}}(s) \neq \mathsf{val}_{\mathcal{A}_\ell}(s)$ or if s does not occur in $\mathcal{A}_\ell$, and*
   (b) $\mathsf{mod}_{\mathcal{A}_{\ell+1}}(s) := \mathsf{mod}_{\mathcal{A}_\ell}(s)$ *otherwise.*
   Let $\mathcal{A}^\mathsf{b}$ be a resulting ABox to which no derivation rule is applicable.
3. *Set S to be equal to the set of pairs $\langle s, t \rangle$ of individuals such that s is directly blocked in $\mathcal{A}^\mathsf{b}$ by t and s is validly blocked in $\mathcal{A}^\mathsf{b}$.*
4. *Set $\mathsf{mod}_{\mathcal{A}^\mathsf{b}}(s) := \mathsf{false}$ for each individual s in $\mathcal{A}^\mathsf{b}$.*
5. *If an individual s exists such that s is core blocked in $\mathcal{A}^\mathsf{b}$ by t but $\langle s, t \rangle \notin S$, then set $\mathcal{A}^\mathsf{a} := \mathcal{A}^\mathsf{b}$ and go to Step 2.*
6. *Return $\mathcal{A}^\mathsf{b}$.*

Roughly speaking, our algorithm first applies the derivation rules as usual, with the difference that core blocking is used (this is because $S = \emptyset$ in Step 1). After computing a candidate pre-model $\mathcal{A}^\mathsf{b}$ in Step 2, in Step 3 the algorithm updates S to the set of pairs of valid blocks, and in Step 4 it marks all individuals in $\mathcal{A}^\mathsf{b}$ as not changed. In Step 5, the algorithm checks whether $\mathcal{A}^\mathsf{b}$ contains invalid blocks. If that is the case, the process is repeated; but then, S-core blocking ensures that only those blocks are considered that are known to be valid or for which at least one of the individuals has changed since the last validation. Theorem 1 shows that the calculus is sound, complete, and terminating.

Theorem 1. *Let $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ be a $\mathcal{SROIQ}$ knowledge base and $\mathcal{C}$ the set of HT-clauses for $\mathcal{K}$.*

1. *The hypertableau calculus with core blocking terminates.*

2. *If $\mathcal{C}$ and $\mathcal{A}$ are satisfiable, then $\bot \notin \mathcal{A}^{\mathrm{b}}$ for some $\mathcal{A}^{\mathrm{b}}$ computed by the calculus.*
3. *If $\mathcal{C}$ and $\mathcal{A}$ are unsatisfiable, then $\bot \in \mathcal{A}^{\mathrm{b}}$ for each $\mathcal{A}^{\mathrm{b}}$ computed by the calculus.*

Proof (Sketch). For the first claim, assume that $\mathcal{A}^{\mathrm{b}}$ is an ABox computed in Step 2 such that, whenever s is directly blocked in $\mathcal{A}^{\mathrm{b}}$ by t for core blocking, then s is directly blocked in $\mathcal{A}^{\mathrm{b}}$ by t for standard blocking. Each individual s is then validly blocked in $\mathcal{A}^{\mathrm{b}}$, so $\langle s, t \rangle \in S$ at Step 3 and the condition at Step 5 is not satisfied, so the calculus terminates. Thus, in the worst case, core blocking reduces to standard blocking, which implies a bound on the size of $\mathcal{A}^{\mathrm{b}}$ in the usual way [4]. Furthermore, if an individual t does not validly block s in an ABox $\mathcal{A}^{\mathrm{b}}$, then t can be considered again as a blocker for s only after $\mathsf{val}_{\mathcal{A}^{\mathrm{b}}}(s)$ or $\mathsf{val}_{\mathcal{A}^{\mathrm{b}}}(t)$ changes. Since $\mathcal{A}^{\mathrm{b}}$ is bounded in size, $\mathsf{val}_{\mathcal{A}^{\mathrm{b}}}(s)$ and $\mathsf{val}_{\mathcal{A}^{\mathrm{b}}}(t)$ can change only a bounded number of times; hence, t is considered as a candidate blocker for s only a finite number of times, which implies termination.

The second claim holds in the same way as in [4]. Finally, for the third claim, given an ABox $\mathcal{A}^{\mathrm{b}}$ computed by the calculus such that $\bot \notin \mathcal{A}^{\mathrm{b}}$, we unravel $\mathcal{A}^{\mathrm{b}}$ into an interpretation in the standard way [4]. From the definition of unraveling in [4], one can see that, for each blockable individual s, the ABox $\mathsf{val}_{\mathcal{A}^{\mathrm{b}}}(s)$ contains the assertions that correspond to the part of the unraveled interpretation involving s. Since s is validly blocked in $\mathcal{A}^{\mathrm{b}}$, all the relevant restrictions are satisfied for s. Since all blocks are valid in $\mathcal{A}^{\mathrm{b}}$, the unraveled interpretation is a model of $\mathcal{C}$ and $\mathcal{A}$. □

3.3 Core Blocking Policies

We now present two policies for identifying core assertions. Each policy can be used with either single or pairwise core blocking.

The *simple core policy* is inspired by the following observation. Let $\mathcal{A}$ be a potentially infinite ABox obtained by applying the hypertableau calculus without blocking to an $\mathcal{EL}$ knowledge base $\mathcal{K}$, and let s and t be two individuals introduced by applying the $\geqslant$-rule to assertions of the form $(\geqslant n\, r.B)(s')$ and $(\geqslant m\, r'.B)(t')$. Then, $\mathcal{L}_{\mathcal{A}}(s) = \mathcal{L}_{\mathcal{A}}(t)$; in fact, the concept labels $\mathcal{L}_{\mathcal{A}}(s)$ and $\mathcal{L}_{\mathcal{A}}(t)$ depend only on the concept B. The policy thus makes such assertions $B(s)$ and $B(t)$ core in the hope that, if a knowledge base is sufficiently "$\mathcal{EL}$-like," then s would validly block t.

Definition 4. *The* simple core *policy marks all assertions as not core unless they are covered by one of the following rules.*

- *Each assertion $B(c_j)$ derived by applying the $\geqslant$-rule to an assertion of the form $(\geqslant n\, r.B)(a)$ is marked as core.*
- *Each assertion α' derived by the $\approx$-rule from an assertion α via merging is marked as core if and only if α is core.*
- *If an ABox contains α as a noncore assertion but some derivation rule derives α as a core assertion, the former assertion is replaced with the latter.*

Simple core blocking generates very small cores, but it can be imprecise and can therefore lead to frequent validation of blocks. For example, if s and t are individuals introduced by applying the $\geqslant$-rule as above, then inferences involving the predecessor of s can cause the propagation of new concepts to s, which might invalidate blocking.

Furthermore, if the knowledge base contains nondeterministic concepts, then nondeterministic inferences involving s and t may cause $\mathcal{L}_\mathcal{A}(s)$ and $\mathcal{L}_\mathcal{A}(t)$ to diverge, which can also invalidate blocking. We therefore define the following, stronger notion of cores.

Definition 5. *The* complex core *policy is the extension of the simple core policy in which, whenever the* Hyp*-rule derives an assertion $\sigma(V_j)$ using a mapping σ and an HT-clause $\gamma = \bigwedge_{i=1}^{m} U_i \to \bigvee_{j=1}^{n} V_j$, the assertion $\sigma(V_j)$ is marked as core if and only if $\sigma(V_j)$ is a concept assertion and*

- *$n > 1$, or*
- *$\sigma(V_j)$ is of the form $B(s)$ with s a successor of $\sigma(w)$ for some variable w in γ.*

The complex core policy is motivated by the fact that, when $\mathcal{EL}$-style algorithms are extended to expressive but deterministic DLs such as Horn-$\mathcal{SHIQ}$ [10], the concepts that are propagated to an individual from its predecessor uniquely determine the individual's label, so we mark all such assertions as core.

4   Empirical Evaluation

We implemented the different core blocking strategies in our HermiT reasoner and carried out a preliminary empirical evaluation. For the evaluation, we selected several ontologies commonly used in practice. We classified each ontology and tested the satisfiability of all concepts from the ontologies with the different blocking strategies. Our main measurement is the number individuals in the final pre-model since this number directly relates to the amount of memory required by the reasoner.

We conducted our tests on a 2.6 GHz Windows 7 Desktop machine with 8 GB of RAM. We used Java 1.6 allowing for 1 GB of heap space in each test. All tested ontologies, a version of HermiT that supports core blocking, and Excel spreadsheets containing test results are available online.[1]

Figures 1–4 contain concepts on the x-axis; however, concept names are not shown due to the high number of concepts. The concepts are ordered according to the performance under the standard blocking strategy reasoner. The y-axis either displays the number of individuals in the pre-models or the reasoning times in milliseconds. All reasoning times exclude loading and preprocessing times, since these are independent of the blocking strategy. Some figures employ a logarithmic scale to improve readability. The label *standard pairwise* refers to the standard pairwise anywhere blocking strategy, *complex pairwise* refers to pairwise core blocking with the complex core policy, etc.

Tables 1–2 show average measurements taken while testing the satisfiability of all concepts in an ontology. The meaning of various rows is as follows: *final pre-model size* shows the average number of individuals in the final pre-model; *finally blocked* shows the average number of blocked individuals in the final pre-model; and *number of validations* shows the average number of validations before a pre-model was found in which all blocks are valid; *time in ms* shows the average time to test concept satisfiability; and *validation part* shows the percentage of this time taken to validate blocks. Finally, all tables show the time needed to classify the ontology in the format *hours:minutes:seconds*.

---

[1] http://www.hermit-reasoner.com/coreBlocking.html
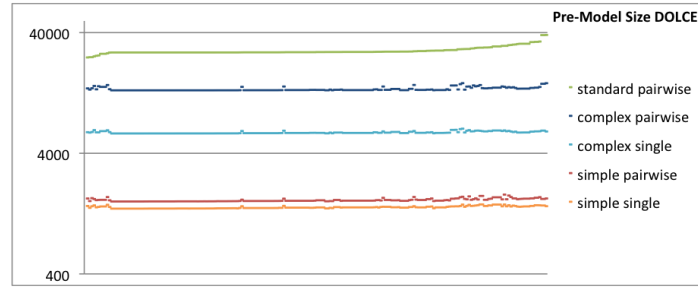
Fig. 1. The number of individuals in the pre-models for all concepts in DOLCE
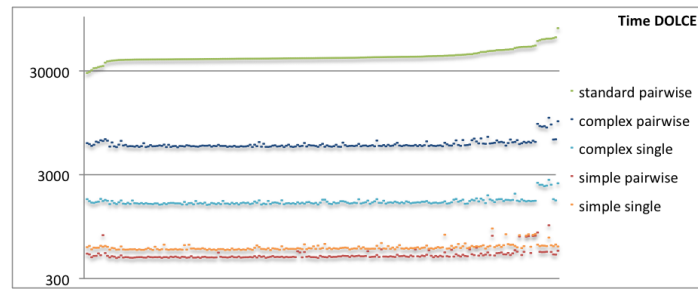
Fig. 2. The reasoning times in ms for testing the satisfiability of all concepts in DOLCE

DOLCE is a small but complex $\mathcal{SHOIQ}(\mathcal{D})$ ontology containing 209 concepts and 1,537 axioms that produce 2,325 HT-clauses. Core blocking works particularly well on DOLCE. The pre-model sizes (see Figure 1) and the reasoning times (see Figure 2) for all core blocking variants are consistently below those obtained with the standard anywhere blocking strategy. The simple single core blocking strategy gives the smallest pre-models but the reasoning times are slightly smaller for the simple pairwise strategy. This is because the simple single strategy produces more invalid blocks and, consequently, requires more expansion and (expensive) validation cycles before a final pre-model is found (see Table 1). Overall, the strategies work very well because DOLCE does not seem to be very highly constrained and many blocks are valid immediately.

GALEN is the original version of the GALEN medical ontology dating from about 10 years ago. Apart from CB [10], which implements an extension of an $\mathcal{EL}$-style algorithm to Horn-$\mathcal{SHIQ}$ [10], HermiT is currently the only reasoner that can classify this ontology. GALEN is a Horn-$\mathcal{SHIF}$ ontology containing 2,748 concepts and 4,979 axioms that produce 8,189 HT-clauses, and it normally requires pairwise blocking. GALEN is unusual in that it contains 2,256 "easy" concepts that are satisfied in very small pre-models (< 200 individuals) and 492 "hard" concepts that are satisfied in very large pre-models ($> 35,000$ individuals) for the standard blocking strategy. The classification times in Table 2 take all concepts into account; in all other cases we omit the measurements for the "easy" concepts since they do not show much difference between the different blocking strategies and just clutter the presentation. As for DOLCE, the

Table 1. Average measurements over all concepts in DOLCE and the classification time

	standard pairwise	complex pairwise	complex single	simple pairwise	simple single
final pre-model size	28,310	13,583	5,942	1,634	1,426
finally blocked	19,319	9,341	4,241	1,207	1,046
number of validations	—	1.03	1.06	1.09	2.09
time in ms	41,821	5,970	1,663	511	601
validation part	—	2.17%	3.98%	48.84%	65.63%
classification time	01:18:32	00:24:03	00:08:43	00:03:45	00:05:29
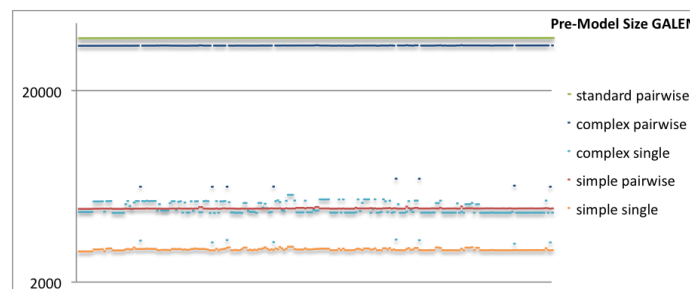
Fig. 3. The number of individuals in the pre-models for the (hard) concepts in GALEN

simple single core blocking strategy produces the most significant reduction in model size (see Figure 3). Although this strategy requires the most validation rounds, and these take up 86% of the overall reasoning time, this strategy is still the fastest (see Figure 4) since the reduction in model sizes compensates for the expensive block validations.

The only optimization in HermiT that needs adapting in order to work with core blocking is the *blocking cache*: once a pre-model for a concept is constructed, parts of the pre-model are reused in the remaining subsumption tests [4]. This dramatically reduces the overall classification time. The blocking cache can only be used on ontologies without nominals; in out test suite only GALEN falls into that category. Although the blocking cache could in principle be adapted for use with core blocking, this has not yet been implemented, so we switched this optimization off.

The foundational model of anatomy (FMA) is a domain ontology about human anatomy [2]. The ontology is one of the largest OWL ontologies available, containing 41,648 concepts and 122,617 $\mathcal{ALCOIF}(\mathcal{D})$ axioms, and it is transformed into 125,346 HT-clauses and 3,740 ABox assertions. We initially tried to take the same measurements for FMA as for the other ontologies; however, after 20 hours we processed only about 10% of the concepts (5,288 out of 41,648), so we aborted the test. Only the single simple core blocking strategy was able to process all 5,288 concepts. The pairwise simple core strategy stayed within the memory limit, but was significantly slower and suffered from 5 timeouts due to our imposition of a 2 minute time limit per concept. The standard blocking strategy exceeded either the memory or the time limit on 56 concepts, the pairwise complex core strategy on 70, and the single complex core strategy on 37 concepts. Therefore, we produced complete measurements only with single simple core
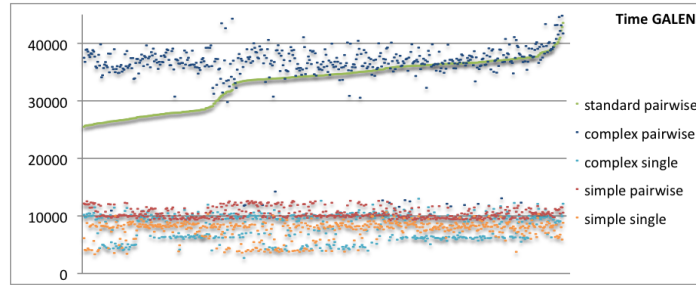
Fig. 4. The reasoning times for testing the satisfiability of the (hard) concepts in GALEN

Table 2. Average measurements over (hard) concepts in GALEN and the classification time

	standard pairwise	complex pairwise	complex single	simple pairwise	simple single
final pre-model size	37,747	33,557	4,876	4,869	2,975
finally blocked	19,290	19,726	2,234	1,896	1,247
number of validations	—	9.18	12.65	8.87	13.91
time in ms	33,293	36,213	8,050	10,485	7,608
validation part	—	27.47%	74.91%	81.47%	86.78%
classification time	03:50:01	04:35:12	01:07:18	01:27:50	01:02:44

blocking, using which HermiT was able to classify the entire ontology in about 5.5 hours, discovering 33,431 unsatisfiable concepts. The ontology thus seems to contain modeling errors that went undetected so far due to lack of adequate tool support. The unsatisfiability of all of these concepts was detected before blocking validation was required. The sizes of the ABoxes constructed while processing unsatisfiable concepts is included in the final pre-model size in Table 3, although these are not strictly pre-models since they contain a clash.

We also tested how much memory is necessary to construct all pre-models for DOLCE and GALEN under different blocking strategies. Starting with 16MB, we doubled the memory until the tested strategy could build all pre-models. The simple and complex core blocking strategies require as little as 64MB and 128MB of memory, respectively, whereas the standard blocking technique requires 512MB.

5   Discussion

In this paper we presented several novel blocking strategies that can improve the performance of DL reasoners by significantly reducing the size of the pre-models generated during satisfiability tests. Although we expected complex core blocking to work better on knowledge bases in expressive DLs, the evaluation shows that the simple core policy clearly outperforms the complex core policy regarding space and time on all tested ontologies. On more complex ontologies, the memory requirement with core blocking seems to decrease significantly.

Table 3. Average measurements over FMA with the single simple core strategy

final pre-model size	1,747	finally blocked	1,074
time in ms	518	validation part	0.00%
number of validations	0.2	classification time	05:31:23

On ontologies where very few individuals are blocked (e.g., Wine) the new strategies cannot really reduce the sizes of the pre-models [3]; however, they do not seem to have a negative effect on the reasoning times either.

The current publicly available version of HermiT (1.2.2) uses simple single core blocking as its default blocking strategy for ontologies with nominals; for ontologies without nominals it uses standard anywhere blocking with the blocking cache optimization, an optimization that has not yet been extended to core blocking.

Blocking validation is not highly optimized in our prototypical implementation. This is most apparent for the single simple core strategy that causes the most invalid blocks and where block validation takes 86% of the time for GALEN. Only the significant model size reductions allows this strategy to nevertheless be the fastest. We believe that we can significantly improve the performance in the future. We identified the two most common reasons for invalid blocks: the $\geqslant$-rule is applicable to an assertion from $\mathsf{val}_{\mathcal{A}_\ell}(s)$ of a blocked individual, or the Hyp-rule is applicable to the assertions from the temporary ABox of the predecessor of a directly blocked individual. Testing for these two cases first should reduce the overall time of validity tests. Finally, we shall adapt the blocking cache technique to core blocking.

References

1. Horrocks, I., Sattler, U.: Optimised reasoning for $\mathcal{SHIQ}$. In: Proc. of ECAI-02. (2002)
2. Golbreich, C., Zhang, S., Bodenreider, O.: The foundational model of anatomy in owl: Experience and perspectives. J. of Web Semantics: Science, Services and Agents on the World Wide Web **4**(3) (2006) 181–195
3. Glimm, B., Horrocks, I., Motik, B.: Optimized dl reasoning via core blocking. Technical report, University of Oxford (2010) http://www.comlab.ox.ac.uk/files/2743/paper.pdf.
4. Motik, B., Shearer, R., Horrocks, I.: Hypertableau reasoning for description logics. J. of Artificial Intelligence Research **173**(14) (2009) 1275–1309
5. Horrocks, I., Kutz, O., Sattler, U.: The even more irresistible $\mathcal{SROIQ}$. In: Proc. KR-06. (2006) 57–67
6. Baumgartner, P., Schmidt, R.A.: Blocking and Other Enhancements for Bottom-Up Model Generation Methods. In: Proc. of IJCAR-06. Volume 4130 of LNCS. (2006) 125–139
7. Ding, Y., Haarslev, V.: Tableau Caching for Description Logics with Inverse and Transitive Roles. In: Proc. DL. (2006)
8. Goré, R., Widmann, F.: Sound global state caching for $\mathcal{ALC}$ with inverse roles. In: Proc. of TABLEAUX 2009. LNCS (2009) 205–219
9. Donini, F.M., Massacci, F.: EXPTIME tableaux for $\mathcal{ALC}$. Artificial Intelligence Journal **124**(1) (2000) 87–138
10. Kazakov, Y.: Consequence-driven reasoning for horn SHIQ ontologies. In: Proc. of IJCAI-09. (2009) 2040–2045

Correcting Access Restrictions to a Consequence

Martin Knechtel[1] and Rafael Peñaloza[2]

[1] SAP Research Center Dresden, Germany
martin.knechtel@sap.com
[2] Theoretical Computer Science TU Dresden, Germany
penaloza@tcs.inf.tu-dresden.de

Abstract. Recent research has shown that annotations are useful for representing access restrictions to the axioms of an ontology and their implicit consequences. Previous work focused on computing a consequence's access restriction efficiently from the restrictions of its implying axioms. However, a security administrator might not be satisfied since the intended restriction differs from the one obtained through these methods. In this case, one is interested in finding a minimal set of axioms which need changed restrictions. In this paper we look at this problem and present algorithms based on ontology repair for solving it. Our first experimental results on large scale ontologies show that our methods perform well in practice.

1 Introduction

Description Logics (DL) [1] have been successfully used to model a wide variety of real-world application domains. The relevant portions of these domains are described through a DL ontology and highly optimized reasoners can then be used to deduce facts implicitly described in the ontology. In information systems with a huge ontology it is desirable to restrict the access of users to only a portion of the whole ontology, selected in accordance to an appropriate criterion. Motivations might be reducing information overload, filtering with respect to a trust level, or controlled access following a strict policy. For the access control scenario, each axiom is assigned a privacy level and each user is assigned a security clearance. A user can then see only those axioms whose privacy level is exceeded by the clearance of the user. One naive approach would be to maintain a separate sub-ontology obtained from one big ontology for each possible security clearance which means that any update in the ontology needs to be propagated to each of the sub-ontologies, and any change in the privacy levels or security clearances may result in a full recomputation of the sub-ontologies. Moreover, this would require separate reasoning for each sub-ontology. In order to avoid this, one rather keeps only the big ontology and stores the access information for axioms and users so that they can be retrieved easily. The approach proposed in [2] is to use a labeling lattice $(L, \leq)$. Every axiom and user gets a label in L assigned, and the sub-ontology accessible to a user with label ℓ is the set of all axioms whose label is greater than or equal to ℓ. In [2] it was also shown that

any implicit consequence c from the ontology can be assigned a label, called a *boundary*, such that deciding whether a user has access to c requires again only a computationally cheap label comparison.

DL systems consist of an ontology which represents explicit knowledge and a reasoner which makes implicit consequences of this knowledge explicit. The explicit and implied knowledge is exploited by the application by interacting with the DL system. A correct access labeling of an ontology is a difficult task. Indeed, several seemingly harmless axioms might possibly be combined to deduce information that is considered private. On the other hand, an over-restrictive labeling of axioms may cause public information to be inaccessible to some users. If the knowledge engineer finds that the boundary for a given consequence differs from the desired one, then she would like to automatically receive suggestions on how to modify the labeling function and correct this error. In this paper we present some methods in this direction. We assume that the knowledge engineer knows the exact boundary ℓ_g that the consequence c should receive, and propose a set $\mathcal{S}$ of axioms of minimal cardinality such that if all the axioms in $\mathcal{S}$ are relabeled to ℓ_g, then the boundary of c will be ℓ_g. We call $\mathcal{S}$ a *change set*.

We show that the main ideas from axiom-pinpointing [11, 10, 8, 4, 3] can be exploited in the computation of a change set and present a hitting set tree-based black-box approach that yields the desired set. Our experimental results at the end of the paper show that our algorithms behave well in practice.

2   Preliminaries

To keep our presentation and results as general as possible, we impose only minimal restrictions to our ontology language. We just assume that an *ontology* is a finite set, whose elements are called *axioms*, such that every subset of an ontology is itself an ontology. If $\mathcal{O}' \subseteq \mathcal{O}$ and $\mathcal{O}$ is an ontology, then $\mathcal{O}'$ is called a *sub-ontology* of $\mathcal{O}$. A *monotone consequence relation* $\models$ is a binary relation between ontologies $\mathcal{O}$ and *consequences* c such that if $\mathcal{O} \models c$, then for every ontology $\mathcal{O}' \supseteq \mathcal{O}$ it holds that $\mathcal{O}' \models c$. If $\mathcal{O} \models c$, we say that c *follows from* $\mathcal{O}$ or that $\mathcal{O}$ *entails* c. An ontology language specifies which sets of axioms are admitted as ontologies. Consider, for instance, a Description Logic $\mathcal{L}$. Then, an ontology is a finite set of general concept inclusion axioms (GCIs) of the form $C \sqsubseteq D$, with C, D $\mathcal{L}$-concept descriptions and assertion axioms of the form $C(a)$, with C an $\mathcal{L}$-concept description and a an individual name. Examples of consequences are subsumption relations $A \sqsubseteq B$ for concept names A, B.

If $\mathcal{O} \models c$, we may be interested in finding the axioms responsible for this fact. A sub-ontology $\mathcal{S} \subseteq \mathcal{O}$ is called a *MinA* for $\mathcal{O},c$ if $\mathcal{S} \models c$ and for every $\mathcal{S}' \subset \mathcal{S}, \mathcal{S}' \not\models c$. The dual notion of a MinA is that of a diagnosis. A *diagnosis* for $\mathcal{O},c$ is a sub-ontology $\mathcal{S} \subseteq \mathcal{O}$ such that $\mathcal{O} \setminus \mathcal{S} \not\models c$ and $\mathcal{O} \setminus \mathcal{S}' \models c$ for all $\mathcal{S}' \subset \mathcal{S}$.

For a lattice $(L, \leq)$ and a set $K \subseteq L$, we denote as $\bigoplus_{\ell \in K} \ell$ and $\bigotimes_{\ell \in K} \ell$ the *join* (least upper bound) and *meet* (greatest lower bound) of K, respectively. We consider that ontologies are *labeled* with elements of the lattice. More formally,

for an ontology $\mathcal{O}$ there is a labeling function lab that assigns a *label* $\mathsf{lab}(a) \in L$ to every element a of $\mathcal{O}$. We will often use the notation $L_{\mathsf{lab}} := \{\mathsf{lab}(a) \mid a \in \mathcal{O}\}$.

For a user labeled with $\ell \in L$, we denote as $\mathcal{O}_{\geq \ell}$ the sub-ontology $\mathcal{O}_{\geq \ell} := \{a \in \mathcal{O} \mid \mathsf{lab}(a) \geq \ell\}$ visible for him. The sub-ontologies $\mathcal{O}_{\leq \ell}, \mathcal{O}_{=\ell}, \mathcal{O}_{\neq \ell}, \mathcal{O}_{\not\geq \ell}$, and $\mathcal{O}_{\not\leq \ell}$ are defined analogously. This notion is extended to sets of labels in the natural way, e.g. $\mathcal{O}_{=K} := \{a \in \mathcal{O} \mid \mathsf{lab}(a) = \ell \text{ for some } \ell \in K\}$. Conversely, for a sub-ontology $\mathcal{S} \subseteq \mathcal{O}$, we define $\lambda_{\mathcal{S}} := \bigotimes_{a \in \mathcal{S}} \mathsf{lab}(a)$ and $\mu_{\mathcal{S}} := \bigoplus_{a \in \mathcal{S}} \mathsf{lab}(a)$. An element $\ell \in L$ is called *join prime relative to* L_{lab} if for every $K_1, \ldots, K_n \subseteq L_{\mathsf{lab}}$, $\ell \leq \bigoplus_{i=1}^{n} \lambda_{K_i}$ implies that there is $i, 1 \leq i \leq n$ such that $\ell \leq \lambda_{K_i}$. For instance, in Figure 1, ℓ_1 and ℓ_4 are the only elements that are not join prime relative to $L_{\mathsf{lab}} = \{\ell_1, \ldots, \ell_5\}$, since $\ell_1 \leq \ell_2 \oplus \ell_4$ but neither $\ell_1 \leq \ell_2$ nor $\ell_1 \leq \ell_4$ and similarly $\ell_4 \leq \ell_5 \oplus \ell_3$ but neither $\ell_4 \leq \ell_5$ nor $\ell_4 \leq \ell_3$. Join prime elements relative to L_{lab} are called *user labels*. The set of all user labels is denoted as U. When dealing with labeled ontologies, the reasoning problem of interest consists on the computation of a boundary for a consequence c. Intuitively, the boundary divides the user labels ℓ of U according to whether $\mathcal{O}_{\geq \ell}$ entails c or not.

Definition 1 (Boundary). *Let $\mathcal{O}$ be an ontology and c a consequence. An element $\nu \in L$ is called a* boundary *for $\mathcal{O},c$ if for every join prime element relative to L_{lab} ℓ it holds that $\ell \leq \nu$ iff $\mathcal{O}_{\geq \ell} \models c$.*

Given a user label ℓ_u, we will say that the user *sees* a consequence c if $\ell_u \leq \nu$ for some boundary ν. The following lemma relating MinAs and boundaries was shown in [2].

Lemma 1. *If $\mathcal{S}_1, \ldots, \mathcal{S}_n$ are all MinAs for $\mathcal{O},c$, then $\bigoplus_{i=1}^{n} \lambda_{\mathcal{S}_i}$ is a boundary for $\mathcal{O},c$.*

A dual result, which relates the boundary with the set of diagnoses, also exists. The proof follows easily from the definitions given in this section.

Lemma 2. *If $\mathcal{S}_1, \ldots, \mathcal{S}_n$ are all diagnoses for $\mathcal{O},c$, then $\bigotimes_{i=1}^{n} \mu_{\mathcal{S}_i}$ is a boundary for $\mathcal{O},c$.*

Example 1. Let $(L_d, \leq_d)$ be the lattice shown in Figure 1, and $\mathcal{O}$ a labeled ontology from a marketplace in the Semantic Web with the following axioms

> $a_1 : EUecoService \sqcap HighperformanceService(ecoCalculatorV1)$
> $a_2 : HighperformanceService$
> $\quad \sqsubseteq ServiceWithLowCustomerNr \sqcap LowProfitService$
> $a_3 : EUecoService \sqsubseteq ServiceWithLowCustomerNr \sqcap LowProfitService$
> $a_4 : ServiceWithLowCustomerNr \sqsubseteq ServiceWithComingPriceIncrease$
> $a_5 : LowProfitService \sqsubseteq ServiceWithComingPriceIncrease$

where the function lab assigns to each axiom the labels as shown in Figure 1. This ontology entails $c : ServiceWithComingPriceIncrease(ecoCalculatorV1)$. The MinAs for $\mathcal{O},c$ are $\{a_1, a_2, a_4\}, \{a_1, a_2, a_5\}, \{a_1, a_3, a_4\}, \{a_1, a_3, a_5\}$, and its diagnoses are $\{a_1\}, \{a_2, a_3\}, \{a_4, a_5\}$. Using Lemma 2, we can compute the boundary as $\mu_{\{a_1\}} \otimes \mu_{\{a_2, a_3\}} \otimes \mu_{\{a_4, a_5\}} = \ell_1 \otimes \ell_2 \otimes \ell_4 = \ell_3$. Valid user labels are $\ell_0, \ell_2, \ell_3, \ell_5$ which represent user roles as illustrated. Only for ℓ_0 and ℓ_3, c is visible.
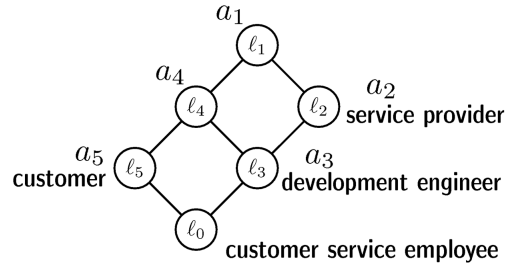
Fig. 1. Lattice $(L_d, \leq_d)$ with 4 user labels and an assignment of 5 axioms to labels

3 Modifying the Boundary

Once the boundary for a consequence c has been computed, it is possible that the knowledge engineer or the security administrator considers this solution erroneous. For instance, the boundary may express that a given user u is able to deduce c, although this was not intended. Alternatively, the boundary may imply that c is a very confidential consequence, only visible to a few, high-clearance users, while in reality c should be more publicly available.

Example 2. The boundary ℓ_3 computed in Example 1 expresses that the consequence c can only be seen by the development engineers and customer service employees (see Figure 1). It could be, however, that c is not expected to be accessible to development engineers, but rather to customers. In that case, we wish to modify the boundary of c to ℓ_5.

The problem we face is how to change the labeling function so that the computed boundary corresponds to the desired label in the lattice. This problem can be formalized and approached in several different ways. In our approach, we fix a goal label ℓ_g and try to modify the labeling of as few axioms as possible so that the boundary equals ℓ_g.

Definition 2. *Let $\mathcal{O}$ be an ontology,* lab *a labeling function, $\mathcal{S} \subseteq \mathcal{O}$ and $\ell_g \in L$ the* goal label. *The* modified assignment $\mathsf{lab}_{\mathcal{S},\ell_g}$ *is given by*

$$\mathsf{lab}_{\mathcal{S},\ell_g}(a) = \begin{cases} \ell_g, & \text{if } a \in \mathcal{S}, \\ \mathsf{lab}(a), & \text{otherwise.} \end{cases}$$

A sub-ontology $\mathcal{S} \subseteq \mathcal{O}$ is called a change set *(CS) for ℓ_g if the boundary for $\mathcal{O}$,c under the labeling function $\mathsf{lab}_{\mathcal{S},\ell_g}$ equals ℓ_g.*

Obviously, the original ontology $\mathcal{O}$ is always a CS set for any goal label if $\mathcal{O} \models c$. However, we are interested in performing minimal changes to the labeling function. Hence, we search for a CS of minimum cardinality.

Let ℓ_g denote the goal label and ℓ_c the computed boundary for c. If $\ell_g \neq \ell_c$, we have three cases: either (1) $\ell_g < \ell_c$, (2) $\ell_c < \ell_g$, or (3) ℓ_g and ℓ_c are incomparable.

In our example, the three cases are given by ℓ_g being ℓ_0, ℓ_4, and ℓ_5, respectively. Consider the first case, where $\ell_g < \ell_c$. Then, from Lemma 2 it follows that any diagnosis $\mathcal{S}$ is a CS for ℓ_g: since $\ell_g < \ell_c$, then for every diagnosis $\mathcal{S}'$, $\ell_g < \mu_{\mathcal{S}'}$. But then, under the new labeling $\mathsf{lab}_{\mathcal{S},\ell_g}$ we get that $\mu_{\mathcal{S}} = \ell_g$. And hence, when the greatest lower bound of all $\mu_{\mathcal{S}'}$ is computed, we obtain ℓ_g as a boundary. Using an analogous argument and Lemma 1, it is possible to show that if $\ell_c < \ell_g$, then every MinA is a CS for ℓ_g. The third case can be solved using a combination of the previous two: if ℓ_g and ℓ_c are incomparable, we can first set as a partial goal $\ell'_g := \ell_g \otimes \ell_c$. Thus, we can first solve the first case, to set the boundary to ℓ'_g, and then, using the second approach, modify this new boundary once more to ℓ_g. Rather than actually performing this task as a two-step computation, we can simply compute a MinA and a diagnose. The union of these two sets yields a CS. Unfortunately, the CS computed this way is not necessarily of minimum cardinality, even if the smallest diagnosis or MinA is used, as shown in the following example.

Example 3. Let $\mathcal{O},c$ and lab be as in Example 1. We then know that $\ell_c := \ell_3$ is a boundary for $\mathcal{O},c$. Suppose now that the goal label is $\ell_g := \ell_4$. Since $\ell_c < \ell_g$, we know that any MinA is a CS. Since all MinAs for $\mathcal{O},c$ have exactly three elements, any CS produced this way will have cardinality three. However, $\{a_2\}$ or $\{a_3\}$ are also valid CS, whose cardinalities are obviously smaller.

To understand why the minimality of MinAs is not sufficient for obtaining a minimum CS, we can look back to Lemma 1. This lemma says that in order to find a boundary, we need to compute the join of all $\lambda_{\mathcal{S}}$, with $\mathcal{S}$ a MinA, and $\lambda_{\mathcal{S}}$ the meet of the labels of all axioms in $\mathcal{S}$. But then, for any axiom $a \in \mathcal{S}$ such that $\ell_g \leq \mathsf{lab}(a)$, modifying this label to ℓ_g will have no influence in the result of $\lambda_{\mathcal{S}}$. In Example 3, there is a MinA $\{a_1, a_2, a_4\}$, where two axioms, namely a_1 and a_4 have a label greater or equal to $\ell_g = \ell_4$. Thus, the only axiom that needs to be relabeled is in fact a_2, which yields the minimum CS $\{a_2\}$ shown in the example. Basically, we consider every axiom $a \in \mathcal{O}$ such that $\ell_g \leq \mathsf{lab}(a)$ as *fixed* in the sense that it is superfluous for any CS. For this reason, we will deal with a generalization of MinAs and diagnoses, that we call IAS and RAS, respectively.

Definition 3 (IAS,RAS). *A minimal inserted axiom set (IAS) for ℓ_g is a subset $I \subseteq \mathcal{O}_{\ngeq \ell_g}$ such that $\mathcal{O}_{\geq \ell_g} \cup I \models c$ and for every $I' \subset I : \mathcal{O}_{\geq \ell_g} \cup I' \not\models c$. A minimal removed axiom set (RAS) for ℓ_g is a subset $R \subseteq \mathcal{O}_{\nleq \ell_g}$ such that $\mathcal{O}_{\nleq \ell_g} \setminus R \not\models c$ and for every $R' \subset R : \mathcal{O}_{\nleq \ell_g} \setminus R' \models c$.*

The following theorem justifies the use of IAS and RAS when searching for a CS of minimum cardinality.

Theorem 1. *Let ℓ_c be a boundary for $\mathcal{O},c$, ℓ_g the goal label, and m_R, m_I and m_U the cardinalities of the smallest RAS, the smallest IAS and the smallest union of an IAS and a RAS for ℓ_g, respectively. Then, for every IAS I and RAS R for ℓ_g it holds:*

  – if $\ell_g < \ell_c$ and $|R| = m_R$, then R is a CS of minimum cardinality,

- if $\ell_c < \ell_g$ and $|I| = m_I$, then I is a CS of minimum cardinality,
- if ℓ_c and ℓ_g are incomparable and $|R \cup I| = m_U$, then $I \cup R$ is a CS of minimum cardinality.

4   Computing a Minimal Change Set

Naïvely the smallest CS can be found by computing all CS and selecting the smallest. As explained above, the task of computing all CS is related to computing all diagnoses and all MinAs, which has been widely studied in recent years, and there exist black-box implementations based on the hitting set tree (HST) algorithm [7, 12]. Our approach to compute a minimal CS follows similar ideas. The HST algorithm makes repeated calls to an auxiliary procedure that computes a single CS. Further CS are found by building a tree, where nodes are labeled with CS and edges with axioms. If the CS labeling a node has n axioms ($\mathcal{S} := \{a_1, \ldots, a_n\}$), then this node will have n children: the edge to the i-th child labeled with a_i, the child labeled with a CS that is not allowed to contain neither a_i nor any ancestor's edge label. This ensures that each node is labeled with a CS distinct from those of its predecessors.

For the auxiliary procedure to compute a single CS, we will use two sub procedures extracting RAS and IAS, respectively. In Algorithm 1 we present a variation of the logarithmic MinA extraction procedure presented in [5] that is able to compute an IAS or stop once this has reached a size n and return a partial IAS. We also show the RAS variant in Algorithm 2. Given a goal label ℓ_g, if we want to compute a IAS or a partial IAS of size exactly n for a consequence c, then we would make a call to extract-partial-IAS($\mathcal{O}_{\geq \ell_g}, \mathcal{O}_{\not\geq \ell_g}, c, n$). Similarly, a call to extract-partial-RAS($\mathcal{O}_{\not\leq \ell_g}, \mathcal{O}_{\not\leq \ell_g}, c, n$) yields a RAS of size $\leq n$ or a partial RAS of size exactly n. The cardinality limit will be used to avoid unnecessary computations when looking for the smallest CS.

Given the procedures to extract RAS and IAS, Algorithm 3 extracts a CS. In order to label a node, we compute a CS with extract-partial-CS($\mathcal{O}, \mathsf{lab}, c, \ell_g, H, n$), where H is the set of all labels attached to edges on the way from the node to the root of the tree. Note that axioms in H are removed from the search space to extract the IAS and RAS. Furthermore, axioms in the IAS are considered as *fixed* for the RAS computation. The returned set is a CS of size $\leq n$ or a partial CS of size n.

Example 4. Returning to our running example, suppose now that we want to modify the label of consequence c to $\ell_g = \ell_5$. Algorithm 3 starts by making a call to extract-partial-IAS($\mathcal{O}_{\geq \ell_5}, \mathcal{O}_{\not\geq \ell_5}, c$).[1] A possible output for this call is $I = \{a_3\}$. We can then call extract-partial-RAS($\mathcal{O}_{\not\leq \ell_5} \setminus I, \mathcal{O}_{\not\leq \ell_5} \setminus I, c$), which may output e.g. the set $R = \{a_1\}$. Thus, globally the algorithm returns $\{a_3, a_1\}$, which can be easily verified to be a CS for ℓ_5.

One of the advantages of the HST algorithm is that the labels of any node are always ensured not to contain the label of any of its predecessor nodes. In

---

[1] For this example, we ignore the cardinality limit, as we want to find only one CS.

Algorithm 1 Compute (partial) IAS

Procedure extract-partial-IAS($\mathcal{O}_{\text{fix}}, \mathcal{O}_{\text{test}}, c, n$)
Input: $\mathcal{O}_{\text{fix}}$: fixed axioms; $\mathcal{O}_{\text{test}}$: axioms; c: consequence; n: limit
Output: first n elements of a minimal $\mathcal{S} \subseteq \mathcal{O}_{\text{test}}$ such that $\mathcal{O}_{\text{fix}} \cup \mathcal{S} \models c$
 1: **Global** $l := 0, n$
 2: **return** extract-partial-IAS-r($\mathcal{O}_{\text{fix}}, \mathcal{O}_{\text{test}}, c$)
Subprocedure extract-partial-IAS-r($\mathcal{O}_{\text{fix}}, \mathcal{O}_{\text{test}}, c$)

 1: **if** $n = l$ **then**
 2:    **return** $\emptyset$
 3: **if** $|\mathcal{O}_{\text{test}}| = 1$ **then**
 4:    $l := l + 1$
 5:    **return** $\mathcal{O}_{\text{test}}$
 6: $\mathcal{S}_1, \mathcal{S}_2 := \mathsf{halve}(\mathcal{O}_{\text{test}})$ (partition $\mathcal{O}_{\text{test}}$ so that $||\mathcal{S}_1| - |\mathcal{S}_2|| \leq 1$)
 7: **if** $\mathcal{O}_{\text{fix}} \cup \mathcal{S}_1 \models c$ **then**
 8:    **return** extract-partial-IAS-r($\mathcal{O}_{\text{fix}}, \mathcal{S}_1, c$)
 9: **if** $\mathcal{O}_{\text{fix}} \cup \mathcal{S}_2 \models c$ **then**
10:    **return** extract-partial-IAS-r($\mathcal{O}_{\text{fix}}, \mathcal{S}_2, c$)
11: $\mathcal{S}_1' := $ extract-partial-IAS-r($\mathcal{O}_{\text{fix}} \cup \mathcal{S}_2, \mathcal{S}_1, c$)
12: $\mathcal{S}_2' := $ extract-partial-IAS-r($\mathcal{O}_{\text{fix}} \cup \mathcal{S}_1', \mathcal{S}_2, c$)
13: **return** $\mathcal{S}_1' \cup \mathcal{S}_2'$

---

particular this means that even if we compute a partial CS, the algorithm will still correctly find all CS that do not contain any of the partial CS found during the execution. Since we are interested in finding the CS of minimum cardinality, we can set the limit n to the size of the smallest CS found so far. This limit is initially fixed to the size of the ontology. If extract-partial-CS outputs a set with fewer elements, we are sure that this is indeed a full CS, and our new smallest known CS. The HST algorithm will not find all CS in this way, but we can be sure that one CS with the minimum cardinality will be found. The idea of limiting cardinality for finding the smallest CS can be taken a step further by not expanding each node for all the axioms in it, but rather only on the first $n - 1$, where n is the size of the smallest CS found so far. This further reduces the search space by decreasing the branching factor of the search tree. Notice that the highest advantage of this second optimization appears when the HST is constructed in a depth-first fashion. In that case, a smaller CS found further below in the tree will reduce the branching factor of all its predecessors. So the cardinality limit reduces the search space in two dimensions: (1) the computation of a single CS is limited to n axioms and (2) only $n - 1$ axioms are expanded from each node. The following theorem shows that such a variant of the HST algorithm is correct.

Theorem 2. *Let $\mathcal{O}$ be an ontology, c a consequence with $\mathcal{O} \models c$, and ℓ_g a goal label. If m is the minimum cardinality of all CS for ℓ_g, the HST algorithm described above outputs a CS $\mathcal{S}$ such that $|\mathcal{S}| = m$.*

Algorithm 2 Compute (partial) RAS

Procedure extract-partial-RAS($\mathcal{O}_{\mathrm{nonfix}}, \mathcal{O}_{\mathrm{test}}, c, n$)
Input: $\mathcal{O}_{\mathrm{nonfix}}$: axioms; $\mathcal{O}_{\mathrm{test}} \subseteq \mathcal{O}_{\mathrm{nonfix}}$: axioms; c: consequence; n: limit
Output: first n elements of a minimal $\mathcal{S} \subseteq \mathcal{O}_{\mathrm{test}}$ such that $\mathcal{O}_{\mathrm{nonfix}} \setminus \mathcal{S} \not\models c$

1: **Global** $l := 0, \mathcal{O}_{\mathrm{nonfix}}, n$
2: **return** extract-partial-RAS-r($\emptyset, \mathcal{O}_{\mathrm{test}}, c$)

Subprocedure extract-partial-RAS-r($\mathcal{O}_{\mathrm{hold}}, \mathcal{O}_{\mathrm{test}}, c$)

1: **if** $n = l$ **then**
2:     **return** $\emptyset$
3: **if** $|\mathcal{O}_{\mathrm{test}}| = 1$ **then**
4:     $l := l + 1$
5:     **return** $\mathcal{O}_{\mathrm{test}}$
6: $\mathcal{S}_1, \mathcal{S}_2 := \mathsf{halve}(\mathcal{O}_{\mathrm{test}})$ (partition $\mathcal{O}_{\mathrm{test}}$ so that $||\mathcal{S}_1| - |\mathcal{S}_2|| \leq 1$)
7: **if** $\mathcal{O}_{\mathrm{nonfix}} \setminus (\mathcal{O}_{\mathrm{hold}} \cup \mathcal{S}_1) \not\models c$ **then**
8:     **return** extract-partial-RAS-r($\mathcal{O}_{\mathrm{hold}}, \mathcal{S}_1, c$)
9: **if** $\mathcal{O}_{\mathrm{nonfix}} \setminus (\mathcal{O}_{\mathrm{hold}} \cup \mathcal{S}_2) \not\models c$ **then**
10:    **return** extract-partial-RAS-r($\mathcal{O}_{\mathrm{hold}}, \mathcal{S}_2, c$)
11: $\mathcal{S}_1' := $ extract-partial-RAS-r($\mathcal{O}_{\mathrm{hold}} \cup \mathcal{S}_2, \mathcal{S}_1, c$)
12: $\mathcal{S}_2' := $ extract-partial-RAS-r($\mathcal{O}_{\mathrm{hold}} \cup \mathcal{S}_1', \mathcal{S}_2, c$)
13: **return** $\mathcal{S}_1' \cup \mathcal{S}_2'$

Proof. The described algorithm outputs a CS since the globally stored and finally returned $\mathcal{S}$ is only modified when the output of extract-partial-CS has size strictly smaller than the limit n, and hence only when this is indeed a CS itself. Suppose now that the output $\mathcal{S}$ is such that $m < |\mathcal{S}|$, and let $\mathcal{S}_0$ be a CS such that $|\mathcal{S}_0| = m$, which exists by assumption. Then, every set obtained by calls to extract-partial-CS has size strictly greater than m, since otherwise, $\mathcal{S}$ and n would be updated. Consider now an arbitrary set $\mathcal{S}'$ found during the execution through a call to extract-partial-CS, and let $\mathcal{S}_n' := \{a_1, \ldots, a_n\}$ be the first n elements of $\mathcal{S}'$. Since $\mathcal{S}'$ is a (partial) CS, it must be the case that $\mathcal{S}_0 \not\subseteq \mathcal{S}_n'$ since every returned CS is minimal in the sense that no axiom might be removed to obtain another CS. Then, there must be an $i, 1 \leq i \leq n$ such that $a_i \notin \mathcal{S}_0$. But then, $\mathcal{S}_0$ will still be a CS after axiom $\{a_i\}$ has been removed. Since this argument is true for all nodes, it is in particular true for all leaf nodes, but then they should not be leaf nodes, since a new CS, namely $\mathcal{S}_0$ can still be found by expanding the HST, which contradicts the fact that $\mathcal{S}$ is the output of the algorithm. $\qquad\square$

Example 5. Returning to our running example, suppose that we want to set the label of c to $\ell_g = \ell_0$. Algorithm 3 first calls extract-partial-RAS($\mathcal{O}_{\not\leq \ell_0}, \mathcal{O}_{\not\leq \ell_0}, c, 5$). A possible output of this call is $R = \{a_2, a_3\}$. The tree now branches through a_2 and a_3. In the first case it calls extract-partial-RAS($\mathcal{O}_{\not\leq \ell_0}, \mathcal{O}_{\not\leq \ell_0} \setminus \{a_2\}, c, 2$), which could yield $R = \{a_4, a_5\}$. This might be a partial CS since its size equals the cardinality limit. The next call extract-partial-RAS($\mathcal{O}_{\not\leq \ell_0}, \mathcal{O}_{\not\leq \ell_0} \setminus \{a_2, a_4\}, c, 2$) yields the smallest $R = \{a_1\}$, and the HST terminates. Notice that if $\{a_1\}$ had been the first change set found, the process would have immediately terminated.

Algorithm 3 Compute (partial) Change Set

Procedure extract-partial-CS$(\mathcal{O}, \mathsf{lab}, c, \ell_g, H, n)$

1: $\ell_c := \mathsf{hst\text{-}boundary}(\mathcal{O}, c)$ function defined in [2]

2: **return** extract-partial-CS$(\mathcal{O}, \mathsf{lab}, c, \ell_g,$
    $\ell_g \not< \ell_c \wedge \mathcal{O}_{\geq \ell_g} \not\models c,$
    $\ell_g \not> \ell_c \wedge \mathcal{O}_{\not\leq \ell_g} \models c,\ H, n)$

Procedure extract-partial-CS$(\mathcal{O}, \mathsf{lab}, c, \ell_g, \mathsf{is}_I, \mathsf{is}_R, H, n)$

Input: $\mathcal{O}, \mathsf{lab}$: labeled ontology; c: consequence; ℓ_g: goal label; is_I: decision to compute IAS; is_R: decision to compute RAS; H: HST edge labels; n: limit

Output: first n elements of a minimal CS $\mathcal{S} \subseteq \mathcal{O}$

1: **if** $1 \geq n$ or $\mathsf{is}_I \wedge \mathcal{O}_{\geq \ell_g} \cup (\mathcal{O}_{\not\geq \ell_g} \setminus H) \not\models c$ or $\mathsf{is}_R \wedge H \models c$ **then**

2:     **return** $\emptyset$ (HST normal termination)

3: **if** is_I **then**

4:     $I := \mathsf{extract\text{-}partial\text{-}IAS}(\mathcal{O}_{\geq \ell_g}, \mathcal{O}_{\not\geq \ell_g} \setminus H, c, n)$

5: **if** is_R and $\mathcal{O}_{\not\leq \ell_g} \setminus I \models c$ **then**

6:     $R := \mathsf{extract\text{-}partial\text{-}RAS}(\mathcal{O}_{\not\leq \ell_g} \setminus I, \mathcal{O}_{\not\leq \ell_g} \setminus (I \cup H), c, n - |I|)$

7: **return** $I \cup R$
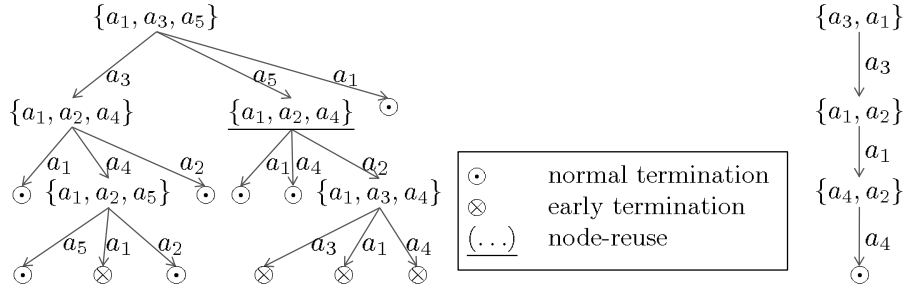
Fig. 2. Hitting Set Trees to compute all MinAs (left) and a minimal change set for $\ell_g = \ell_5$ (right)

Efficient implementations of the original version of the HST algorithm rely on several optimizations. Two standard optimizations described in the literature are node-reuse and early path termination (see, e.g. [7, 12, 2]). Node-reuse keeps a history of all nodes computed so far in order to avoid useless (and usually expensive) calls to the auxiliary procedure that computes a new node. Early path termination, on the other hand, prunes the hitting set tree by avoiding expanding nodes when no new information can be derived from further expansion. In order to avoid unnecessary confusion, we have described the modified HST algorithm without including these optimizations. However, it should be clear that both, node-reuse and early path termination, can be included in the algorithm without destroying its correctness. The implementation used for our experiments contain these two optimizations.

Figure 2 shows the expansion of the HST trees when computing all MinAs and all diagnoses, in comparison with the one obtained for computing a minimal change set for $\ell_g = \ell_5$, using the ontology and consequences of Example 1.

This paper's results are a continuation of work in [9], where we had not one Hitting Set Tree Algorithm but two separately for the smallest IAS and the smallest RAS. This paper's variant is guaranteed to find the smallest CS, as given in the Proof above. For a CS consisting of an IAS and a RAS, computing a smallest of both does not necessarily yield the smallest CS, as the following example shows. Assume $\{a_1, a_2\}, \{a_2, a_3\}$ are the smallest RAS and $\{a_1, a_4\}$ is the smallest IAS, then $\{a_1, a_2, a_4\}$ is the smallest CS, but choosing one smallest IAS and one smallest RAS might yield a CS of cardinality 4. In [9] we also investigated the performance gain by taking not only advantage of fixing a subset of the axioms and limiting cardinality but also by taking the labels of the remaining axioms into account.

5   Empirical Evaluation

We implemented and evaluated our algorithms empirically with large practical ontologies. The test system is identical to one used previous work in [2], so we describe it here very briefly. The two labeling lattices used are $(L_d, \leq_d)$, already introduced in Figure 1, and the linear order $(L_l, \leq_l)$ with 6 elements $L_l = L_d = \{\ell_0, \ldots, \ell_5\}$ with $\leq_l := \{(\ell_n, \ell_{n+1}) \mid \ell_n, \ell_{n+1} \in L_l \wedge 0 \leq n \leq 5\}$. We used the two ontologies O^{Snomed} and O^{Funct} with different expressivity and types of consequences for our experiments. The Systematized Nomenclature of Medicine, Clinical Terms (Snomed ct) is a comprehensive medical and clinical ontology which is built using the Description Logic (DL) $\mathcal{EL}+$. From the January/2005 release of the DL version, which contains 379,691 concept names, 62 object property names, and 379,704 axioms, and entails more than five million subsumptions, we used a sampled set of 27,477 positive subsumptions. O^{Funct} is an OWL-DL ontology for functional description of mechanical engineering solutions [6]. It has 115 concept names, 47 object property names, 16 data property names, 545 individual names, 3,176 axioms, and the DL expressivity is $\mathcal{SHOIN}(\mathbf{D})$. Its 716 consequences are 12 subsumption and 704 instance relationships (class assertions).

We computed the boundary ℓ_c of each consequence c of the ontologies with the algorithms described in [2] and then computed the change set for goal boundary $\ell_g = \ell_3$. Consequences where $\ell_c = \ell_g$ were not considered. Thus, from the 716 consequences in O^{Funct}, we have 415 remaining with labeling lattice $(L_d, \leq_d)$ and 474 remaining with $(L_l, \leq_l)$. From the 27,477 consequences in O^{Snomed} we have 23,695 remaining with labeling lattice $(L_d, \leq_d)$ and 25,897 with $(L_l, \leq_l)$.

Table 1 contains results for the 4 combinations of the two ontologies and the two labeling lattices. For each of them we tested our algorithm against the basic approach of computing all MinAs and diagnoses. We limit the number of computed MinAs and CS to 10, so our algorithms might not find the smallest change set before reaching the limit. We measure the quality of the presented variants given those limitations at execution time. Table 1 lists the ratio of correct solutions where at least 1 correct change set was computed, and the ratio of optimal solutions where the limit was not reached during the computation

Ont.	Lattice	Variant	Runtime Limit per goal	Time (minutes)	Ratio of correct solutions	Ratio of optimal solutions
O^{Funct}	nonlinear	all diagnoses and MinAs	≤ 10 MinA	44.05	96%	47%
		a minimal CS	≤ 10 (partial) CS	8.66	100%	98%
	linear	all diagnoses and MinAs	≤ 10 MinA	54.46	98%	49%
		a minimal CS	≤ 10 (partial) CS	8.61	100%	99%
O^{Snomed}	nonlinear	all diagnoses and MinAs	≤ 10 MinA	184.76	100%	75%
		a minimal CS	≤ 10 (partial) CS	10.51	100%	100%
	linear	all diagnoses and MinAs	≤ 10 MinAs	185.35	100%	75%
		a minimal CS	≤ 10 (partial) CS	28.14	100%	98%

Table 1. Results comparing our with the reference approach in 4 test settings

and thus yielded the smallest change set possible. Notice however that the ratio of cases with the minimal change set successfully computed might be higher, including those where the limitation was reached but the minimal change set was already found.

Computing all MinAs is clearly outperformed by our optimized approach. To conclude, fixed sub-ontologies and cardinality limit are optimizations with reasonable impact.

6   Conclusions

Previous work has studied labeled ontologies and methods to compute boundaries for their consequences. In this paper we considered scenarios where a security administrator is not satisfied with the access restriction level computed from the access restriction levels of the implying axioms. Based on ontology repair techniques we developed algorithms to compute a change set of minimal cardinality, which contains axioms to be relabeled in order to yield a consequence's access restriction. The base problem, finding the smallest MinA and diagnosis without computing all of them might be interesting beyond our application domain. Our algorithms take advantage of (1) fixing a subset of the axioms which are not part of the search space, and (2) limiting cardinality of change sets to be computed in the Hitting Set Tree to the smallest known change set. We implemented the algorithms and have first experimental results on large-scale ontologies which show that our ideas yield tangible improvements in both the execution time and the quality of the solution.

As future work we intend to study the problem of finding change sets for several consequences (each with its own goal label) simultaneously. We will also look at more flexible restrictions on the goal label and other criteria for the minimality of change sets for example not counting the amount of changed axiom label assignments but the distance of the new from the old label in the lattice or the amount of other consequence's boundaries changed.

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, 2007.

2. F. Baader, M. Knechtel, and R. Peñaloza. A generic approach for large-scale ontological reasoning in the presence of access restrictions to the ontology's axioms. In *Proc. of ISWC'09*, volume 5823 of *LNCS*, pages 49–64, 2009.

3. F. Baader and R. Peñaloza. Automata-based axiom pinpointing. *Journal of Automated Reasoning*, 2010. Special Issue: IJCAR 2008. To appear.

4. F. Baader and R. Peñaloza. Axiom pinpointing in general tableaux. *Journal of Logic and Computation*, 2010. Special Issue: Tableaux'07. To appear.

5. F. Baader and B. Suntisrivaraporn. Debugging SNOMED CT using axiom pinpointing in the description logic $\mathcal{EL}^+$. In *Proc. of KR-MED'08*, 2008.

6. A. Gaag, A. Kohn, and U. Lindemann. Function-based solution retrieval and semantic search in mechanical engineering. In *Proc. of ICED'09*, 2009.

7. A. Kalyanpur, B. Parsia, M. Horridge, and E. Sirin. Finding all justifications of OWL DL entailments. In *Proc. of ISWC'07 + ASWC'07*, 2007.

8. A. Kalyanpur, B. Parsia, E. Sirin, and J. A. Hendler. Debugging unsatisfiable classes in OWL ontologies. *J. Web Sem.*, 3(4):268–293, 2005.

9. M. Knechtel and R. Peñaloza. A Generic Approach for Correcting Access Restrictions to a Consequence. In *Proc. of ESWC'10*, 2010. To appear.

10. T. Meyer, K. Lee, R. Booth, and J. Z. Pan. Finding maximally satisfiable terminologies for the description logic $\mathcal{ALC}$. In *Proc. of AAAI'06*, 2006.

11. S. Schlobach and R. Cornet. Non-standard reasoning services for the debugging of description logic terminologies. In *Proc. of IJCAI'03*, pages 355–362, 2003.

12. B. Suntisrivaraporn. *Polynomial-Time Reasoning Support for Design and Maintenance of Large-Scale Biomedical Ontologies*. PhD thesis, TU Dresden, 2008.

The modular structure of an ontology:
an empirical study[*]

Chiara Del Vescovo, Bijan Parsia, Uli Sattler, and Thomas Schneider

School of Computer Science, University of Manchester, UK
{delvescc,bparsia,sattler,schneider}@cs.man.ac.uk

1   Introduction

Why modularize an ontology?   In software engineering, modularly structured systems are desirable, all other things being equal. Given a well-designed modular program, it is generally easier to process, modify, and analyze it and to reuse parts by exploiting the modular structure. As a result, support for modules (or components, classes, objects, packages, aspects) is a commonplace feature in programming languages.

Ontologies are computational artefacts akin to programs and, in notable examples, can get quite large as well as complex, which suggests that exploiting modularity might be fruitful, and research into modularity for ontologies has been an active area for ontology engineering. Recently, a lot of effort has gone into developing *logically sensible* modules, that is, modules which offer strong logical guarantees for intuitive modular properties. One such guarantee is called *coverage* and means that the module captures all the ontology's knowledge about a given set of terms (signature)—a kind of dependancy isolation. A module in this sense is therefore a subset of the axioms in an ontology that provides coverage for a signature, and each possible signature determines such a module. Coverage is provided by modules based on conservative extensions, but also by efficiently computable approximations, such as modules based on syntactic locality [1].

The task of extracting one such module given a signature, which we call GetOne in this section, is well understood and starting to be deployed in standard ontology development environments, such as Protégé 4,[1] and online.[2] The extraction of locality-based modules has already been effectively used in the field for ontology reuse [2] as well as a subservice for incremental reasoning [3].

While GetOne is an important and useful service, it, by itself, tells us nothing about the modular structure of the ontology as a whole. The modular structure is determined by the set of *all* modules and their inter-relations, or at least a suitable subset thereof. We call the task of a-posteriori determining the modular structure of an ontology GetStruct and, in order to determine that structure, we investigate here the task GetAll of extracting all modules. While GetOne is well-understood and often computationally cheap, GetAll has hardly been examined

---

[1] http://www.co-ode.org/downloads/protege-x
[2] http://owl.cs.manchester.ac.uk/modularity

for module notions with strong logical guarantees, with the work described in [4] being a promising exception. GetOne also requires the user to know in advance the right set of terms to input to the extractor: we call this a *seed* signature for the module and note that one module can have several such seed signatures. Since there are non-obvious relations between the final signature of a module and its seed signature, users are often unsure how to generate a proper request and confused by the results. If they had access to the overall modular structure of the ontology determined by GetAll, they could use it to guide their extraction choices. In general, supported by the experience described in [4], we believe that, by revealing the modular structure of an ontology, we can obtain information about its topicality, connectedness, structure, superfluous parts, or agreement between actual and intended modeling. Our use-cases include: for ontology engineers, the possibility of checking the ontology design—for example, if the module relative to some terms corresponds to the intuitive "knowledge encapsulation" about that term; for end users, the possibility to support the understanding of what the ontology deals with, and where the topic they want to focus on is placed within the ontology.

In the worst case, the number of all modules of an ontology is exponential in the number of terms or axioms in the ontology, in fact in the minimum of these numbers. Hence, it is possibly the case that ontologies have too many modules to extract all of them, even with an optimized extraction methodology. Even with only polynomially many modules, there may be too many for direct user inspection. Then, some other form of analysis would have to be designed.

We report on experiments to obtain or estimate this number and to evaluate the modular structure of an ontology where we succeeded to compute it.

Related work. One solution to GetStruct is described in [4,5] via partitions related to $\mathcal{E}$-connections. The resulting modules are disjoint, and this technique is of limited applicability—when it succeeds, it divides an ontology into three kinds of modules: (A) those which import vocabulary from others, (B) those whose vocabulary is imported, and (C) isolated parts. In experiments and user experience, the numbers of parts extracted were quite low and often corresponded usefully to user understanding. For instance, the tutorial ontology Koala, consisting of 42 logical axioms, is partitioned into one A-module about animals and three B-modules about genders, degrees and habitats.

It has also been shown in [4] that certain combinations of these parts provide coverage. For Koala, such a combination would still be the whole ontology. In general, partitions were observed to be too coarse grained; sometimes extraction resulted in a single partition even though the ontology seemed well structured. Furthermore, the robustness properties of the parts (e.g., under vocabulary extension) are not as well-understood as those of locality-based modules. However, partitions share efficient computability with locality-based modules.

Another approach to GetStruct is described in [6]. It underlies the tool Mod-Onto, which aims at providing support for working with ontology modules that is similar to, and borrows intuitions from, software modules. This approach is logic-based and a-posteriori but, to the best of our knowledge, it has not been ex-

amined whether such modules provide coverage in the above sense. Furthermore, ModOnto does not aim at obtaining *all* modules from an ontology.

Another procedure for partitioning an ontology is described in [7]. However, this method only takes the concept hierarchy of the ontology into account and can therefore not provide the strong logical guarantee of coverage.

Among the a-posteriori approaches to GetOne, some provide logical guarantees such as coverage, and others do not. The latter are not of interest for this paper. The former are usually restricted to DLs of low expressivity, where deciding conservative extensions—which underly coverage—is tractable. Prominent examples are the module extraction feature of CEL [8] and the system MEX [9]. However, we aim at an approach that covers DLs up to OWL 2.

There are a number of logic-based approaches to modularity that function a-priori, i.e., the modules of an ontology have to be specified in advance by features that are added to the underlying (description) logic and whose semantics is well-defined. These approaches often support distributed reasoning; they include C-OWL [10], $\mathcal{E}$-connections [11], Distributed Description Logics [12], and Package-Based Description Logics [13]. Even in these cases, however, we may want to understand the modular structure of the syntactically delineated parts. Furthermore, with imposed structure, it is not always clear that that structure is correct. Decisions about modular structure have to be taken early in the modeling which may enshrine misunderstandings. Examples were reported in [4], where user attempts to capture the modular structure of their ontology by separating the axioms into separate files were totally at odds with the analyzed structure.

Overview of the experiments and results. In the following, we will report on experiments performed to extract all modules from several ontologies as a first solution candidate for GetAll. We have considered three notions of modules based on syntactic locality—they all provide coverage, but differ in the size of the modules and in other useful properties of modules, see [14]—and extracted such modules for all subsets of the terms in the respective ontology. At this stage, we are mainly interested in module *numbers* rather than sizes or interrelations: the main concern is whether the suspected combinatorial explosion occurs. In order to test the latter, we have sampled subsets of each ontology and performed a full modularization on each subontology, measuring the relation between module number and subontology size for each ontology. We have also tried different approaches to reduce the number of modules to the most "interesting" ones.

An extended version of this paper and additional material for the evaluation of the experiments, such as spreadsheets and charts, are available online [15,16].

2 Preliminaries

Underlying description logics. We assume the reader to be familiar with OWL and the underlying description logics (DLs) [17,18]. We consider an ontology to be a finite set of axioms, which are of the form $C \sqsubseteq D$ or $C \equiv D$, where C, D are (possibly complex) concepts, or $R \sqsubseteq S$, where R, S are (possibly inverse) roles. Since we are interested in the logical part of an ontology, we

disregard non-logical axioms. However, it is easy to add the corresponding an-
notation and declaration axioms in retrospect once the logical part of a module
has been extracted. This is included in the publicly available implementation of
locality-based module extraction in the OWL API.[3]

Let $\mathsf{N_C}$ be a set of concept names, and $\mathsf{N_R}$ a set of role names. A *signa-
ture* Σ is a set of terms, i.e., $\Sigma \subseteq \mathsf{N_C} \cup \mathsf{N_R}$. We can think of a signature
as specifying a topic of interest. Axioms that only use terms from Σ can be
thought of as "on-topic", and all other axioms as "off-topic". For instance,
if $\Sigma = \{\mathsf{Animal}, \mathsf{Duck}, \mathsf{Grass}, \mathsf{eats}\}$, then $\mathsf{Duck} \sqsubseteq \exists\mathsf{eats}.\mathsf{Grass}$ is on-topic, while
$\mathsf{Duck} \sqsubseteq \mathsf{Bird}$ is off-topic.

Any concept or role name, ontology, or axiom that uses only terms from Σ is
called a Σ-*concept*, Σ-*role*, Σ-*ontology*, or Σ-*axiom*. Given any such object X,
we call the set of terms in X the *signature of X* and denote it with $\widetilde{X}$.

Conservative extensions and locality. Conservative extensions (CEs) cap-
ture the above described encapsulation of knowledge. A CE-based module for a
signature Σ of an ontology $\mathcal{O}$ preserves all entailments η in $\mathcal{O}$ that can be for-
mulated using symbols Σ only. For more precise definitions, see e.g., [19,20,15].

Unfortunately, CEs are hard or even impossible to decide for many DLs, see
[21,19,14]. Therefore, approximations have been devised. We focus on *syntactic
locality* (here for short: locality). Locality-based modules can be efficiently com-
puted and provide coverage, that is, they capture *all* the relevant entailments,
but not necessarily *only* those [1,22]. Although locality is defined for the DL
$\mathcal{SHIQ}$, it is straightforward to extend it to $\mathcal{SHOIQ}(D)$ (see [1,22]), and the
implementation of locality-based module extraction in the OWL API. We are
using the notion of locality from [14].

Definition 1. An axiom α is called *syntactically $\bot$-local ($\top$-local) w.r.t. signa-
ture Σ* if it is of the form $C^\bot \sqsubseteq C$, $C \sqsubseteq C^\top$, $R^\bot \sqsubseteq R$ $(R \sqsubseteq R^\top)$, or $\mathsf{Trans}(R^\bot)$
$(\mathsf{Trans}(R^\top))$, where C is an arbitrary concept, R is an arbitrary role name,
$R^\bot \notin \Sigma$ $(R^\top \notin \Sigma)$, and $C^\bot$ and $C^\top$ are from $\mathrm{Bot}(\Sigma)$ and $\mathrm{Top}(\Sigma)$ as defined in
Figure 2 (a) (Figure 2 (b)).

It has been shown in [1] that $\mathcal{M} \subseteq \mathcal{O}$ and all axioms in $\mathcal{O} \setminus \mathcal{M}$ being $\bot$-local
(or all axioms being $\top$-local) w.r.t. $\Sigma \cup \widetilde{M}$ is sufficient for $\mathcal{M}$ to be a CE-based
module for Σ of $\mathcal{O}$. The converse does not hold in general.

It is described in [1] how to obtain modules of $\mathcal{O}$ for $\top$- and $\bot$-locality.
We are using the notions of $\top$-, $\bot$-, $\top\bot^*$- and $\bot\top^*$-modules from [14, Def.
4]. That is, given an ontology $\mathcal{O}$, a *seed signature* Σ and a module notion
$x \in \{\top, \bot, \top\bot^*, \bot\top^*\}$, we denote the x-module of $\mathcal{O}$ w.r.t. Σ by $x\text{-mod}(\Sigma, \mathcal{O})$. If
we do not specify x, we generally speak of a *locality-based* module. It is straight-
forward to show that $\top\bot^*\text{-mod}(\Sigma, \mathcal{O}) = \bot\top^*\text{-mod}(\Sigma, \mathcal{O})$ for each $\mathcal{O}$ and Σ. In
contrast, $\top$- and $\bot$-modules do not have to be equal—in fact, the former are
usually larger than the latter. Through the nesting, $\top\bot^*\text{-mod}(\Sigma, \mathcal{O})$ is always
contained in $\top\text{-mod}(\Sigma, \mathcal{O})$ and $\bot\text{-mod}(\Sigma, \mathcal{O})$. Finally, we want to point out that,
for $\mathcal{M} = x\text{-mod}(\Sigma, \mathcal{O})$, neither $\Sigma \subseteq \widetilde{\mathcal{M}}$ nor $\widetilde{\mathcal{M}} \subseteq \Sigma$ needs to hold.

---

[3] http://owlapi.sourceforge.net

(a) $\perp$-Locality

Let $A^\perp, R^\perp \notin \Sigma$, $C^\perp \in \text{Bot}(\Sigma)$, $C_{(i)}^\top \in \text{Top}(\Sigma)$, $\bar{n} \in \mathbb{N} \setminus \{0\}$

$$\text{Bot}(\Sigma) ::= A^\perp \mid \perp \mid \neg C^\top \mid C \sqcap C^\perp \mid C^\perp \sqcap C \mid \exists R.C^\perp \mid \geqslant \bar{n}\, R.C^\perp \mid \geqslant \bar{n}\, R^\perp.C$$

$$\text{Top}(\Sigma) ::= \top \mid \neg C^\perp \mid C_1^\top \sqcap C_2^\top \mid \geqslant 0\, R.C$$

(b) $\top$-Locality

Let $A^\top, R^\top \notin \Sigma$, $C^\perp \in \text{Bot}(\Sigma)$, $C_{(i)}^\top \in \text{Top}(\Sigma)$, $\bar{n} \in \mathbb{N} \setminus \{0\}$

$$\text{Bot}(\Sigma) ::= \perp \mid \neg C^\top \mid C \sqcap C^\perp \mid C^\perp \sqcap C \mid \geqslant \bar{n}\, R.C^\perp$$

$$\text{Top}(\Sigma) ::= A^\top \mid \top \mid \neg C^\perp \mid C_1^\top \sqcap C_2^\top \mid \geqslant \bar{n}\, R^\top.C^\top \mid \geqslant 0\, R.C$$

Figure 1. Syntactic locality conditions

The following property of locality-based modules has been shown in [1] for $x \in \{\perp, \top\}$. The transfer to nested modules is straightforward.

Proposition 2. *Let $\mathcal{O}$ be an ontology, Σ, Σ' be a signatures, $x \in \{\perp, \top, \top\perp^*\}$; let $\mathcal{M} = x\text{-mod}(\Sigma, \mathcal{O})$ and $\Sigma \subseteq \Sigma' \subseteq \Sigma \cup \widetilde{\mathcal{M}}$. Then $x\text{-mod}(\Sigma', \mathcal{O}) = \mathcal{M}$.*

Genuine modules. In order to limit the overall number of modules, we introduce the notion of a *genuine module*. Intuitively, a given module $\mathcal{M}$ of an ontology is *fake* if it can be partitioned into a set $\{\mathcal{M}_1, \ldots, \mathcal{M}_n\}$ of smaller modules such that each "relevant" entailment of $\mathcal{M}$ follows from some $\mathcal{M}_i$.

Since the definition of relevance of an entailment within a module is still in progress, we use a computable approximation, described in Definition 3. We first introduce some useful notions. Let $\mathcal{O}$ be an ontology and $\mathfrak{M}$ be the set of all modules of $\mathcal{O}$. An atomic concept C is called *top-level for $\mathcal{M}$ (bottom-level for $\mathcal{M}$)* if $\mathcal{O} \models A \sqsubseteq C$ ($\mathcal{O} \models C \sqsubseteq A$) for all atomic concepts $A \in \widetilde{M}$. A set $\{\Sigma_1, \ldots, \Sigma_n\}$ of signatures is called *$\mathcal{M}$-almost pairwise disjoint* if every two signatures Σ_i, Σ_j with $i \neq j$ are disjoint or share at most one symbol, which is an atomic concept, and if the set of all these shared atomic concepts contains at most one top-level and at most one bottom-level concept for $\mathcal{M}$.

Definition 3. A module $\mathcal{M} \in \mathfrak{M}$ is *fake* if there exist modules $\mathcal{M}_1, \ldots, \mathcal{M}_n \in \mathfrak{M}$ such that $\mathcal{M} = \mathcal{M}_1 \uplus \cdots \uplus \mathcal{M}_n$ and the set $\{\widetilde{\mathcal{M}_1}, \ldots, \widetilde{\mathcal{M}_n}\}$ is *$\mathcal{M}$-almost pairwise disjoint*. Otherwise $\mathcal{M}$ is called *genuine*.

In particular, if a module is fake, then it consists of disjoint modules whose signatures *almost* disjoint. For example, in Koala, we have a fake module about habitat that consists of a rainforest and a dryforest submodule, which only overlap in the term habitat and do not share any other terms and no axioms. Fake modules are uninteresting because $\mathcal{M}$ being fake means that different seed signatures of the $\mathcal{M}_i$ do not interact with each other. Given that often the overall number of modules appears to grow exponentially with the size of the subontology, a natural question arising is whether this is only caused by the fact that there are exponentially many fake modules.

3 Description of the experiments

Ontologies. We performed the experiments on several existing ontologies that we consider to be well designed and sufficiently diverse. We used Koala, Mereology, University, People, miniTambis, OWL-S, Tambis and Galen, whose sizes (axioms + terms) range from $42 + 25$ to $4{,}528 + 3{,}161$. See [15] for an overview of sizes and expressivities as well as an explanation of the choice criteria.

Full modularization. Let $\mathcal{O}$ be the ontology to be modularized. Our goal is to find all modules of $\mathcal{O}$, i.e., to compute $\{x\text{-mod}(\Sigma, \mathcal{O}) \mid \Sigma \in \widetilde{\mathcal{O}}\}$. In order to keep track of the seed signatures, we seek an algorithm which, given $\mathcal{O}$ as input, returns a representation of all pairs $(\Sigma, \mathcal{M})$ with $\Sigma \subseteq \widetilde{\mathcal{O}}$ and $\mathcal{M} = x\text{-mod}(\Sigma, \mathcal{O})$.

The most naïve procedure is to simply traverse through all seed signatures Σ, extract the corresponding module and add it to the output. Since there are exponentially many seed signatures, this is not feasible—even for Koala, 2^{25} runs of even the easiest test is unrealistic. Fortunately, we have good reasons to believe that there are significantly fewer modules than seed signatures in realistic ontologies: first, Proposition 2 says that, given the locality-based module $\mathcal{M} = x\text{-mod}(\Sigma, \mathcal{O})$, every seed signature Σ' that extends Σ and is a subset of $\Sigma \cup \widetilde{\mathcal{M}}$ yields the same module $\mathcal{M}$. Second, even if two seed signatures Σ and Σ' are not in such a relationship, the modules for Σ and Σ' can still coincide.

It should be noted, however, that there are very simple families of ontologies that already have exponentially many genuine modules, for instance the taxonomies $\mathcal{T}_n = \{B \sqsubseteq A\} \cup \{C_i \sqsubseteq B \mid i = 1, \ldots, n\}$, or the ontologies $\mathcal{O}_n = \{B_i \sqsubseteq A,\ C_i \sqsubseteq B_i \mid i = 1, \ldots, n\} \cup \{B_i \sqsubseteq \neg B_j \mid 1 \leqslant i < j \leqslant n\}$. More examples are given in [15]. However, we have not been able to construct any example with exponentially many genuine modules for inferred concept hierarchies of *bounded width*. In contrast, there are ontologies of arbitrary size that have exactly one module or at most quadratically many modules. Thus, while the worst case number of modules is high, it is not analytically impossible that real ontologies would have a reasonable number of modules. Unfortunately, empirically, as discussed in Section 4, this does not seem to be the case. We are not aware of any systematic study about theoretically possible module numbers.

Since a module can have several seed signatures, we represent a module as a pair consisting of $\mathcal{M}$ and the set $\mathcal{S}$ of all *minimal* seed signatures Σ for which $\mathcal{M}$ is a module. Whenever a module for a new seed signature Σ' is to be computed, we first check whether Σ' satisfies $\Sigma \subseteq \Sigma' \subseteq \Sigma \cup \widetilde{\mathcal{M}}$ for some already extracted module $\mathcal{M}$ and some associated minimal seed signature Σ. Only if this is not the case, the module $\mathcal{M}' = x\text{-mod}(\Sigma', \mathcal{O})$ is computed. If $\mathcal{M}'$ coincides with some already extracted module $\mathcal{M}$, then Σ' is added to the set of minimal seed signatures associated with $\mathcal{M}$; otherwise the pair $(\{\Sigma'\}, \mathcal{M}')$ is added to the set of extracted modules. This is performed by Algorithm 1, which calls Alg. 2. For soundness and completeness of Algorithm 1 and optimizations, see [15].

Sampling via subsets. In preliminary testing it soon became apparent that even our optimized algorithm would not reasonably terminate on even fairly small ontologies. Since we have a search space exponential in the size of the

---

Algorithm 1 Extract all x-modules

1: **Input:**   an ontology $\mathcal{O}$ with signature $\widetilde{\mathcal{O}}$
2: **Output:** a set $\mathfrak{M} = \{(\mathcal{S}_1, \mathcal{M}_1), \ldots, (\mathcal{S}_n, \mathcal{M}_n)\}$
            of all x-modules of $\mathcal{O}$,
            associated with their sets of
            minimal seed signatures (SSigs)

3: {*Start*: extract x-modules for all singleton SSigs}
4: $\mathfrak{M} \leftarrow \emptyset$
5: **for all** $t \in \widetilde{\mathcal{O}}$ **do**
6:    $\mathcal{M} \leftarrow$ extract x-module of $\mathcal{O}$ w.r.t. $\{t\}$
7:    **call** integrate($\mathfrak{M}, \{t\}, \mathcal{M}$)
8: **end for**

9: {*Extension*: iteratively add single terms to SSigs}
10: **while** $\mathfrak{M}$ contains $(\mathcal{S}, \mathcal{M})$ with marked $\Sigma \in \mathcal{S}$ **do**
11:    $(\mathcal{S}, \mathcal{M}) \leftarrow$ some elem. of $\mathfrak{M}$ with marked $\Sigma \in \mathcal{S}$
12:    $\Sigma \leftarrow$ some marked element of $\mathcal{S}$
13:    **for all** $t \in \widetilde{\mathcal{O}} \setminus (\Sigma \cup \widetilde{\mathcal{M}})$ **do**
14:       $\mathcal{M}' \leftarrow$ extract x-module of $\mathcal{O}$ w.r.t. $\Sigma \cup \{t\}$
15:       **call** integrate($\mathfrak{M}, \Sigma \cup \{t\}, \mathcal{M}'$)
16:    **end for**
17:    unmark Σ in $(\mathcal{S}, \mathcal{M})$
18: **end while**
19: **return** $\mathfrak{M}$

Algorithm 2
integrate($\mathfrak{M}, \Sigma, \mathcal{M}$)

---

for all $(\mathcal{S}', \mathcal{M}') \in \mathfrak{M}'$ **do**
  if $\mathcal{M} = \mathcal{M}'$ **then**
    $\mathcal{S}' \leftarrow \mathcal{S}' \cup \{\Sigma\}$
    mark Σ in $(\mathcal{S}', \mathcal{M}')$
    return
  end if
end for
$\mathfrak{M} \leftarrow \mathfrak{M} \cup (\{\Sigma\}, \mathcal{M})$
mark Σ in $(\{\Sigma\}, \mathcal{M})$
return

---

ontology and potentially exponentially many modules, it was not clear whether the problem was that our algorithm was not sufficiently optimized (so that the search space dominated) or that the output was impossible to generate. Since it is pointless to try to optimize an algorithm for a function whose output is exponentially large in the size of the typical input, it is imperative to determine whether real-world ontologies do have an exponential number of modules. This last question is one goal of the experiments described in this paper.

In order to test the hypothesis that real-life ontologies have an exponential number of modules, we have sampled subsets of different sizes from the ontologies considered. By fully modularizing each of these subsets, we can draw conclusions about the asymptotic relation between its size and the number of modules obtained. Randomly generated subsets would tend to contain unrelated axioms, taken out of the context in which they have been included by the ontology developers. Since unrelated axioms, or ontologies with many unrelated terms, generally yield many modules, it would be harder to justify the hypothesis that real-world ontologies tend to have significantly less than exponentially many modules if we used arbitrary, less coherent subsets.

We have therefore chosen to let each subset be a module for a randomly generated signature—although we are aware that such subsets are more modular

than necessary because ontologies are not normally developed modularly. But this is not a problem: it can only cause us to *understate* the number of modules.

We have sampled 10 signatures of each size between 0 and a threshold of 50 (or ontology's signature size if that was smaller). In some cases where the subset sizes were not optimally distributed (e.g., when small subsets were missing), we sampled 30 signatures of each size. For these signatures, we have extracted the $\top\bot^*$-modules, excluding duplicates, and ordered them by size. Then we have fully modularized all subsets in descending order, aborting when a single modularization took longer than a preset timeout of 20, 60 or 600 minutes, see Section 4 for an explanation of that choice. For each subset, we counted the number of all modules and of its genuine modules. See [15] for computer specifications.

4   Results

Module numbers for full modularization. Figure 2 shows the full modularization of Koala and Mereology for the module types $\top$, $\bot$ and $\top\bot^*$. In the case of $\top\bot^*$, we also determined genuine modules, denoted by $\top\bot^*_g$. In addition to the number of modules, we have listed the runtime and four aggregations of module sizes, where "size" refers to the number of logical axioms. Since the number of axioms is a syntax-dependent measure, we plan to include other measures, such as the number of terms and the sum of the sizes of all axioms, in future work.

	Koala				Mereology			
	$\top$	$\bot$	$\top\bot^*$	$\top\bot^*_g$	$\top$	$\bot$	$\top\bot^*$	$\top\bot^*_g$
#Modules	**12**	**520**	**3,660**	**2,143**	**40**	**552**	**1952**	**272**
Time [s]	*0*	*1*	*9*	*34*	*0*	*6*	*158*	*158*
Min size	29	6	0	0	18	0	0	0
Avg size	35	27	23	23	26	25	20	22
Max size	42	42	42	42	40	40	40	38
Std. dev.	4	6	6	6	6	7	8	8

$\top\bot^*_g = $ *genuine* $\top\bot^*$ modules.   "Size" $=$ number of logical axioms.

Figure 2. Full modularization of Koala and Mereology

For both ontologies, the number of modules increases from $\top$- via $\bot$- to $\top\bot^*$ modules as expected: as mentioned before, $\top$-modules tend to be bigger, and therefore more modules coincide in this case. However, $\top$-modules are too coarse-grained: most of them comprise almost the whole ontology, and all have a size of at least 29 (69% of Koala) or 18 (41% of Mereology).

The extracted $\bot$-modules yield a more fine-grained picture, although all their sizes for Koala are still above 6 (14%). We already pay for this with an increase in the number of modules by a factor of more than 43 (Koala) and 14 (Mereology). With $\top\bot^*$, smaller modules are included, but for the price of another increase

in module numbers by a factor of 7 (Koala) and 3.5 (Mereology). For a more fine-grained modularization, we also pay with an increased extraction time. See [15] for comments on the observed differences between Koala and Mereology.

Attempts to fully modularize ontologies larger than Koala and Mereology with the described algorithm did not succeed. We cancelled each such computation after several hours, when thousands of modules have been extracted.

Although Koala and Mereology have much fewer modules than the theoretical upper bound of 2^{25}, we still get too many for (regular) inspection by ontology users. We have therefore tried two more ways to reduce their modules to fewer "interesting" ones. Both approaches showed no significant impact, see [15].

Module numbers for subset sampling. After carrying out the subset sampling technique described in Section 3, we are strongly convinced that most of the ontologies examined exhibit the feared exponential behavior. Figure 3 shows scatterplots of the number of $\top\bot^*$ modules (genuine $\top\bot^*$ modules) versus the size of the subset for People and Koala. Each chart shows an exponential trendline, which is the least-squares fit through the data points by using the equation $m = ce^{bn}$, where n is the size of the subset, m is the number of modules, e is the base of the natural logarithm, and c, b are constants. This equation and the corresponding determination coefficient (R^2 value) are given beneath each chart. Spreadsheets with the underlying data, as well as spreadsheets and charts for the other ontologies, can be found at [16]. The R^2 values and trendline equations for the examined ontologies are summarized in Figure 4, where we also included the estimated number of modules for the full ontology as per the equation, the timeout used and the overall runtime.
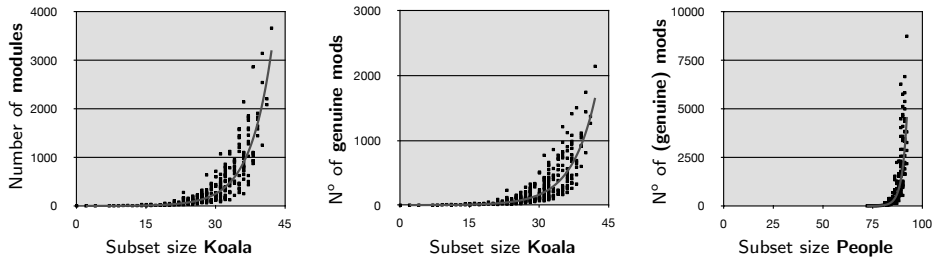
Figure 3. Numbers of modules versus subset sizes for Koala and People

The scatterplots and determination coefficients for the first six ontologies in Figure 4 provide strong evidence that the number of modules depends exponentially on the size of the subset.

Weight analysis for Koala. Even if we consider only genuine modules, there are ontologies that have exponentially many of them. In order to focus on even fewer, "interesting" modules, we have devised the measures *cohesion* and *pulling power*. Thy are based on the number of seed signatures (SSigs) of a module $\mathcal{M}$ and the number of terms in $\widetilde{\mathcal{M}}$. An SSig Σ of $\mathcal{M}$ is called *minimal* (MSSig) if

Ontology	Confidence R_{m}^2	R_{g}^2	Trendline equation m	g	Estimate m	g	Timeout [min]	Runtime [min]
People	.95	.95	both $2 \cdot 10^{-13} e^{.41n}$		10^6	10^6	20	148
Mereology	.87	.94	$1.2 e^{.16n}$	$1.1 e^{.13n}$	10^3	10^2	—	4
Koala	.90	.88	$.45 e^{.21n}$	$.50 e^{.19n}$	10^3	10^3	—	4
Galen	.94	.86	$1.2 e^{.24n}$	$1.6 e^{.16n}$	NaN	NaN	60	288
University	.84	.83	$1.7 e^{.19n}$	$1.6 e^{.14n}$	10^4	10^3	20	354
OWL-S	.82	.84	$.0027 e^{.17n}$	$.0032 e^{.16n}$	10^{17}	10^{17}	60	73
Tambis	.75	.70	$1.1 e^{.22n}$	$1.4 e^{.13n}$	10^{58}	10^{33}	600	681
miniTambis	.47	.52	$2.6 e^{.18n}$	$2.5 e^{.14n}$	10^{14}	10^{10}	600	963

m, g $\top\bot^*$ modules, genuine $\top\bot^*$ modules
$R_{\mathrm{m}}^2, R_{\mathrm{g}}^2$ Determination coefficient of fitted trendlines
Estimate   Module numbers for full ontology as per trendline
NaN   Estimate is larger than 10^{142}

Figure 4. Witnesses for exponential behavior

there is no signature $\Sigma' \subset \Sigma$ that is an SSig of $\mathcal{M}$. If we ignore terms not present in the module, we speak of a *real MSSig* for $\mathcal{M}$: this is a signature $\Sigma' = \Sigma \cap \widetilde{\mathcal{M}}$ where Σ is an MSSig for $\mathcal{M}$. Let r, s, m be the number of real MSSigs for $\mathcal{M}$, the size of the smallest MSSig for $\mathcal{M}$, and the size of $\widetilde{\mathcal{M}}$.

The *cohesion* of $\mathcal{M}$ measures how strongly the terms in $\mathcal{M}$ are held together, as indicated by the number of seed signatures for $\mathcal{M}$. More precisely, the cohesion of $\mathcal{M}$ is defined to be the ratio r/s. The *pulling power* of $\mathcal{M}$ measures how many terms are needed in an MSSig to "pull" all terms into $\mathcal{M}$ that we find there. We define the *pulling power* of $\mathcal{M}$ to be the ratio m/s.

As a first draft, we define the *weight* of a module $\mathcal{M}$ to be the product of its cohesion and pulling power: $w = \frac{r \cdot m}{s^2}$. We computed the weight of all 3660 modules of Koala. The 11 heaviest modules and their set differences yield a partition of almost the whole ontology into 10 parts, each of which consists of terms that intuitively form a topic (subconcepts included): Animal; Person and isHardWorking; Student; Parent; Koala and Marsupial; TasmanianDevil; Quokka; Habitat; Degree; Gender. These topics reflect the core parts of the ontology. Axioms that do not occur among the heaviest modules tend to be those that we intuitively would call less important for the ontology, e.g., RainForest $\sqsubseteq$ Forest. The first 11 (34) modules cover 39 (42) out of all 42 logical axioms.

The next step will be to refine this measure and apply it to more ontologies and to find ways to extract heavy-weight modules separately.

5   Discussion and outlook

The fundamental conclusion is clear: the number of modules, even when we restrict our attention to genuine modules, is exponential in the size of the ontology for real ontologies. Our most reasonable estimates of the total number of modules in small to midsize ontologies (i.e., anything over 100 axioms) show

that full modularization is practically impossible. As we are computing locality based modules, which tend to be larger than conservative extension based modules, our results give us a lower bound on the number of modules.

It is, of course, possible that there are principled ways to reduce the target number of modules. We could use a coarser approximation, though that would be hard to justify on logical grounds. Attempts to use "less minimal" modules or to heuristically merge modules have exhibited bad behavior, with a strong tendency to collapse to very few modules that comprise most of the ontology.

We believe that this conclusion is robust, even with the failure of our experiments on Tambis and miniTambis to uncover exponential behavior. As we said in Section 4, our expectation is that a longer timeout will reveal the problematic behavior. We also suspect a connection between the relatively low number of modules for these two ontologies and the fact that they have a large number of unsatisfiable concepts. For details, see [15]. The ratio between genuine and fake modules can be seen as a measure of axiomatic richness, at least indicating how strongly the axioms in the ontology connect its terms: the fewer of its modules are fake, the more "mutually touching" its terms are.

Attempts at estimating the module number statistically were unhelpful too. We could randomly draw a small number of seed signatures, compute their modules and use that number to estimate the number of all modules. We convinced ourselves using elementary stochastics that we cannot get a confident estimate by sampling only a small proportion of all seed signatures. See [15] for details.

While the outcome of the experiments is discouraging from the point of view of using the complete modularization in order to analyze the ontology, it does suggest several interesting lines of future work. First, we have already seen several features of ontologies that correlate well with a large or small number of modules. However, except for the phenomenon seen in Mereology, we do not have a verified explanation. Thus, for example, we need to get a precise picture of the relationship between justificatory and modular structure. Second, even if we cannot compute all modules, we may be able to compute a better approximation of their number. Given that signature sampling did not seem to help, we intend to explore sources of module number increase or reduction, such as the shape of the inferred concept hierarchy and patterns of axioms. Methodologically, it seems that artificial ontologies should be used, e.g., for confirmation of the relationship between justificatory structure and module number. Third, our preliminary experiments aimed at computing *heavy weight* ontologies are promising: our weights seem to capture nicely the cohesion and pulling power of a module, and the resulting heavy modules seem to correlate nicely with topics. We are currently investigating whether it is possible to compute all heavy modules without computing all modules, and also looking into a suitable notion of *building blocks* of modules. The latter concept is closely related to fake and genuine modules, which we are also investigating in more detail.

References

 1. Cuenca Grau, B., Horrocks, I., Kazakov, Y., Sattler, U.: Modular reuse of ontolo-

gies: Theory and practice. J. Artif. Intell. Res. **31** (2008) 273–318

2. Jimeno, A., Jiménez-Ruiz, E., Berlanga, R., Rebholz-Schuhmann, D.: Use of shared lexical resources for efficient ontological engineering. In: SWAT4LS-08. Volume 435 of CEUR. (2008)

3. Cuenca Grau, B., Halaschek-Wiener, C., Kazakov, Y.: History matters: Incremental ontology reasoning using modules. In: Proc. of ISWC/ASWC-07. Volume 4825 of LNCS. (2007) 183–196

4. Cuenca Grau, B., Parsia, B., Sirin, E., Kalyanpur, A.: Modularity and web ontologies. In: Proc. of KR-06. (2006) 198–209

5. Cuenca Grau, B., Parsia, B., Sirin, E.: Combining OWL ontologies using $\mathcal{E}$-connections. JWebSem **4**(1) (2006) 40–59

6. Bezerra, C., de Freitas, F.L.G., Zimmermann, A., Euzenat, J.: ModOnto: A tool for modularizing ontologies. In: Proc. WONTO-08. Volume 427 of CEUR. (2008)

7. Stuckenschmidt, H., Klein, M.: Structure-based partitioning of large concept hierarchies. In: Proc. of ISWC-04. Volume 3298 of LNCS. (2004) 289–303

8. Suntisrivaraporn, B.: Module extraction and incremental classification: A pragmatic approach for $\mathcal{EL}^+$ ontologies. In: Proc. of ESWC-08. Volume 5021 of LNCS. (2008) 230–244

9. Konev, B., Lutz, C., Walther, D., Wolter, F.: Logical difference and module extraction with CEX and MEX. In: Proc. of DL 2008. Volume 353 of CEUR. (2008)

10. Stuckenschmidt, H., van Harmelen, F., Bouquet, P., Giunchiglia, F., Serafini, L.: Using C-OWL for the alignment and merging of medical ontologies. In: Proc. KR-MED. Volume 102 of CEUR. (2004) 88–101

11. Kutz, O., Lutz, C., Wolter, F., Zakharyaschev, M.: E-connections of abstract description systems. Artificial Intelligence **156**(1) (2004) 1–73

12. Borgida, A., Serafini, L.: Distributed description logics: Assimilating information from peer sources. J. Data Semantics **1** (2003) 153–184

13. Bao, J., Voutsadakis, G., Slutzki, G., Honavar, V.: Package-based description logics. [23] 349–371

14. Sattler, U., Schneider, T., Zakharyaschev, M.: Which kind of module should I extract? In: DL 2009. Volume 477 of CEUR. (2009)

15. Del Vescovo, C., Parsia, B., Sattler, U., Schneider, T.: The modular structure of an ontology: an empirical study. Technical report, University of Manchester `http://www.cs.man.ac.uk/%7Eschneidt/publ/modstrucreport.pdf` .

16. Materials: `http://owl.cs.manchester.ac.uk/modproj/meat-experiment` .

17. Horrocks, I., Patel-Schneider, P.F., van Harmelen, F.: From $\mathcal{SHIQ}$ and RDF to OWL: The making of a web ontology language. JWebSem **1**(1) (2003) 7–26

18. Horrocks, I., Kutz, O., Sattler, U.: The even more irresistible $\mathcal{SROIQ}$. In: Proc. of KR-06. (2006) 57–67

19. Konev, B., Lutz, C., Walther, D., Wolter, F.: Formal properties of modularization. [23] 25–66

20. Lutz, C., Walther, D., Wolter, F.: Conservative extensions in expressive description logics. In: Proc. of IJCAI-07. (2007) 453–458

21. Ghilardi, S., Lutz, C., Wolter, F.: Did I damage my ontology? A case for conservative extensions in description logics. In: Proc. of KR-06. (2006) 187–197

22. Jiménez-Ruiz, E., Cuenca Grau, B., Sattler, U., Schneider, T., Berlanga Llavori, R.: Safe and economic re-use of ontologies: A logic-based methodology and tool support. In: Proc. of ESWC-08. Volume 5021 of LNCS. (2008) 185–199

23. Stuckenschmidt, H., Parent, C., Spaccapietra, S., eds.: Modular Ontologies: Concepts, Theories and Techniques for Knowledge Modularization. Volume 5445 of LNCS. Springer (2009)

Optimization Techniques
for Fuzzy Description Logics

Nikolaos Simou[1], Theofilos Mailis[1], Giorgos Stoilos[2], and Giorgos Stamou[1]

[1] Department of Electrical and Computer Engineering,
National Technical University of Athens,
Zographou 15780, Greece
{nsimou,theofilos,gstam}@image.ntua.gr
[2] Oxford University Computing Laboratory , Parks Road
Wolfson Building, Oxford OX1 3QD, United Kingdom
giorgos.stoilos@comlab.ox.ac.uk

Abstract. Sophisticated uncertainty representation and reasoning are
necessary for the alignment and integration of Web data from different
sources. For this purpose the extension of the Description Logics using
fuzzy set theory has been proposed, resulting to fuzzy Description Logics
(DLs). However, despite the fact that since the initial proposal a lot of
work has been done in the area, the practicability of very expressive
fuzzy DLs still remains open, due to the absence of practically scalable
systems. This paper presents optimization techniques that can improve
the performance of fuzzy-DL systems' reasoning.

1 Introduction

Fuzzy ontologies are envisioned to be very useful in the Semantic Web. Further-
more, the need for handling fuzzy and uncertain information is crucial to the
Web, since information and data along it may often be uncertain or imperfect.
This requirement for uncertainty representation has led W3C to set up the Un-
certainty Reasoning for the World Wide Web XG[3]. Currently $FiRE$[4] [13] and
$FuzzyDL$[5] [1] are the only existing systems for very expressive fuzzy description
logics (DLs) supporting the f_{KD}-$\mathcal{SHIN}$ and fuzzy $\mathcal{SHIf}$ languages respectively.
Furthermore, the $DeLorean$ reasoner that supports $f_{KD} - \mathcal{SROIQ}$ was recently
proposed in literature [3]. This reasoner does not implement a fuzzy tableau al-
gorithm but an algorithm that reduces a fuzzy knowledge base to a crisp one [2]
using $Pellet$ [12] for reasoning.

Despite the fact that the first proposal for fuzzy DLs was made by Straccia
in 1998 [16], since then little work has been done in order to permit the use
of expressive fuzzy DLs in realistic applications. The theoretical complexity of
the tableau reasoning algorithm for f-$\mathcal{SHIN}$, presented in [14], is 2-Nexptime

---

[3] http://www.w3.org/2005/Incubator/urw3/

[4] http://www.image.ece.ntua.gr/~nsimou/FiRE/

[5] http://gaia.isti.cnr.it/~straccia/software/fuzzyDL/fuzzyDL.html

that is very expensive. Therefore an implementation directly based on this very expensive algorithm, would result to a reasoner that could not be applied to real case scenarios. This problem was handled in crisp DLs, that also suffer from high complexity, by the use of optimization techniques [8, 18]. Using these techniques, a theoretically expensive computation can be converted to an equivalent of practically lower complexity. As a result many optimized reasoners were implemented for expressive DLs like FaCT++ [17], Racer [5] and Pellet [12] that can handle effectively large and expressive knowledge bases.

Regarding optimization techniques for fuzzy DLs there is only the work of Haarslev et al. [6] that presents an optimized prototype system supporting $\mathcal{ALC}$ extended with uncertainty. This system uses fuzzy, probabilistic and possibilistic functions, while the optimizations presented are quite general. Our work, on the other hand, focuses on optimization techniques only for fuzzy DLs and more specifically for the expressive DL f_{KD}-$\mathcal{SHIN}$. We present in detail the proposed optimization techniques, we discuss their applicability to fuzzy DLs using other fuzzy operators than those used in f_{KD}-$\mathcal{SHIN}$, and we optimize the operation of the greatest lower bound that is the main reasoning service of fuzzy DLs. The main contributions of this paper are the following:

1. It presents novel optimization techniques that can be applied to fuzzy DLs.
2. It provides an experimental evaluation of the proposed optimization techniques.

The rest of the paper is organized as follows. The following section introduces f_{KD}-$\mathcal{SHIN}$, section 3 presents the proposed optimizations techniques and section 4 illustrates the evaluation of our proposal. Finally, section 5 concludes the paper and provides a discussion on the achieved results and possible future work.

2 The Fuzzy DL f_{KD}-$\mathcal{SHIN}$

In this section, we briefly present the syntax and semantics of DL f_{KD}-$\mathcal{SHIN}$ which is a fuzzy extension of the DL $\mathcal{SHIN}$ [9]. Similarly to crisp description logic languages, a fuzzy description logic language consists of an alphabet of distinct concepts names ($\mathbf{C}$), role names ($\mathbf{R}$) and individual names ($\mathbf{I}$), together with a set of constructors to construct concept and role descriptions. If R is a role then R^- is also a role, namely the inverse of R. f_{KD}-$\mathcal{SHIN}$-concepts are inductively defined as follows,

1. If $C \in \mathbf{C}$, then C is a f_{KD}-$\mathcal{SHIN}$-concept,
2. If C and D are concepts, R is a role, S is a simple role and $n \in \mathbb{N}$, then $(\neg C)$, $(C \sqcup D)$, $(C \sqcap D)$, $(\forall R.C)$, $(\exists R.C)$, $(\geq nS)$ and $(\leq nS)$ are also f_{KD}-$\mathcal{SHIN}$-concepts.

In contrast to crisp DLs, the semantics of fuzzy DLs are provided by a *fuzzy interpretation* [15]. A fuzzy interpretation is a pair $\mathcal{I} = \langle \Delta^{\mathcal{I}}, \cdot^{\mathcal{I}} \rangle$ where $\Delta^{\mathcal{I}}$ is a non-empty set of objects and $\cdot^{\mathcal{I}}$ is a fuzzy interpretation function, which maps

an individual name a to elements of $\mathsf{a}^{\mathcal{I}} \in \Delta^{\mathcal{I}}$ and a concept name A (role name R) to a membership function $\mathsf{A}^{\mathcal{I}} : \Delta^{\mathcal{I}} \to [0, 1]$ ($R^{\mathcal{I}} : \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}} \to [0, 1]$).

By using fuzzy set theoretic operations the fuzzy interpretation function can be extended to give semantics to complex concepts, roles and axioms. $\mathrm{f}_{KD}\text{-}\mathcal{SHIN}$ uses the standard fuzzy operators of $1 - x$ for fuzzy negation, max, min for fuzzy union and intersection respectively and Kleenes Dienes implication [10].

A $\mathrm{f}_{KD}\text{-}\mathcal{SHIN}$ knowledge base Σ is a triple $\langle \mathcal{T}, \mathcal{R}, \mathcal{A} \rangle$, where $\mathcal{T}$ is a fuzzy $TBox$, $\mathcal{R}$ is a fuzzy $RBox$ and $\mathcal{A}$ is a fuzzy $ABox$. The $TBox$ is a finite set of fuzzy concept axioms which are of the form $C \sqsubseteq D$ called fuzzy concept inclusion axioms and $C \equiv D$ called fuzzy concept equivalence axioms, where C is a concept name and D an $\mathrm{f}_{KD}\text{-}\mathcal{SHIN}$ concept. Similarly, the $RBox$ is a finite set of fuzzy role axioms of the form $\mathsf{Trans}(R)$ called fuzzy transitive role axioms and $R \sqsubseteq S$ called fuzzy role inclusion axioms. Finally, the $ABox$ is a finite set of fuzzy assertions of the form $\langle a : C \bowtie n \rangle$, $\langle (a, b) : R \bowtie n \rangle$, where $\bowtie$ stands for $\geq, >, \leq$ or $<$, or $a \neq b$, for $a, b \in \mathbf{I}$. Furthermore, the symbols $\rhd$ and $\lhd$ are used as a placeholder for the inequalities $\geq, >$ and $\leq, <$ respectively. An assertion is called positive if defined by $\rhd$ while it is called negative if defined by $\lhd$. Intuitively, a fuzzy assertion of the form $\langle a : C \geq n \rangle$ means that the membership degree of a to the concept C is at least equal to n.

As in crisp DLs, the main reasoning services of $\mathrm{f}_{KD}\text{-}\mathcal{SHIN}$ are *entailment*, *ABox consistency* and *subsumption*. Furthermore, since a fuzzy $ABox$ might contain many positive assertions for an individual, without forming a contradiction, two additional reasoning services exist in $\mathrm{f}_{KD}\text{-}\mathcal{SHIN}$ to compute what is the best lower and upper truth-value bounds of a fuzzy assertion. The *greatest lower bound* (*glb*) and the *least upper bound* (*lub*) of an assertion with respect to a knowledge base have been defined in [15].

The reasoning services in fuzzy DLs are reduced to *ABox* consistency. This problem in the majority of expressive DLs is solved with the use of tableaux algorithms [7] that operate by decomposing complex concepts contained in an *ABox* according to their semantics. This procedure is made by expansion rules that differ for each DL constructor. The main objective of tableaux algorithms is to create a tableau structure that will be an abstraction of a model of an *ABox* $\mathcal{A}$ [9]. In a similar way, a tableau algorithm is used in fuzzy DLs to construct a fuzzy tableau for a fuzzy *ABox* $\mathcal{A}$ [14].

The tableau algorithm, presented by Stoilos et al. [14] for $\mathrm{f}_{KD}\text{-}\mathcal{SHIN}$, operates in completion forests similar to the $\mathcal{SHIN}$ algorithm [9]. A completion forest consists of a set of completion trees that are connected as the defined assertions of an *ABox* $\mathcal{A}$ specify. Each node x is labelled with a set $\mathcal{L}(x)$, which contains membership triples of the form $\langle C, \bowtie, n \rangle$, where C a $\mathrm{f}_{KD}\text{-}\mathcal{SHIN}$ concept that appears within $\mathcal{A}$ and $n \in [0, 1]$. Similarly, each edge $\langle x, y \rangle$ is labelled with a set $\mathcal{L}(\langle x, y \rangle)$ which contains membership triples of the form $\langle R, \bowtie, n \rangle$, where R is an $\mathrm{f}_{KD}\text{-}\mathcal{SHIN}$ role that occurs in $\mathcal{A}$. The algorithm expands the tree either by expanding the set $\mathcal{L}(x)$, of a node x with new triples, or by adding new leaf nodes. The expansion of the completion forest is determined from the tableau expansion rules that apply for the membership triples of a node. If

for example $\langle C_1 \sqcap C_2, \rhd, n \rangle \in \mathcal{L}(x)$, and $\{\langle C_1, \rhd, n \rangle, \langle C_2, \rhd, n \rangle\} \not\subseteq \mathcal{L}(x)$ then $\mathcal{L}(x) \to \mathcal{L}(x) \cup \{\langle C_1, \rhd, n \rangle, \langle C_2, \rhd, n \rangle\}$. It is very important to note at this point that tableau expansion rules for fuzzy DLs depend on the type of assertion, hence a positive conjunction is treated differently from a negative one. Finally, intuitively a clash is contained in a node when there are two conjugated triples i.e $\mathcal{L}(x) = \{\langle C, >, n \rangle \langle C, <, l \rangle\}$, with $n > l$. For a detailed presentation of the f_{KD}-$\mathcal{SHIN}$ tableau algorithm, the interested reader is referred to [14].

3 Optimizations techniques

3.1 Degrees Normalization

The *ABox* in fuzzy DLs contains concepts assertions, in which an individual participates in a concept with a degree, as well as role assertions, in which two individuals are related through a role with a degree. Due to this extension, in fuzzy DLs we can end up with an *ABox* in which an individual participates in the same concept with different degrees without forming a contradiction. Since the *ABox* of a fuzzy knowledge base in many cases is automatically generated [11], the existence of multiple assertions that can degrade the performance of reasoning is possible. Therefore, we can end up with a node

$$\mathcal{L}(x) = \{\langle C, \rhd_i, n_i \rangle, \langle C, \lhd_j, \ell_j \rangle\},$$

where C is a f_{KD}-$\mathcal{SHIN}$ concept, $n_i, \ell_j \in [0, 1]$ are degrees and $1 \le i \le k, 1 \le j \le m$. This situation is particularly problematic for many reasons. Firstly, in order to check for a clash we need to perform a proper number of checks, which here is $k \times m$. Subsuquently, if the node is clash-free and C is the complex concept $\exists R.A$, then one needs to apply rule $\exists_\rhd$ k times creating k different edges $\langle x, y_i \rangle$ with $\mathcal{L}(y_i) = \{\langle A, \rhd_i, n_i \rangle\}$.

This situation can be solved more effectively by normalizing the participation degrees in the membership triples of a node that use the same concept, reducing the assertions of this concept in a node to at most 2. In other words, we only allow the greatest positive assertion and the least negative assertion of a concept in a node, i.e.

$$\mathcal{L}(x) = \{\langle C, \rhd, d_{max} \rangle \langle C, \lhd, d_{min} \rangle\}.$$

If we furthermore extend this idea to concept assertions that include the negation of a concept, we end up with the rules illustrated in Table 1 for degrees normalization in a node.

Additionally, during the process of degrees normalization, we can introduce some additional rules based on the expansion rules of f_{KD}-$\mathcal{SHIN}$ in order to detect a contradiction. Lets assume node

$$\mathcal{L}(x) = \{\langle C, >, n \rangle \langle \neg C, >, l \rangle\}$$

and $n + l \ge 1$ which means that there is a clash that can be detected without applying the f_{KD}-$\mathcal{SHIN}$ rule of negation.

Table 1. Rules for degrees normalization

Assertion 1	Assertion 2	Condition	Action
$\langle C, \rhd, n \rangle \in \mathcal{L}(x)$	$\langle C, \rhd, m \rangle \in \mathcal{L}(x)$	$n \rhd m$	Delete $\langle C, \rhd, m \rangle$
$\langle C, >, n \rangle \in \mathcal{L}(x)$	$\langle C, \geq, m \rangle \in \mathcal{L}(x)$	$n \geq m$	Delete $\langle C, \geq, m \rangle$
$\langle C, \lhd, n \rangle \in \mathcal{L}(x)$	$\langle C, \lhd, m \rangle \in \mathcal{L}(x)$	$n \lhd l$	Delete $\langle C, \rhd, m \rangle$
$\langle C, <, n \rangle \in \mathcal{L}(x)$	$\langle C, \leq, m \rangle \in \mathcal{L}(x)$	$n \leq m$	Delete $\langle C, \geq, m \rangle$
$\langle C, \lhd, n \rangle \in \mathcal{L}(x)$	$\langle \neg C, \rhd, m \rangle \in \mathcal{L}(x)$	$n \leq 1 - m$	Delete $\langle \neg C, \rhd, m \rangle$
		$n > 1 - m$	Delete $\langle C, \lhd, n \rangle$
$\langle C, \rhd, n \rangle \in \mathcal{L}(x)$	$\langle \neg C, \lhd, m \rangle \in \mathcal{L}(x)$	$n \geq 1 - m$	Delete $\langle \neg C, \lhd, m \rangle$
		$n < 1 - m$	Delete $\langle C, \rhd, n \rangle$

The technique of degrees normalization can be also extended to the role assertions of a fuzzy *ABox*. Degrees normalization is easily implemented and for a node x that contains n membership triples the possible checks that need to be done are the combinations per 2 i.e. $\frac{n!}{2!(n-2)!}$ which means that this technique is of polynomial complexity. Additionally, the rules for early clash detection can be modified according to the fuzzy complement used for the interpretation of negation, permiting in that way its use in fuzzy DLs that use different fuzzy logics. The only disadvantage of this optimization technique is that it is strongly depended on the knowledge base. In other words, it is possible that the use of degrees normalization technique for some knowledge bases (more specifically for knowledge bases that do not contain membership triples of the same concept) will have no result in the performance.

3.2 ABox Partitioning

An optimization technique that was applied in crisp reasoners and can be also used in fuzzy reasoners to boost up their performance is *ABox* partitioning [4, 6]. This technique is based on the fact that tableau expansion rules have specific effect on the tableau structure. Hence, a tableau expansion rule can either add (i) a new neighbour node to the node of examination, (ii) new membership triples in this node or finally (iii) new membership triples to neighbouring nodes. Therefore, due to this property of the f_{KD}-$\mathcal{SHIN}$ constructors, the assertional component of a fuzzy knowledge base can be divided in smaller partitions, which can be examined independently. Let's examine the following example in order to understand the benefits from the *ABox* partitioning technique.

Example 1. Let us assume the completion forest shown in Fig. 1 where A, B, C, D, E, F are f_{KD}-$\mathcal{SHIN}$ concepts and R, S, L are f_{KD}-$\mathcal{SHIN}$ roles.

Assuming that B is a complex f_{KD}-$\mathcal{SHIN}$ concept, we will examine the different ways that the tableau expansion rules affect this completion forest, with respect to the different forms that B can have and the different forms of inequalities ($\rhd, \lhd$) that can appear in membership triples with B. Consequently we distinguish the following cases: If B consists of:
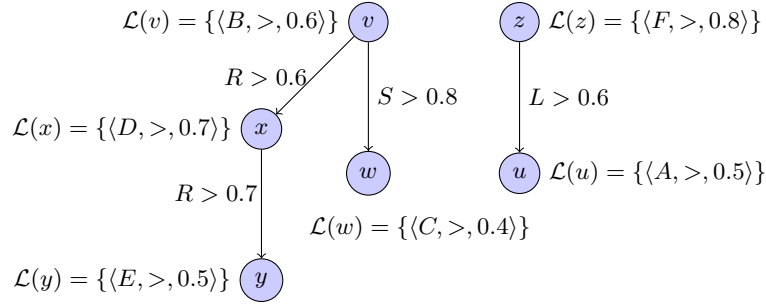
Fig. 1. The completion forest of two *ABox* partitions.

- constructors of the form $\neg, \sqcap, \sqcup$, then only node v is affected.
- constructors $\exists$ and $\geq$ and the membership triple contains $\triangleright$ or constructors $\forall$ and $\leq$ and the inequality is $\triangleleft$, then new neighbor nodes are created for node v.
- constructor $\forall$ and inequality $\triangleright$ or constructor $\exists$ and inequality $\triangleleft$, then all existing and (possibly) new neighbors of v are affected can be affected (depending on the role S that participates in the concept $B = \forall S.C$ and whether there exists R with $R \stackrel{*}{\sqsubseteq} S$).
- constructor $\leq$ and inequality $\triangleright$, or $\geq$ and inequality $\triangleleft$, then nodes w, x and also the possible new neighbors of v are affected.

Finally, if we consider R^- as the inverse role of R then again the expansion of a rule with inverse roles will only affect the neighbors of v.

As we can observe, in any case the consistency of node v is independent of nodes z and u. In other words, the ABox of the Example 1 can be partitioned in two smaller ABoxes that can be examined independently. If both partitions are consistent then the ABox will be consistent as well, while if even one of the partitions is inconsistent then the ABox will be inconsistent.

In that way the storage requirements of tableau are considerably reduced, since the nodes that are not connected to the node that is examined, and therefore do not directly affect its consistency, can be omitted. Additionally, the storage requirements can be further reduced because after the expansion of a consistent partition of the ABox, the partition can be discarded from the completion forest. Formally given ABox partitions are evaluated as folows.

Definition 1. *(Connection Relation) We inductively define the connection relation between two individuals $a, b \in \mathbf{I}$ w.r.t. an ABox $\mathcal{A}$ (denoted with $\leftrightsquigarrow_{\mathcal{A}}$) as follows:*

$$a \leftrightsquigarrow_{\mathcal{A}} b \iff \begin{cases} R\,(a,b) \triangleright d \in \mathcal{A} & \text{for some role } R \text{ and } d \in [0,1] \text{ or} \\ R\,(b,a) \triangleright d \in \mathcal{A} & \text{for some role } R \text{ and } d \in [0,1] \text{ or} \\ a \leftrightsquigarrow_{\mathcal{A}} c \text{ and } c \leftrightsquigarrow_{\mathcal{A}} b & \text{for some individual } c \in \mathbf{I} \end{cases} \quad (1)$$

Definition 2. *For an ABox $\mathcal{A}$ and* the set of individuals in it **I***, for each $a \in$ **I** the set $[a]_{\mathcal{A}}$ contains a and all the individuals related to it w.r.t $\mathcal{A}$.*

$$[a]_{\mathcal{A}} = \{a\} \bigcup \{b \mid b \in \mathbf{I} \text{ and } a \leftrightsquigarrow_{\mathcal{A}} b\} \tag{2}$$

Definition 3. *We denote with $\mathcal{A}_{[a]}$ the partition of the ABox $\mathcal{A}$ that contains only individuals in $[a]_{\mathcal{A}}$:*

$$\mathcal{A}_{[a]} = \{C(b) \bowtie d \mid C(b) \bowtie d \in \mathcal{A} \text{ and } b \in [a]_{\mathcal{A}}\} \bigcup \tag{3}$$
$$\{R(b,c) \rhd d \mid R(b,c) \rhd d \in \mathcal{A} \text{ and } b, c \in b \in [a]_{\mathcal{A}}\}$$

Definition 4. *The set $\mathbb{A}$ is the smallest subset of the powerset of $\mathcal{A}$ such that it applies:*

$$a \in \mathbf{I} \implies \mathcal{A}_{[a]} \in \mathbb{A} \tag{4}$$

Theorem 1. *It holds that:*

1. $\bigcup\limits_{\mathcal{A}_i \in \mathbb{A}} \mathcal{A}_i = \mathcal{A}$,
2. $\mathcal{A}_i \bigcap \mathcal{A}_j = \emptyset$, for each pair $\mathcal{A}_i, \mathcal{A}_j \in \mathbb{A}$ such that $\mathcal{A}_i \neq \mathcal{A}_j$,
3. $\mathcal{A}$ is consistent w.r.t. a TBox $\mathcal{T}$ iff each $\mathcal{A}_i \in \mathbb{A}$ is consistent w.r.t. a TBox $\mathcal{T}$.

ABox partitioning is a very effective optimization technique. It is of polynomial complexity and it can be applied to any fuzzy DL without nominals independently from the fuzzy operators used to provide the interpretations. The extreme case in which *ABox* partitioning does not boost up the performance of reasoning is when all the individuals are connected with each other. Additionally, *ABox* partitioning is very important because the consistency of a node can be examined independently of the others nodes contained in an *ABox*, a fact that is very useful for greatest lower bound reasoning service (see Section 3.3).

3.3   Optimized GLB

One of the most interesting and important reasoning services offered by fuzzy DLs is computing the greatest lower bound of some individual a to some concept C. Formally for a fuzzy knowledge base Σ and a crip assertion φ, the *greatest lower bound (glb)* of φ w.r.t. Σ is $glb(\Sigma, \varphi) = \sup\{n \mid \Sigma \models \varphi \geq n\}$, where $\sup \emptyset = 0$ while the *least upper bound lub* of c w.r.t. Σ is $lub(\Sigma, \varphi) = \inf\{n \mid \Sigma \models \varphi \leq n\}$, where $\inf \emptyset = 1$. A decision procedure for solving greatest lower and least upper bounds was proposed by Straccia [15]. More precisely, one first defines the set of "relative" degrees as complemented values (for membership degree 0.4, the complemented value is $C_{0.4} = 1 - 0.4 = 0.6$) and the degrees 0, 0.5 and 1 form the set of membership degrees $N^{\Sigma} = \{n, 1 - n \mid \{(a : C) \bowtie n, ((a, b) : R) \bowtie n\} \cap \mathcal{A} \neq \emptyset\}$. Then, in order to evaluate the *glb* of an assertion φ one evaluates the greatest $n \in N^{\Sigma}$ such that $\Sigma \models \varphi \geq n$. An optimization in the search space, proposed in [15], is to use binary search algorithm reducing in that way the satisfiability checks required. To better understand the operation for the evaluation of *glb* let's consider the following example.

Example 2. Let Σ be a satisfiable fuzzy knowledge base with $N^\Sigma = \{0,\ldots,0.5,\ldots,1\}$ that contains the following nodes

$$\mathcal{L}(x) = \{\langle (E \sqcap D), \geq, 0.6\rangle, \langle \exists R.(\forall R^-.C), \geq 0.8\rangle\}$$

$$\mathcal{L}(y) = \{\langle (A \sqcap B), \geq, 0.7\rangle\}$$

and $glb(\Sigma, (x : C))$ is asked.

Since *glb* is asked $\Sigma \models \varphi \geq n, \forall n \in N^\Sigma$ must be solved to find the greatest n. We apply the binary search algorithm, assuming that 0.5 is the middle if we sort the elements of N^Σ. Therefore, we evaluate if $\Sigma \models (x : C) \geq 0.5$ and in case it is (i.e. $\Sigma \cup (x : C) < 0.5$ is unsatisfiable) we move on to higher degree until $\Sigma \not\models (x : C) \geq n$ that indicates that the previous degree is the $glb(\Sigma, (x : C))$, differently (i.e. $\Sigma \models (x : C) \geq n, \forall n \in N^\Sigma$) $glb(\Sigma, (x : C)) = 1$.

1. Add membership triple $\langle C, <, 0.5\rangle$ to node x.

$$\mathcal{L}(x) = \{\langle (E \sqcap D), \geq, 0.6\rangle, \langle \exists R.(\forall R^-.C), \geq, 0.8\rangle \langle C, <, 0.5\rangle\}$$

$$\mathcal{L}(y) = \{\langle (A \sqcap B), \geq, 0.7\rangle\}$$

2. Application of $(E \sqcap D), \geq, 0.6$

$$\mathcal{L}(x) = \{\langle E, \geq, 0.6\rangle, \langle D, \geq, 0.6\rangle, \langle \exists R.(\forall R^-.C), \geq, 0.8\rangle \langle C, <, 0.5\rangle\}$$

$$\mathcal{L}(y) = \{\langle (A \sqcap B), \geq, 0.7\rangle\}$$

3. Application of $\langle \exists R.(\forall R^-.C), \geq, 0.8\rangle$

$$\mathcal{L}(x) = \{\langle E, \geq, 0.6\rangle, \langle D, \geq, 0.6\rangle, \langle \exists R.(\forall R^-.C), \geq, 0.8\rangle, \langle C, <, 0.5\rangle, \langle C, \geq, 0.8\rangle\}$$

$$\mathcal{L}(x, y) = \{\langle R, \geq, 0.8\rangle\}$$

$$\mathcal{L}(x) = \{\langle \forall R^-.C, \geq, 0.8\rangle\}$$

$$\mathcal{L}(y) = \{\langle A, \geq, 0.7\rangle, \langle B, \geq, 0.7\rangle\}$$

4. Clash detected i.e. $\Sigma \models (x : C) \geq 0.5$ and we move on to next $n \in N^\Sigma$ selected by binary search by adding membership triple $\langle C, <, n\rangle$ to node x.
5. Application of $(E \sqcap D), \geq, 0.6$

$$\mathcal{L}(x) = \{\langle E, \geq, 0.6\rangle, \langle D, \geq, 0.6\rangle, \langle \exists R.(\forall R^-.C), \geq, 0.8\rangle \langle C, <, n\rangle\}$$

$$\mathcal{L}(y) = \{\langle (A \sqcap B), \geq, 0.7\rangle\}$$

$\vdots$

In order to improve the performance of the algorithm for the evaluation of *glb* we propose the use of two techniques. Firstly, since this check refers to a specific individual *ABox* partitioning can be used in order to examine only the *ABox* partition $\mathcal{A}'$ in which the individual of assertion φ is contained. (Note that node y in the above example is unnecessary.) It is important to note at this point that the knowledge base must be consistent, differently the partition selected may be an consistent partition of an inconsistent *ABox* that will give incorrect results. By selecting a partition $\mathcal{A}'$ a new set of membership degrees only for it with $N^{\mathcal{A}'} \subseteq N^{\Sigma}$ is evaluated that in most cases contains a significantly smaller amount of degrees resulting to less satisfiability checks.

Furthermore, as we can observe from the previous example, when satisfiability for *glb* is solved the assertions of the examined node are expanded in the same way regardless of the new membership triple that is added each time. The performance of *glb* can be further improved by preventing this recurrent expansion of the membership triples in the original *ABox*. This can be achieved by expanding the $\mathcal{A}'$ partition resulting to a completion forest F in which no expansion rule can apply, which is then cached. Then, the membership triple that results from the assertion examined for *glb* φ is added to the cached completion forest i.e. $F \cup \varphi$ and the resulting completion forest is expanded. In that way, we get the same satisfiability result without the recurrent expansion of the membership triples contained in $\mathcal{A}'$, a fact which makes the *glb* computation much faster.

The described optimizations for *glb* are very effective since they reduce the search space of tableau independently of the fuzzy knowledge base used. Additionally, they are applicable to any fuzzy DL since they do not depend on the fuzzy operators used and they are easily implemented. Finally, despite the fact that the storage requirements may increase due to caching, the overall storage requirements of tableau remains low compared to unoptimized *glb*.

4   Results

Our evaluation focuses on the performance of greatest lower bound reasoning service. More specifically, we evaluate the performance of global greatest lower bound i.e. the greatest lower bound of all the individuals in a fuzzy knowledge base with all the defined concepts of the $TBox$. The $TBox$ used is acyclic and it contains 43 defined concepts of f$_{KD}$-$\mathcal{SHIN}$ expressiveness. All the experiments performed using FiRE under Linux on a Core 2 Duo 2G machine with 2Gb memory. We examined the performance of this reasoning service using fuzzy knowledge bases of different sizes by adjusting the size of individuals, the results are illustrated in Table 2.

As we can observe the optimization techniques dramatically reduce the time required for the evaluation of global greatest lower bound in all cases. More specifically, unoptimized FiRE cannot perform global *glb* for more than about 1200 individuals because the system runs out of memory. This is because the size of the tableau increases proportionally to the number of individuals in the knowledge base. On the other hand, optimized FiRE using *ABox* partitioning is

able to reduce the storage requirements making in that way the problem almost scalable. Furthermore the optimized use of *glb* service avoids the recurrence satisfiability test saving in that way space and time. However, it is very important to note in the specific knowledge base *ABox* partitioning operates very well, fact that boosts the overall performance. In the worst case scenario that *ABox* partitioning cannot apply the space and time required remain very large.

Table 2. Performance of global *glb* in knowledge bases of different size. The response time is in milliseconds

Individuals	Unoptimized FiRE	Optimized FiRE
500	1.436.127	277.006
1000	3.992.231	651.342
1550	Out of Memory	984.966
2140	Out of Memory	1326.072

5   Conclusions

In this paper optimizations techniques that can boost the performance of fuzzy DLs were presented. Our main objective was to present novel optimization techniques that can apply to fuzzy DLs. We first made an introduction to the fuzzy DL f_{KD}-$\mathcal{SHIN}$ and we then presented degrees normalization, *ABox* partitioning and an optimized method for the evaluation of the greatest lower bound, which is a very important reasoning service for fuzzy DLs. After that, we performed an evaluation of the proposed optimization techniques using fuzzy reasoning engine FiRE in which they are implemented. Evaluation of FiRE using the optimization techniques showed that reasoning in fuzzy DLs can be very effective. More specifically optimized FiRE reduces the storage requirements making in that way the global greatest lower bound problem almost scalable.

As far as future directions are concerned, we intend to further investigate on optimization techniques for very expressive fuzzy DLs.

References

1. F. Bobillo and U. Straccia. fuzzydl: An expressive fuzzy description logic reasoner. In *In Proceedings of the 17th IEEE International Conference on Fuzzy Systems (FUZZ-IEEE 2008)*, pages 923–930, 2008.
2. Fernando Bobillo, Miguel Delgado, and Juan Gómez-Romero. A crisp representation for fuzzy $\mathcal{SHOIN}$ with fuzzy nominals and general concept inclusions. pages 174–188, 2008.
3. J. Gmez-Romero F. Bobillo, M. Delgado. Delorean: A reasoner for fuzzy owl 1.1. In *In Proceedings of the 4th International Workshop on Uncertainty Reasoning for the Semantic Web (URSW 2008)*, pages 923–930, 2008.

4. Volker Haarslev and Ralf Moller. Expressive abox reasoning with number restrictions, role hierarchies, and transitively closed roles. In *In: Proceedings of Seventh International Conference on Principles of Knowledge Representation and Reasoning (KR2000)*, pages 273–284. Morgan Kaufmann, 2000.

5. Volker Haarslev and Ralf Möller. RACER System Description. In *IJCAR-01*, volume 2083, 2001.

6. Volker Haarslev, Hsueh-Ieng Pai, and Nematollaah Shiri. Optimizing tableau reasoning in alc extended with uncertainty. In *Proceedings of the 2007 International Workshop on Description Logics (DL-2007)*, pages 307–314, 2007.

7. Reiner Hähnle. Tableaux and related methods. In Alan Robinson and Andrei Voronkov, editors, *Handbook of Automated Reasoning*, pages 103–137. Elsevier Science Publishers, 2001.

8. I. Horrocks and P. F. Patel-Schneider. Optimising description logic subsumption. *Journal of Logic and Computation*, 9(3):267–293, 1999.

9. I. Horrocks, U. Sattler, and S. Tobies. Reasoning with Individuals for the Description Logic $\mathcal{SHIQ}$. In David MacAllester, editor, *CADE-2000*, number 1831 in LNAI, pages 482–496. Springer-Verlag, 2000.

10. G. J. Klir and B. Yuan. *Fuzzy Sets and Fuzzy Logic: Theory and Applications*. Prentice-Hall, 1995.

11. N. Simou, Th. Athanasiadis, G. Stoilos, and S. Kollias. Image indexing and retrieval using expressive fuzzy description logics. *Signal, Image and Video Processing*, 2(4):321–335.

12. Evren Sirin, Bijan Parsia, Bernardo Cuenca Grau, Aditya Kalyanpur, and Yarden Katz. Pellet: A practical OWL-DL reasoner. *Journal of Web Semantics*, 5:51–53, 2007.

13. Giorgos Stoilos, Nikos Simou, Giorgos Stamou, and Stefanos Kollias. Uncertainty and the semantic web. *IEEE Intelligent Systems*, 21(5):84–87, 2006.

14. Giorgos Stoilos, Giorgos Stamou, Vassilis Tzouvaras, Jeff Z. Pan, and Ian Horrocks. Reasoning with very expressive fuzzy description logics. *Journal of Artificial Intelligence Research*, 30(5):273–320, 2007.

15. U. Straccia. Reasoning within fuzzy description logics. *Journal of Artificial Intelligence Research*, 14:137–166, 2001.

16. Umberto Straccia. A fuzzy description logic. In *AAAI '98/IAAI '98: Proceedings of the fifteenth national/tenth conference on Artificial intelligence/Innovative applications of artificial intelligence*, pages 594–599. American Association for Artificial Intelligence, 1998.

17. Dmitry Tsarkov and Ian Horrocks. FaCT++ description logic reasoner: System description. In *Proc. of the Int. Joint Conf. on Automated Reasoning (IJCAR 2006)*, volume 4130 of *Lecture Notes in Artificial Intelligence*, pages 292–297. Springer, 2006.

18. Dmitry Tsarkov, Ian Horrocks, and Peter F. Patel-Schneider. Optimizing terminological reasoning for expressive description logics. *J. of Automated Reasoning*, 39(3):277–316, 2007.

Role-depth Bounded Least Common Subsumers by Completion for $\mathcal{EL}$- and Prob-$\mathcal{EL}$-TBoxes

Rafael Peñaloza and Anni-Yasmin Turhan

TU Dresden, Institute for Theoretical Computer Science

Abstract. The least common subsumer (lcs) w.r.t general $\mathcal{EL}$-TBoxes does not need to exists in general due to cyclic axioms. In this paper we present an algorithm for computing role-depth bounded $\mathcal{EL}$-lcs based on the completion algorithm for $\mathcal{EL}$. We extend this computation algorithm to a recently introduced probabilistic variant of $\mathcal{EL}$: Prob-$\mathcal{EL}^{01}$.

1 Introduction

The least common subsumer (lcs) inference yields a concept description, that generalizes a collection of concepts by extracting their commonalities. This inference was introduced in [8]. Most prominently the lcs is used in the bottom-up construction of knowledge bases [5], where a collection of individuals is selected for which a new concept definition is to be introduced in the TBox. This is can be achieved by first generalizing each selected individual into a concept description (by computing the most specific concept) and then applying the lcs to these concept descriptions. Further applications of the lcs include similarity-based Information Retrieval or learning from examples.

The lightweight Description Logic $\mathcal{EL}$ and many of its extensions enjoy the nice property that concept subsumption and classification of TBoxes can be computed in polynomial time [3]. Thus, despite of its limited expressiveness, $\mathcal{EL}$ is used in many practical applications – most prominently in the medical ontology SNOMED [15] – and is the basis for the EL profile of the OWL 2.0 standard.

However, some practical applications such as medical or context-aware applications need to represent information that holds only with a certain probability. For instance, context-aware applications may need to represent sensor data in their ontology, which is correct only with a certain probability. This sort of information can be represented by the probabilistic DLs recently introduced in [12], which allows to represent subjective probabilities. These DLs are based on Halpern's probabilistic FOL variant called Type-2 [9] and they allow to assign probabilistic information to concepts (and roles) and not, as in other probabilistic DLs, to concept axioms [11, 10]. In particular, in [12] the DL Prob-$\mathcal{EL}^{01}$ was introduced, which allows to express limited probability values for $\mathcal{EL}$-concepts, and it was shown that instance checking is in PTime.

If in applications different information sources supply varying information on the same topic, the generalization of this information by the lcs gives a description of what the sources agree upon. For both, $\mathcal{EL}$ and Prob-$\mathcal{EL}$, the computation

of the lcs is a desirable task. Unfortunately, the lcs w.r.t. general TBoxes does not need to exist in this setting (see [1]), due to cyclic definitions in the TBox.

In this paper we present practical algorithms for computing the lcs up to a certain role-depth for $\mathcal{EL}$ and Prob-$\mathcal{EL}^{01}$. The concept obtained is still a generalization of the input concepts, but not necessarily the least one w.r.t. subsumption. Our computation algorithms are based on the completion algorithms for classification in $\mathcal{EL}$ and Prob-$\mathcal{EL}^{01}$ and thus can be implemented on top of reasoners for these two DLs. Due to space limitations most of the proofs can be found in [14].

2 $\mathcal{EL}$ and Prob-$\mathcal{EL}$

Starting from two disjoint sets N_C and N_R of *concept* and *role names*, respectively, $\mathcal{EL}$-*concept descriptions* are built using the concept *top* ($\top$) and the constructors conjunction ($\sqcap$), and existential restriction ($\exists r.C$). We will often call concept descriptions simply *concepts* for brevity. The semantics of $\mathcal{EL}$ is defined with the help of interpretations $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ consisting of a non-empty *domain* $\Delta^{\mathcal{I}}$ and an *interpretation function* $\cdot^{\mathcal{I}}$ that assigns binary relations on $\Delta^{\mathcal{I}}$ to role names and subsets of $\Delta^{\mathcal{I}}$ to concepts.

A TBox is a set of *concept inclusion axioms* of the form $C \sqsubseteq D$, where C, D are concept descriptions. An interpretation $\mathcal{I}$ *satisfies* the concept inclusion $C \sqsubseteq D$, denoted as $\mathcal{I} \models C \sqsubseteq D$ iff $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$. $\mathcal{I}$ is a *model* of a TBox $\mathcal{T}$ if it satisfies all axioms in $\mathcal{T}$. A concept C is *subsumed by* a concept D w.r.t. $\mathcal{T}$ (denoted $C \sqsubseteq_{\mathcal{T}} D$) if, for every model $\mathcal{I}$ of $\mathcal{T}$ it holds that $\mathcal{I} \models C \sqsubseteq D$.

We now introduce Prob-$\mathcal{EL}^{01}$, a probabilistic logic that extends $\mathcal{EL}$ with the probabilistic constructors $P_{>0}$ and $P_{=1}$. Intuitively, the concepts $P_{>0}C$ and $P_{=1}C$ express that the probability of C being satisfied is greater than 0, and equal to 1, respectively. This logic was first introduced, along with more expressive probabilistic DLs in [12]. Formally, Prob-$\mathcal{EL}^{01}$ concepts are constructed as

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

where A is a concept name, r is a role name, and $*$ is one of $\{> 0, = 1\}$.

In contrast to previously introduced probabilistic DLs, uncertainty in Prob-$\mathcal{EL}^{01}$ is expressed by assigning probabilities to concepts, instead of axioms. Thus, the semantics of Prob-$\mathcal{EL}^{01}$ generalize the interpretation-based semantics of $\mathcal{EL}$ towards the possible worlds semantic used by Halpern [9]. A *probabilistic interpretation* is of the form

$$\mathcal{I} = (\Delta^{\mathcal{I}}, W, (\mathcal{I}_w)_{w \in W}, \mu),$$

where $\Delta^{\mathcal{I}}$ is the (non-empty) *domain*, W is a set of *worlds*, μ is a discrete probability distribution on W, and for each world $w \in W$, $\mathcal{I}_w$ is a classical $\mathcal{EL}$ interpretation with domain $\Delta^{\mathcal{I}}$. The probability that a given element of the domain $d \in \Delta^{\mathcal{I}}$ belongs to the interpretation of a concept name A is given by

$$p_d^{\mathcal{I}}(A) := \mu(\{w \in W \mid d \in A^{\mathcal{I}_w}\}).$$

The functions $\mathcal{I}_w$ and $p_d^{\mathcal{I}}$ are extended to complex concepts in the usual way for the classical $\mathcal{EL}$ constructors, where the extension to the new constructors P_* is defined as

$$(P_{>0}C)^{\mathcal{I}_w} := \{d \in \Delta^{\mathcal{I}} \mid p_d^{\mathcal{I}}(C) > 0\},$$
$$(P_{=1}C)^{\mathcal{I}_w} := \{d \in \Delta^{\mathcal{I}} \mid p_d^{\mathcal{I}}(C) = 1\}.$$

A probabilistic interpretation $\mathcal{I}$ *satisfies* a concept inclusion $C \sqsubseteq D$, denoted as $\mathcal{I} \models C \sqsubseteq D$ if for every $w \in W$ it holds that $C^{\mathcal{I}_w} \subseteq D^{\mathcal{I}_w}$. It is a *model* of a TBox $\mathcal{T}$ if it satisfies all concept inclusions in $\mathcal{T}$. Let C, D be two Prob-$\mathcal{EL}^{01}$ concepts and $\mathcal{T}$ a TBox. We say that C is *subsumed by* D w.r.t. $\mathcal{T}$ (denoted as $C \sqsubseteq_{\mathcal{T}} D$) if for every model $\mathcal{I}$ of $\mathcal{T}$ it holds that $\mathcal{I} \models C \sqsubseteq D$.

Intuitively, the different worlds express the different possibilities for the domain elements to be interpreted (in the sense of crisp $\mathcal{EL}$ interpretations), and the probability of a concept C being satisfied by a given individual a is given by the probabilities of the different worlds in which a belongs to C.

An interesting property of this logic is that subsumption between concepts can be decided in polynomial time [12]. Moreover, as we will see in the following section, an algorithm for deciding subsumption can be obtained by extending the completion algorithm for (crisp) $\mathcal{EL}$.

3 Completion-based Subsumption Algorithms

We briefly sketch the completion algorithms for deciding subsumption in $\mathcal{EL}$ [3] and in Prob-$\mathcal{EL}^{01}$ [12]. Completion-based methods compute not only subsumption relations for a pair of concept names, but *classify* the whole TBox.

3.1 Completion-based Subsumption Algorithm for $\mathcal{EL}$

Given an $\mathcal{EL}$-TBox $\mathcal{T}$, we use $\mathsf{BC}_{\mathcal{T}}$ to denote the set of *basic concepts for $\mathcal{T}$*, i.e., the smallest set of concept descriptions which contains (1) $\top$ and (2) all concept names used in $\mathcal{T}$. A normal form for $\mathcal{EL}$-TBoxes can be defined as follows.

Definition 1 (Normal Form for $\mathcal{EL}$-TBoxes). *An $\mathcal{EL}$-TBox $\mathcal{T}$ is in* normal form *if all concept inclusions have one of the following forms, where $C_1, C_2, D \in \mathsf{BC}_{\mathcal{T}}$:*
$$C_1 \sqsubseteq D, \quad C_1 \sqcap C_2 \sqsubseteq D, \quad C_1 \sqsubseteq \exists r.C_2 \quad \text{or} \quad \exists r.C_1 \sqsubseteq D.$$

Any $\mathcal{EL}$-TBox $\mathcal{T}$ can be transformed into a normalized TBox $\mathcal{T}'$ by introducing new concept names. $\mathcal{EL}$-TBoxes can be transformed into normal form by applying the normalization rules displayed in Figure 1 exhaustively. These rules replace the GCI on the left-hand side of the rules with the set of GCIs on the right-hand side of the rule.

Let $\mathcal{T}$ be a TBox in normal form to be classified and let $\mathsf{R}_{\mathcal{T}}$ denote the set of all role names appearing in $\mathcal{T}$. The completion algorithm works on two kinds on *completion sets*: $S(C)$ and $S(C, r)$ for each $C \in \mathsf{BC}_{\mathcal{T}}$ and $r \in \mathsf{R}_{\mathcal{T}}$, which

NF1 $C \sqcap \hat{D} \sqsubseteq E \longrightarrow \{\; \hat{D} \sqsubseteq A, C \sqcap A \sqsubseteq E \;\}$

NF2 $\exists r.\hat{C} \sqsubseteq D \longrightarrow \{\; \hat{C} \sqsubseteq A, \exists r.A \sqsubseteq D \;\}$

NF3 $\hat{C} \sqsubseteq \hat{D} \longrightarrow \{\; \hat{C} \sqsubseteq A, A \sqsubseteq \hat{D} \;\}$

NF4 $B \sqsubseteq \exists r.\hat{C} \longrightarrow \{\; B \sqsubseteq \exists r.A, A \sqsubseteq \hat{C} \;\}$

NF5 $B \sqsubseteq C \sqcap D \longrightarrow \{\; B \sqsubseteq C, B \sqsubseteq D \;\}$

where $\hat{C}, \hat{D} \notin \mathsf{BC}_{\mathcal{T}}$ and A is a new concept name.

Fig. 1. $\mathcal{EL}$ normalization rules

CR1 If $C' \in S(C)$, $C' \sqsubseteq D \in \mathcal{T}$, and $D \notin S(C)$
then $S(C) := S(C) \cup \{D\}$

CR2 If $C_1, C_2 \in S(C)$, $C_1 \sqcap C_2 \sqsubseteq D \in \mathcal{T}$, and $D \notin S(C)$
then $S(C) := S(C) \cup \{D\}$

CR3 If $C' \in S(C)$, $C' \sqsubseteq \exists r.D \in \mathcal{T}$, and $D \notin S(C, r)$
then $S(C, r) := S(C, r) \cup \{D\}$

CR4 If $D \in S(C, r)$, $D' \in S(D)$, $\exists r.D' \sqsubseteq E \in \mathcal{T}$, and $E \notin S(C)$
then $S(C) := S(C) \cup \{E\}$

Fig. 2. $\mathcal{EL}$ completion rules

contain concept names from $\mathsf{BC}_{\mathcal{T}}$. The intuition is that the completion rules make implicit subsumption relationships explicit in the following sense:

- $D \in S(C)$ implies that $C \sqsubseteq_{\mathcal{T}} D$,
- $D \in S(C, r)$ implies that $C \sqsubseteq_{\mathcal{T}} \exists r.D$.

By $\mathsf{S}_{\mathcal{T}}$ we denote the set containing all completion sets of $\mathcal{T}$. In the algorithm, the completion sets are initialized as follows:

- $S(C) := \{C, \top\}$ for each $C \in \mathsf{BC}_{\mathcal{T}}$,
- $S(C, r) := \emptyset$ for each $r \in \mathsf{R}_{\mathcal{T}}$.

The sets $S(C)$ and $S(C, r)$ are extended by applying the completion rules shown in Figure 2 until no more rule applies. After the completion has terminated, the subsumption relation between two basic concepts A and B can be tested by checking whether $B \in S(A)$. Soundness and completeness of the $\mathcal{EL}$-completion algorithm has been shown in [4] as well as that it runs in polynomial time. This algorithm has recently been extended for a probabilistic variant of $\mathcal{EL}$, which we introduce next.

PCR1 If $C' \in S_*(C,v)$, $C' \sqsubseteq D \in \mathcal{T}$, and $D \notin S_*(C,v)$
then $S_*(C,v) := S_*(C,v) \cup \{D\}$

PCR2 If $C_1, C_2 \in S_*(C,v)$, $C_1 \sqcap C_2 \sqsubseteq D \in \mathcal{T}$, and $D \notin S_*(C,v)$
then $S_*(C,v) := S_*(C,v) \cup \{D\}$

PCR3 If $C' \in S_*(C,v)$, $C' \sqsubseteq \exists r.D \in \mathcal{T}$, and $D \notin S_*(C,r,v)$
then $S_*(C,r,v) := S_*(C,r,v) \cup \{D\}$

PCR4 If $D \in S_*(C,r,v)$, $D' \in S_{\gamma(v)}(D, \gamma(v))$, $\exists r.D' \sqsubseteq E \in \mathcal{T}$, and $E \notin S_*(C,v)$
then $S_*(C,v) := S_*(C,v) \cup \{E\}$

PCR5 If $P_{>0}A \in S_*(C,v)$, and $A \notin S_*(C, P_{>0}A)$
then $S_*(C, P_{>0}A) := S_*(C, P_{>0}A) \cup \{A\}$

PCR6 If $P_{=1}A \in S_*(C,v)$, $v \neq 0$, and $A \notin S_*(C,v)$
then $S_*(C,v) := S_*(C,v) \cup \{A\}$

PCR7 If $A \in S_*(C,v)$, $v \neq 0$, $P_{>0}A \in \mathcal{P}_0^{\mathcal{T}}$, and $P_{>0}A \notin S_*(C,v')$
then $S_*(C,v') := S_*(C,v') \cup \{P_{>0}A\}$

PCR8 If $A \in S_*(C,1)$, $P_{=1}A \in \mathcal{P}_1^{\mathcal{T}}$, and $P_{=1}A \notin S_*(C,v)$
then $S_*(C,v) := S_*(C,v) \cup \{P_{=1}A\}$

Fig. 3. Prob-$\mathcal{EL}^{01}$ completion rules

3.2 Completion-based Subsumption Algorithm for Prob-$\mathcal{EL}$

In Prob-$\mathcal{EL}^{01}$, basic concepts also include the probabilistic constructors; that is, the set $\mathsf{BC}_{\mathcal{T}}$ of Prob-$\mathcal{EL}^{01}$ basic concepts for $\mathcal{T}$ is the smallest set that contains (1) $\top$, (2) all concept names used in $\mathcal{T}$, and (3) all concepts of the form P_*A, where A is a concept name in $\mathcal{T}$.

Definition 2 (Normal Form for Prob-$\mathcal{EL}^{01}$-TBoxes). *A Prob-$\mathcal{EL}^{01}$-TBox* $\mathcal{T}$ *is in* normal form *if all its axioms are of one of the following forms*

$$C \sqsubseteq D, \quad C_1 \sqcap C_2 \sqsubseteq D, \quad C \sqsubseteq \exists r.A, \quad \text{or} \quad \exists r.A \sqsubseteq D,$$

where $C, C_1, C_2, D \in \mathsf{BC}_{\mathcal{T}}$ *and* A *is a new concept name.*

The normalization rules in Figure 1 can also be used to transform a Prob-$\mathcal{EL}^{01}$-TBox into this extended notion of normal form. We denote as $\mathcal{P}_0^{\mathcal{T}}$ and $\mathcal{P}_1^{\mathcal{T}}$ the set of all concepts of the form $P_{>0}A$ and $P_{=1}A$, respectively, occurring in a normalized TBox $\mathcal{T}$.

The completion algorithm for Prob-$\mathcal{EL}^{01}$ follows the same idea as the algorithm for $\mathcal{EL}$, but uses several completion sets to deal with the information of what needs to be satisfied in the different worlds of a model. We define the set of worlds $V := \{0, \varepsilon, 1\} \cup \mathcal{P}_0^{\mathcal{T}}$, where the probability distribution μ assigns a probability of 0 to the world 0, and the uniform probability $1/(|V| - 1)$ to all other worlds. For each concept name A, role name r and world v, we store the

completion sets $S_0(A, v), S_\varepsilon(A, v), S_0(A, r, v)$, and $S_\varepsilon(A, r, v)$. These completion sets are simple generalizations of the completion sets for crisp $\mathcal{EL}$. Intuitively, $D \in S_0(C, v)$ implies $C \sqsubseteq D$ if $v = 0$, $C \sqsubseteq P_{=1}D$ if $v = 1$, and $C \sqsubseteq P_{>0}D$, otherwise. Likewise, $D \in S_\varepsilon(C, v)$ implies $P_{>0}C \sqsubseteq D$ if $v = 0$, $P_{>0}C \sqsubseteq P_{=1}D$ if $v = 1$, and $P_{>0}C \sqsubseteq P_{>0}D$, otherwise.

The algorithm initializes the sets as follows for every $A \in \mathsf{BC}_\mathcal{T}$ and $r \in \mathsf{R}_\mathcal{T}$:

- $S_0(A, 0) = \{\top, A\}$ and $S_0(A, v) = \{\top\}$ for all $v \in V \setminus \{0\}$,
- $S_\varepsilon(A, \varepsilon) = \{\top, A\}$ and $S_\varepsilon(A, v) = \{\top\}$ for all $v \in V \setminus \{\varepsilon\}$,
- $S_0(A, r, v) = S_\varepsilon(A, r, v) = \emptyset$ for all $v \in V$.

These sets are then extended by exhaustively applying the rules shown in Figure 3, where $* \in \{0, \varepsilon\}$ and $\gamma : V \to \{0, \varepsilon\}$ is defined by $\gamma(0) = 0$, and $\gamma(v) = \varepsilon$ for all $v \in V \setminus \{0\}$.

The first four rules are simple adaptations of the completion rules for $\mathcal{EL}$, while the last four rules deal with probabilistic concepts. This algorithm terminates in polynomial time. After termination it holds that, for every pair of concept names A, B, $B \in S_0(A, 0)$ if and only if $A \sqsubseteq_\mathcal{T} B$ [12].

4 Computing Least Common Subsumers using Completion

The least common subsumer was first mentioned in [8] and has since been investigated for several DLs. However, most lcs computation algorithms were devised for concept descriptions only or for unfoldable TBoxes (see e.g. [5]) and are not capable of handling general TBoxes. In case of $\mathcal{EL}$ the lcs has been investigated for cyclic TBoxes: the lcs does not need to exist w.r.t. descriptive semantics [2], which is the usual semantics for DLs. One approach to compute the lcs even in the presence of GCIs is to use different semantics for the underlying DL, e.g., greatest fixed-point semantics have been employed in [1, 7]. A different approach was followed in [6, 16], where the lcs was investigated for unfoldable TBoxes written in a "small" DL using concepts from an expressive general background TBox.

All approaches for proving the (non-)existence of the lcs or devising computation algorithms for the lcs are built on a characterization of subsumption for the respective DL and for the underlying TBox formalism. For instance, the lcs algorithm for $\mathcal{EL}$-concept descriptions [5] uses homomorphisms between so-called $\mathcal{EL}$-description trees. The work on the lcs w.r.t. cyclic $\mathcal{EL}$-TBoxes [1, 2] uses (synchronized) simulations between $\mathcal{EL}$-description graphs to characterize subsumption. In this paper we use the completion algorithm from [3] as the underlying characterization of subsumption to obtain a role-depth bounded lcs in $\mathcal{EL}$.

Formally the lcs is defined as follows. Let $\mathcal{T}$ be a TBox and C, D concept descriptions in the DL $\mathcal{L}$, then the $\mathcal{L}$-concept description L is the *least common subsumer (lcs)* of C, D w.r.t. $\mathcal{T}$ (written $lcs_\mathcal{T}(C, D)$) iff

1. $C \sqsubseteq_{\mathcal{T}} L$ and $D \sqsubseteq_{\mathcal{T}} L$, and
2. for all $\mathcal{L}$-concept descriptions E it holds that,
   $C \sqsubseteq_{\mathcal{T}} E$ and $D \sqsubseteq_{\mathcal{T}} E$ implies $L \sqsubseteq_{\mathcal{T}} E$.

Note, that the lcs is defined w.r.t. to a certain DL $\mathcal{L}$. In cases where the lcs is computed for concept descriptions, we can simply use an empty TBox $\mathcal{T}$. Due to the associativity of the lcs operator, the lcs can be defined as a n-ary operation. However, we stick to its binary version for simplicity of the presentation.

4.1 Role-depth bounded lcs in $\mathcal{EL}$

As mentioned, the lcs does not need to exist due to cycles in the TBox. Consider the TBox $\mathcal{T} = \{A \sqsubseteq \exists r.A \sqcap C,\ B \sqsubseteq \exists r.B \sqcap C\}$. The lcs of A and B is then $C \sqcap \exists r.(C \sqcap \exists r.(C \sqcap \exists r.(C \sqcap \cdots$ and cannot be expressed by a finite concept description. To avoid such infinite nestings, we limit the role-depth of the concept description to be computed. Let C, D be $\mathcal{EL}$-concept descriptions, then the *role-depth* of a concept description C (denoted $rd(C)$) is:

- 0 for concept names and $\top$
- $max(rd(C), rd(D))$ for a conjunction $C \sqcap D$, and
- $1 + rd(C)$ for existential restrictions of the form $\exists r.C$.

Now we can define the lcs with limited role-depth for $\mathcal{EL}$.

Definition 3 (Role-depth bounded $\mathcal{EL}$-lcs). *Let $\mathcal{T}$ be an $\mathcal{EL}$-TBox and C, D $\mathcal{EL}$-concept descriptions and $k \in \mathbb{N}$. Then the $\mathcal{EL}$-concept description L is the role-depth bounded $\mathcal{EL}$-least common subsumer of C, D w.r.t. $\mathcal{T}$ and role-depth k (written k-lcs(C, D)) iff*

1. *$rd(L) \leq k$,*
2. *$C \sqsubseteq_{\mathcal{T}} L$ and $D \sqsubseteq_{\mathcal{T}} L$, and*
3. *for all $\mathcal{EL}$-concept descriptions E with $rd(E) \leq k$ it holds that,*
   $C \sqsubseteq_{\mathcal{T}} E$ and $C \sqsubseteq_{\mathcal{T}} E$ implies $L \sqsubseteq_{\mathcal{T}} E$.

4.2 Computing the Role-depth Bounded $\mathcal{EL}$- lcs

The computation algorithm for the role-depth bounded lcs w.r.t. general $\mathcal{EL}$-TBoxes, constructs the concept description from the set of completion sets. More precisely, it combines and intersects the completion sets in the same fashion as in the cross-product computation in the lcs algorithm for $\mathcal{EL}$-concept descriptions from [5].

However, the completion sets may contain concept names that were introduced during normalization. The returned lcs-concept description should only contain concept names that appear in the initial TBox, thus we need to "de-normalize" the concept descriptions obtained from the completion sets.

De-normalizing $\mathcal{EL}$-concept Descriptions. The *signature of a concept description* C (denoted $\mathsf{sig}(C)$) is the set of concept names and role names that appear in C. The *signature of a TBox* $\mathcal{T}$ (denoted $\mathsf{sig}(\mathcal{T})$) is the set of concept names and role names that appear in $\mathcal{T}$.

 Clearly, the signature of $\mathcal{T}$ may be extended during normalization. To capture the relation between $\mathcal{T}$ and its normalized variant, we introduce the notion of a *conservative extension* as in [13].

Definition 4 ($\mathsf{sig}(\mathcal{T})$-inseparable, conservative extension). *Let $\mathcal{T}_1, \mathcal{T}_2$ be $\mathcal{EL}$-TBoxes.*

- $\mathcal{T}_1$ *and* $\mathcal{T}_2$ *are* $\mathsf{sig}(\mathcal{T}_1)$*-inseparable w.r.t. concept inclusion in $\mathcal{EL}$, iff for all $\mathcal{EL}$-concept descriptions C, D with $\mathsf{sig}(C) \cup \mathsf{sig}(D) \subseteq \mathsf{sig}(\mathcal{T}_1)$, we have $\mathcal{T}_1 \models C \sqsubseteq D$ iff $\mathcal{T}_2 \models C \sqsubseteq D$.*
- $\mathcal{T}_2$ *is a* conservative extension *of $\mathcal{T}_1$ w.r.t. concept inclusion in $\mathcal{EL}$, if*
  - $\mathcal{T}_1 \subseteq \mathcal{T}_2$*, and*
  - $\mathcal{T}_1$ *and* $\mathcal{T}_2$ *are* $\mathsf{sig}(\mathcal{T}_1)$*-inseparable w.r.t. concept inclusion in $\mathcal{EL}$.*

However, the extension of the signature by normalization according to the normalization rules from Figure 1 does not affect subsumption tests for $\mathcal{EL}$-concept descriptions formulated w.r.t. the initial signature of $\mathcal{T}$.

Lemma 1. *Let $\mathcal{T}$ be an $\mathcal{EL}$-TBox and $\mathcal{T}'$ the TBox obtained from $\mathcal{T}$ by applying the $\mathcal{EL}$ normalization rules, C, D be $\mathcal{EL}$-concept descriptions with $\mathsf{sig}(C) \subseteq \mathsf{sig}(\mathcal{T})$ and $\mathsf{sig}(D) \subseteq \mathsf{sig}(\mathcal{T}')$ and D' be the concept description obtained by replacing all names $A \in \mathsf{sig}(\mathcal{T}') \setminus \mathsf{sig}(\mathcal{T})$ from D with $\top$. Then $C \sqsubseteq_{\mathcal{T}'} D$ iff $C' \sqsubseteq_{\mathcal{T}} D'$.*

Proof. Since $\mathcal{T}'$ is a conservative extension of $\mathcal{T}$ w.r.t. concept inclusion in $\mathcal{EL}$, it is implied that $\mathcal{T}$ and $\mathcal{T}'$ are $\mathsf{sig}(\mathcal{T})$*-inseparable w.r.t. concept inclusion in $\mathcal{EL}$.* Thus the claim follows directly. $\qquad\square$

 Lemma 1 guarantees that subsumption relations w.r.t. the normalized TBox $\mathcal{T}'$ between C and D, also hold w.r.t. the original TBox $\mathcal{T}$ for C and D', which is basically obtained from D by removing the names introduced by normalization, i.e., concept names from $\mathsf{sig}(\mathcal{T}') \setminus \mathsf{sig}(\mathcal{T})$.

A Computation Algorithm for k-lcs. We assume that the role-depth of each input concept of the lcs has a role-depth less or equal to k. This assumption is motivated by the applications of the lcs on the one hand and on the other by the simplicity of presentation, rather than a technical necessity. The algorithm for computing the role-depth bounded lcs of two $\mathcal{EL}$-concept descriptions is depicted in Algorithm 1. It consists of the procedure k-lcs, which calls the recursive procedure k-lcs-r.

 The procedure k-lcs first adds concept definitions for the input concept descriptions to (a copy of) the TBox and transforms this TBox into the normalized TBox $\mathcal{T}'$. Next, it calls the procedure apply-completion-rules, which applies the $\mathcal{EL}$ completion rules exhaustively to the TBox $\mathcal{T}'$, and stores the obtained set of

---

Algorithm 1 Computation of a role-depth bounded $\mathcal{EL}$-lcs.

Procedure k-lcs $(C, D, \mathcal{T}, k)$
Input: C, D: $\mathcal{EL}$-concept descriptions; $\mathcal{T}$: $\mathcal{EL}$-TBox; k: natural number
Output: $k\text{-}lcs(C, D)$: role-depth bounded $\mathcal{EL}$-lcs of C and D w.r.t $\mathcal{T}$ and k.

1: $\mathcal{T}' :=$ normalize$(\mathcal{T} \cup \{A \equiv C, B \equiv D\})$
2: $\mathsf{S}_{\mathcal{T}'} :=$ apply-completion-rules$(\mathcal{T}')$
3: $L :=$ k-lcs-r $(A, B, \mathsf{S}_{\mathcal{T}'}, k)$
4: **if** $L = A$ **then return** C
5: **else if** $L = B$ **then return** D
6: **else return** remove-normalization-names(L)
7: **end if**

Procedure k-lcs-r (A, B, S, k)
Input: A, B: concept names; S: set of completion sets; k: natural number
Output: $k\text{-}lcs(A, B)$: role-depth bounded $\mathcal{EL}$-lcs of A and B w.r.t $\mathcal{T}$ and k.

1: **if** $B \in S(A)$ **then return** B
2: **else if** $A \in S(B)$ **then return** A
3: **end if**
4: common-names $:= S(A) \cap S(B)$
5: **if** $k = 0$ **then return** $\displaystyle\bigsqcap_{P \in \text{common–names}} P$
6: **else return** $\displaystyle\bigsqcap_{P \in \text{common–names}} P \sqcap$
$$\bigsqcap_{r \in \mathsf{R}_{\mathcal{T}}} \Big(\bigsqcap_{(E,F) \ \in \ S(A,r) \times S(B,r)} \exists r.\ \text{k-lcs-r} \ (E, F, \mathsf{S}, k-1) \Big)$$
7: **end if**

---

completion sets in S. Then it calls the function k-lcs-r with the concept names A and B for the input concepts, the set of completion sets S, and the role-depth limit k. The result is then de-normalized and returned (lines 4 to 6). More precisely, in case a complex concept description is returned from k-lcs-r, the procedure remove-normalization-names removes concept names that were added during the normalization of the TBox.

The function k-lcs-r gets a pair of concept names, a set of completion sets and a natural number as inputs. First, it tests whether one of the input concepts subsumes the other w.r.t. $\mathcal{T}'$. In that case the name of the subsuming concept is returned. Otherwise the set of concept names that appear in the completion sets of both input concepts is stored in common-names (line 4).[1] In case the role-depth bound is reached ($k = 0$), the conjunction of the elements in common-names is returned. Otherwise, the elements in common-names are conjoined with a conjunction over all roles $r \in \mathsf{R}_{\mathcal{T}}$, where for each r and each element of the cross-product over the r-successors of the current A and B a recursive call to k-lcs-r is made with the role-depth bound reduced by 1 (line 6). This conjunction is then returned to k-lcs.

---

[1] Note, that the intersection $S(A) \cap S(B)$ is never empty, since both sets contain $\top$.

For $L = \mathsf{k\text{-}lcs}(C, D, \mathcal{T}, k)$ it holds by construction that $rd(L) \leq k$.[2] We now show that the result of the function $\mathsf{k\text{-}lcs}$ is a common subsumer of the input concept descriptions.

Lemma 2. *Let C and D be $\mathcal{EL}$-concept descriptions, $\mathcal{T}$ an $\mathcal{EL}$-TBox, $k \in \mathbb{N}$ and $L = \mathsf{k\text{-}lcs}(C, D, \mathcal{T}, k)$. Then $C \sqsubseteq_{\mathcal{T}} L$ and $D \sqsubseteq_{\mathcal{T}} L$.*

Lemma 1 justifies to replace "normalization names" in the concept description constructed from the normalization sets in the fashion described earlier and still preserve the subsumption relationships. Lemma 2 can be shown by induction on k. For the full proof see [14].

Next, we show that the result of the function $\mathsf{k\text{-}lcs}$ obtained for $\mathcal{EL}$-concept descriptions C and D is the least (w.r.t. subsumption) concept description of role-depth up to k that subsumes the input concepts, see [14].

Lemma 3. *Let C and D be $\mathcal{EL}$-concept descriptions, $\mathcal{T}$ an $\mathcal{EL}$-TBox, $k \in \mathbb{N}$ and $L = \mathsf{k\text{-}lcs}(C, D, \mathcal{T}, k)$ and E an $\mathcal{EL}$-concept description with $rd(E) \leq k$. If $C \sqsubseteq_{\mathcal{T}} E$ and $D \sqsubseteq_{\mathcal{T}} E$, then $L \sqsubseteq_{\mathcal{T}} E$.*

We obtain together with Lemma 2 and Lemma 3 that all conditions of Definition 3 are fulfilled for $\mathsf{k\text{-}lcs}(C, D, \mathcal{T}, k)$.

Theorem 1. *Let C and D be $\mathcal{EL}$-concept descriptions, $\mathcal{T}$ an $\mathcal{EL}$-TBox, $k \in \mathbb{N}$, then $\mathsf{k\text{-}lcs}(C, D, \mathcal{T}, k) \equiv k\text{-}lcs(C, D)$.*

For cases where $\mathsf{k\text{-}lcs}$ returns a concept description with role-depth of less than k we conjecture that it is the exact lcs.

The complexity of the overall method is exponential. However, if a compact representation of the lcs with structure sharing is used, the lcs-concept descriptions can be represented polynomially. In contrast to the lcs algorithm for $\mathcal{EL}$-concept descriptions, the algorithm $\mathsf{k\text{-}lcs}$ does not need to copy concepts[3] that are referenced several times, but proceeds by structure sharing by re-using the completion sets. Thus completion-based algorithm is even advantageous for unfoldable $\mathcal{EL}$-TBoxes such as Snomed.

Moreover, if a $k\text{-}lcs$ is too general and a bigger role depth of the $k\text{-}lcs$ is desired, the completion of the TBox does not have to be redone for a second computation. The completion sets can simply be "traversed" further.

4.3 Computing the Role-depth Bounded Prob-$\mathcal{EL}^{01}$-lcs

The computation of the role-depth bounded Prob-$\mathcal{EL}^{01}$-lcs follows the same steps as in Section 4.2. First, it adds concept definitions for the input concepts to the TBox and normalizes it. It then applies the completion rules from Figure 3 exhaustively to produce the set of completion sets S. It then calls a variation of the function $\mathsf{k\text{-}lcs\text{-}r}$ that can deal with probabilistic concepts. The new function

---

[2] Recall our assumption: the role-depth of each input concept is less or equal to k.

[3] as typically done during unfolding

k-lcs-r is identical to the one presented in Algorithm 1, except that in line 6 it now returns:

$$\prod_{P \in \mathsf{common-names}} P \ \sqcap \prod_{r \in \mathsf{R}_\mathcal{T}} (\prod_{(E,F) \in S_0(A,r,0) \times S_0(B,r,0)} \exists r.\mathsf{k\text{-}lcs\text{-}r}(E,F,\mathsf{S},k-1) \sqcap$$
$$\prod_{(E,F) \in S_0^{>0}(A,r) \times S_0^{>0}(B,r)} P_{>0}(\exists r.\mathsf{k\text{-}lcs\text{-}r}(E,F,\mathsf{S},k-1)) \sqcap$$
$$\prod_{(E,F) \in S_0(A,r,1) \times S_0(B,r,1)} P_{=1}(\exists r.\mathsf{k\text{-}lcs\text{-}r}(E,F,\mathsf{S},k-1))),$$

where $S_0^{>0}(A,r) := \bigcup_{v \in V \setminus \{0\}} S_0(A,r,v)$. The result is then de-normalized by removing all concept names that were introduced during the normalization phase. The correctness of this procedure can be shown in a similar way as it was done for $\mathcal{EL}$ before.

Theorem 2. *Let C and D be Prob-$\mathcal{EL}^{01}$-concept descriptions, $\mathcal{T}$ a Prob-$\mathcal{EL}^{01}$-TBox, and $k \in \mathbb{N}$; then $\mathsf{k\text{-}lcs}(C,D,\mathcal{T},k) \equiv k\text{-}lcs(C,D)$.*

Again, the proof is given in [14].

5   Conclusions

In this paper we have presented a practical method for computing the role-depth bounded lcs of $\mathcal{EL}$-concepts w.r.t. a general TBox. Our approach is based on the completion sets that are computed during classification of a TBox. Thus, any of the available implementation of the $\mathcal{EL}$ completion algorithm can be easily extended to an implementation of the lcs computation algorithm. We also showed that the same idea can be adapted for the computation of the lcs in the probabilistic DL Prob-$\mathcal{EL}^{01}$.

As future work, we want to investigate the computation of the most specific Prob-$\mathcal{EL}^{01}$ concept (msc) that describes a given individual in an ABox. We also plan to investigate the bottom-up constructions (i. e. lcs and msc computations) in more expressive probabilistic DLs. One possible extension is by studying Prob-$\mathcal{ALE}$. A second approach is to weaken the restrictions imposed in Prob-$\mathcal{EL}^{01}$, allowing for different probabilistic constructors.

References

1. F. Baader. Least common subsumers and most specific concepts in a description logic with existential restrictions and terminological cycles. In *Proc. of the 18th Int. Joint Conf. on Artificial Intelligence (IJCAI-03)*, 2003.
2. F. Baader. Terminological cycles in a description logic with existential restrictions. In *Proc. of the 18th Int. Joint Conf. on Artificial Intelligence (IJCAI-03)*, 2003.
3. F. Baader, S. Brandt, and C. Lutz. Pushing the $\mathcal{EL}$ envelope. In *Proc. of the 19th Int. Joint Conf. on Artificial Intelligence (IJCAI-05)*, 2005.

4. F. Baader, S. Brandt, and C. Lutz. Pushing the $\mathcal{EL}$ envelope. LTCS-Report LTCS-05-01, Chair for Automata Theory, Institute for Theoretical Computer Science, Dresden University of Technology, Germany, 2005. See http://lat.inf.tu-dresden.de/research/reports.html.

5. F. Baader, R. Küsters, and R. Molitor. Computing least common subsumer in description logics with existential restrictions. In *Proc. of the 16th Int. Joint Conf. on Artificial Intelligence (IJCAI-99)*, 1999.

6. F. Baader, B. Sertkaya, and A.-Y. Turhan. Computing the least common subsumer w.r.t. a background terminology. *Journal of Applied Logics*, 2007.

7. S. Brandt. *Standard and Non-standard Reasoning in Description Logics*. PhD thesis, Institute for Theoretical Computer Science, TU Dresden, 2006.

8. W. W. Cohen, A. Borgida, and H. Hirsh. Computing least common subsumers in description logics. In *Proc. of the 10th Nat. Conf. on Artificial Intelligence (AAAI-92)*, 1992.

9. J. Y. Halpern. An analysis of first-order logics of probability. *Artificial Intelligence*, 46:311–350, 1990.

10. P. Klinov, B. Parsia, and U. Sattler. On correspondences between probabilistic first-order and description logics. In *Proc. of the 2008 Description Logic Workshop (DL 2009)*, volume 477 of *CEUR*, 2009.

11. T. Lukasiewicz. Expressive probabilistic description logics. *Artificial Intelligence*, 172(6-7):852–883, 2008.

12. C. Lutz and L. Schröder. Probabilistic description logics for subjective probabilities. In *Proc. of the 12th Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR-10)*, 2010. To appear.

13. C. Lutz and F. Wolter. Deciding inseparability and conservative extensions in the description logic $\mathcal{EL}$. *Journal of Symbolic Computation*, 45(2):194 – 228, 2010.

14. R. Peñaloza and A.-Y. Turhan. Completion-based computation of least common subsumers with limited role-depth for $\mathcal{EL}$ and prob-$\mathcal{EL}^{01}$. LTCS-Report LTCS-10-02, Chair f. Automata Theory, Inst. for Theoretical Computer Science, TU Dresden, Germany, 2010. See http://lat.inf.tu-dresden.de/research/reports.html.

15. K. Spackman. Managing clinical terminology hierarchies using algorithmic calculation of subsumption: Experience with snomed-rt. *Journal of the American Medical Informatics Assoc.*, 2000. Fall Symposium Special Issue.

16. A.-Y. Turhan. *On the Computation of Common Subsumers in Description Logics*. PhD thesis, TU Dresden, Institute for Theoretical Computer Science, 2007.

Query Rewriting in *DL-Lite*$_{horn}^{(\mathcal{HN})}$ *

Elena Botoeva, Alessandro Artale, and Diego Calvanese

KRDB Research Centre
Free University of Bozen-Bolzano
I-39100 Bolzano, Italy
`lastname@inf.unibz.it`

Abstract. In this paper we present practical algorithms for query answering and knowledge base satisfiability checking in *DL-Lite*$_{horn}^{(\mathcal{HN})}$, a logic from the extended *DL-Lite* family that contains horn concept inclusions and number restriction. This logic is the most expressive DL that is shown to be FOL-rewritable. The algorithms we present are based on the rewriting technique so that reasoning over the TBox and over the ABox can be done independently of each other, and the inference problems are reduced to first order query evaluation. This allows for employing relational database technology for the final query evaluation and gives optimal data complexity.

1 Introduction

Query answering is the main reasoning task in the setting of ontology based data access [1,2] and data integration [3], where large amounts of data are stored in external databases, and accessed through a conceptual layers provided by an ontology (expressed in terms of a description logic knowledge base). Query answering in this case requires reasoning, and to perform it efficiently in practice the underlying description logic has to be 'lite' enough, to be more precise it should enjoy first-order rewritability: it should be possible to rewrite a query q posed over the ontology in terms of a new query that can be directly evaluated over the data, and that provides the same answers as those provided by q.

The *DL-Lite* family [4] is a family of description logics that enjoy such nice computational properties, i.e., data complexity of query answering is in AC^0. For *DL-Lite*$_{core}$ and *DL-Lite*$_{core}^{\mathcal{F}}$ [1], the basic logics of the *DL-Lite* family, a polynomial query answering algorithm was established in [4], and later extended to *DL-Lite*$_{\mathcal{A}}$ [1]. The idea of the algorithm is to first rewrite the query by taking into account the assertions in the TBox and then to evaluate the rewritten query over the ABox. Based on this approach, several systems were implemented, notably QuOnto [6,7].

DL-Lite$_{horn}^{(\mathcal{HN})}$ is a more expressive logic than *DL-Lite*$_{\mathcal{A}}$, which contains number restrictions (as opposed to global functionality assertions), allows for conjunction of basic concepts on the left-hand side of concept inclusions, and for role inclusions that

---

* This work has been partially supported by the EU project Ontorule (ICT-231875).

^{1} Notice that we adopt here the naming convention for logics introduced in [5].

however cannot interact with maximum number restriction. As shown in [5], also *DL-Lite$_{horn}^{(\mathcal{HN})}$* is FOL-rewritable, i.e., a similar approach to the one discussed above can be used for query answering over ontologies. However, the algorithm presented in [5] is not immediately implementable, since it would generate queries that would be extremely difficult to process and optimize by a DBMS. Indeed as demonstrated by recent experiments with ontology based data access systems [2], commercial relational DBMSs are not designed and optimized to process complex queries (where, e.g., joins are performed over unions), and for such kinds of queries performance degrades dramatically when the size of the data increases.

Therefore, in this paper we address the problem of devising an algorithm for answering unions of conjunctive queries in *DL-Lite$_{horn}^{(\mathcal{HN})}$* that is based on rewriting, and where the rewriting step generates again a union of conjunctive queries. Such an algorithm can be directly implemented in a system like QUONTO by extending the current algorithm for *DL-Lite$_{\mathcal{A}}$*. Moreover, since current DBMSs are optimized for the evaluation of conjunctive queries, they can process the queries generated by our rewriting algorithm more efficiently than queries generated by an algorithm that tries to delegate complex operations to the DBMS.

Summing up, our contributions are the following:

- We present an algorithm for answering unions of conjunctive queries posed to *DL-Lite$_{horn}^{(\mathcal{HN})}$* knowledge bases that is in AC^0 for data complexity. We employ the query rewriting approach, that is, query answering is performed in two steps. First, the initial query is rewritten using the TBox. Then, the rewritten query is evaluated over the ABox. The main advantage of this approach is that the part of the process requiring TBox reasoning is independent of the ABox, and the part of the process requiring access to the ABox can be carried out by an SQL engine.

- We provide an algorithm that checks satisfiability of *DL-Lite$_{horn}^{(\mathcal{HN})}$* knowledge bases by evaluating a first-order query over the ABox and that is in AC^0 for data complexity. Thus, the knowledge base satisfiability problem is reduced to query evaluation and again the TBox and the ABox are processed independently of each other.

2   The Description Logic *DL-Lite$_{horn}^{(\mathcal{HN})}$*

In this section we present the logic *DL-Lite$_{horn}^{(\mathcal{HN})}$* and give other preliminary definitions.

The language of *DL-Lite$_{horn}^{(\mathcal{HN})}$* contains atomic concept and role names, respectively denoted by A and P, possibly with subscripts. *Basic roles* and *concepts*, denoted respectively by R and B, possibly with subscripts, are defined as $R ::= P \mid P^-$ and $B ::= \bot \mid A \mid\, \geq k\,R$, where k is a positive integer. $\geq k\,R$ is called a *number restriction*.

A *DL-Lite$_{horn}^{(\mathcal{HN})}$* *TBox*, $\mathcal{T}$, is a finite set of *concept* and *role inclusion axioms* of the form $B_1 \sqcap \cdots \sqcap B_n \sqsubseteq B$ and $R_1 \sqsubseteq R_2$, respectively, and *role constraints* $\mathsf{Dis}(R_1, R_2)$, $\mathsf{Asym}(P)$, $\mathsf{Sym}(P)$, $\mathsf{Irr}(P)$ and $\mathsf{Ref}(P)$. The TBox $\mathcal{T}$ may also contain occurrences of qualified number restrictions $\geq k\,R.B$ on the right-hand side of concept inclusions. The TBox assertions must satisfy the following conditions:

(inter) if R has a proper sub-role in $\mathcal{T}$ (i.e., $R' \sqsubseteq R$, $R \not\sqsubseteq R'$ for some R'), then $\mathcal{T}$ does not contain occurrences of number restrictions $\geq k\,R$ or $\geq k\,R^-$ with $k \geq 2$ on the left-hand side of concept inclusions.

(exists) if $\geq k\,R.B$ occurs in $\mathcal{T}$, then $\mathcal{T}$ does not contain occurrences of $\geq k'\,R$ or $\geq k'\,R^-$, for $k' \geq 2$, on the left-hand side of concept inclusions.

Note that disjointness between concepts B_1 and B_2 is expressed as $B_1 \sqcap B_2 \sqsubseteq \perp$. Also, *DL-Lite*$_{horn}^{(\mathcal{HN})}$ allows for expressing local cardinality constraints, e.g., the assertion $A \sqcap \geq 3\,R \sqsubseteq \perp$ is equivalent to $A \sqsubseteq \leq 2\,R$.

An *ABox* $\mathcal{A}$ is a finite set of *membership assertions* of the form $A(a)$, $\neg A(a)$, $P(a, b)$, and $\neg P(a, b)$. $\mathcal{T}$ and $\mathcal{A}$ constitute the *knowledge base* $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$.

In the following, we use R^- to denote P^- if $R = P$ and P if $R = P^-$, $R(x, y)$ to denote $P(x, y)$ if $R = P$, and $P(y, x)$ if $R = P^-$. For a TBox $\mathcal{T}$, let $\sqsubseteq^{\pm}$ denote the closure under inverses of the subrole relation: $R \sqsubseteq^{\pm} R' \in \mathcal{T}$ iff $R \sqsubseteq R' \in \mathcal{T}$ or $R^- \sqsubseteq R'^- \in \mathcal{T}$.

The formal semantics relies on the standard notion of interpretation [8]. Here we adopt the *unique name assumption* (UNA). In the following we assume that TBoxes do not contain role constraints $\mathsf{Asym}(P)$, $\mathsf{Sym}(P)$, $\mathsf{Irr}(P)$, $\mathsf{Ref}(P)$ and qualified number restrictions: we can get rid of them as described in [5].

In this work we concentrate on two reasoning tasks for *DL-Lite*$_{horn}^{(\mathcal{HN})}$, to which other reasoning tasks can be reduced: knowledge base satisfiability and query answering. The *KB satisfiability problem* is to check, given a KB $\mathcal{K}$, whether $\mathcal{K}$ admits at least one model. To define the query answering problem, we first provide some definitions.

A *conjunctive query (CQ) q over a KB* $\mathcal{K}$ is a first-order formula of the form: $q(\boldsymbol{x}) = \exists \boldsymbol{y}.conj(\boldsymbol{x}, \boldsymbol{y})$, where $conj(\boldsymbol{x}, \boldsymbol{y})$ is a conjunction of atoms of the form $A(t)$ and $P(t_1, t_2)$, and t, t_1, t_2 are either constants in $\mathcal{K}$ or variables in $\boldsymbol{x}$ and $\boldsymbol{y}$, $\boldsymbol{x}$ are the free variables of q, also called *distinguished* variables. A *union of conjunctive queries (UCQ)* q is a formula of the form $q(\boldsymbol{x}) = \bigvee_{i=1,\ldots,n} \exists \boldsymbol{y}_i.conj_i(\boldsymbol{x}, \boldsymbol{y}_i)$, with $conj_i(\boldsymbol{x}, \boldsymbol{y}_i)$ as before. Given a query q (either a CQ or a UCQ) and an interpretation $\mathcal{I}$, we denote by $q^{\mathcal{I}}$ the set of tuples of elements of $\Delta^{\mathcal{I}}$ obtained by evaluating q in $\mathcal{I}$. The answer to q over a KB $\mathcal{K}$ is the set $ans(q, \mathcal{K})$ of tuples $\boldsymbol{a}$ of constants appearing in $\mathcal{K}$ such that $\boldsymbol{a}^{\mathcal{I}} \in q^{\mathcal{I}}$, for every model $\mathcal{I}$ of $\mathcal{K}$. Each such tuple is called *certain answer*. Observe that, if $\mathcal{K}$ is unsatisfiable, then $ans(q, \mathcal{K})$ is trivially the set of all possible tuples of constants in $\mathcal{K}$ whose arity is the one of the query. We denote such a set by *AllTup*$(q, \mathcal{K})$. The *query answering problem* is defined as follows: given a KB $\mathcal{K}$ and a query q (either a CQ or a UCQ) over $\mathcal{K}$, compute the set $ans(q, \mathcal{K})$.

3 Knowledge Base Satisfiability

To check KB satisfiability, we exploit the notions of canonical interpretation and closure of negative inclusions. First, we define negative and positive inclusion assertions, and the database interpretation.

We call *negative inclusions (NI)* assertions of the form $B_1 \sqcap \cdots \sqcap B_n \sqsubseteq \perp$ and $\mathsf{Dis}(R_1, R_2)$. Other assertions are called *positive inclusions (PI)*. $\mathcal{T}_{\mathcal{N}}$ denotes the set of all negative inclusions in $\mathcal{T}$, and $\mathcal{T}_{\mathcal{P}}$ the set of positive inclusions in $\mathcal{T}$. Obviously,

$\mathcal{T} = \mathcal{T}_{\mathcal{N}} \cup \mathcal{T}_{\mathcal{P}}$. Note that NIs include also those assertions that express functionality of roles, e.g. $\geq 2\,R \sqsubseteq \perp$, and more in general maximum number restrictions.

The *database interpretation* $db(\mathcal{A}) = \langle \Delta^{db(\mathcal{A})}, \cdot^{db(\mathcal{A})} \rangle$ of an ABox $\mathcal{A}$ is defined as follows: $\Delta^{db(\mathcal{A})}$ is the nonempty set consisting of all constants occurring in $\mathcal{A}$; $a^{db(\mathcal{A})} = a$, for each constant a; $A^{db(\mathcal{A})} = \{a \mid A(a) \in \mathcal{A}\}$, for each atomic concept A; $P^{db(\mathcal{A})} = \{(a_1, a_2) \mid P(a_1, a_2) \in \mathcal{A}\}$, for each atomic role P.

The canonical interpretation is an interpretation constructed according to the notion of chase [9]. Following [4] we can construct the chase of a KB starting from the ABox, and applying positive inclusions to sets of membership assertions. In the definition of chase, we concentrate here on the differences with respect to the definition of chase given in [4]. We remark, however, that for the application of the chase rules we assume to have a total (lexicographic) ordering on the assertions and on all the constants (including the newly introduced ones). The *chase* of $\mathcal{K}$ is the set of membership assertions $chase(\mathcal{K}) = \bigcup_{j \in \mathbb{N}} \mathcal{S}_j$, where $\mathcal{S}_0 = \mathcal{A}$, $\mathcal{S}_{j+1} = \mathcal{S}_j \cup S_j^{new}$, and S_j^{new} is the set of new membership assertions obtained from $\mathcal{S}_j$ according to the chase rules cr1, cr2, cr3:

cr1   if $I = B_1 \sqcap \cdots \sqcap B_n \sqsubseteq A$, $S' = S_{B_1}(a) \cup \cdots \cup S_{B_n}(a)$, and $A(a) \notin \mathcal{S}_j$, then $S_j^{new} = \{A(a)\}$,

cr2   if $I = B_1 \sqcap \cdots \sqcap B_n \sqsubseteq \geq k\,R$ and $S' = S_{B_1}(a) \cup \cdots \cup S_{B_n}(a)$, $k_1 = \sharp\{b \mid R(a, b) \in S\} < k$, then $S_j^{new} = \{R(a, b_{k_1+1}), \ldots, R(a, b_k)\}$, where $b_{k_1+1}, \ldots, b_k$ are $k - k_1$ new constants in Γ_N that follow lexicographically the constants introduced in the previous steps,

cr3   if $I = R_1 \sqsubseteq^{\pm} R_2$, $S' = \{R_1(a, b)\}$, and $R_1(a, b) \notin \mathcal{S}_j$ then $S_j^{new} = \{R_2(a, b)\}$,

where S' is the first (in lexicographic order) set of membership assertions in $\mathcal{S}_j$ s.t. there exists a PI applicable[2] to it, I is the first such PI, and we use $S_B(a)$ to denote $\{A(a)\}$ if $B = A$ and $\{R(a, b_1), \ldots, R(a, b_k)\}$ if $B = \geq k\,R$, with $b_1, \ldots, b_k$ some constants. We denote by $chase_i(\mathcal{K})$ the portion of the chase obtained after i applications of the chase rules.

The canonical interpretation is the interpretation $can(\mathcal{K}) = \langle \Delta^{can(\mathcal{K})}, \cdot^{can(\mathcal{K})} \rangle$ where $\Delta^{can(\mathcal{K})}$ is the set of constants occurring in $chase(\mathcal{K})$, $a^{can(\mathcal{K})} = a$, for each constant a, $A^{can(\mathcal{K})} = \{a \mid A(a) \in chase(\mathcal{K})\}$, for each atomic concept A, and $P^{can(\mathcal{K})} = \{(a_1, a_2) \mid P(a_1, a_2) \in chase(\mathcal{K})\}$, for each atomic role P. The canonical interpretation is constructed in such a way that all the positive inclusions of the TBox are satisfied, so a knowledge base where the TBox contains only positive inclusions is always satisfiable. The following lemma establishes a notable property of $can(\mathcal{K})$.

Lemma 1. *Let $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ be a DL-Lite$_{horn}^{(\mathcal{HN})}$ KB, and let $\mathcal{T}_{\mathcal{P}}$ be the set of positive inclusion assertions in $\mathcal{T}$. Then $can(\mathcal{K})$ is a model of $\langle \mathcal{T}_{\mathcal{P}}, \mathcal{A} \rangle$.*

In order to check satisfiability of *DL-Lite$_{horn}^{(\mathcal{HN})}$* KBs, negative inclusions must be considered. Thus, if a negative inclusion in the TBox is violated by membership assertions of the ABox, then the knowledge base is inconsistent and, therefore, unsatisfiable. Besides, an interaction of positive and negative inclusions may cause inconsistency. So we need to consider all NIs implied by the TBox.

---

[2] The notion of *applicability* of a PI suitably extends the one in [4].

Definition 2. *Let $\mathcal{T}$ be a TBox. We call* NI-closure *of $\mathcal{T}$, denoted by $cln(\mathcal{T})$, the following set of assertions defined inductively:*

1. $\mathcal{T}_{\mathcal{N}} \subseteq cln(\mathcal{T})$.
2. *if* $B_1 \sqcap \cdots \sqcap B_n \sqcap A \sqsubseteq \bot \in cln(\mathcal{T})$, $n \geq 0$ *and* $B_1' \sqcap \cdots \sqcap B_m' \sqsubseteq A$ *is in $\mathcal{T}$, then also* $B_1 \sqcap \cdots \sqcap B_n \sqcap B_1' \sqcap \cdots \sqcap B_m' \sqsubseteq \bot \in cln(\mathcal{T})$.
3. *if* $B_1 \sqcap \cdots \sqcap B_n \sqcap \geq k\,R \sqsubseteq \bot \in cln(\mathcal{T})$, $n \geq 0$ *and* $B_1' \sqcap \cdots \sqcap B_m' \sqsubseteq \geq k'\,R$ *is in $\mathcal{T}$ for $k' \geq k$, then also* $B_1 \sqcap \cdots \sqcap B_n \sqcap B_1' \sqcap \cdots \sqcap B_m' \sqsubseteq \bot \in cln(\mathcal{T})$.
4. *if* $B_1 \sqcap \cdots \sqcap B_n \sqcap \geq 1\,R \sqsubseteq \bot \in cln(\mathcal{T})$, $n \geq 0$ *and* $R' \sqsubseteq^{\pm} R$ *is in $\mathcal{T}$, then also* $B_1 \sqcap \cdots \sqcap B_n \sqcap \geq 1\,R' \sqsubseteq \bot \in cln(\mathcal{T})$.
5. *if* $\mathsf{Dis}(R, R_1)$ *or* $\mathsf{Dis}(R_1, R) \in cln(\mathcal{T})$ *and* $R' \sqsubseteq^{\pm} R$ *is in $\mathcal{T}$, then also* $\mathsf{Dis}(R', R_1) \in cln(\mathcal{T})$.
6. *if either $\geq 1\,R \sqsubseteq \bot$, or $\geq 1\,R^- \sqsubseteq \bot$, or $\mathsf{Dis}(R, R)$ is in $cln(\mathcal{T})$, then all three assertions are in $cln(\mathcal{T})$.*

Note, that in rule 4 it is enough to consider $k = 1$ because the condition (inter) ensures that concepts of the form $\geq k\,R$, for $k \geq 2$, do not occur in the left-hand side of concept inclusions when R appears in the right-hand side of a role inclusion. Also we don't need to add both inclusions $\mathsf{Dis}(R, R_1)$ and $\mathsf{Dis}(R_1, R)$ to $cln(\mathcal{T})$, since to trigger rule 5 one of them is sufficient.

The canonical interpretation can also be exploited for checking satisfiability of a KB containing negative inclusions. To establish that they are satisfied by $can(\mathcal{K})$, it suffices to verify that the interpretation $db(\mathcal{A})$ satisfies $cln(\mathcal{T})$.

Lemma 3. *Let $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ be a DL-Lite$_{horn}^{(\mathcal{HN})}$ KB. Then $can(\mathcal{K})$ is a model of $\mathcal{K}$ if and only if $db(\mathcal{A})$ is a model of $\langle cln(\mathcal{T}), \mathcal{A} \rangle$.*

The proof follows the line of that of Lemma 12 in [4], but we need to take into account the modified definition of chase, and hence of $can(\mathcal{K})$. Now we can show that to check satisfiability of a KB it is is sufficient (and necessary) to look at $db(\mathcal{A})$ (provided we have computed $cln(\mathcal{T})$). More precisely, the next theorem shows that a contradiction on a *DL-Lite$_{horn}^{(\mathcal{HN})}$* KB may hold only if a membership assertion in the ABox contradicts a negative inclusion in the closure $cln(\mathcal{T})$.

Theorem 4. *Let $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ be a DL-Lite$_{horn}^{(\mathcal{HN})}$ KB. Then $\mathcal{K}$ is satisfiable if and only if $db(\mathcal{A})$ is a model of $\langle cln(\mathcal{T}), \mathcal{A} \rangle$.*

Having these results, we can formulate the satisfiability problem in terms of evaluation of a first order query over the database (interpretation). In order to do so we define a translation δ from assertions in $cln(\mathcal{T})$ to FOL formulas encoding their violation:

$$\delta(B_1 \sqcap \cdots \sqcap B_n \sqsubseteq \bot) = \exists x(\gamma_{B_1}(x) \wedge \cdots \wedge \gamma_{B_n}(x))$$
$$\delta(\mathsf{Dis}(R_1, R_2)) = \exists x, y(\rho_{R_1}(x, y) \wedge \rho_{R_2}(x, y))$$

where:

$$\gamma_{B_i}(x) = A(x), \text{ if } B_i = A;$$
$$\gamma_{B_i}(x) = \exists y_1, \ldots, y_k(P(x, y_1) \wedge \cdots \wedge P(x, y_k) \wedge \bigwedge_{j<l} y_j \neq y_l), \text{ if } B_i = \geq k\,P;$$
$$\gamma_{B_i}(x) = \exists y_1, \ldots, y_k(P(y_1, x) \wedge \cdots \wedge P(y_k, x) \wedge \bigwedge_{j<l} y_j \neq y_l), \text{ if } B_i = \geq k\,P^-;$$
$$\rho_{R_i}(x, y) = P(x, y), \text{ if } R_i = P; \qquad \rho_{R_i}(x, y) = P(y, x), \text{ if } R_i = P^-.$$

Algorithm Consistent($\mathcal{K}$)
Input: *DL-Lite$_{horn}^{(\mathcal{HN})}$* KB $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$
Output: *true* if $\mathcal{K}$ is satisfiable, *false* otherwise
$q_{unsat} = \bot$;
for each $I \in cln(\mathcal{T})$ **do** $q_{unsat} = q_{unsat} \vee \delta(I)$;
if $q_{unsat}{}^{db(\mathcal{A})} = \emptyset$ **then return** *true*; **else return** *false*;

Fig. 1. The algorithm Consistent

The algorithm Consistent, depicted in Fig. 1, takes as input a *DL-Lite$_{horn}^{(\mathcal{HN})}$* KB, computes $db(\mathcal{A})$ and $cln(\mathcal{T})$, and evaluates over $db(\mathcal{A})$ the Boolean FOL query obtained by taking the union of all FOL formulas returned by the application of the above function δ to every assertion in $cln(\mathcal{T})$. In the algorithm, the symbol $\bot$ indicates a predicate whose evaluation is false in every interpretation. Therefore, when $\mathcal{K}$ contains no negative inclusions, $q_{unsat}{}^{db(\mathcal{A})} = \bot^{db(\mathcal{A})}$, and Consistent($\mathcal{K}$) returns true.

One can show that the algorithm Consistent terminates and a KB $\mathcal{K}$ is satisfiable if and only if Consistent($\mathcal{K}$) = *true*. As a direct consequence, we get the following theorem, which provides an alternative proof of an analogous result shown in [5].

Theorem 5. *In DL-Lite$_{horn}^{(\mathcal{HN})}$, knowledge base satisfiability is FOL-rewritable.*

Now we can characterize the computational complexity of the algorithm Consistent.

Theorem 6. *The algorithm* Consistent *is* AC^0 *in the size of the ABox and runs in exponential time in the size of the TBox.*

The following example demonstrates the worst case behaviour of the algorithm, where the size of $cln(\mathcal{T})$ is exponential in the size of $\mathcal{T}$.

Example 7. Let us consider the TBox $\mathcal{T}$ consisting of the assertions: $A'_1 \sqsubseteq A_1, \ldots,$ $A'_n \sqsubseteq A_n$, $A_1 \sqcap \cdots \sqcap A_n \sqsubseteq \bot$. Then $cln(\mathcal{T})$ contains 2^n negative inclusion assertions of the form $B_1 \sqcap \cdots \sqcap B_n \sqsubseteq \bot$, where each B_i is either A_i or A'_i. Moreover, none of the assertions can be omitted: for every negative inclusion $I = B_1 \sqcap \cdots \sqcap B_n \sqsubseteq \bot$ there is an ABox $\mathcal{A}$ such that $\delta(I)^{db(\mathcal{A})} = \emptyset$. It is enough to take $\mathcal{A} = \{B_1(a), \ldots, B_n(a)\}$ with a a fresh object name. Therefore, in order to ensure that the algorithm detects unsatisfiability of $\langle \mathcal{T}, \mathcal{A} \rangle$, all the possible negative inclusions must be present in $cln(\mathcal{T})$ and its size is exponential in the size of $\mathcal{T}$.

Note that the algorithm Consistent can be turned into a non-deterministic PTIME algorithm for checking unsatisfiability of *DL-Lite$_{horn}^{(\mathcal{HN})}$* KBs.

4 Query Answering

The aim of this section is to devise an algorithm for answering unions of CQs in *DL-Lite$_{horn}^{(\mathcal{HN})}$* that is based on query reformulation, and hence can be easily implemented: in the rewriting step the assertions of the TBox are compiled into the query, then the resulting query is evaluated over the ABox without considering the TBox.

Let $Q = \bigcup_i q_i$ be a UCQ. We say that an argument of an atom in a query is *bound* if it corresponds either to a constant, or to a distinguished variable, or to a shared variable, that is, a variable occurring at least twice in the query body. Instead, an argument of an atom in a query is *unbound* if it corresponds to a nondistinguished nonshared variable. The symbol '_' is used to represent unbound variables.

Let $Q_{\mathcal{T}}^R$ denote the set of natural numbers containing 1 and all the numerical parameters k for which the concept $\geq k\, R$ occurs in $\mathcal{T}$. The *extension* $ext(\mathcal{T})$ of $\mathcal{T}$ contains

- $\geq k'\, R \sqsubseteq\ \geq k\, R$, for all $k, k' \in Q_{\mathcal{T}}^R$ such that $k' > k$ and $k' > k'' > k$ for no $k'' \in Q_{\mathcal{T}}^R$ and R is either a direct or inverse role in $\mathcal{T}$, and
- $\geq k\, R \sqsubseteq\ \geq k\, R'$, for all $k \in Q_{\mathcal{T}}^R$ and $R \sqsubseteq^{\pm} R' \in \mathcal{T}$.

A positive concept inclusion I is *applicable to an atom* $B(x)$ if I has B on the right-hand side, where $B(x) = A(x)$ if $B = A$, $B(x) = E_k P(x)$ if $B =\ \geq k\, P$, and $B(x) = E_k P^-(x)$ if $B =\ \geq k\, P^-$. A positive role inclusion I is *applicable to an atom* $P(x_1, x_2)$ if I has either P or P^- on the right-hand side. We indicate with $gr(g, I)$ the atom obtained from the atom g by applying the applicable inclusion I. Formally:

Definition 8. *Let $I \in ext(\mathcal{T})$ be a positive inclusion assertion that is applicable to the atom g. Then, $gr(g, I)$ is the formula defined as follows:*

1. *if $g = A(x)$ and $I = B_1 \sqcap \cdots \sqcap B_n \sqsubseteq A$, then $gr(g, I) = B_1(x) \wedge \cdots \wedge B_n(x)$,*
2. *if $g = E_k R(x)$ and $I = B_1 \sqcap \cdots \sqcap B_n \sqsubseteq\ \geq k'\, R$, $k' \geq k$, then $gr(g, I) = B_1(x) \wedge \cdots \wedge B_n(x)$,*
3. *if $g = E_1 R(\_)$ and $I = B_1 \sqcap \cdots \sqcap B_n \sqsubseteq\ \geq k\, R$, then $gr(g, I) = B_1(x) \wedge \cdots \wedge B_n(x)$, for a fresh variable x,*
4. *if $g = E_k R(x)$ and $I = P_1 \sqsubseteq^{\pm} R$, then $gr(g, I) = E_k P_1(x)$,*
5. *if $g = E_k R(x)$ and $I = P_1 \sqsubseteq^{\pm} R^-$, then $gr(g, I) = E_k P_1^-(x)$,*
6. *if $g = P(x_1, x_2)$ and $I = P_1 \sqsubseteq^{\pm} P$, then $gr(g, I) = P_1(x_1, x_2)$,*
7. *if $g = P(x_1, x_2)$ and $I = P_1 \sqsubseteq^{\pm} P^-$, then $gr(g, I) = P_1(x_2, x_1)$.*

We also define the *most general unifier*, mgu, between two atoms g_1, g_2 of a query q that unify. In the case where g_1 and g_2 are respectively of the form $A(x)$ and $A(z)$, or $P(x, y)$ and $P(z, w)$, the mgu is defined as usual (see [4]), taking also into account the possible presence of inequalities $x \neq z$ (and $y \neq w$). However, we allow also an atom $P(x, y)$ to unify with $E_1 P(z)$ and $E_1 P^-(w)$. Moreover, when one of the atoms is of the form $E_k R(x)$, then the mgu is defined as follows: let $g_1 = E_{k_1} R(x)$ and $g_2 = E_{k_2} R(z)$, $k = max(k_1, k_2)$, and $x \neq z$ does not occur in q; or $g_1 = E_k R(x)$ and $g_2 = E_1 R^-(\_)$, then $mgu = E_k R(x)$.

In Fig. 2 we provide the algorithm PerfectRef, which reformulates a UCQ taking into account the PIs of a TBox $\mathcal{T}$. In the algorithm, $q[g/g']$ denotes the CQ obtained from a CQ q by replacing the atom g with a new atom g'. The function $remdup$ removes from the body of a CQ duplicated atoms, or if a query contains two atoms of the form $E_{k_1} R(x)$ and $E_{k_2} R(x)$, then it removes the atom with the smaller k_i. Furthermore, τ is a function that takes as input a CQ q and returns a new CQ obtained by replacing each occurrence of an unbound variable in q with the symbol _, whereby $P(x, \_)$ becomes $E_1 P(x)$, $P(\_, x)$ becomes $E_1 P^-(x)$ (and $P(\_, \_)$ becomes $E_1 P(\_)$). Finally, *reduce* is a function that takes as input a CQ q and two atoms g_1 and g_2 that unify and occur

Algorithm PerfectRef($Q, \mathcal{T}$)
Input: union of conjunctive queries Q, *DL-Lite*$_{horn}^{(\mathcal{HN})}$ TBox $\mathcal{T}$
Output: union of conjunctive queries PR
$PR := \{remdup(\tau(q)) \mid q \text{ is a CQ in } Q\}$;
repeat   (1)
    $PR' := PR$;
    for each $q \in PR'$
    (a) **for each** g in q
            for each PI I in $ext(\mathcal{T})$
                if I is applicable to g
                then $PR := PR \cup \{remdup(q[g/gr(g, I)])\}$;
    (b) **for each** g_1, g_2 in q
            if g_1 and g_2 unify
            then $PR := PR \cup \{remdup(\tau(reduce(q, g_1, g_2)))\}$;
until $PR' = PR$;
for each $q \in PR$ (2)
    for each g in q
        if g is of the form $E_k R(x)$, $k \geq 2$, **then** replace g with $\gamma_{\geq k\ R}(x)$;
        if g is of the form $E_1 R(x)$ **then** replace g with $R(x, \_)$;
return PR.

Fig. 2. The algorithm PerfectRef

in the body of q, and returns a CQ q' obtained by applying to q the most general unifier between g_1 and g_2.

Informally, part (1) of the algorithm reformulates the query by replacing and unifying atoms, and accumulates the new queries. In this part all possible applications of the PIs, according to Definition 8, are exhausted. Part (2) performs unfolding of the atoms $E_k R(x)$. Notice that it is sufficient to perform unfolding of the atoms $E_k R(x)$ in the very end for the following reasons: first, if $k \geq 2$, then only role inclusions could be applied to the atoms of the form $R(x, y)$; however condition (inter) ensures that in this case there are no such inclusions. If $k = 1$, then such role inclusions have already been aplied in step (1). Second, one could reduce some of the atoms $R(x, y)$, but it would not produce new answers, since all variables in the new R-atoms are bound.

This algorithm is an extension of the PerfectRef algorithm devised for the logic *DL-Lite$_{\mathcal{A}}$* [4]. The extended version has to deal with inequality atoms and with atoms of the form $E_k R(x)$ that later need to be unfolded. In contrast with the algorithm for *DL-Lite$_{\mathcal{A}}$* the query may grow, so one also needs to take care of redundant atoms. Notice also that the number of CQs in PR may be exponential in the size of the TBox (and not only in the length of q), due to the fact that horn inclusions may cause a single CQ to become of length linear in the size of the TBox. Here, we assume that numbers in the TBox are coded in unary.

Now, to compute the answers to Q over the KB $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$, we need to evaluate the set of conjunctive queries PR produced by the algorithm PerfectRef over the ABox $\mathcal{A}$ considered as a relational database. The algorithm computing $ans(Q, \mathcal{K})$ is exactly the same as in [4], so we do not present it here. Correctness of the above described query-answering technique is established in the following. We start by observing that,

as in [4], query answering can in principle be done by evaluating the query over the model $can(\mathcal{K})$.

Theorem 9. *Let $\mathcal{K}$ be a satisfiable DL-Lite$_{horn}^{(\mathcal{HN})}$ KB, and let Q be a union of conjunctive queries over $\mathcal{K}$. Then, $ans(Q, \mathcal{K}) = Q^{can(\mathcal{K})}$.*

Since $can(\mathcal{K})$ is in general infinite, we cannot compute it and evaluate Q over it. Instead, we compile the TBox into the query, thus simulating the evaluation of the query over $can(\mathcal{K})$ by evaluating a finite reformulation of the query over the ABox considered as a database. The proof of the following lemma is inspired by the one for *DL-Lite$_{\mathcal{R}}$* in [4], but needs to take into account number restrictions and horn inclusion assertions.

Lemma 10. *Let $\mathcal{T}$ be a DL-Lite$_{horn}^{(\mathcal{HN})}$ TBox, Q a UCQ over $\mathcal{T}$, and PR the UCQ returned by $\mathsf{PerfectRef}(Q, \mathcal{T})$. For every DL-Lite$_{horn}^{(\mathcal{HN})}$ ABox $\mathcal{A}$ such that $\langle \mathcal{T}, \mathcal{A} \rangle$ is satisfiable, $ans(Q, \langle \mathcal{T}, \mathcal{A} \rangle) = PR^{db(\mathcal{A})}$.*

Proof. We first introduce the preliminary notion of *witness of a tuple of constants* with respect to a CQ q in Q. Given a *DL-Lite$_{horn}^{(\mathcal{HN})}$* knowledge base $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$, a CQ $q(\boldsymbol{x}) \leftarrow conj(\boldsymbol{x}, \boldsymbol{y})$ over $\mathcal{K}$, and a tuple $\boldsymbol{t}$ of constants occurring in $\mathcal{K}$, a set of membership assertions $\mathcal{G}$ is a *witness* of $\boldsymbol{t}$ w.r.t. q if there exists a substitution σ from the variables $\boldsymbol{y}$ in $conj(\boldsymbol{t}, \boldsymbol{y})$ to constants in $\mathcal{G}$ such that the set of atoms in $\sigma(conj(\boldsymbol{t}, \boldsymbol{y}))$ is equal to $\mathcal{G}$. Then $\boldsymbol{t} \in q^{can(\mathcal{K})}$ iff there exists a witness $\mathcal{G}$ of $\boldsymbol{t}$ w.r.t. q such that $\mathcal{G} \subseteq chase(\mathcal{K})$. The cardinality of a witness $\mathcal{G}$, denoted by $|\mathcal{G}|$, is the number of membership assertions in $\mathcal{G}$.

Let us prove first the statement for a modified version of PerfectRef, where at the end of the algorithm we perform an additional step of unifications as in step (b) of part (1). Let us call PerfectRef$_u$ this extended version of PerfectRef.

We have that $ans(Q, \langle \mathcal{T}, \mathcal{A} \rangle) = Q^{can(\mathcal{K})} = \bigcup_{q \in Q} q^{can(\mathcal{K})}$, and $PR^{db(\mathcal{A})} = \bigcup_{\hat{q} \in PR} \hat{q}^{db(\mathcal{A})}$, where PR is the UCQ returned by PerfectRef$_u(Q, \mathcal{T})$. Hence, we need to show that $\bigcup_{\hat{q} \in PR} \hat{q}^{db(\mathcal{A})} = Q^{can(\mathcal{K})}$. For simplicity, we consider the case where Q consists of a single CQ q.

"$\Leftarrow$" We have to prove that $\hat{q}^{db(\mathcal{A})} \subseteq q^{can(\mathcal{K})}$, for each $\hat{q} \in PR$. Let q_{i+1} be obtained from q_i by some step of the algorithm PerfectRef$_u$. We can show that $q_{i+1}^{can(\mathcal{K})} \subseteq q_i^{can(\mathcal{K})}$ at any step. Since each query of PR is either q or a query obtained from q by repeatedly applying steps (a) and (b) of the algorithm PerfectRef$_u$, then by the rewriting in part (2), and the unification step in the end, it follows that for each $\hat{q} \in PR$, $\hat{q}^{can(\mathcal{K})} \subseteq q^{can(\mathcal{K})}$, by repeatedly applying the property $q_{i+1}^{can(\mathcal{K})} \subseteq q_i^{can(\mathcal{K})}$. Since $db(\mathcal{A}) \subseteq can(\mathcal{K})$ and CQs are monotonic queries, we get $\hat{q}^{db(\mathcal{A})} \subseteq \hat{q}^{can(\mathcal{K})} \subseteq q^{can(\mathcal{K})}$ for each CQ $\hat{q} \in PR$.

"$\Rightarrow$" We have to show that for each tuple $\boldsymbol{t} \in q^{can(\mathcal{K})}$, there exists $\hat{q} \in PR$ such that $\boldsymbol{t} \in \hat{q}^{db(\mathcal{A})}$. First, since $\boldsymbol{t} \in q^{can(\mathcal{K})}$, it follows that there exists a finite number h such that there is a witness $\mathcal{G}_h$ of $\boldsymbol{t}$ w.r.t. q contained in $chase_h(\mathcal{K})$. Moreover, w.l.o.g. we can assume that every rule cr1, cr2 and cr3 used in the construction of $chase(\mathcal{K})$ is necessary in order to generate such a witness $\mathcal{G}_h$. In the following, we say that a set S of membership assertions is an *ancestor* of a set S' of membership assertions in a set $\mathcal{S}$ of

membership assertions, if there exist $S_1, \dots, S_n$ in $\mathcal{S}$, where $S_1 = S$ and $S_n = S'$, such that, for each $j \in \{2, \dots, n\}$, S_j can be generated by applying a chase rule to a subset of S_{j-1}. We also say that S' is a successor of S. Furthermore, for each $i \in \{0, \dots, h\}$, we denote with $\mathcal{G}_i$ the *pre-witness* of $\boldsymbol{t}$ w.r.t. q in $chase_h(\mathcal{K})$, defined as follows:

$$\mathcal{G}_i = \bigcup_{S' \subseteq \mathcal{G}_h} \{\, S \subseteq chase_i(\mathcal{K}) \mid S \text{ is an ancestor of } S' \text{ in } chase_h(\mathcal{K}) \text{ and} \\ \text{there exists no successor of } S \text{ in } chase_i(\mathcal{K}) \\ \text{that is an ancestor of } S' \text{ in } chase_h(\mathcal{K}) \,\}.$$

Now we prove by induction on i that, starting from $\mathcal{G}_h$, we can "go back" through the rule applications and find a query $\hat{q}$ in PR such that the pre-witness $\mathcal{G}_{h-i}$ of $\boldsymbol{t}$ w.r.t. q in $chase_{h-i}(\mathcal{K})$ is also a witness of $\boldsymbol{t}$ w.r.t. $\hat{q}$. To this aim, we prove that there exists $\hat{q} \in PR$ such that $\mathcal{G}_{h-i}$ is a witness of $\boldsymbol{t}$ w.r.t. $\hat{q}$ and $|\hat{q}| = |\mathcal{G}_{h-i}|$, where $|\hat{q}|$ indicates the number of atoms in the CQ $\hat{q}$ except inequality atoms. The claim then follows for $i = h$, since $chase_0(\mathcal{K}) = \mathcal{A}$.

Base step: There exists $\hat{q} \in PR$ such that $\mathcal{G}_h$ is a witness of $\boldsymbol{t}$ w.r.t. $\hat{q}$ and $|\hat{q}| = |\mathcal{G}_h|$. This is an immediate consequence of the fact that (1) $q \in PR$ and (2) PR is closed with respect to step (b) of the algorithm PerfectRef$_u$.

Inductive step: Suppose there exists $\hat{q} \in PR$ such that $\mathcal{G}_{h-i+1}$ is a witness of $\boldsymbol{t}$ w.r.t. $\hat{q}$ in $chase_{h-i+1}(\mathcal{K})$ and $|\hat{q}| = |\mathcal{G}_{h-i+1}|$. Let us assume that $chase_{h-i+1}(\mathcal{K})$ is obtained by applying chase rule cr2 to $chase_{h-i}(\mathcal{K})$ (the proof for rules cr1 and cr3 is analogous). Hence, a PI of the form $B_1 \sqcap \dots \sqcap B_n \sqsubseteq\, \geq k\, R$, where B_j, $1 \leq j \leq n$, are basic concepts and R is a basic role, is applied in $chase_{h-i}(\mathcal{K})$ to a set of membership assertions $S' = S_{B_1}(a) \cup \dots \cup S_{B_n}(a)$, such that $k_1 = \sharp\{b \mid R(a,b) \in chase_{h-i}(\mathcal{K})\} < k$. Therefore, $chase_{h-i+1}(\mathcal{K}) = chase_{h-i}(\mathcal{K}) \cup \{R(a, b_{k_1+1}), \dots, R(a, b_k)\}$, where $b_{k_1+1}, \dots, b_k \in \Gamma_N$ follow lexicographically all constants occurring in $chase_{h-i}(\mathcal{K})$. Since every rule used in the construction of $chase(\mathcal{K})$ is necessary for generating $\mathcal{G}_{h-i+1}$ and the set of membership assertions $\{R(a, b_{k_1+1}), \dots, R(a, b_k)\}$ does not have successors in $chase_{h-i+1}(\mathcal{K})$, so $\{R(a, b_{k_1+1}), \dots, R(a, b_k)\} \subseteq \mathcal{G}_{h-i+1}$.

Let $\{R(a, b_1), \dots, R(a, b_{k_2})\}$, $0 \leq k_2 \leq k_1$, be the minimal set of membership assertions that has successors in $chase_{h-i+1}(\mathcal{K})$. Then, according to the definition of $\mathcal{G}_i$, the set $\{R(a, b_{k_2+1}), \dots, R(a, b_{k_1})\} \subseteq \mathcal{G}_{h-i+1}$. By the inductive assumption, $|\hat{q}| = |\mathcal{G}_{h-i+1}|$, and $\{R(a, b_{k_2+1}), \dots, R(a, b_k)\} \subseteq \mathcal{G}_{h-i+1}$, hence, $\hat{q}$ has to contain the atoms $R(x, y_{k_2+1}), \dots, R(x, y_k)$, where $y_{k_2+1}, \dots, y_k$ appear only in the mentioned predicate atoms and possibly inequalities. We show that in $\hat{q}$ there must be all the possible inequalities $y_{j_1} \neq y_{j_2}$, for $k_2 + 1 \leq j_1 < j_2 \leq k$.

By contradiction, assume that not all variables are related to each other by inequalities, i.e. there are $m \geq 2$ sets of variables $y_{k_2+1}, \dots, y_k$ such that the variables within one set are mutually unequal and variables from different sets are not constrained to be different. It means that the part of $\hat{q}$ with $y_{k_2+1}, \dots, y_k$ looks as follows:

$$R(x, y_{11}), \dots, R(x, y_{1l_1}), y_{11} \neq y_{12}, \dots, y_{1l_1-1} \neq y_{1l_1},$$
$$R(x, y_{21}), \dots, R(x, y_{2l_2}), y_{21} \neq y_{22}, \dots, y_{2l_2-1} \neq y_{2l_2},$$
$$\dots$$
$$R(x, y_{m1}), \dots, R(x, y_{ml_m}), y_{m1} \neq y_{m2}, \dots, y_{ml_m-1} \neq y_{ml_m},$$

where $l_j \geq 1$, $j \in \{1, \dots, m\}$, $\sum l_j = k - k_2$. Therefore, $\hat{q}$ has to be the result of unfolding by part (2) of the algorithm PerfectRef$_u$ of a query q_1 with the corresponding

part $E_{l_1} R(x), \ldots, E_{l_m} R(x)$. However, the algorithm cannot produce such a query: the function *remdup* would keep in q_1 only one such atom $E_{l_{max}} R(x)$ with $l_{max} < k - k_2$, $l_{max} = \max_j \{l_j\}$, which unfolded would contain less than $k - k_2$ atoms of the form $R(x, y_j)$. The contradiction rises from the assumption $m \geq 2$. So, m has to be equal to 1 and all variables $y_{k_2+1}, \ldots, y_k$ appear in pairwise inequalities in $\hat{q}$.

Thus, there exists a query q_1 in PR_1 that contains the atom $E_{k-k_2} R(x), k - k_2 \leq k$, and $\hat{q}$ is obtained from q_1 by part (2) of the algorithm. Then, by step (a) it follows that there exists a query $q_2 = remdup(\hat{q}_1[E_{k-k_2} R(x)/B_1(x) \wedge \cdots \wedge B_n(x)])$ in PR_1 such that $\mathcal{G}_{h-i}$ is a witness of t w.r.t. q_2. Let $\hat{q}_1$ be the result of unfolding q_2 by part (2) of the algorithm, then $\mathcal{G}_{h-i}$ is a witness of t w.r.t. $\hat{q}_1$ as well and $|\hat{q}_1| \geq |\mathcal{G}_{h-i}|$.

If $|\hat{q}_1| > |\mathcal{G}_{h-i}|$ it implies that there exists at least one membership assertion f in $\mathcal{G}_{h-i}$ such that there exist at least two atoms g_1, g_2 in $\hat{q}_1$ that both unify with f. Hence g_1 and g_2 unify, and by step (b) of the algorithm it follows that in PR there exists $q_3 = remdup(\tau(reduce(\hat{q}_1, g_1, g_2)))$, $|q_3| < |\hat{q}_1|$ and $\mathcal{G}_{h-i}$ is a witness of t w.r.t. q_3. If $|q_3| > |\mathcal{G}_{h-i}|$ then by applying the argument consecutively there is a query $\hat{q}_2$ such that $|\hat{q}_2| = |\mathcal{G}_{h-i}|$ and $\mathcal{G}_{h-i}$ is a witness of t w.r.t. $\hat{q}_2$, which proves the claim.

Finally, we observe that unification of atoms of the query at the end of PerfectRef$_u$ does not add new answers to the set $ans(Q, \langle \mathcal{T}, \mathcal{A} \rangle)$ since all variables in the new R atoms introduced in part (2) are bound. Therefore, the theorem holds also for the algorithm PerfectRef.   □

Based on the above property, we are finally able to establish correctness of the algorithm Answer and its computational complexity.

Theorem 11. *Let $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ be a DL-Lite$_{horn}^{(\mathcal{HN})}$ KB, Q a UCQ over $\mathcal{T}$, and t a tuple of constants in $\mathcal{K}$. Then, $t \in ans(Q, \mathcal{K})$ if and only if $t \in$ Answer$(Q, \mathcal{K})$.*

Theorem 12. *The algorithm Answer is exponential in the size of the TBox, and AC^0 in the size of the ABox (data complexity).*

Note that we can modify PerfectRef to get an algorithm for the decision problem associated with the query answering problem that runs in nondeterministic polynomial time in combined complexity: it nondeterministically returns one of the CQs from the reformulation of the input query (polynomially many rewriting steps) and in polynomial time checks whether a tuple is in the answer to this CQ. The corresponding version of Answer also runs in nondeterministic polynomial time.

5   Conclusions

This paper presents practical algorithms for query answering and knowledge base satisfiability in *DL-Lite*$_{horn}^{(\mathcal{HN})}$. They are based on the same idea as those devised for the original *DL-Lite* logics. The distinguishing feature of these algorithms is a separation between TBox and ABox reasoning, which enables an interesting modularization of query answering: the part of the process requiring TBox reasoning is independent of the ABox, and the part of the process requiring access to the ABox can be carried out by an SQL engine. We showed that the algorithms are sound and complete.

The complexity of the developed algorithms is AC^0 w.r.t. data complexity, NP in the size of the TBox and NP w.r.t. combined complexity.

References

1. Poggi, A., Lembo, D., Calvanese, D., De Giacomo, G., Lenzerini, M., Rosati, R.: Linking data to ontologies. J. on Data Semantics **X** (2008) 133–173
2. Rodríguez-Muro, M.: Tools and Techniques for Ontology Based Data Access in Lightweight Description Logics. PhD thesis, KRDB Research Centre for Knowledge and Data, Free University of Bozen-Bolzano (2010)
3. Lenzerini, M.: Data integration: A theoretical perspective. In: Proc. of the 21st ACM SIGACT SIGMOD SIGART Symp. on Principles of Database Systems (PODS 2002). (2002) 233–246
4. Calvanese, D., De Giacomo, G., Lembo, D., Lenzerini, M., Rosati, R.: Tractable reasoning and efficient query answering in description logics: The *DL-Lite* family. J. of Automated Reasoning **39**(3) (2007) 385–429
5. Artale, A., Calvanese, D., Kontchakov, R., Zakharyaschev, M.: The *DL-Lite* family and relations. J. of Artificial Intelligence Research **36** (2009) 1–69
6. Acciarri, A., Calvanese, D., De Giacomo, G., Lembo, D., Lenzerini, M., Palmieri, M., Rosati, R.: QuOnto: QUerying ONTOlogies. In: Proc. of the 20th Nat. Conf. on Artificial Intelligence (AAAI 2005). (2005) 1670–1671
7. Poggi, A., Rodríguez-Muro, M., Ruzzi, M.: Ontology-based database access with DIG-Mastro and the OBDA Plugin for Protégé. In Clark, K., Patel-Schneider, P.F., eds.: Proc. of the 4th Int. Workshop on OWL: Experiences and Directions (OWLED 2008 DC). (2008)
8. Baader, F., Calvanese, D., McGuinness, D., Nardi, D., Patel-Schneider, P.F., eds.: The Description Logic Handbook: Theory, Implementation and Applications. Cambridge University Press (2003)
9. Abiteboul, S., Hull, R., Vianu, V.: Foundations of Databases. Addison Wesley Publ. Co. (1995)

The Complexity of Satisfiability for
Sub-Boolean Fragments of $\mathcal{ALC}$

Arne Meier[1] and Thomas Schneider[2]

[1] Leibniz Universität Hannover, Germany, `meier@thi.uni-hannover.de`
[2] University of Manchester, UK, `schneider@cs.man.ac.uk`

Abstract. The standard reasoning problem, concept satisfiability, in the basic description logic $\mathcal{ALC}$ is PSPACE-complete, and it is EXPTIME-complete in the presence of unrestricted axioms. Several fragments of $\mathcal{ALC}$, notably logics in the $\mathcal{FL}$, $\mathcal{EL}$, and DL-Lite families, have an easier satisfiability problem; sometimes it is even tractable. All these fragments restrict the use of Boolean operators in one way or another. We look at systematic and more general restrictions of the Boolean operators and establish the complexity of the concept satisfiability problem in the presence of axioms. We separate tractable from intractable cases.

1 Introduction

Standard reasoning problems of description logics, such as satisfiability or subsumption, have been studied extensively. Depending on the expressivity of the logic and the reasoning problem, the complexity of reasoning for DLs ranging from logics below the basic description logic $\mathcal{ALC}$ to the OWL DL standard $\mathcal{SROIQ}$ is between tractable and NEXPTIME.

For $\mathcal{ALC}$, concept satisfiability is PSPACE-complete [27] and, in the presence of unrestricted axioms, it is EXPTIME-complete due to the correspondence with propositional dynamic logic [25, 29, 14]. Since the standard reasoning tasks are interreducible in the presence of all Boolean operators, subsumption has the same complexity.

Several fragments of $\mathcal{ALC}$, such as logics in the $\mathcal{FL}$, $\mathcal{EL}$ or DL-Lite families, are well-understood. They often restrict the use of Boolean operators, and it is known that their reasoning problems are often easier than for $\mathcal{ALC}$. For instance, concept subsumption with respect to acyclic and cyclic TBoxes, and even with GCIs is tractable in the logic $\mathcal{EL}$, which allows only conjunctions and existential restrictions, [4, 9], and it remains tractable under a variety of extensions such as nominals, concrete domains, role chain inclusions, and domain and range restrictions [5, 6]. However, the presence of universal quantifiers breaks tractability: Subsumption in $\mathcal{FL}_0$, which allows only conjunction and universal restrictions, is coNP-complete [22] and increases to PSPACE-complete with respect to cyclic TBoxes [3, 18] and to EXPTIME-complete with GCIs [5, 17]. In [12, 13], concept satisfiability and subsumption for several logics below and above $\mathcal{ALC}$ that extend $\mathcal{FL}_0$ with disjunction, negation and existential restrictions and

other features, is shown to be tractable, NP-complete, coNP-complete or PSPACE-complete. Subsumption in the presence of general axioms is EXPTIME-complete in logics containing both existential and universal restrictions plus conjunction or disjunction [15], as well as in $\mathcal{AL}$, where only conjunction, universal restrictions and unqualified existential restrictions are allowed [11]. In DL-Lite, where atomic negation, unqualified existential and universal restrictions, conjunctions and inverse roles are allowed, satisfiability of ontologies is tractable [10]. Several extensions of DL-Lite are shown to have tractable and NP-complete satisfiability problems in [1, 2].

This paper revisits restrictions to the Boolean operators in $\mathcal{ALC}$. Instead of looking at one particular subset of $\{\wedge, \vee, \neg\}$, we are considering all possible sets of Boolean operators, including less commonly used operators such as the binary exclusive or $\oplus$. Our aim is to find for *every* possible combination of Boolean operators whether it makes satisfiability of the corresponding restriction of $\mathcal{ALC}$ hard or easy. Since each Boolean operator corresponds to a Boolean function— *i.e.*, an n-ary function whose arguments and values are in $\{\bot, \top\}$—there are infinitely many sets of Boolean operators determining fragments of $\mathcal{ALC}$. The complexity of the corresponding concept satisfiability problems without theories, which are equivalent to the satisfiability problems for the corresponding fragments of multimodal logic, has already been classified in [16]: it is PSPACE-complete if at least the ternary operator $x \wedge (y \vee z)$ and the constant $\bot$ are allowed, coNP-complete if at least conjunctions and at most conjunctions plus the constant $\bot$ are allowed, and trivial otherwise, *i.e.*, for all other sets of Boolean operators, every modal formula (concept description) is satisfiable. We will put this classification into the context of the above listed results for $\mathcal{ALC}$ fragments.

The tool used in [16] for classifying the infinitely many satisfiability problems was Post's lattice [24], which consists of all sets of Boolean functions closed under superposition. These sets directly correspond to all sets of Boolean operators closed under nesting. Similar classifications have been achieved for satisfiability for classical propositional logic [19], Linear Temporal Logic [7], hybrid logic [20], and for constraint satisfaction problems [26, 28].

In this paper, we classify the concept satisfiability problems with respect to theories for $\mathcal{ALC}$ fragments obtained by arbitrary sets of Boolean operators. We will separate tractable and intractable cases, showing that these problems are

- EXPTIME-hard whenever we allow at least conjunction, disjunction or all self-dual operators, where a Boolean function is called self-dual if negating all its arguments negates its value,
- PSPACE-hard whenever we allow at least negation or both constants $\bot, \top$,
- coNP-hard whenever we allow at least the constant $\bot$,
- trivial, which means that all instances are satisfiable, in all other cases.

We will also put these results into the context of the above listed results for $\mathcal{ALC}$ fragments. This is work in progress which we plan to extend by corresponding upper bounds, restricted use of $\exists, \forall$, and terminological restrictions to TBoxes such as acyclicity and atomic left-hand sides of axioms. Furthermore, not all results carry over straightforwardly to other reasoning problems because some of

the standard reductions use Boolean operators that are not available in every fragment.

2 Preliminaries

Description Logic. We use the standard syntax and semantics of $\mathcal{ALC}$ with the Boolean operators $\sqcap$, $\sqcup$, $\neg$, $\top$, $\bot$ replaced by arbitrary operators o that correspond to Boolean functions f_o of arbitrary arity. Let $\mathsf{N_C}$, $\mathsf{N_R}$ and $\mathsf{N_I}$ be sets of atomic concepts, roles and individuals. Then the set of *concept descriptions*, for short *concepts*, is defined by

$$C := A \mid o(C, \ldots, C) \mid \exists R.C \mid \forall R.C,$$

where $A \in \mathsf{N_C}$, $R \in \mathsf{N_R}$, and o is a Boolean operator. A *general concept inclusion (GCI)* is an axiom of the form $C \sqsubseteq D$ where C, D are concepts. We use "$C \equiv D$" as the usual syntactic sugar for "$C \sqsubseteq D$ and $D \sqsubseteq C$". A *TBox* is a finite set of GCIs without restrictions. An *ABox* is a finite set of axioms of the form $C(x)$ or $R(x, y)$, where C is a concept, $R \in \mathsf{N_R}$ and $x, y \in \mathsf{N_I}$. An *ontology* is the union of a TBox and an ABox. This simplified view suffices for our purposes.

An *interpretation* is a pair $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$, where $\Delta^{\mathcal{I}}$ is a nonempty set and $\cdot^{\mathcal{I}}$ is a mapping from $\mathsf{N_C}$ to $\mathfrak{P}(\Delta^{\mathcal{I}})$, from $\mathsf{N_R}$ to $\mathfrak{P}(\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}})$ and from $\mathsf{N_I}$ to $\Delta^{\mathcal{I}}$ that is extended to arbitrary concepts as follows:

$$o(C_1, \ldots, C_n)^{\mathcal{I}} = \{x \in \Delta^{\mathcal{I}} \mid f_o(\|x \in C_1^{\mathcal{I}}\|, \ldots, \|x \in C_n^{\mathcal{I}}\|) = \top\},$$
$$\text{where } \|x \in C_1^{\mathcal{I}}\| = \top \text{ if } x \in C_1^{\mathcal{I}} \text{ and } \|x \in C_1^{\mathcal{I}}\| = \bot \text{ if } x \notin C_1^{\mathcal{I}},$$
$$\exists R.C^{\mathcal{I}} = \{x \in \Delta^{\mathcal{I}} \mid \{y \in C^{\mathcal{I}} \mid (x, y) \in R^{\mathcal{I}}\} \neq \emptyset\},$$
$$\forall R.C^{\mathcal{I}} = \{x \in \Delta^{\mathcal{I}} \mid \{y \in C^{\mathcal{I}} \mid (x, y) \notin R^{\mathcal{I}}\} = \emptyset\}.$$

An interpretation $\mathcal{I}$ *satisfies* the axiom $C \sqsubseteq D$, written $\mathcal{I} \models C \sqsubseteq D$, if $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$. Furthermore, $\mathcal{I}$ satisfies $C(x)$ or $R(x, y)$ if $x^{\mathcal{I}} \in C^{\mathcal{I}}$ or $(x^{\mathcal{I}}, y^{\mathcal{I}}) \in R^{\mathcal{I}}$. An interpretation $\mathcal{I}$ satisfies a TBox (ABox, ontology) if it satisfies every axiom therein. It is then called a *model* of this set of axioms.

Let B be a finite set of Boolean operators and use $\mathsf{Con}(B)$ and $\mathsf{Ax}(B)$ to denote the set of all concepts and axioms using only operators in B. The following decision problems are of interest for this paper.

Concept satisfiability $\mathrm{CSAT}(B)$:
Given a concept $C \in \mathsf{Con}(B)$, is there an interpretation $\mathcal{I}$ s.t. $C^{\mathcal{I}} \neq \emptyset$?

TBox satisfiability $\mathrm{TSAT}(B)$:
Given a TBox $\mathcal{T} \subseteq \mathsf{Ax}(B)$, is there an interpretation $\mathcal{I}$ s.t. $\mathcal{I} \models \mathcal{T}$?

TBox-concept satisfiability $\mathrm{TCSAT}(B)$:
Given $\mathcal{T} \subseteq \mathsf{Ax}(B)$ and $C \in \mathsf{Con}(B)$, is there an $\mathcal{I}$ s.t. $\mathcal{I} \models \mathcal{T}$ and $C^{\mathcal{I}} \neq \emptyset$?

Ontology satisfiability $\mathrm{OSAT}(B)$:
Given an ontology $\mathcal{O} \subseteq \mathsf{Ax}(B)$, is there an interpretation $\mathcal{I}$ s.t. $\mathcal{I} \models \mathcal{O}$?

Ontology-concept satisfiability $\mathrm{OCSAT}(B)$:
   Given $\mathcal{O} \subseteq \mathsf{Ax}(B)$ and $C \in \mathsf{Con}(B)$, is there an $\mathcal{I}$ s.t. $\mathcal{I} \models \mathcal{O}$ and $C^{\mathcal{I}} \neq \emptyset$?

These problems are interreducible independently of B in the following way:

$$\mathrm{CSAT}(B) \leq_{\mathrm{m}}^{\log} \mathrm{OSAT}(B)$$

$$\mathrm{TSAT}(B) \leq_{\mathrm{m}}^{\log} \mathrm{TCSAT}(B) \leq_{\mathrm{m}}^{\log} \mathrm{OSAT}(B) \equiv_{\mathrm{m}}^{\log} \mathrm{OCSAT}(B)$$

The reasons are: a concept C is satisfiable iff the ontology $\{a : C\}$ is satisfiable, for some individual a; a terminology $\mathcal{T}$ is satisfiable iff a fresh atomic concept A is satisfiable w.r.t. $\mathcal{T}$; C is satisfiable w.r.t. $\mathcal{T}$ iff $\mathcal{T} \cup \{a : C\}$ is satisfiable, for a fresh individual a.

Complexity Theory. We assume familiarity with the standard notions of complexity theory as, e. g., defined in [23]. In particular, we will make use of the classes P, NP, coNP, PSPACE, and EXPTIME, as well as logspace reductions $\leq_{\mathrm{m}}^{\log}$.

Boolean operators. This study aims at being complete with respect to Boolean operators, which correspond to Boolean functions. A set of Boolean functions is called a *clone* if it is closed under superpositions of functions, *i.e.*, nesting of operators. The lattice of all clones has been established in [24], see [8] for a more succinct but complete presentation. Via the inclusion structure, lower and upper complexity bounds carry over to higher and lower clones. We will therefore only state our results for minimal and maximal clones.

   Given a finite set B of functions, the smallest clone containing B is denoted by $[B]$. The set B is called a *base* of $[B]$, but $[B]$ often has other bases as well. On the operator side, $[B]$ consists of all operators obtained by nesting operators in B into each other. For example, nesting of binary conjunction yields conjunctions of arbitrary arity. The table below lists all clones that we will refer to, using the following definitions. A Boolean function f is called *self-dual* if $f(\overline{x_1}, \ldots, \overline{x_n}) = \overline{f(x_1, \ldots, x_n)}$, *c-reproducing* if $f(c, \ldots, c) = c$, and *c-separating* if there is an $1 \leq i \leq n$ s.t. for each $(b_1, \ldots, b_n) \in f^{-1}(c)$ $b_i = c$ for $c \in \{\top, \bot\}$. The symbol $\oplus$ denotes the binary exclusive or.

Clone	Description	Base
BF	all Boolean functions	$\{\wedge, \neg\}$
M	All monotone functions	$\{\wedge, \vee, \bot, \top\}$
S_{11}	$\top$-separating, monotone function	$\{x \wedge (y \vee z), \bot\}$
D	self-dual functions	$\{(x \wedge \overline{y}) \vee (x \wedge \overline{z}) \vee (\overline{y} \wedge \overline{z})\}$
E	conjunctions and constants	$\{\wedge, \bot, \top\}$
E_0	conjunctions and $\bot$	$\{\wedge, \bot\}$
V_0	disjunctions and $\bot$	$\{\vee, \bot\}$
R_1	$\top$-reproducing functions	$\{\vee, x \oplus y \oplus \top\}$
R_0	$\bot$-reproducing functions	$\{\wedge, \oplus\}$
N_2	negation	$\{\neg\}$
I	identity functions and constants	$\{\mathrm{id}, \bot, \top\}$
I_0	identity functions and $\bot$	$\{\mathrm{id}, \bot\}$

Auxiliary results. The following lemmata contain technical results that will be useful to formulate our main results. We use $\star\mathrm{SAT}(B)$ to speak about any of the four satisfiability problems $\mathrm{TSAT}, \mathrm{TCSAT}, \mathrm{OSAT}$ and OCSAT introduced above.

Lemma 1. *Let B be a finite set of Boolean functions. If $\mathsf{N}_2 \subseteq [B]$, then it holds that $\star\mathrm{SAT}(B) \equiv_{\mathrm{m}}^{\log} \star\mathrm{SAT}(B \cup \{\top, \bot\})$.*

Proof. It is easy to observe that the concepts $\top$ and $\bot$ can be simulated by fresh atomic concepts T and B, using the axioms $\neg T \sqsubseteq T$ and $B \sqsubseteq \neg B$. □

Lemma 2. *Let B be a finite set of Boolean functions. Then it holds that $\mathrm{TCSAT}(B) \leq_{\mathrm{m}}^{\log} \mathrm{TSAT}(B \cup \{\top\})$.*

Proof. It can be easily shown that $\langle C, \mathcal{T} \rangle \in \mathrm{TCSAT}(B)$ iff $\langle \mathcal{T} \cup \{\top \sqsubseteq \exists R.C\} \rangle \in \mathrm{TSAT}(B \cup \{\top\})$, where R is a fresh relational symbol. For "$\Rightarrow$" observe that for the satisfying interpretation $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ there must be a world w' where C holds and then from every world $w \in \Delta^{\mathcal{I}}$ there can be an R-edge from w to w' to satisfy $\mathcal{T} \cup \{\top \sqsubseteq \exists R.C\}$. For "$\Leftarrow$" note that for a satisfying interpretation $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ all axioms in $\mathcal{T} \cup \{\top \sqsubseteq \exists R.C\}$ are satisfied. In particular the axiom $\top \sqsubseteq \exists R.C$. Hence there must be at least one world w' s.t. $w' \models C$. Thus $\mathcal{I} \models \mathcal{T}$ and $C^{\mathcal{I}} \supseteq \{w'\} \neq \emptyset$. □

Furthermore, we observe that, for each set B of Boolean functions with $\top, \bot \in [B]$, we can simulate the negation of an atomic concept using a fresh atomic concept A and role R_A: if we add the axioms $A \equiv \exists R_A.\top$ and $A' \equiv \forall R_A.\bot$ to the given terminology $\mathcal{T}$, then each model of $\mathcal{T}$ has to interpret A' as the complement of A.

3 Complexity results for CSAT

The following classification of concept satisfiability has been obtained in [16].

Theorem 3 ([16]). *Let B be a finite set of Boolean functions.*

1. *If $\mathsf{S}_{11} \subseteq [B]$, then $\mathrm{CSAT}(B)$ is PSPACE-complete.*
2. *If $[B] \in \{\mathsf{E}, \mathsf{E}_0\}$, then $\mathrm{CSAT}(B)$ is coNP-complete.*
3. *If $[B] \subseteq \mathsf{R}_1$, then $\mathrm{CSAT}(B)$ is trivial.*
4. *Otherwise $\mathrm{CSAT}(B) \in \mathrm{P}$.*

Part (1) is in contrast with the coNP-completeness of $\mathcal{ALU}$ satisfiability [27] because the operators in $\mathcal{ALU}$ can express the canonical base of S_{11}. The difference is caused by the fact that $\mathcal{ALU}$ allows only *unqualified* existential restrictions. Part (2) generalises the coNP-completeness of $\mathcal{ALE}$ satisfiability, where hardness is proven in [12] without using atomic negation. It is in contrast with the tractability of $\mathcal{AL}$ satisfiability [13], again because of the unqualified restrictions. Part (3) generalises the known fact that every $\mathcal{EL}$, $\mathcal{FL}_0$, and $\mathcal{FL}^-$ concept is satisfiable. The results for logics in the DL-Lite family cannot be put into this context because DL-Lite quantifiers are unqualified.

4   Complexity Results for TSAT, TCSAT, OSAT, OCSAT

In this section we will completely classify the above mentioned satisfiability problems for their tractability with respect to sub-Boolean fragments and put them into context with existing results for fragments of $\mathcal{ALC}$.

Main results. Due to the interreducibilities stated in Section 2, it suffices to show lower bounds for TSAT and upper bounds for OCSAT.

Theorem 4. *Let B be a finite set of Boolean functions.*

1. *If $\wedge \in B$ or $\vee \in B$, then TCSAT(B) is EXPTIME-hard.*
   If also $\top \in B$, then even TSAT(B) is EXPTIME-hard.
2. *If all functions in B are self-dual, then TSAT(B) is EXPTIME-hard.*
3. *If $\neg \in B$ or $\{\top, \bot\} \subseteq B$, then TSAT$(B)$ is PSPACE-hard.*
4. *If all functions in B are $\bot$-reproducing, then TSAT(B) is trivial.*

5. *If $\bot \in B$, then TCSAT(B) is coNP-hard.*
6. *If all functions in B are $\top$-reproducing, then OCSAT(B) is trivial.*

Proof. Parts 1.–6. are formulated as Lemmas 9, 10, 11, 7, 8, 12, and are proven below. The second part of (1.) follows from Lemma 9 in combination with Lemma 2. $\qquad\square$

As a consequence of Theorem 4 in combination with Lemma 6 in [21] , we obtain the following two corollaries that generalise the results to arbitrary bases for all four satisfiability problems.

Corollary 5. *Let B be a finite set of Boolean functions and $\star$SAT$'$ one of the problems TCSAT, OSAT and OCSAT.*

1. *If $\mathsf{E}_0 \subseteq [B]$ or $\mathsf{V}_0 \subseteq [B]$, and $[B] \subseteq \mathsf{M}$, then $\star$SAT$'(B)$ is EXPTIME-hard.*
2. *If $[B] = \mathsf{D}$ or $[B] = \mathsf{BF}$, then $\star$SAT$'(B)$ is EXPTIME-hard.*
3. *If $\mathsf{N}_2 \subseteq [B]$ or $\mathsf{I} \subseteq [B]$, then $\star$SAT$'(B)$ is PSPACE-hard.*
4. *If $[B] = \mathsf{I}_0$, then $\star$SAT$'(B)$ is coNP-hard.*
5. *If $[B] \subseteq \mathsf{R}_1$, then $\star$SAT$'(B)$ is trivial.*

Corollary 6. *Let B be a finite set of Boolean functions.*

1. *If $\mathsf{E} \subseteq [B]$ or $\mathsf{V} \subseteq [B]$, and $[B] \subseteq \mathsf{M}$, then TSAT(B) is EXPTIME-hard.*
2. *If $[B] = \mathsf{D}$ or $[B] = \mathsf{BF}$, then TSAT(B) is EXPTIME-hard.*
3. *If $\mathsf{N}_2 \subseteq [B]$ or $\mathsf{I} \subseteq [B]$, then TSAT(B) is PSPACE-hard.*
4. *If $[B] \subseteq \mathsf{R}_0$, or $[B] \subseteq \mathsf{R}_1$, then TSAT$(B)$ is trivial.*

Part (1) generalises the EXPTIME-hardness of subsumption for $\mathcal{FL}_0$ and $\mathcal{AL}$ with respect to GCIs [15, 11, 17]. It is in contrast to the tractability of subsumption with respect to GCIs in $\mathcal{EL}$ because our result does not separate the two types of restriction, because $\mathcal{EL}$ has only existential restriction, and our results do not (yet) consider existential, resp., universal restrictions separately. This undermines the observation that, for negation-free fragments, the choice

of the quantifier affects tractability and not the choice between conjunction and disjunction. Again, DL-Lite cannot be put into this context because of the unqualified restrictions.

Parts (2)–(4) (resp. (2) and (3) for Corollary 6) show that satisfiability with respect to theories is already intractable for even smaller sets of Boolean operators. One reason is that sets of axioms already contain limited forms of implication and conjunction. This also causes the results of this analysis to differ from similar analyses for related logics in that hardness already holds for bases of clones that are comparatively low in Post's lattice.

Due to Post's lattice, our analysis is complete for dividing the fragments into tractable and intractable cases.

Proofs of the main results.

Lemma 7. *Let B be a finite set of Boolean functions s.t. B contains only $\top$-reproducing functions. Then $\mathrm{OCSAT}(B)$ is trivial.*

Proof. According to Post's lattice, every B that does not fall under Theorem 4 (1)–(4)+(6) contains only $\top$-reproducing functions. Hence the following interpretation satisfies any instance $(\mathcal{O}, C)$: $\mathcal{I} = (\{w\}, \cdot^{\mathcal{I}})$ s.t. $A^{\mathcal{I}} = \{w\}$ for each atomic concept A, $r^{\mathcal{I}} = \{(w, w)\}$ for each role r, and $a^{\mathcal{I}} = w$ for each individual a. It then holds trivially that $\mathcal{I} \models \mathcal{O}$ and $C^{\mathcal{I}} = \{w\} \neq \emptyset$. $\square$

Lemma 8. *Let B be a finite set of Boolean functions s.t. B contains only $\bot$-reproducing functions. Then $\mathrm{TSAT}(B)$ is trivial.*

Proof. The intepretation $\mathcal{I} = (\{w\}, \cdot^{\mathcal{I}})$ with $A^{\mathcal{I}} = \emptyset$ for each atomic concept A, and $r^{\mathcal{I}} = \{(w, w)\}$ for each role r satisfies any instance $\mathcal{T}$ for $\mathrm{TSAT}(B)$, where B contains only $\bot$-reproducing functions. This follows from the observation that for each axiom $A \sqsubseteq B$ in $\mathcal{T}$ both sides are always falsified by $\mathcal{I}$ (because every atomic concept is falsified, and we only have $\bot$-reproducing operators as connectives). This can be shown by an easy induction on the concept structure. Please note that we need to construct a looping node concerning the transition relations due to the fact that we need to falsify axioms with $\forall r.\bot$ on the left side for some relation r. If we set $r^{\mathcal{I}} = \emptyset$ then this expression would be satisfied and would contradict our argumentation for the axiom $\forall r.\bot \sqsubseteq \bot$. Moreover this construction cannot fulfill wrongly the left side of an axiom because of the absence of $\top$ and as no atomic concept has instances with w. $\square$

Lemma 9. *Let B be a finite set of Boolean functions with $\wedge \in B$, or $\vee \in B$. Then $\mathrm{TCSAT}(B)$ is EXPTIME-hard. If all self-dual functions can be expressed in B, then $\mathrm{TSAT}(B)$ is EXPTIME-hard.*

Proof. The cases $\wedge \in B$ and $\vee \in B$ follow from [15]. The remaining case for the self-dual functions follows from Lemma 1, as all self-dual functions in combination with the constants $\top, \bot$ (to which we have access as $\neg$ is self-dual) can express any arbitrary Boolean function. $\square$

Lemma 10. *Let B be a finite set of Boolean functions s.t. $\{\bot, \top\} \subseteq B$. Then* TSAT$(B)$ *is* PSPACE-*hard.*

Proof. To prove PSPACE-hardness, we state a $\leq_{\mathrm{cd}}$-reduction from QBF-3-SAT to TSAT(B) and only allow $\bot$ and $\top$ as available functions in B. Let $\varphi \equiv \eth_1 x_1 \eth_2 x_2 \cdots \eth_n x_n (C_1 \wedge \cdots \wedge C_m)$ be a quantified Boolean formula and $\eth_i \in \{\exists, \forall\}$. In the following we construct a TBox $\mathcal{T} \subseteq \mathsf{Ax}(B)$ s.t. $\varphi \equiv \top$ if and only if $\mathcal{T} \in$ TSAT(B), where B consists only of $\top$ and $\bot$.

We are first adding the following axioms to the TBox $\mathcal{T}$ using atomic concepts $d_0, \ldots, d_n, x_1, \ldots, x_n, x'_1, \ldots, x'_n$ and roles $R_r, R_1, \ldots, R_n, S, R_{x_1}, \ldots, R_{x_n}$, $R_{d_1}, \ldots, R_{d_n}$, $R_{C_1}, \ldots, R_{C_m}, P_{11}, P_{21}, P_{31}, \ldots, P_{1m}, P_{2m}, P_{3m}$. The atomic concepts d_i stand for levels, x_i and x'_i for assigning truth values to the variables.

Initial starting point:
$$\{\top \sqsubseteq \exists S.d_0\} \tag{1}$$
x_i is the negation of x'_i:
$$\{x_i \equiv \exists R_{x_i}.\top \mid 1 \leq i \leq n\} \cup \{x'_i \equiv \forall R_{x_i}.\bot \mid 1 \leq i \leq n\} \tag{2}$$
in each level d_i we have R_{i+1}-successors where x_{i+1} and x'_{i+1} hold:
$$\{d_i \sqsubseteq \exists R_{i+1}.x_{i+1} \mid 0 \leq i < n\} \cup \{d_i \sqsubseteq \exists R_{i+1}.x'_{i+1} \mid 0 \leq i < n\} \tag{3}$$
the levels d_i are disjoint and we have succeeding levels:
$$\{d_i \sqsubseteq \forall R_{i+1}.d_{i+1} \mid 0 \leq i < n\} \cup$$
$$\{d_i \sqsubseteq \exists R_{d_i}.\top, d_j \sqsubseteq \forall R_{d_i}.\bot \mid 0 \leq i < j \leq n\} \tag{4}$$
x_i and x'_i carry over:
$$\{x_i \sqsubseteq \forall R_j.x_i \mid 1 \leq i < j \leq n\} \cup \{x'_i \sqsubseteq \forall R_j.x'_i \mid 1 \leq i < j \leq n\} \tag{5}$$

Now $\mathcal{T}$ is consistent, and each of its models contains a tree-like substructure similar to the one depicted in Figure 2. The *root* of this substructure is an instance of d_0. The individuals at depth n counting from the root are called *leaves*.

Please note that each individual in $\Delta^{\mathcal{I}}$ is an instance of either x_i or x'_i because of axiom (2). In particular, this holds for the leaves. Furthermore, this enforcement does not contradict the level-based labeling of the x_i—e.g., the atomic concepts x_i and x'_i "labeled in d_0" are *not* carried forward to the next levels because axiom (5) states this carry only if $j > i$.

Fig. 1. clause $C_6 \equiv \overline{x}_2 \vee x_3 \vee x_5$

In the remaining part, we need to ensure the following, where C_j is an arbitrary clause in φ. Each leaf w is an instance of the atomic concept C_j if and only if the combination of the x_i-values in w satisfies the clause C_j. In order to achieve this, we again use two complementary atomic propositions C_j and C'_j. The C'_j must be enforced in all leaves where *all* literals of C_j are set to false. For a literal $\ell \in \{x_1, \overline{x}_1, \ldots, x_n, \overline{x}_n\}$, use $\widetilde{\ell}$ to denote the atomic concept x_i if $\ell = \overline{x}_i$ and x'_i if $\ell = x_i$. The correct labeling of the leaves by the C_j and C'_j
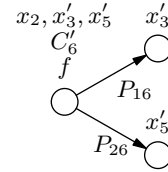
is ensured by adding the following axioms to $\mathcal{T}$, which enforce substructures as depicted for the example in Figure 1:

$$\left\{ \widetilde{l}_{1j} \sqsubseteq \exists P_{1j}.\top, \;\; \widetilde{l}_{2j} \sqsubseteq \forall P_{1j}.\widetilde{l}_{2j}, \;\; \exists P_{1j}.\widetilde{l}_{2j} \sqsubseteq \exists P_{2j}.\top, \right.$$
$$\left. \widetilde{l}_{3j} \sqsubseteq \forall P_{2j}.\widetilde{l}_{3j}, \;\; \exists P_{2j}.\widetilde{l}_{3j} \sqsubseteq C_j', \; \middle| \; C_j = l_{1j} \lor l_{2j} \lor l_{3j} \text{ in } \varphi \right\} \cup \quad (6)$$

$$\{ C_j' \sqsubseteq f \mid 1 \leq j \leq m \} \cup \quad (7)$$

$$\{ f \sqsubseteq \exists F.\top, \;\; f' \sqsubseteq \forall F.\bot \} \cup \quad (8)$$

$$\{ C_j \equiv \exists R_{C_j}.\top, \;\; C_j' \equiv \forall R_{C_j}.\bot \mid 1 \leq j \leq m \} \quad (9)$$

Finally we need to ensure that all concepts C_j are true in the leaves depending on the quantifications $\partial_1 x_1 \partial_2 x_2 \cdots \partial_n x_n$. For this purpose, we add the following axioms to the TBox $\mathcal{T}$ which ensure that, starting at the root, we run through each variable level of the tree as required by the quantification in φ, and reach only leaves that are no instances of f, *i.e.*, that are instances of f':

$$\{ d_0 \sqsubseteq \partial_1 R_1.\partial_2 R_2.\cdots \partial_n R_n.f' \} \quad (10)$$

Claim. $\varphi \equiv \top$ iff $\mathcal{T} \in \mathrm{TSAT}(\{\top, \bot\})$.

Proof. "$\Leftarrow$": Let $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ be an interpretation s.t. $\mathcal{I} \models \mathcal{T}$. Due to axiom (1), there exists an individual w_0 that is an instance of d_0. Because of axioms (3) and (4), there are at least two different R_1-successors of d_0, one being an instance of x_1 and the other of x_1' (axiom (5) in combination with axiom (4) ensure that these successors are fresh individuals). *Every* other R_1-successor is an instance of either x_1 or x_1', due to axiom (2). Other possible R_j-edges for $2 \leq j \leq n$ will not affect our argumentation as we will see in the following.

Repeated application of axioms (3) and (4) shows that this structure becomes a complete binary tree of depth n with (at least) 2^n leaves. Each leaf represents one of all possible Boolean combinations of x_i and x_i' for

Fig. 2. Essential part of the interpretation for the qBf $\varphi = \exists x_1 \forall x_2 \exists x_3 (x_1 \lor \neg x_2 \lor x_3) \land (\neg x_1 \lor \neg x_2 \lor \neg x_3)$.

$1 \leq i \leq n$. Due to axioms (3) and (4), every possible combination does occur. In addition, axiom (9) and (7) ensure the following: each leaf is an instance of

either C_j or C'_j, for each $1 \leq j \leq m$; if a leaf is an instance of at least one such C_j, it is also an instance of f.

Axiom (10) allows us to conclude that all relevant leaves that represent the assignments $\theta_i \colon \{x_1, \ldots, x_n\} \to \{0, 1\}$ for which $\theta_i \models C_1 \wedge \cdots \wedge C_m$ must hold, are instances of the proposition f'. Additional R_j-edges, as mentioned above, do not contradict the argumentation. Hence every relevant leaf must be an instance of every C_j because otherwise it were an instance of C'_j and thus of f'. Therefore, at least one literal in each clause is labeled and thereby satisfied. Hence $\varphi \equiv \top$.

Note that only those leaves that correspond to an assignment satisfying C_j can be instances of C_j. To clarify this fact, consider a clause $C_j = l_{1j} \vee l_{2j} \vee l_{3j}$ that is not satisfied by some assignment $\theta \colon \{x_1, \ldots, x_n\} \to \{\top, \bot\}$, and some leaf w is (erroneously) an instance of C_j. As $\theta \not\models C_j$, it holds that $\theta \not\models l_{ij}$ for $1 \leq i \leq 3$. Thus l'_{ij} must be labeled in w in order for axiom (2) to be satisfied. Now axiom (6) enforces R_{1j}- and R_{2j}-edges to successors satisfying $\widetilde{l}_{2j}$ and $\widetilde{l}_{3j}$. Finally, these propositions and transitions lead to w being an instance of C'_j. This is not possible because C_j and C'_j are disjoint due to axiom (9).

"$\Rightarrow$": Let n be the number of variables in φ. In [21] we show by induction on n: if $\varphi = \exists x_1 \forall x_2 \cdots \eth x_n (C_1 \wedge \cdots \wedge C_m) \equiv \top$, then $\mathcal{T} \in \text{TSAT}(\{\top, \bot\})$.

$\diamond$

As the number of axioms in $\mathcal{T}$ is polynomially bounded and the terminology is consistent if and only if the quantified Boolean formula φ is satisfiable, the lemma applies. $\qquad\square$

Lemma 11. TSAT($\{\neg\}$) *is* PSPACE-*hard.*

Proof. From Lemma 1 we can simulate $\top$ and $\bot$ with fresh atomic concepts. Then the argumentation follows similarly to Lemma 10. $\qquad\square$

Lemma 12. TCSAT($\{\bot\}$) *is* coNP-*hard.*

Proof. In contrast to Lemma 10, the instances of TCSAT(l_0) consist of a concept C and a TBox $\mathcal{T} \subseteq \mathsf{Ax}(\{\bot\})$. Both do not contain the concept $\top$. Now we adapt the proof of Lemma 10 to this new setting as follows: in all axioms containing $\top$, we replace $\top$ with a fresh atomic concept t. This is unproblematic except for axiom (1), where we need to enforce d_0 to have an instance. For this purpose, we remove the axiom $\top \sqsubseteq \exists S.d_0$ from $\mathcal{T}$ and set $C = d_0$. Additionally, we need to adopt axiom (10) to $d_0 \sqsubseteq \forall R_1.\forall R_2. \cdots \forall R_n.f'$ to match the desired reduction from TAUT. Please note, that with this construction it is not possible to state a reduction from QBF-3-SAT, because an interpretation where whenever we want to branch existentially, a respective individual with neither x_i nor x'_i labeled can be added without interfering the axioms, in particular axiom (2). $\qquad\square$

5   Conclusion

With Corollaries 5 and 6, we have separated the problems TSAT, TCSAT, OSAT and OCSAT for $\mathcal{ALC}$ fragments obtained by arbitrary sets of Boolean operators

into tractable and intractable cases. We have shown that these problems are on the one hand for TSAT

- EXPTIME-hard whenever we allow the constant $\top$ in combination with at least conjunction or disjunction,
- EXPTIME-hard whenever all Boolean self-dual functions can be expressed,
- PSPACE-hard whenever we allow at least negation or both constants $\bot, \top$,
- trivial in all other cases.

On the other hand for the remaining three satisfiability problems we reached EXPTIME-hardness even for only disjunction or conjunction (without the constant $\top$), and got coNP-hard cases whenever we allow at least the constant $\bot$ (hence the $\bot$-reproducing cases that are trivial for TSAT drop to intractable for these problems).

According to the Figures 4 and 5 in [21], which arrange our results in Post's lattice, this classification covers all sets of Boolean operators closed under nesting.

We have also shown how our results, and the direct transfer of the results in [16] to concept satisfiability, generalise known results for the $\mathcal{FL}$ and $\mathcal{EL}$ family and other fragments of $\mathcal{ALC}$. Furthermore, due to the presence of arbitrary axioms, the overall picture differs from similar analyses for related logics in that hardness already holds for small sets of inexpressive Boolean operators.

It remains for future work to find matching upper bounds for the hardness results, to look at fragments with only existential or universal restrictions, and to restrict the background theories to terminologies with atomic left-hand sides of concept inclusion axioms with and without cycles. Furthermore, since the standard reasoning tasks are not always interreducible if the set of Boolean operators is restricted, a similar classification for other decision problems such as concept subsumption is pending.

Acknowledgements

References

[1] A. Artale, D. Calvanese, R. Kontchakov, and M. Zakharyaschev. DL-Lite in the light of first-order logic. In *Proc. AAAI*, pages 361–366, 2007.

[2] A. Artale, D. Calvanese, R. Kontchakov, and M. Zakharyaschev. Adding weight to DL-Lite. In *Proc. DL*, http://CEUR-WS.org, 2009.

[3] F. Baader. Using automata theory for characterizing the semantics of terminological cycles. *Ann. Math. Artif. Intell.*, 18(2-4):175–219, 1996.

[4] F. Baader. Terminological cycles in a description logic with existential restrictions. In *Proc. IJCAI*, pages 325–330, 2003.

[5] F. Baader, S. Brandt, and C. Lutz. Pushing the $\mathcal{EL}$ envelope. In *Proc. IJCAI*, pages 364–369, 2005.

[6] F. Baader, S. Brandt, and C. Lutz. Pushing the $\mathcal{EL}$ envelope further. In *Proc. OWLED DC*, 2008.

[7] M. Bauland, T. Schneider, H. Schnoor, I. Schnoor, and H. Vollmer. The complexity of generalized satisfiability for Linear Temporal Logic. *LMCS*, 5(1), 2009.

[8] E. Böhler, N. Creignou, S. Reith, and H. Vollmer. Playing with Boolean blocks, part I: Post's lattice with applications to complexity theory. *ACM-SIGACT Newsletter*, 34(4):38–52, 2003.

[9] S. Brandt. Polynomial time reasoning in a description logic with existential restrictions, GCI axioms, and—what else? In *Proc. ECAI*, pages 298–302, 2004.

[10] D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, and R. Rosati. DL-Lite: Tractable description logics for ontologies. In *Proc. AAAI*, pages 602–607, 2005.

[11] F. M. Donini. Complexity of reasoning. In *Description Logic Handbook*, pages 96–136. Cambridge University Press, 2003.

[12] F. M. Donini, M. Lenzerini, D. Nardi, B. Hollunder, W. Nutt, and A. Marchetti-Spaccamela. The complexity of existential quantification in concept languages. *AI*, 53(2-3):309–327, 1992.

[13] F. M. Donini, M. Lenzerini, D. Nardi, and W. Nutt. The complexity of concept languages. *Inf. Comput.*, 134(1):1–58, 1997.

[14] F. M. Donini and F. Massacci. EXPTIME tableaux for $\mathcal{ALC}$. *AI*, 124(1):87–138, 2000.

[15] Robert Givan, David McAllester, Carl Wittny, and Dexter Kozen. Tarskian set constraints. *Information and Computation*, 174:105–131, 2002.

[16] Edith Hemaspaandra, Henning Schnoor, and Ilka Schnoor. Generalized modal satisfiability. *CoRR*, abs/0804.2729, 2008.

[17] M. Hofmann. Proof-theoretic approach to description-logic. In *Proc. LICS*, pages 229–237, 2005.

[18] Y. Kazakov and H. de Nivelle. Subsumption of concepts in $\mathcal{FL}_0$ for (cyclic) terminologies with respect to descriptive semantics is PSPACE-complete. In *Proc. DL*, http://www.CEUR-WS.org, 2003.

[19] H. Lewis. Satisfiability problems for propositional calculi. *Math. Sys. Theory*, 13:45–53, 1979.

[20] A. Meier, M. Mundhenk, T. Schneider, M. Thomas, V. Weber, and F. Weiss. The complexity of satisfiability for fragments of hybrid logic — Part I. In *Proc. MFCS*, volume 5734 of *LNCS*, pages 587–599, 2009.

[21] A. Meier and T. Schneider. The complexity of satisfiability for sub-Boolean fragments of $\mathcal{ALC}$. *CoRR*, http://arxiv.org/abs/1001.4255, 2010.

[22] B. Nebel. Terminological reasoning is inherently intractable. *AI*, 43(2):235–249, 1990.

[23] C. H. Papadimitriou. *Computational Complexity*. Addison-Wesley, 1994.

[24] E. Post. The two-valued iterative systems of mathematical logic. *Annals of Mathematical Studies*, 5:1–122, 1941.

[25] V. R. Pratt. A practical decision method for propositional dynamic logic: Preliminary report. In *STOC*, pages 326–337. ACM, 1978.

[26] T. J. Schaefer. The complexity of satisfiability problems. In *Proc. STOC*, pages 216–226. ACM Press, 1978.

[27] Manfred Schmidt-Schauß and Gert Smolka. Attributive concept descriptions with complements. *AI*, 48(1):1–26, 1991.

[28] Henning Schnoor. *Algebraic Techniques for Satisfiability Problems*. PhD thesis, Leibniz University of Hannover, 2007.

[29] M. Y. Vardi and P. Wolper. Automata-theoretic techniques for modal logics of programs. *JCSS*, 32(2):183–221, 1986.

Towards Formal Comparison of Ontology Linking, Mapping and Importing

Martin Homola[1,2] and Luciano Serafini[1]

[1] Fondazione Bruno Kessler,
Via Sommarive 18, 38123 Trento, Italy

[2] Comenius University,
Faculty of Mathematics, Physics and Informatics,
Mlynská dolina, 84248 Bratislava, Slovakia

`homola@fmph.uniba.sk`, `serafini@fbk.eu`

Abstract. Multiple distributed and modular ontology representation frameworks have recently appeared. They typically extend Description Logics (DL), with new constructs to represent relations between entities across several ontologies. Three kinds of constructs appear in the literature: link properties, found in $\mathcal{E}$-connections, semantic mapping, found in Distributed Description Logics (DDL), and semantic imports, used in Package-based Description Logics (P-DL). In this work, we aim towards formal comparison of the expressive power of these frameworks, and thus also the ontology combination paradigms that they instantiate. Reduction from DDL to $\mathcal{E}$-connections is already known. We present two new reductions, from P-DL to DDL and vice versa. These results show that there are similarities between these frameworks. However, due to the fact that none of the reductions is unconditional, it cannot be claimed that any of the three approaches is strictly more expressive than another.

1 Introduction

There are multiple reasons behind the research in modular and distributed ontologies. Introduction of modularity into ontology engineering, inspired by modular software engineering, calls for organizing ontologies into modules which could then be reused and combined in novel ways and thus the whole ontology engineering process will be facilitated and simplified. Another motivation comes from the Semantic Web vision, where ontologies are seen as a central ingredient, but on the other hand, decentralization and duplicity of knowledge sources is seen as very important too [1]. The centralized, monolithic treatment of ontologies, predominant in the mainstream theoretical research on ontology representation [2], is not sufficient; a novel decentralized and distributed ontology representation approach is of demand, that would deal with duplicity, temporal unavailability and also occasional inconsistence of the various ontological data sources.

The effort to achieve such an ontological representation breaks down into two problems: what are the requirements for an encapsulated and reusable ontology module, and how to combine such modules and enable reasoning with

the combined ontology. There are two main research directions. The first is to combine the modules simply by means of union and to find out the requirements under which such combination is feasible [3,4,5]. The second direction, on which we focus in this paper, imposes none or only very basic requirements on the modules and provides new constructs in order to combine the modules. Representation frameworks of this kind typically work with multiple DL [2] ontologies, and they provide novel constructs to interlink these ontologies. Each framework employs slightly different constructs; three ontology combination paradigms are distinguished: ontology linking, ontology mapping and ontology importing.

Ontology linking allows individuals from distinct ontologies to be coupled with *links*. A strict requirement is that the domains of the ontologies that are being combined are disjoint. See Fig. 1 a), where an ontology of companies is inter-linked with another ontology of products using the link produces. Links allow for complex concepts to be constructed, and they basically act as cross-ontology roles. The linking paradigm is employed by $\mathcal{E}$-connections [6].

Ontology mapping, in contrast, allows to relate ontologies on the same domain or on partially overlapping domains. Special *mapping* constructs indicate how elements from different ontologies are semantically related. Concepts, roles and individuals are possibly related. Mapping enables for knowledge reuse and also for resolution of semantic heterogeneity between ontologies that model the same domain but each from a different perspective. See Fig. 1 b), where mapping is expressed between two ontologies which possibly cover same entities but one models business relations and the other one models legal relations of these entities. In this paper we take a look at DDL [7], another notable instance of this paradigm are Integrated Distributed Description Logics [8].

Ontology importing allows a subset of concepts, relations and individuals defined in one ontology to be imported into another ontology where they are then reused. The importing takes care of propagating also the semantic relations that exist between these entities in their home ontology into the ontology that imports them. In Fig. 1 c), several roles are imported from a dedicated ontology module that deals with partonomy. Importing has been studied by Pan et al. [9]. The P-DL framework [10] falls under this paradigm.

This work aims towards comparison of the expressive power of these approaches. Some comparisons have already appeared in the literature. Some of them stay on qualitative or intuitive level [11,12]. Formal comparisons of the expressive power are less frequent. To our best knowledge, only reduction that is known is between DDL and $\mathcal{E}$-connections [13,6]. In this paper, we extend these efforts by producing two new reductions, from P-DL to DDL (with a specifically adjusted semantics), and vice versa. These results show, that there are certain similarities between P-DL and DDL, however, the semantics of P-DL is stronger, and it cannot be claimed that one of the frameworks is more expressive than the other. Similarly, one has to be careful when interpreting the reduction between DDL and $\mathcal{E}$-connections [13,6], because also this result assumes a DDL semantics which is not currently considered the standard one.
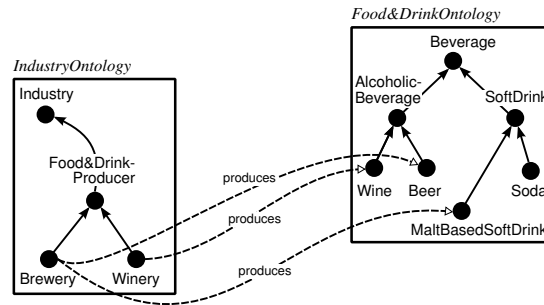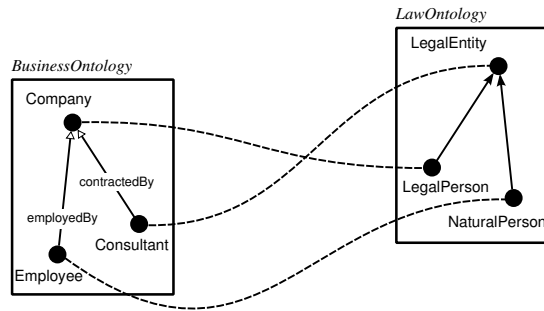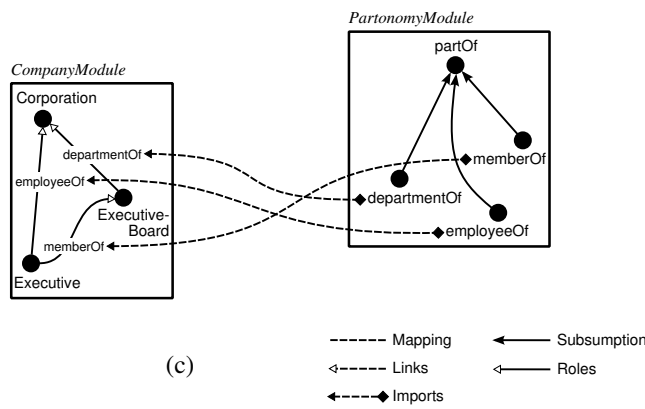
Fig. 1. Ontology combination paradigms: a) ontology linking; b) ontology mapping; c) ontology importing.

On the other hand, these results provide us with valuable insight on the relation between the paradigms of ontology linking, ontology mapping and ontology import, and can possibly guide the user that is to pick an appropriate formalism for a particular application.

2   Distributed Description Logics

DDL [7,14,15] follow the ontology mapping paradigm. They allow to connect multiple DL KB with bridge rules, a new kind of axioms that represents the mapping. An important point is that bridge rules are directed. Typically, DDL are built over $\mathcal{SHIQ}$. For the lack of space we cannot introduce $\mathcal{SHIQ}$ fully, please see the paper by Horrocks et al. [16].

Assume an index set I. Sets $N_\mathrm{C} = \{N_{\mathrm{C}i}\}_{i\in I}$, $N_\mathrm{R} = \{N_{\mathrm{R}i}\}_{i\in I}$ and $N_\mathrm{I} = \{N_{\mathrm{I}i}\}$ will be used for concept, role and individual names respectively. A distributed TBox is a family of TBoxes $\mathfrak{T} = \{\mathcal{T}_i\}_{i\in I}$, a distributed RBox is a family of RBoxes $\mathfrak{R} = \{\mathcal{R}_i\}_{i\in I}$, a distributed ABox is a family of ABoxes $\mathfrak{A} = \{\mathcal{A}_i\}_{i\in I}$, such that $\mathcal{K} = \langle \mathcal{T}_i, \mathcal{R}_i, \mathcal{A}_i \rangle$ is a $\mathcal{SHIQ}$ KB built over symbols from $N_{\mathrm{C}i}$, $N_{\mathrm{R}i}$, and $N_{\mathrm{I}i}$. By $i : \phi$ we denote that ϕ is a formula from $\mathcal{K}_i$. A bridge rule from i to j is an expression of one of the forms:

$$i : X \xrightarrow{\sqsubseteq} j : Y \ , \qquad i : X \xrightarrow{\sqsupseteq} j : Y \ , \qquad i : a \longmapsto j : b \ ,$$

where X and Y are either both concepts or both roles, and a, b are two individuals, in the respective language. The expression $i : X \xrightarrow{\equiv} j : Y$ is a syntactic shorthand for the pair of bridge rules $i : X \xrightarrow{\sqsubseteq} j : Y$ and $i : X \xrightarrow{\sqsupseteq} j : Y$. The symbol $\mathfrak{B}$ stands for a set of bridge rules, such that $\mathfrak{B} = \bigcup_{i,j\in I, i\neq j} \mathfrak{B}_{ij}$ where $\mathfrak{B}_{ij}$ contains only bridge rules from i to j. A DDL KB over I is $\mathfrak{K} = \langle \mathfrak{T}, \mathfrak{R}, \mathfrak{A}, \mathfrak{B} \rangle$ with all four components ranging over I.

A distributed interpretation $\mathfrak{I} = \langle \{\mathcal{I}_i\}_{i\in I}, \{r_{ij}\}_{i,j\in I, i\neq j} \rangle$ consists of local interpretations $\{\mathcal{I}_i\}_{i\in I}$ and domain mappings $\{r_{ij}\}_{i,j\in I, i\neq j}$ (notation: $r_{ij}(d) = \{d' \mid \langle d, d' \rangle \in r_{ij}\}$ and $r_{ij}(D) = \bigcup_{d\in D} r_{ij}(d)$). For each $i \in I$ the local interpretation is of the form $\mathcal{I}_i = \langle \Delta^{\mathcal{I}_i}, \cdot^{\mathcal{I}_i} \rangle$. In DDL, $\Delta^{\mathcal{I}_i}$ may also be empty. In such a case we call $\mathcal{I}_i$ a hole and denote it by $\mathcal{I}_i = \mathcal{I}^\epsilon$. A distributed interpretation $\mathfrak{I}$ satisfies elements of $\mathfrak{K}$ (denoted by $\mathfrak{I} \models_\epsilon \cdot$) according to the following clauses:

1. $\mathfrak{I} \models_\epsilon i : \phi$ if $\mathcal{I}_i \models \phi$;
2. $\mathfrak{I} \models_\epsilon \mathcal{T}_i$ if $\mathfrak{I} \models_\epsilon i : \phi$ for each $\phi \in \mathcal{T}_i$;
3. $\mathfrak{I} \models_\epsilon \mathfrak{T}$ if $\mathfrak{I} \models_\epsilon \mathcal{T}_i$ for each $i \in I$;
4. $\mathfrak{I} \models_\epsilon \mathcal{R}_i$ if $\mathfrak{I} \models_\epsilon i : \phi$ for each $\phi \in \mathcal{R}_i$;
5. $\mathfrak{I} \models_\epsilon \mathfrak{R}$ if $\mathfrak{I} \models_\epsilon \mathcal{R}_i$ for each $i \in I$;
6. $\mathfrak{I} \models_\epsilon \mathcal{A}_i$ if $\mathfrak{I} \models_\epsilon \phi$ for each $\phi \in \mathcal{A}_i$;
7. $\mathfrak{I} \models_\epsilon \mathfrak{A}$ if $\mathfrak{I} \models_\epsilon \mathcal{A}_i$ for each $i \in I$;
8. $\mathfrak{I} \models_\epsilon i : X \xrightarrow{\sqsubseteq} j : Y$ if $r_{ij}\left(X^{\mathcal{I}_i}\right) \subseteq Y^{\mathcal{I}_j}$;
9. $\mathfrak{I} \models_\epsilon i : X \xrightarrow{\sqsupseteq} j : Y$ if $r_{ij}\left(X^{\mathcal{I}_i}\right) \supseteq Y^{\mathcal{I}_j}$;
10. $\mathfrak{I} \models_\epsilon i : a \longmapsto j : b$ if $b^{\mathcal{I}_j} \in r_{ij}\left(a^{\mathcal{I}_i}\right)$;
11. $\mathfrak{I} \models_\epsilon \mathfrak{B}$ if $\mathfrak{I} \models_\epsilon \phi$ for all axioms $\phi \in \mathfrak{B}$.

A distributed interpretation $\mathfrak{I}$ is a distributed model, also ϵ-model, of $\mathfrak{K}$, if $\mathfrak{I} \models_\epsilon \mathfrak{T}$, $\mathfrak{I} \models_\epsilon \mathfrak{R}$, $\mathfrak{I} \models_\epsilon \mathfrak{A}$ and $\mathfrak{I} \models_\epsilon \mathfrak{B}$ (denoted $\mathfrak{I} \models_\epsilon \mathfrak{K}$). $\mathfrak{I}$ is a d-model of $\mathfrak{K}$, if $\mathfrak{I} \models_\epsilon \mathfrak{K}$ and it contains no hole (denoted $\mathfrak{I} \models_d \mathfrak{K}$). A DDL KB $\mathfrak{K}$ is i-consistent, if it has an ϵ-model with $\mathcal{I}_i \neq \mathcal{I}^\epsilon$; it is globally consistent if it has a d-model.

A concept C is ϵ-satisfiable (d-satisfiable) with respect to $\mathcal{K}_i$ of $\mathfrak{K}$, if there is an ϵ-model (d-model) $\mathfrak{I}$ of $\mathfrak{K}$ such that $C^{\mathcal{I}_i} \neq \emptyset$. A formula $i : \phi$ is ϵ-entailed (d-entailed) with respect to $\mathfrak{K}$, if in every ϵ-model (d-model) of $\mathfrak{K}$ we have $\mathfrak{I}$ satisfies $i : \phi$. This is denoted by $\mathfrak{K} \models_\epsilon i : \phi$ ($\mathfrak{K} \models_d i : \phi$).

The semantics corresponding to ϵ-satisfiability and ϵ-entailment is the actual state of the art in DDL. It enjoys many reasonable properties [7,14]. We will call this semantics DDL_ϵ. The notions of d-satisfiability and d-entailment (the semantics DDL_d) are rather auxiliary. As usual, entailment of subsumption formulae and (un)satisfiability are interreducible and both reducible into deciding i-consistence of a KB (DDL_ϵ) or global consistence of a KB (DDL_d).

DDL_ϵ and DDL_d permit any domain relations [7], but also alternate semantics with restricted domain relations have been investigated [17,18]. DDL_ϵ which requires domain relations to be partial functions is denoted by $\mathrm{DDL}_\epsilon(\mathrm{F})$. DDL_ϵ with injective domain relations is denoted by $\mathrm{DDL}_\epsilon(\mathrm{I})$. DDL_ϵ with composition-ally consistent domain relations ($r_{ij} \circ r_{jk} = r_{ik}$ for any distinct $i,j,k \in I$) is denoted by $\mathrm{DDL}_\epsilon(\mathrm{CC})$. DDL_ϵ with restricted compositionality ($r_{ij} \circ r_{jk} = r_{ik}$ for any distinct $i,j,k \in I$, if there is a directed path of bridge rules from i to j and from j to k) is denoted by $\mathrm{DDL}_\epsilon(\mathrm{RC})$. DDL_ϵ with role-preserving domain relation (if $(x,y) \in R^{\mathcal{I}_i}$ than $r_{ij}(x) \neq \emptyset \implies r_{ij}(y) \neq \emptyset$), for each R that appears on the left hand side of any bridge rule from i to j, will be denoted by $\mathrm{DDL}_\epsilon(\mathrm{RP})$. Variants with DDL_d and combinations are possible (e.g., later on we will discuss $\mathrm{DDL}_\epsilon(\mathrm{F,I,RC,RP})$, the semantics with partially functional, injective, restricted-compositional and role-preserving domain relations).

In addition, we will assume that local alphabets are mutually disjoint, and only atomic concepts are used in bridge rules. This is a normal form of DDL which is equivalent to the full version without these restrictions.

3   Package-based Description Logics

P-DL instantiate the ontology import paradigm. They allow a subset of terms to be imported from one DL KB to another (in P-DL these ontology modules are called packages). The review is based on the recent publication of Bao et al. [19] which builds on top of $\mathcal{SHOIQ}$ resulting into P-DL language $\mathcal{SHOIQP}$. The space does not permit us to introduce $\mathcal{SHOIQ}$ formally, please see the work of Horrocks [20].

A package based ontology is any $\mathcal{SHOIQ}$ ontology $\mathcal{P}$ which partitions into a finite set of packages $\{P_i\}_{i \in I}$, using an index set I. Each P_i uses its own alphabet of terms $N_{\mathrm{C}i} \uplus N_{\mathrm{R}i} \uplus N_{\mathrm{I}i}$ (concept, role and individual names, respectively). The alphabets are not mutually disjoint, but for any term t there is a unique home package of t, denoted by $\mathrm{home}(t)$. The set of home terms of a package $P_i \in \mathcal{P}$ is denoted by Δ_{S_i}. A term t occurring in P_i is a *local term* in P_i if $\mathrm{home}(t) = i$,

otherwise it is a *foreign term* in P_i. If t is a foreign term in P_j, and home$(t) = i$, then we write $P_i \xrightarrow{t} P_j$. If $P_i \xrightarrow{t} P_j$ for any term t, then also the package P_j imports the package P_i (denoted $P_i \to P_j$). By $\xrightarrow{*}$ we denote the transitive closure of $\to$ and by P_j^* the set $P_j \cup \{P_i | i \xrightarrow{*} j\}$.

When $P_i \to P_j$, the symbol $\top_i$ occurring within P_j represents the imported domain of P_i. Also in this case a novel *contextualized negation* constructor $\neg_i$ is applicable within P_j $\neg_i$ (it is however a mere syntactic sugar and can be ruled out as we always have $\neg_i C \equiv \top_i \sqcap \neg_j C$).

A distributed interpretation of $\mathcal{P}$ is a pair $\mathcal{I} = \left\langle \{\mathcal{I}_i\}_{i \in I}, \{r_{ij}\}_{i \xrightarrow{*} j} \right\rangle$, such that each $\mathcal{I}_i = \left\langle \Delta^{\mathcal{I}_i}, \cdot^{\mathcal{I}_i} \right\rangle$ is an interpretation of the local package P_i and each $r_{ij} \subseteq \Delta^{\mathcal{I}_i} \times \Delta^{\mathcal{I}_j}$ is a domain relation between $\Delta^{\mathcal{I}_i}$ and $\Delta^{\mathcal{I}_j}$. A distributed interpretation $\mathcal{I}$ is a model of $\{P_i\}_{i \in I}$, if the following conditions hold:

1. there is at least one $i \in I$ such that $\Delta^{\mathcal{I}_i} \neq \emptyset$;
2. $\mathcal{I}_i \models P_i$;
3. r_{ij} is an injective partial function, and r_{ii} is the identity function;
4. if $i \xrightarrow{*} j$ and $j \xrightarrow{*} k$, then $r_{ik} = r_{ij} \circ r_{jk}$ (compositional consistency);
5. if $i \xrightarrow{t} j$, then $r_{ij}(t^{\mathcal{I}_i}) = t^{\mathcal{I}_j}$;
6. if $i \xrightarrow{R} j$ and $(x, y) \in R^{\mathcal{I}_i}$ than $r_{ij}(x) \neq \emptyset \implies r_{ij}(y) \neq \emptyset$ (role preserving).

The three main reasoning tasks for P-DL are consistency of KB, concept satisfiability and concept subsumption entailment with respect to a KB. These are always defined with respect to a so called witness package $P_w \in \mathcal{P}$. A package-based ontology $\mathcal{P}$ is consistent as witnessed by a package P_w of $\mathcal{P}$, if there exists a model $\mathcal{I}$ of P_w^* such that $\Delta^{\mathcal{I}_w} \neq \emptyset$. In this case we also say that $\mathcal{P}$ is w-consistent. A concept C is satisfiable as witnessed by a package P_w of $\mathcal{P}$, if there exists a model $\mathcal{I}$ of P_w^* such that $C^{\mathcal{I}_w} \neq \emptyset$. A subsumption formula $C \sqsubseteq D$ is valid as witnessed by a package P_w of $\mathcal{P}$ (denoted $\mathcal{P} \models C \sqsubseteq_w D$), if for every model $\mathcal{I}$ of P_w^* we have $C^{\mathcal{I}_w} \subseteq D^{\mathcal{I}_w}$.

4 $\mathcal{E}$-connections

The $\mathcal{E}$-connections framework represents the ontology linking paradigm. Although many flavours of $\mathcal{E}$-connections are known, for sake of simplicity we introduce the language $\mathcal{C}^{\mathcal{E}}(\mathcal{SHIQ})$ [6,21,22]. This language allows to connect multiple ontologies expressed in $\mathcal{SHIQ}$ [16] with links.

Assume a finite index set I. For $i \in I$ let N_{Ci} and N_{Ii} be pairwise disjoint sets of concepts names and individual names respectively. For $i, j \in I$, i and j not necessarily distinct, let ϵ_{ij} be sets of properties, not necessarily mutually disjoint, but disjoint with respect to $N_{C_k}^{m_k}$ and $N_{I_k}^{m_k}$. An ij-property axiom is of the form $P_1 \sqsubseteq P_2$, where $P_1, P_2 \in \epsilon_{ij}$. An ij-property box $\mathcal{R}_{ij}$ is a finite set of ij-property axioms. The combined property box $\mathcal{R}$ contains all the property boxes for each $i, j \in I$.

Given some $i \in I$, $A \in N_{\mathrm{C}i}$, two i-concepts C and D, and a j-concept Z, and $P, S \in \epsilon_{ij}$, S is simple (see [16,22]), the following are also i-concepts:

$$\bot_i | \top_i | A | \neg C | C \sqcap D | C \sqcup D | \exists P.Z | \forall P.Z | \geqslant n\, S.Z | \leqslant n\, S.Z \ .$$

A combined TBox is a tuple $\mathcal{K} = \{\mathcal{K}_i\}_{i \in I}$ where each $\mathcal{K}_i$ is a finite set of i-local GCI axioms of the form $C \sqsubseteq D$, where C, D are i-concepts. A combined ABox $\mathcal{A} = \{\mathcal{A}_i\}_{i \in I} \cup \mathcal{A}_{\mathcal{E}}$ is a set of local ABoxes $\mathcal{A}_i$, each comprising of a finite number of i-local concept assertions $C(a)$ and role assertions $R(a,b)$, where C is an i-concept, $R \in \epsilon_{ii}$, $a, b \in N_{\mathrm{I}i}$. $\mathcal{A}_{\mathcal{E}}$ is a finite set of object assertions, each of the form $a \cdot E \cdot b$, where $E \in \epsilon_{ij}$, $a \in N_{\mathrm{I}i}$, $b \in N_{\mathrm{I}i}$. A combined KB is a triple $\Sigma = \langle \mathcal{K}, \mathcal{R}, \mathcal{A} \rangle$, where each component ranges over the same index set I.

Now we focus on the semantics. Given some KB $\Sigma = \langle \mathcal{K}, \mathcal{R}, \mathcal{A} \rangle$ with index set I, a combined interpretation is a triple $\mathcal{I} = \langle \{\Delta^{\mathcal{I}_i}\}_{i \in I}, \{\cdot^{\mathcal{I}_i}\}_{i \in I}, \{\cdot^{\mathcal{I}_{ij}}\}_{i,j \in I} \rangle$, where $\Delta^{\mathcal{I}_i} \neq \emptyset$, for $i \in I$, and $\Delta^{\mathcal{I}_i} \cap \Delta^{\mathcal{I}_j} = \emptyset$, for $i, j \in I, i \neq j$. The interpretation functions provide denotation for i-concepts ($\cdot^{\mathcal{I}_i}$) and for ij-properties ($\cdot^{\mathcal{I}_{ij}}$).

Each ij-property $P \in \epsilon_{ij}$ is interpreted by $P^{\mathcal{I}_{ij}} \subseteq \Delta^{\mathcal{I}_i} \times \Delta^{\mathcal{I}_j}$. $\mathcal{I}$ satisfies an ij-property axiom $P_1 \sqsubseteq P_2$, if $P_1{}^{\mathcal{I}_{ij}} \subseteq P_2{}^{\mathcal{I}_{ij}}$. Each i-concept C is interpreted by $C^{\mathcal{I}_i} \subset \Delta^{\mathcal{I}_i}$; $\top_i = \Delta^{\mathcal{I}_i}$ and $\bot_i = \emptyset$, and denotation of complex i-concepts must satisfy the constraints as given in Table 1. $\mathcal{I}$ satisfies an i-local GCI $C \sqsubseteq D$ (denoted by $\mathcal{I} \models C \sqsubseteq D$), if $C^{\mathcal{I}_i} \subseteq D^{\mathcal{I}_i}$. $\mathcal{I}$ satisfies an i-local concept assertion $C(a)$ (denoted $\mathcal{I} \models C(a)$), if $a^{\mathcal{I}_i} \in C^{\mathcal{I}_i}$; it satisfies an i-local role assertion $R(a,b)$ (denoted $\mathcal{I} \models R(a,b)$), if $\langle a^{\mathcal{I}_i}, b^{\mathcal{I}_i} \rangle \in R^{\mathcal{I}_{ii}}$; $\mathcal{I}$ satisfies an object assertion $a \cdot E \cdot b$ (denoted $\mathcal{I} \models a \cdot E \cdot b$), $a \in N_{\mathrm{I}i}$, $b \in N_{\mathrm{I}j}$, $E \in \rho_{ij}$, if $\langle a^{\mathcal{I}_i}, b^{\mathcal{I}_j} \rangle \in E^{\mathcal{I}_{ij}}$.

X	$X^{\mathcal{I}_i}$
$\neg C$	$\Delta^{\mathcal{I}_i} \setminus C^{\mathcal{I}_i}$
$C \sqcap D$	$C^{\mathcal{I}_i} \cap D^{\mathcal{I}_i}$
$C \sqcup D$	$C^{\mathcal{I}_i} \cup D^{\mathcal{I}_i}$
$\forall P.Z$	$\{x \in \Delta^{\mathcal{I}_i} \mid (\forall y)\ (x,y) \in P^{\mathcal{I}_{ij}} \implies y \in Z^{\mathcal{I}_j}\}$
$\exists P.Z$	$\{x \in \Delta^{\mathcal{I}_i} \mid (\exists y)\ (x,y) \in P^{\mathcal{I}_{ij}} \wedge y \in Z^{\mathcal{I}_j}\}$
$\geqslant n\, S.Z$	$\mathcal{I}_i = \{x \in \Delta^{\mathcal{I}_i} \mid \sharp\{y \mid (x,y) \in S^{\mathcal{I}_{ij}}\} \geq n\}$
$\leqslant n\, S.Z$	$\{x \in \Delta^{\mathcal{I}_i} \mid \sharp\{y \mid (x,y) \in S^{\mathcal{I}_{ij}}\} \leq n\}$

Table 1. Semantic constraints on complex i-concepts in $\mathcal{E}$-connections.

Finally, a combined interpretation $\mathcal{I}$ is a model of $\Sigma = \langle \mathcal{K}, \mathcal{R}, \mathcal{A} \rangle$ (denoted by $\mathcal{I} \models \Sigma$), if $\mathcal{I}$ satisfies every axiom in $\mathcal{K}$, $\mathcal{R}$ and $\mathcal{A}$. An i-concept is satisfiable with respect to Σ, if Σ has a combined model $\mathcal{I}$, such that $C^{\mathcal{I}_i} \neq \emptyset$. We have $\Sigma \models C \sqsubseteq D$, for two i-concepts C and D, if in each combined model $\mathcal{I}$ of Σ, $C^{\mathcal{I}_i} \subseteq D^{\mathcal{I}_i}$. Both reasoning tasks are inter-reducible, as usual.

5   Relating DDL to P-DL

In this section, we present two reductions. First is from P-DL into DDL, and later on also vice versa from DDL into P-DL. As we have learned, P-DL use a rather strongly restricted semantics, hence we will use $DDL_\epsilon(F,I,RC,RP)$ in order to make the reduction possible.

Theorem 1. *Given a $\mathcal{SHIQP}$ ontology $\mathcal{P} = \{P_i\}_{i\in I}$, with the importing relation $P_i \xrightarrow{t} P_j$. Let us construct a DDL KB $\mathfrak{K} = \langle \mathfrak{T}, \mathfrak{R}, \mathfrak{A}, \mathfrak{B} \rangle$ over I:*

- *$\mathcal{T}_i := \{\phi \mid \phi \in P_i$ is a GCI axiom$\} \cup \{\top_i \equiv \top\}$, for each $i \in I$;*
- *$\mathcal{R}_i := \{\phi \mid \phi \in P_i$ is a RIA axiom$\}$, for each $i \in I$;*
- *$\mathcal{A}_i := \{\phi \mid \phi \in P_i$ is an ABox assertion$\}$, for each $i \in I$;*
- *$\mathfrak{B}_{ij} := \{i : C \xrightarrow{\equiv} j : C \mid P_i \xrightarrow{C} P_j\} \cup \{i : R \xrightarrow{\equiv} j : R \mid P_i \xrightarrow{R} P_j\} \cup \{i : a \longmapsto j : a \mid P_i \xrightarrow{a} P_j\}$, for each $i, j \in I$.*

Given any $i \in I$, $\mathcal{P}$ is i-consistent if and only if $\mathfrak{K}$ is i-consistent under $DDL_\epsilon(F,I,RC,RP)$.

It trivially follows that also the decision problems of satisfiability and subsumption entailment are reducible, since they are reducible into *i*-consistence.

Corollary 1. *Deciding satisfiability and subsumption entailment in P-DL reduces into deciding i-consistence in DDL under the semantics $DDL_\epsilon(F,I,RC,RP)$.*

We will now show, that under this strong semantics of DDL, also a reduction in the opposite direction is possible.

Theorem 2. *Let $\mathfrak{K} = \langle \mathfrak{T}, \mathfrak{R}, \mathfrak{A}, \mathfrak{B} \rangle$ be a DDL KB over some index set I. We construct a package-based ontology $\mathcal{P} = \{P_i\}_{i\in I}$. Each package P_j contains a union of the following components:*

1. *$\mathcal{T}_j \cup \{C \sqsubseteq G \mid i : C \xrightarrow{\sqsubseteq} j : G \in \mathfrak{B}\} \cup \{G \sqsubseteq C \mid i : C \xrightarrow{\sqsupseteq} j : G \in \mathfrak{B}\}$;*
2. *$\mathcal{R}_j \cup \{R \sqsubseteq S \mid i : R \xrightarrow{\sqsubseteq} j : S \in \mathfrak{B}\} \cup \{S \sqsubseteq R \mid i : R \xrightarrow{\sqsupseteq} j : S \in \mathfrak{B}\}$;*
3. *$\mathcal{A}_j \cup \{a = b \mid i : a \longmapsto j : b \in \mathfrak{B}\}$.*

In addition, $\mathcal{P}$ uses the following imports:

- *$P_i \xrightarrow{C} P_j$ if either $i : C \xrightarrow{\sqsubseteq} j : G \in \mathfrak{B}$ or $i : C \xrightarrow{\sqsupseteq} j : G \in \mathfrak{B}$, for any $i, j \in I$, for any i-concept C and for any j-concept G;*
- *$P_i \xrightarrow{R} P_j$ if either $i : R \xrightarrow{\sqsubseteq} j : S \in \mathfrak{B}$ or $i : R \xrightarrow{\sqsupseteq} j : S \in \mathfrak{B}$, for any $i, j \in I$, for any i-role R and for any j-role S;*
- *$P_i \xrightarrow{a} P_j$ if $i : a \longmapsto j : b \in \mathfrak{B}$ for any $i, j \in I$, for any i-individual a and for any j-individual b.*

Given any $i \in I$, $\mathcal{P}$ is i-consistent if and only if $\mathfrak{K}$ is i-consistent under $DDL_\epsilon(F,I,RC,RP)$.

Also the remaining decision problems are reducible, because they are reducible into *i*-consistence in DDL.

Corollary 2. *Deciding satisfiability of concepts and subsumption entailment with respect in $DDL_\epsilon(F,I,RC,RP)$ reduces into deciding i-consistence in P-DL.*

As we see, P-DL are closely related to $DDL_\epsilon(F,I,RC,RP)$. More precisely, these results show, that it is possible to implement importing with bridge rules, and vice versa, it is possible to simulate bridge rules with imports and additional local axioms, *if the requirements placed on domain relations are the same.*

These results do not imply, however, that the two frameworks have the same expressivity, for two reasons. First, in terms of expressive power, none of the reductions is complete. DDL does not handle nominals, hence P-DL ontologies with nominals cannot be reduced. On the other hand, we deal with simplified version of DDL in this paper. We did not provide any reduction for DDL ontologies with heterogeneous bridge rules [15], and as these bridge rules essentially require domain relations that are not functional, it is hard to imagine that any reduction is possible.

The second reason is the fact that the DDL semantics used by the reductions is considerably stronger than what is commonly understood as appropriate DDL semantics. $DDL_\epsilon(RC)$, a weaker version of $DDL_\epsilon(F,I,RC,RP)$ has been studied [18], and it has been showed that it does not satisfy all the desiderata commonly placed on DDL [7,14]. More specifically, this semantics may behave unexpectedly, in the presence of an accidental inconsistency in one of the ontologies that are combined [18]. The problem also occurs with the stronger semantics $DDL_\epsilon(F,I,RC,RP)$, and we will show that it also occurs in P-DL.

Example 1. Consider a package-based ontology $\mathcal{P}$ consisting of three packages P_1, P_2 and P_3 with the following imports:

$$P_1 \xrightarrow{C} P_2 \ , \qquad\qquad P_2 \xrightarrow{D} P_3 \ , \qquad\qquad P_1 \xrightarrow{E} P_3 \ .$$

This is a very basic P-DL setting, each of the three packages imports one concept. Furthermore, let us assume that all tree packages are empty (i.e., $P_1 = P_2 = P_3 = \emptyset$). This means, that the three concepts C, D and E are unrelated. It is easy to show, that if P_2 becomes inconsistent, for some reason, then the imported concept D becomes unsatisfiable in P_3. This is quite intuitive, since D is imported from P_2. However, as we will show, if P_2 becomes inconsistent, also E becomes unsatisfiable in P_3 which we consider rather unintuitive, since this concept is imported from P_1 and is unrelated to D.

Let P_2 be inconsistent. For simplicity, let us assign $P_2 := \{\top \sqsubseteq \bot\}$. Due to the first two imports we get that $P_1 \xrightarrow{*} P_2$ and $P_2 \xrightarrow{*} P_3$, and so the semantics implies that in any model of $\mathcal{P}$, it must be the case that $r_{13} = r_{12} \circ r_{23}$. However, since P_2 is inconsistent, $\Delta^{\mathcal{I}_2} = \emptyset$ in every model, and hence $r_{12} \circ r_{23} = \emptyset$, and hence also $r_{13} = \emptyset$. And so, we also get $E^{\mathcal{I}_3} = r_{13}(E^{\mathcal{I}_1}) = \emptyset$.

The DDL_ϵ semantics does not exhibit this problem [14]. On the other hand, DDL_ϵ has problems with transitive propagation of the effects of bridge rules, which is not a problem for $DDL_\epsilon(RC)$, nor $DDL_\epsilon(F,I,RC,RP)$ [18]. In the P-DL setting this reformulates as the problem of transitive propagation of the

imported semantic relations, and due to correspondence between P-DL and DDL_ϵ(F,I,RC,RP), we now have a formal proof that it never appears in P-DL. This clearly marks the difference between the two approaches.

6   On Relation of DDL and $\mathcal{E}$-connections

The relation of DDL and $\mathcal{E}$-connections has been studied in the literature. It is known, that under certain assumptions, it is possible to reduce a DDL KB into $\mathcal{E}$-connections and reason equivalently in the latter formalism. More specifically, for a DDL KB with bridge rules between concepts and between individuals, the reasoning problems associated with d-entailment are reducible [6,13].

Theorem 3 ([6,13]). *Assume a DDL KB $\mathfrak{K} = \langle \mathfrak{T}, \mathfrak{R}, \mathfrak{A}, \mathfrak{B} \rangle$ with index set I and alphabet $N_\mathrm{C} = \{N_{\mathrm{C}i}\}_{i \in I}$, $N_\mathrm{R} = \{N_{\mathrm{R}i}\}_{i \in I}$ and $N_\mathrm{I} = \{N_{\mathrm{I}i}\}_{i \in I}$ which contains no bridge rules between roles. Let $N_\mathrm{C}' = N_\mathrm{C}$, $N_\mathrm{I}' = N_\mathrm{I}$, $\epsilon_{ii}' = N_{\mathrm{R}i}$, for each $i \in I$. Let $\epsilon_{ij}' = \{E_{ij}\}$, for each $i, j \in I$, $i \neq j$, where E_{ij} is a new symbol.*

Let us construct $\Sigma' = \langle \mathcal{K}', \mathcal{R}', \mathcal{A}' \rangle$, a $\mathcal{C}^{\mathcal{E}}(\mathcal{SHIQ})$ KB built over the index set I and the vocabulary N_C', N_I' and $\{\epsilon_{ij}'\}_{i,j \in I}$, as follows:

- *$\mathcal{K}_i' := \mathcal{T}_i \cup \{\exists E_{ij}.C \sqsubseteq G \mid j : C \xrightarrow{\sqsubseteq} i : G \in \mathfrak{B}\} \cup \{H \sqsubseteq \exists E_{ij}.D \mid j : D \xrightarrow{\sqsupseteq} i : H \in \mathfrak{B}\}$, for each $i \in I$;*
- *$\mathcal{R}_i' := \mathcal{R}_i$, for each $i \in I$;*
- *$\mathcal{A}_i' := \mathcal{A}_i$, for each $i \in I$;*
- *$\mathcal{A}_{\mathcal{E}}' := \{a \cdot E_{ij} \cdot b \mid i : a \longmapsto j : b \in \mathfrak{B}\}$.*

It follows, that $\mathfrak{K}$ is globally consistent if and only if Σ' is consistent.

As a straightforward consequence of the theorem, also the remaining reasoning tasks related to global consistency are reducible.

Corollary 3. *Deciding d-satisfiability and d-entailment of subsumption with respect to a DDL KB reduces into deciding the consistence of KB in $\mathcal{E}$-connections.*

This shows, that there is some similarity between bridge rules in DDL and links in $\mathcal{E}$-connections. The reduction especially shows, what is the exact relation between bridge rules on one side, and links on the other one. However it cannot be claimed that $\mathcal{E}$-connections are more expressive than DDL based on this result. This is for two reasons.

First, the reduction is for the DDL_d semantics. In DDL, however, "the semantics" is currently DDL_ϵ, which allows holes and satisfies a number of desired properties [7,14]. In particular, DDL_ϵ offers effective means to deal with accidental inconsistency, and hence it is possible to reason with a DDL KB, even if some local ontologies are inconsistent. $\mathcal{E}$-connections offer no such features.

Second, to our best knowledge it is not possible to reduce the richer flavours of DDL that include either bridge rules between roles or heterogeneous bridge rules between concepts and roles [15].

These findings are well in line with the different purpose for which each of the formalisms was designed. $\mathcal{E}$-connections work with carefully crafted local

modules with non-overlapping modeling domains, while DDL allow to connect overlapping ontologies in which part of the terminology is modeled on both sides although possibly differently.

7 Conclusion

The novel results of this paper are the reductions between P-DL and DDL (with a specifically adjusted semantics) and vice versa. This result suggests that these two frameworks have many similarities, and under certain assumptions, imports used in P-DL can be expressed by bridge rules used in DDL, and the other way around. On the other hand, these results also point out, that the actual difference between the two formalisms is in the fact that P-DL uses much stronger semantics than DDL. These semantics significantly differ, and hence it cannot be claimed that one of these frameworks is more expressive than the other.

Similarly, it cannot be claimed that $\mathcal{E}$-connections are more expressive than DDL, based on the reduction given by Kutz et al. [6,13]. This reduction is not given for the DDL semantics with holes, which is currently considered the most appropriate one, but for a simplified semantics instead. In addition, for some more complex DDL constructs no reduction is known in the literature. What the reduction does show, is how the concept of bridge rules and the concept of links (used in $\mathcal{E}$-connections) are related.

We conclude, that these results are well in line with the particular purpose for which each of the formalisms has been designed and is suitable for. $\mathcal{E}$-connections are particular well suited for combining multiple ontologies with separated local domains. This separation of domains, which has to be maintained, may also serve as a reasonable guide during modular ontology development.

On the other hand, sometimes we wish to combine and integrate also ontologies with partially overlapping domains. Especially in heterogeneous distributed knowledge environments, such as the Semantic Web, this is unavoidable. In such applications, DDL seems to be the most suitable, offering a versatile apparatus of ontology mapping, which allows concepts, roles and individuals to be associated freely according to the need of the application. DDL is robust enough to deal with accidental inconsistency and offers means to resolve possible heterogeneity in the modeling approaches used by different ontologies.

Finally, P-DL offers an ontology importing paradigm, which is very familiar to importing in software engineering. Thus P-DL is intuitive and easily understood also by users without deep understanding of ontology integration issues. Its strong semantics deals with many modeling issues, such as transitive propagation of imported relations, however, as the reduction suggests, it may possibly behave unexpectedly in the presence of accidental inconsistency in the system.

References

1. Berners-Lee, T., Hendler, J., Lassila, O.: The semantic web. Scientific American **284**(5) (2001) 34–43
2. Baader, F., Calvanese, D., McGuinness, D., Nardi, D., Patel-Schneider, P., eds.: The Description Logic Handbook. Cambridge University Press (2003)
3. Cuenca Grau, B., Horrocks, I., Kazakov, Y., Sattler, U.: A logical framework for modularity of ontologies. In: IJCAI 2007. (2007)
4. Ghilardi, S., Lutz, C., Wolter, F.: Did I damage my ontology? A case for conservative extensions in description logics. In: KR 2006, AAAI Press (2006)
5. Lutz, C., Walther, D., Wolter, F.: Conservative extensions in expressive description logics. In: IJCAI 2007. (2007)
6. Kutz, O., Lutz, C., Wolter, F., Zakharyaschev, M.: $\mathcal{E}$-connections of abstract description systems. Artificial Intelligence **156**(1) (2004) 1–73
7. Borgida, A., Serafini, L.: Distributed description logics: Assimilating information from peer sources. Journal of Data Semantics **1** (2003) 153–184
8. Zimmermann, A.: Integrated distributed description logics. In: DL-2007. Volume 250 of CEUR WS. (2007)
9. Pan, J.Z., Serafini, L., Zhao, Y.: Semantic import: An approach for partial ontology reuse. In: WoMo-06. Volume 232 of CEUR WS. (2006)
10. Bao, J., Honavar, V.: Ontology language extensions to support collaborative ontology building. In: ISWC2004. Volume 3298 of LNCS., Springer (2004)
11. Wang, Y., Haase, P., Bao, J.: A survey of formalisms for modular ontologies. In: IJCAI'07, SWeCKA Workshop. (2007)
12. Bao, J., Caragea, D., Honavar, V.: On the semantics of linking and importing in modular ontologies. In: ISWC 2006. Volume 4273 of LNCS., Springer (2006)
13. Cuenca Grau, B., Kutz, O.: Modular ontology languages revisited. In: IJCAI'07, SWeCKA Workshop. (2007)
14. Serafini, L., Borgida, A., Tamilin, A.: Aspects of distributed and modular ontology reasoning. In: IJCAI-05. (2005)
15. Ghidini, C., Serafini, L.: Mapping properties of heterogeneous ontologies. In: WoMo-06. Volume 232 of CEUR WS. (2006)
16. Horrocks, I., Sattler, U., Tobies, S.: Practical reasoning for expressive description logics. In: LPAR'99. Number 1705 in LNAI, Springer (1999) 161–180
17. Homola, M., Serafini, L.: Towards distributed tableaux reasoning procedure for DDL with increased subsumption propagation between remote ontologies. In: KR 2008 Ontology Workshop, Australian Computer Society (2008)
18. Homola, M., Serafini, L.: Augmenting subsumption propagation in distributed description logics. Applied Artificial Intelligence **24** (2010) 137–174
19. Bao, J., Voutsadakis, G., Slutzki, G., Honavar, V.: Package-based description logics. In: Modular Ontologies: Concepts, Theories and Techniques for Knowledge Modularization. Volume 5445 of LNCS. Springer (2009) 349–371
20. Horrocks, I., Sattler, U., Tobies, S.: Practical reasoning for very expressive description logics. J. Interest Group in Pure and Applied Logic **8**(3) (2000) 239–264
21. Cuenca Grau, B., Parsia, B., Sirin, E.: Working with multiple ontologies on the semantic web. In: ISWC2004. Volume 3298 of LNCS., Springer (2004)
22. Parsia, B., Cuenca Grau, B.: Generalized link properties for expressive E-connections of description logics. In: AAAI-2005, AAAI Press (2005)

Query Algebra and Query Optimization for Concept Assertion Retrieval

Jeffrey Pound, David Toman, Grant Weddell and Jiewen Wu

Cheriton School of Computer Science, University of Waterloo, Canada
{jpound, david, j55wu, gweddell}@uwaterloo.ca

Abstract. We develop a query algebra that supports efficient assertion retrieval—a natural extension of instance retrieval. The algebra is based on previously developed techniques for indexing concept descriptions. We show how relational-style query processing, including the use of secondary indices, of multiple cascaded indices, and so on, can be used to improve query performance, and also develop general conditions that enable query reformulation.

1   Introduction

Instance retrieval is a well known problem in which individual names from an ABox are retrieved in response to a query. The utility of a list of individual names however, has limitations in the context of end user applications. For example, displaying a list of individual identifiers may carry little useful information for a user of a DL-based information system. In this work, we focus on a generalization of the instance retrieval problem, *concept assertion retrieval*. In the concept assertion retrieval problem, a concept describing ABox individuals is retrieved in addition to the individual names. The concept is a least subsumer in a restricted language syntax specified as a parameter to the query. This parameter, a *projection description*, is used to specify the format of the returned concept description for each individual retrieved.

Concept assertion retrieval enables new possibilities for DL-based information systems as compared to tradition instance retrieval. Queries can now provide syntactically formatted concept descriptions suitable for communicating information about ABox individuals to end users. Also, concept-based ABox representations can allow efficient evaluation of queries by using tree-based search indices. In particular, query optimization may be performed in order to exploit available indices, making query evaluation efficient by avoiding general TBox reasoning in certain scenarios.

In our model, a query consists of a concept C describing individuals of interest, and a projection description Pd describing the desired information about an individual. The queries are processed over a knowledge base $\mathcal{K} = (\mathcal{T}, \mathcal{A})$, where $\mathcal{T}$ is a TBox in a chosen DL dialect, and $\mathcal{A}$ is an ABox containing assertions on individuals of interest. An evaluation of a query produces a set of assertions of the form $a : C_a$ such that $\mathcal{K} \models a : (C \sqcap C_a)$ where C_a is a least subsumer in the language defined by Pd.

As a driving application for efficient concept assertion retrieval, we consider the case of a collection of web objects with DL-based semantic annotations as an ABox, in addition to a terminology encoding general axioms over the concepts used to annotate web objects. In this scenario, a web application may embed concept assertion queries in a dynamic web page. An end user supplies search values for the queries through an interface. The evaluation of the query takes place with the resulting ABox individual names and associated concept descriptions inlined in the web page.

Example 1. Consider the case of an online dealer of photography equipment. As part of a web presence, the dealer maintains (1) a knowledge base $\mathcal{K}$ with a terminology for digital cameras and an ABox of assertions about particular cameras available for purchase through the dealer, and (2) a collection of web pages with embedded queries over this knowledge base. For example, one of the web pages might have a query Q with a query concept C of the form

$$ProductCode = \text{``digicam''} \sqcap Price < 300$$

paired with a projection description Pd of the form

$$(Name? \sqcap \exists Supplier.(OnlineAddress? \sqcap Rating?)). \tag{1}$$

Consequently, when browsing this page, a user sees in place of Q a list of inexpensive digital cameras, with each list element displaying the name of the camera together with a sublist of supplier URL addresses and ratings for that supplier.□

The example illustrates how assertions computed by our query language can resemble nested relations. Note that this is beyond the scope of more general conjunctive query languages. But also note that conjunctive queries can compute joins which are not expressible in our language. However, we believe that this is not really a requirement for browsing applications such as the above which focus on finding particular information about "objects of interest".

Our contributions are as follows:

1. We investigate the query optimization problem for a query algebra used in concept assertion retrieval. We show how concept-based index structures can be used to efficiently evaluate queries.
2. We show how query plans can be composed which eliminate the need for general TBox reasoning, by making use of precomputed information stored in indices.

Subsequent sections are organized as follows. Section 2 focuses on presenting a formal definition of our concept assertion retrieval problem. In Section 3, we show how basic operations for index scanning and projection can be extended to an algebra for manipulating sets of descriptions, and consider index-based query rewriting and index selection in this framework. Section 4 shows how purely relational algebraic expressions can be derived. Our summary comments then follow in Section 5. Note that all lemmas and the main theorem are stated without proof, but that all are straightforward (but tedious) inductions on the structure of various expressions.

1.1 Related Work

Our notion of concept assertion retrieval derives from an earlier notion of instance retrieval by Horrocks et al. [2] and of certain answer descriptions by ourselves in which we introduced the idea of a projection description [6]. We have also incorporated earlier work on an ordering language for DL concepts, introduced in [4], that attempts to distill comparison based reasoning that happens during search. Extensions to this language have also been explored, along with some initial experimental validation of the approach [5, 3].

2 Definitions

We presume the DL dialect $\mathcal{ALC}(\mathbb{S})$ whenever we mention a knowledge base $\mathcal{K}$, concept C, and so on, for the remainder of the paper. However, our results apply to any dialect that has the following definition of $\mathcal{ALC}(\mathbb{S})$ as a fragment. (This requirement can be relaxed without harm: the dialect need not support concept negation.)

Definition 1 (Description Logic $\mathcal{ALC}(\mathbb{S})$). *Let* $\{A, A_1, \ldots\}$, $\{R, R_1, \ldots\}$, $\{f, g, f_1, \ldots\}$ *and* $\{a, b, \ldots\}$ *denote countably infinite and disjoint sets of* concept names, *role names,* concrete features *and* individual names, *respectively. A* concept *is defined by:*

$$C, D ::= \top \mid \bot \mid A \mid \neg C \mid C \sqcap D \mid \exists R.C$$
$$\mid \quad f = k \qquad (equality\ over\ \mathbb{S})$$
$$\mid \quad f < g \qquad (linear\ order\ over\ \mathbb{S})$$

where k is a finite string. A constraint $\mathcal{C}$ *is an* inclusion dependency, concept assertion, *or* role assertion *with the respective forms* $C \sqsubseteq D$, $a : C$ *and* $R(a, b)$. *A* knowledge base $\mathcal{K}$ *is a finite set of constraints. We write $\mathcal{T}$ to denote the inclusion dependencies in $\mathcal{K}$, called a* terminology *or TBox, and write $\mathcal{A}$ to denote the assertions in $\mathcal{K}$, called an ABox (where $\mathcal{K}$ is understood from context in both cases).*

The semantics of $\mathcal{ALC}(\mathbb{S})$ is defined in the standard way based on interpretations of the form $(\triangle \uplus \mathbb{S}, \cdot^{\mathcal{I}})$ where $\mathbb{S}$ is a totally ordered concrete domain of finite strings that serves as range of concrete features. We use standard abbreviations such as $C \sqcup D$ for $\neg(\neg C \sqcap \neg D)$ and $f \leq k$ for $(f = k) \sqcup ((f < g) \sqcap (g = k))$. Also, given a finite set S of $\mathcal{ALC}(\mathbb{S})$ concepts, we write $\sqcap S$ to denote $\top$ if S is empty and the concept $D_1 \sqcap \cdots \sqcap D_n$ otherwise, when $S = \{D_1, ..., D_n\}$.

   Recall from our introductory comments that a user query (C, Pd) consists of a query concept C paired with a so-called projection description Pd. The syntax for a Pd and the sublanguage of concepts in $\mathcal{ALC}(\mathbb{S})$ that are induced by a Pd are defined as follows.

Definition 2 (Projection Description). *Let f, R and C be a concrete feature, role and concept, respectively. A* projection description Pd *is defined by the grammar:*

$$Pd \quad ::= \quad C? \mid f? \mid Pd_1 \sqcap Pd_2 \mid \exists R.Pd \tag{2}$$

Definition 3 (Induced Concepts). *Let Pd be a projection description. We define the sets $\mathcal{L}_{|Pd}$ and $\mathcal{L}_{|Pd}^{\mathrm{TUP}}$, the $\mathcal{L}$ concepts generated by Pd and $\mathcal{L}$ tuple concepts generated by Pd, respectively, as follows:*

$$\mathcal{L}_{|Pd} \;=\; \{\sqcap S \mid S \subseteq_{\mathrm{fin}} \mathcal{L}_{|Pd}^{\mathrm{TUP}}\}, \; and$$

$$\mathcal{L}_{|Pd}^{\mathrm{TUP}} \;=\; \begin{cases} \{C, \top\} & if\ Pd = \text{``}C?\text{''}; \\ \{f = k \mid k \in \mathbb{S}\} \cup \{\top\} & if\ Pd = \text{``}f?\text{''}; \\ \{C_1 \sqcap C_2 \mid C_1 \in \mathcal{L}_{|Pd_1}^{\mathrm{TUP}} \wedge C_2 \in \mathcal{L}_{|Pd_2}^{\mathrm{TUP}}\} & if\ Pd = \text{``}Pd_1 \sqcap Pd_2\text{''};\ and \\ \{\exists R.C \mid C \in \mathcal{L}_{|Pd_1}\} & if\ Pd = \text{``}\exists R.Pd_1\text{''}. \end{cases}$$

Thus, for a given Pd, any concept occurring in $\mathcal{L}_{|Pd}$ satisfies a syntactic format *conforming* to Pd independently of any terminology $\mathcal{T}$. Among all possible elements of $\mathcal{L}_{|Pd}$ are the *most informative* concepts for a given concept.

Definition 4 (Least Subsuming Concepts). *Let C, S and $\mathcal{K}$ be a concept, set of concepts and knowledge base, respectively. We write $\lfloor S \rfloor_{\mathcal{K}}(C)$ to denote the set of concepts $D \in S$ that are a least subsumer of C in S with respect to $\mathcal{K}$, that is, where $\mathcal{K} \models C \sqsubseteq D$, and for which there is no other concept $D' \in S$ such that $\mathcal{K} \models C \sqsubseteq D'$, $\mathcal{K} \models D' \sqsubseteq D$ and $\mathcal{K} \not\models D \sqsubseteq D'$.*

Lemma 1. *Let $\mathcal{K}$ be an $\mathcal{ALC}(\mathbb{S})$ knowledge base and Pd a projection description. Then the following hold for any concept C:*

1. *$\lfloor \mathcal{L}_{|Pd} \rfloor_{\mathcal{K}}(C)$ is non-empty;*
2. *$\mathcal{K} \models C_1 \equiv C_2$, for any $\{C_1, C_2\} \subseteq \lfloor \mathcal{L}_{|Pd} \rfloor_{\mathcal{K}}(C)$; and*
3. *$\lfloor \lfloor \mathcal{L}_{|Pd} \rfloor_{\mathcal{K}}(C) \rfloor_{\{\}}(\bot)$ is singleton.*

Parts *1* and *2* of Lemma 1 ensure that at least one least subsuming concept exists in $\mathcal{L}_{|Pd}$ and that they are are semantically equivalent with respect to a given $\mathcal{K}$. Note that some such $\mathcal{L}$ restriction of $\mathcal{ALC}(\mathbb{S})$ is essential to ensure part *1*, e.g., that a more general fragment that simply excludes concept negation from $\mathcal{ALC}(\mathbb{S})$ may not have this property [1]. Also note that, although $\mathcal{L}_{|Pd}$ is infinite in general, for any fixed and finite terminology $\mathcal{T}$ and concept C, the language $\mathcal{L}_{|Pd}$ restricted to the symbols used in $\mathcal{T}$ and C is necessarily finite.

Part *3* of Lemma 1 ensures that, among the least subsuming concepts in $\mathcal{L}_{|Pd}$ with respect to $\mathcal{K}$, there is a unique least subsuming concept that is *the most informative* when no knowledge of $\mathcal{K}$ is presumed. For example, let $\mathcal{K} = \{A \sqsubseteq (f = 1)\}$ and $Pd = (A? \sqcap f?)$. Then $\lfloor \mathcal{L}_{|Pd} \rfloor_{\mathcal{K}}(A) = \{A \sqcap (f = 1), A \sqcap \top\}$, and $\lfloor \{A \sqcap (f = 1), A \sqcap \top\} \rfloor_{\{\}}(\bot) = \{A \sqcap (f = 1)\}$.

To simplify notation in the remainder of the paper, we write $\pi_{Pd,\mathcal{K}}(C)$ as shorthand for the concept C_1 such that $\lfloor \lfloor \mathcal{L}_{|Pd} \rfloor_{\mathcal{K}}(C) \rfloor_{\{\}}(\bot) = \{C_1\}$. The formal semantics of a user query now follows.

Definition 5 (Query Semantics). *Let $\mathcal{K}$ be an $\mathcal{ALC}(\mathbb{S})$ knowledge base and $Q = (C, Pd)$ a user query over $\mathcal{K}$. Then Q computes the ABox*

$$\{a : \pi_{Pd,\mathcal{K}}(\sqcap \{D \mid (a : D) \in \mathcal{A}\}) \mid a \text{ occurs in } \mathcal{A} \text{ and } \mathcal{K} \models a : C\}. \qquad (3)$$

This semantics ensures that concept assertion retrieval generalizes instance retrieval. In particular, an instance retrieval query C over $\mathcal{K}$ can be formulated as query $(C, \top?)$ (effectively retrieving no further information about qualifying individual names in $\mathcal{K}$).

In this paper and in our current implementation, we make the simplifying assumption that a knowledge base does not contain any role assertions, a condition justified in, e.g., [2]. We also assume without loss of generality that a knowledge base will have at most one concept assertion in its ABox for any individual name a. Considered together, these assumptions imply that (3) above can be equivalently formulated as

$$\{a : \pi_{Pd, \mathcal{T}}(D) \mid (a : D) \in \mathcal{A} \text{ and } \mathcal{T} \models D \sqsubseteq C\},$$

which suggests two key problems for computing the results of a concept assertion query: computing least subsumers in $\mathcal{L}_{|Pd}$ for an arbitrary projection description Pd, and finding all concept assertions $a : D$ from a potentially large set of concepts assertions, e.g., comprising an ABox, that satisfy a selection condition given by a query concept. We consider these problems in the next section.

3   Indices and Query Algebra

We now introduce a *query algebra* for manipulating sets of concept descriptions. Concept assertions (and, in turn, ABoxes) are therefore *encoded* as concepts by a simple protocol based on the use of the special concrete feature Oid that is reserved for this purpose as follows (and we assume this correspondence for the remainder of the paper):

$$(a : C) \text{ encodes as } ((Oid = \text{``}a\text{''}) \sqcap C). \tag{4}$$

The algebra is centered around the operations for index-based selection [4] and for concept projection [6]; however, additional operators are included that allow basic boolean combinations of queries. We show how expressions in this algebra can describe a variety of *query plans* for evaluating a user query that can vary widely in the cost of their evaluation, and we outline how several standard relational-style *query optimization techniques* can be accommodated in this framework.

3.1   Concept Assertions and the use of Indices

The basic *leaf* operator of our algebra is an *index scan* as introduced in [6]. This assumes that all data, including the original ABox, are stored and organized with the help of so-called *concept trees* [4]. These are search trees in which nodes correspond to concepts and in which search order is defined by an *ordering description* (or Od for short): an expression conforming to the grammar "$Od ::= \text{Un} \mid f : Od \mid D(Od_1, Od_2)$". Intuitively, the productions in this grammar have the respective semantics: *no explicit ordering*, *ordering by the value of a concrete feature f*, and *partition by a description D*. The nesting of these constructs allows, e.g., for lexicographical ordering by several concrete features, etc. (again, see [4] for further details).

Definition 6 (Concept Index). *Let $\mathcal{A}$ and Od be a finite set of concept assertions and an ordering description, respectively. A* concept index *for $\mathcal{A}$ and Od is a concept tree with a node for each element of $\mathcal{A}$, encoded as a concept, that is well-formed with respect to Od. Given a knowledge base $\mathcal{K}$:*

1. *The* primary index $P^{\mathcal{K}}$ *is a concept index for $\mathcal{A}$ and $Oid : \text{Un}$;*
2. *A* secondary index $S^{\mathcal{K}}$ *is a concept index for the result of a user query $(C, Oid? \sqcap Pd)$ and some Od.*

In the second case we write $S^{\mathcal{K}} := (C, Oid? \sqcap Pd) :: Od$ to specify (or declare) such a secondary index over $\mathcal{K}$.

The primary index for a knowledge base always exists and is organized by the *names* (i.e., by the Oid feature in our representation) of the individuals described by the given ABox (hence the ordering description $Oid : \text{Un}$). Such an index provides an efficient way to retrieve the description associated with an individual in the ABox, given the individual's name. Also, the above definition permits the existence of any number (including zero) of secondary indices, that can be organized in various ways to support user queries[1]. Note that secondary indices are *essential* in our approach: they enable query evaluation to avoid (or reduce) the amount of general DL reasoning during query evaluation.

Example 2 (Concept Indices for Digital Cameras). To continue with our running example: we assume three additional secondary indices, in addition to the primary index $P^{\mathcal{CM}}$, are available:

$S_1^{\mathcal{CM}} := (\top, Oid? \sqcap ProductCode?) :: ProductCode : Oid : \text{Un},$
$S_2^{\mathcal{CM}} := (Price < 1000, Oid? \sqcap Price?) :: Price : Oid : \text{Un}, \text{ and}$
$S_3^{\mathcal{CM}} := (\top, Oid? \sqcap Name? \sqcap \exists Supplier.(OnlineAddress? \sqcap Rating?)) :: Oid : \text{Un}.$

The first index, $S_1^{\mathcal{CM}}$, enables an efficient search for individuals by the value of the feature $ProductCode$, the second by $Price$ for products costing under \$1000, and the third index enables search by the individual's name, and also *stores* a more elaborate *projection* of the concept description associated with that individual in the original ABox. □

3.2   Query Algebra

Recall that users specify concept assertion retrieval queries as pairs (C, Pd) where C is a concept that specifies a search condition and Pd is a projection description that specifies the format of the assertions in the answer to the query. To facilitate efficient evaluation of such requests we introduce a more complex *query algebra* to manipulate sets of concepts (usually encoding concept assertions). The algebra allows for the use of indices to speed-up search for qualifying individuals and to retrieve appropriate concepts needed to construct answer concept assertions.

---

[1] Similar to relational systems, multiple specialized indices are typically defined to support queries; this is in contrast to approaches that aspire to developing an "universal" search structure(s) to represent semantic data.

Definition 7 (Query Algebra). *The* Query Algebra *consists of the six opera-tors below, called constant query, index scan, selection, projection, intersection, union, and difference, respectively. Its syntax and semantics are as follows:*

$$
\begin{aligned}
&& (semantics) \\
Q ::={}& C & \{C\} \\
\mid{}& \text{SCAN}_X(Q) & \{D_1 \in X \mid \exists D_2 \in Q : \mathcal{T} \models D_1 \sqsubseteq D_2\} \\
\mid{}& \sigma_C(Q) & \{D \in Q \mid \mathcal{T} \models D \sqsubseteq C\} \\
\mid{}& \pi_{Pd}(Q) & \{\pi_{Pd,\mathcal{T}}(D) \mid D \in Q\} \\
\mid{}& Q_1 \cap Q_2 & \{D_1 \sqcap D_2 \mid D_1 \in Q_1, D_2 \in Q_2, \mathcal{T} \not\models (D_1 \sqcap D_2) \sqsubseteq \bot\} \\
\mid{}& Q_1 \cup Q_2 & \{D_1 \sqcap D_2 \mid D_1 \in Q_1, D_2 \in Q_2, \mathcal{T} \not\models (D_1 \sqcap D_2) \sqsubseteq \bot\} \\
&& \cup\ \{D_1 \sqcap \top \mid D_1 \in Q_1, \forall D_2 \in Q_2 : \mathcal{T} \models (D_1 \sqcap D_2) \sqsubseteq \bot\} \\
&& \cup\ \{\top \sqcap D_2 \mid D_2 \in Q_2, \forall D_1 \in Q_1 : \mathcal{T} \models (D_1 \sqcap D_2) \sqsubseteq \bot\} \\
\mid{}& Q_1 - Q_2 & \{D_1 \in Q_1 \mid \forall D_2 \in Q_2 : \mathcal{T} \models (D_1 \sqcap D_2) \sqsubseteq \bot\}
\end{aligned}
$$

where C is a concept description and X is either a primary index or secondary index. The semantics of the queries is defined the context of the primary index $P^{\mathcal{K}}$ and zero or more secondary indices $\{S_1^{\mathcal{K}}, \ldots, S_n^{\mathcal{K}}\}$ and with respect to a given knowledge base $\mathcal{K}$ with a TBox $\mathcal{T}$.

We say that a query is pure *if all occurrences of the C construct appear only in the scope (i.e., as subexpressions) of the $\text{SCAN}_X(Q)$ operator.*

We use the notation $Q[\mathcal{T}]$ in the remainder of the paper to make the particular TBox used in the above definition of semantics explicit.

Intuitively, each of the operators maps sets of concepts to a set of concepts, with $\text{SCAN}_X(C)$ the only leaf operator that links the algebra to the underlying *concept indices*. While not mandated by our definitions, the argument Q of a given $\text{SCAN}_X(Q)$ operator is expected to be related to the Od part of the specification of the underlying concept index X to facilitate efficient index search. For example, the index $S_1^{\mathcal{CM}}$ from Example 2 can only be efficiently searched with descriptions of the form $(ProductCode = k)$ for some string k.

3.3   Concept Assertion Queries as Algebraic Expressions

In this setting, a given user query (C, Pd) can always be expressed by the alge-braic expression $\pi_{(Oid?\sqcap Pd)}(\sigma_C(\text{SCAN}_{P_{\mathcal{K}}}(\top)))$ in our algebra[2]. However, to ben-efit from the performance gains made possible by secondary indices, the algebra allows a richer space of expressions:

Lemma 2. *Let (C, Pd) be a given query. Then the expression*

$$
\pi_{(Oid?\sqcap Pd)}(\sigma_C((\text{SCAN}_{S_1^{\mathcal{K}}}(C_1) \cap \cdots \cap \text{SCAN}_{S_n^{\mathcal{K}}}(C_n)) \cap (\text{SCAN}_{P^{\mathcal{K}}}(\top))) \tag{5}
$$

is equivalent to the original query, provided that (i) $S_i^{\mathcal{K}} := (D_i, (Oid? \sqcap Pd_i)) :: Od_i$, (ii) $\mathcal{T} \models C \sqsubseteq (D_1 \sqcap \ldots \sqcap D_n)$, and (iii) $C_i = \pi_{(Oid?\sqcap Pd_i),\mathcal{T}}(C)$, for all $0 < i \le n$.

---

[2] Note the explicit request for retrieving the individual's identifier by expanding the original projection description to $(Oid? \sqcap Pd)$.

Conditions (i) and (ii) ensure that the combination (intersection) of the indices $S_i^{\mathcal{K}}$ contains sufficient data to answer the original query. The last condition is necessary to supply a sufficiently *general* search concept to each of the indices. (Note that using the original search concept C instead would lead to loosing answers since the concept assertions stored in the secondary indices are more general than those in the ABox, in general.)

We can also check whether an index that satisfies the conditions in Lemma 2 is useful in pruning the search; it is easy to see, e.g., that indices for which $\mathcal{T} \models \top \sqsubseteq D_i$ and $\mathcal{T} \models \top \sqsubseteq C_i$ always return all ABox individuals and thus cannot be useful in pruning answers to the original query. In practice, the above condition can be refined to judge applicability of an index based, e.g., on *selectivity* (the fraction of individuals retrieved using the particular selection condition C_i).

The general form of (5) can be further simplified using the analogues of relational-style query rewrites that allow the use of *secondary indices* as follows:

Removing Redundant Selections: The selection operation $\sigma_C(\cdot)$ can be removed from (5) to obtain the expression

$$\pi_{(Oid?\sqcap Pd)}((\text{SCAN}_{S_1^{\mathcal{K}}}(C_1) \cap \cdots \cap \text{SCAN}_{S_n^{\mathcal{K}}}(C_n)) \cap (\text{SCAN}_{P^{\mathcal{K}}}(\top)), \quad (6)$$

if $\mathcal{T} \models (C_1 \sqcap \ldots \sqcap C_n) \sqsubseteq C$. Since the primary index $P^{\mathcal{K}}$ is sorted by the names of the individuals (Un), the last intersection operation in the above expression can be efficiently realized by an index nested loop join.

Index-Only Query Evaluation and Simplifying Projections: The expression (6) can be further simplified if one of the secondary indices provides assertions that conform to the final projection description $(Oid? \sqcap Pd)$:

$$\pi_{(Oid?\sqcap Pd)}(\pi_{Oid?}((\text{SCAN}_{S_1^{\mathcal{K}}}(C_1) \cap \cdots \cap \text{SCAN}_{S_{n-1}^{\mathcal{K}}}(C_{n-1}))) \cap (\text{SCAN}_{S_n^{\mathcal{K}}}(C_n)), \quad (7)$$

assuming the projection description in the declaration of $S_n^{\mathcal{K}}$ is the same as $(Oid? \sqcap Pd)$. Note that this rewriting *completely avoids* the use of the primary index.

The rewriting coupled with the ability to store and search efficiently among descriptions yields a path to defining appropriate physical data layout designs in the form of *concept indices* and in turn to *efficient plans for answering concept assertion retrieval queries*; we elaborate on this in Section 4.

Example 3. Recall the running example query (1). With the help of the secondary indices defined in Example 2, we can obtain the following equivalent query expression in our algebra:

$$\pi_{(Oid?\sqcap Pd)}(\pi_{Oid?}(\text{SCAN}_{S_1^{\mathcal{CM}}}(ProductCode = \text{``digicam''}) \cap$$
$$\text{SCAN}_{S_2^{\mathcal{CM}}}(Price < 300)) \cap \text{SCAN}_{S_3^{\mathcal{CM}}}(\top)). \quad (8)$$

The indices $S_1^{\mathcal{CM}}$ and $S_2^{\mathcal{CM}}$ fully qualify the individuals needed to answer the query and can be efficiently accessed using the concepts $(ProductCode = \text{``digicam''})$ and $(Price < 300)$, respectively. The expression then uses the index $S_3^{\mathcal{CM}}$ to form the concept assertions for the answer since $S_3^{\mathcal{CM}}$ stores the (most specific) descriptions conforming to Pd. $\square$

Index only rewriting can be generalized to cases in which the final projection description Pd is *contained* in the combination of the projection descriptions

Pd_i associated with the indices $S_i^{\mathcal{K}}$, i.e, $\pi_{Pd,\mathcal{T}}(D) = \pi_{Pd,\mathcal{T}}(\pi_{(\sqcap Pd_i),\mathcal{T}}(D))$ for all qualifying D [3]. Similarly, general selections can in principle be replaced by boolean combinations of index scans rather than by mere index intersections (the algebra provides the union and set difference operations) and DL reasoning can be used to test for soundness of such a rewrite. This arrangement can support, e.g., horizontal partitioning of indices and other advanced data partitioning schemes.

Now observe that $S_3^{\mathcal{CM}}$ is organized by the ordering description $Oid : \mathrm{Un}$, and therefore that the last intersection in the expression should reduce to an efficient index look-up for each qualifying individual. Here we utilize the explicit representation of the individual names in the descriptions manipulated by the algebra: the name can now be used as a *search condition* for an index.

Example 4. The final algebraic version of our running example thus yields the following expression:

$$\mathrm{SCAN}_{S_3^{\mathcal{CM}}}(\pi_{Oid?}(\mathrm{SCAN}_{S_2^{\mathcal{CM}}}(Price < 300 \,\sqcap$$
$$\pi_{Oid?}(\mathrm{SCAN}_{S_1^{\mathcal{CM}}}(ProductCode = \text{``}digicam\text{''}))))) \qquad (9)$$

Note that all the selections are now performed through an appropriate index scan operation, rather than by explicit set intersections.□

4 On Purely Structural Reasoning

There are a variety of cases in which the operators in our algebra can be evaluated with simple structural subsumption testing in place of general TBox reasoning. In this section, we characterize a general condition in which this holds for various operators in a given concept assertion query Q. Recall from Definition 7 that this happens, for example, when an evaluation of Q must correspond to an evaluation of $Q[\{\,\}]$.

We begin by introducing a notion of *typing* for queries in terms of projection descriptions, and a *normal form* for projection descriptions that suffices for characterizing the relationship between the information content of concept projections and structural subsumption testing.

Definition 8 (Query Typing and Projection Normalization). *Let Q and Pd be a query in the concept assertion algebra and a projection description, respectively. The* type *of Q, written $\alpha(Q)$, is a set of projection descriptions defined as follows:*

$$\alpha(Q) = \begin{cases} \{\top\} & \text{if } Q = \text{``}\mathrm{SCAN}_P(Q_1)\text{''}; \\ \{Pd\} & \text{if } Q = \text{``}\mathrm{SCAN}_{S:=(C,Pd)::Od}(Q_1)\text{''}; \\ \alpha(Q_1) & \text{if } Q = \text{``}\sigma_C(Q_1)\text{''} \text{ or }, \text{``}Q_1 - Q_2\text{''}; \\ \{Pd\} & \text{if } Q = \text{``}\pi_{Pd}(Q_1)\text{''}; \\ \{Pd_1 \sqcap Pd_2 \mid Pd_i \in \alpha(Q_i)\} & \text{if } Q = \text{``}Q_1 \sqcap Q_2\text{''}; \\ \alpha(Q_1) \cup \alpha(Q_2) & \text{if } Q = \text{``}Q_1 \cup Q_2\text{''}; \\ \{C?\} & \text{if } Q = \text{``}C\text{''} \text{ otherwise.} \end{cases}$$

---

[3] This condition is called *projection description refinement*; the full exploration of its properties is beyond the scope of this paper.

Also, the normal form of *Pd, written* $\mathsf{norm}(Pd)$ *is an exhaustive application of the following rules to any subexpression.*

1. $(f = k)? \rightsquigarrow f?$,
2. $(C_1 \sqcap C_2)? \rightsquigarrow (C_1? \sqcap C_2?)$, *and*
3. $\exists R.(Pd_1 \sqcap Pd_2) \rightsquigarrow (\exists R.Pd_1) \sqcap (\exists R.Pd_2)$.

Note that $\alpha(Q)$ denotes a set of projection description. This is necessary to adequately account for our union operator. Also note that $\mathsf{norm}(Pd)$ contains conjunctions only at the top-level and thus can be treated as a set of conjunction-free projection descriptions with component descriptions of the form $C?$ or $f?$ at the end of a (possibly empty) existential role path. For example, $\mathsf{norm}(A? \sqcap \exists R.(B?\sqcap\exists S.(f = 1)?))$ denotes a conjunction of the set of projection descriptions $\{A?, \exists R.B?, \exists R.\exists S.f?\}$.

   With query typing and projection normalization, we are now able to state our main result of the paper:

Theorem 1. *Let* $\mathcal{K}$ *and* Q *be a respective knowledge base and query in the concept assertion algebra. Then* $Q = Q[\{\,\}]$ *if at least one of the following conditions hold for any subquery* Q_1 *of* Q, *where* op *is one of* $\cap$, $\cup$ *or* $-$:

1. $Q_1 = \text{``}C\text{''}$;
2. $Q_1 = \text{``}\text{SCAN}_S(Q_2)\text{''}$ *and* $\mathsf{norm}(Pd_1) \subseteq \mathsf{norm}(Pd_2)$ *for any* $Pd_1 \in \alpha(Q_2)$, *where* S *is defined by* $(C, Pd_2) :: Od$;
3. $Q_1 = \text{``}\sigma_C(Q_2)\text{''}$ *and* $\mathsf{norm}(Pd_1) \subseteq \mathsf{norm}(Pd_2)$ *for any* $Pd_2 \in \alpha(Q_2)$ *and* $Pd_1 \in \alpha(C)$;
4. $Q_1 = \text{``}\pi_{Pd_1}(Q_2)\text{''}$ *and* $\mathsf{norm}(Pd_1) \subseteq \mathsf{norm}(Pd_2)$ *for any* $Pd_2 \in \alpha(Q_2)$;
5. $Q_1 = \text{``}(Q_2 \mathsf{\ op\ } Q_3)\text{''}$ *and both* Q_2 *and* Q_3 *are pure; and*
6. $Q_1 = \text{``}Q_2[\mathcal{K}]\text{''}$.

To see how the theorem applies, consider the following hypothetical query and its evaluation over a knowledge base $\mathcal{K} = \{A \sqsubseteq B\}$ consisting of a single inclusion dependency.

$$\big(\pi_{(Oid?\sqcap B?)}\big(\pi_{(Oid?\sqcap A?\sqcap B?)}\big(\underbrace{((Oid = \text{``}a\text{''}) \sqcap A)}_{\{a:A\}}{}^{(1)} \cup \underbrace{((Oid = \text{``}b\text{''}) \sqcap B)}_{\{b:B\}}{}^{(2)}\big)^{(3)}\big)^{(4)}\big)^{(5)}$$

$$\underbrace{\phantom{((Oid = \text{``}a\text{''}) \sqcap A) \cup ((Oid = \text{``}b\text{''}) \sqcap B)}}_{\{a:A,b:B\}}$$

$$\underbrace{}_{\{a:(A\sqcap B),b:(\top\sqcap B)\}}$$

$$\{a:B,b:B\}$$

The reader can confirm from our definitions that the same evaluation ensues if "$[\{\,\}]$" is inserted at positions (1), (2), (3) and (5) and "$[\mathcal{T}]$" at position (4), that is, that general TBox reasoning is required only for the $\pi_{(Oid?\sqcap A?\sqcap B?)}(\cdot)$ operator. We conclude with a more concrete example relating to our running online digital camera case.

Example 5. Theorem 1 now allows a reformulation of query (9) to the form

$$\text{SCAN}_{S_3^{\mathcal{CM}}}(\pi_{id?}(\text{SCAN}_{S_2^{\mathcal{CM}}}(Price < 300 \cap$$
$$\pi_{Oid?}(\text{SCAN}_{S_1^{\mathcal{CM}}}(ProductCode = \text{``}digicam\text{''})))))[\{\,\}] \qquad (10)$$

that completely avoids TBox reasoning.$\square$

5   Summary and Conclusions

The framework for *concept assertion retrieval* proposed in this paper provides a basis to introducing efficient relational-style query processing to querying the semantic web data. The main cornerstones of the approach are the ability to compute *projections* of general concepts to make properties of individuals syntactically explicit, to store such assertions in efficient tree-based search structures—*indices*, and to use such data structures to efficiently evaluate queries, in particular to sidestep the need for general DL reasoning at query evaluation time.

Future research can use the proposed query algebra to develop additional tools and techniques facilitating efficient query execution, for example:

- Optimization techniques that determine optimal (or nearly optimal reformulations of user queries in the algebra or its extensions; and
- Tools that allow the users to determine what indices to create for a given set of queries.

Another direction of research is whether more complex user queries, e.g., an equivalent of conjunctive queries, can be accommodated by modest extensions to the proposed framework.

A preliminary implementation of the proposed query algebra has been completed and an experimental evaluation of engineering feasibility is underway. The full source code for this implementation, along with an evaluation workload and test data is available online at `http://projection-alcd.googlecode.com/`.

References

1. Franz Baader, Baris Sertkaya, and Anni-Yasmin Turhan. Computing the least common subsumer w.r.t. a background terminology. *J. App. Logic*, 5(3):392–420, 2007.
2. Ian Horrocks, Lei Li, Daniele Turi, and Sean Bechhofer. The Instance Store: DL Reasoning with Large Numbers of Individuals. In Volker Haarslev and Ralf Möller, editors, *Description Logics*, volume 104 of *CEUR Workshop Proceedings*, 2004.
3. Jeffery Pound. Ordering, indexing, and searching semantic data: A terminology aware index structure. University of Waterloo, MMath Thesis. 2008.
4. Jeffrey Pound, Lubomir Stanchev, David Toman, and Grant Weddell. On Ordering Descriptions in a Description Logic. In *20th International Workshop on Description Logics*, pages 123–134. CEUR-WS vol. 250, 2007.
5. Jeffrey Pound, Lubomir Stanchev, David Toman, and Grant Weddell. On Ordering and Indexing Metadata for the Semantic Web. In *21st International Workshop on Description Logics*. CEUR-WS vol. 353, 2008.
6. Jeffrey Pound, David Toman, Grant Weddell, and Jiewen Wu. Concept Projection in Algebras for Computing Certain Answer Descriptions. In *22nd International Workshop on Description Logics*. CEUR-WS vol. 477, 2009.

On the feasibility of Description Logic knowledge bases with rough concepts and vague instances

C. Maria Keet

KRDB Research Centre, Free University of Bozen-Bolzano, Italy, `keet@inf.unibz.it`

Abstract. A usage scenario of bio-ontologies is hypothesis testing, such as finding relationships or new subconcepts in the data linked to the ontology. Whilst validating the hypothesis, such knowledge is uncertain or vague and the data is often incomplete, which DL knowledge bases do not take into account. In addition, it requires scalability with large amounts of data. To address these requirements, we take the $\mathcal{SROIQ}(D)$ and *DL-Lite* family of languages and their application infrastructures augmented with notions of rough sets. Although one can represent only little of rough concepts in *DL-Lite*, useful aspects can be dealt with in the mapping layer that links the concepts in the ontology to queries over the data source. We discuss the trade-offs and demonstrate validation of the theoretical assessment with the HGT application ontology about horizontal gene transfer and its 17GB database by taking advantage of the Ontology-Based Data Access framework. However, the prospects for *comprehensive and usable* rough DL knowledge bases are not good, and may require both sophisticated modularization and scientific workflows to achieve systematic use of rough ontologies.

1 Introduction

Various extensions of DLs and integration of DLs with other formalisms have been proposed, including to represent and reason over vague knowledge. To date, useful results have been obtained with fuzzy ontologies [1], but this is much less so for rough ontologies that aim to combine a standard DL with one of the formalisations of rough sets. In particular, [2–7] diverge in commitment as to which aspects of rough sets are included in the ontology language and the authors are concerned with the theory instead of demonstrating successful use of the rough DL in applications and ontology engineering. However, it has been noted within the Semantic Web context that scientist want to use ontologies together with data, such as hypothesizing that some subconcept exists and subsequently to validate this either in the laboratory or against the instances already represented in the knowledge base [8]. Such a hypothesised new concept is assumed to have a set-extension in the knowledge base and one would want to be able to match those instances with the right combination of object and data properties of the putative concept, i.e., taking a 'guessed' collection of attributes that is subsequently experimentally validated against the data and shown to be correct, or not; e.g., [9]. Such guessing includes dealing with incomplete or otherwise vague data, hence, for which some sort of *rough ontology* may be useful. Ideally, for all relevant individuals belonging to the putative concept, each value of the selected properties is distinct, but this may not be the case due to the limited data or insufficiency of the selected properties

so that some individuals are indistinguishable from each other and therewith instantiating a rough concept. Despite the vagueness, it still can be useful in the ontology engineering process to include such a rough concept in the ontology. To support such usage of ontologies, one needs a language with which one can represent, at least, rough concepts as the intensional representation of the corresponding rough set and a way to (persistently) relate the data to the rough concepts. As it turns out, there is no perfect DL language, reasoner, and ontology development tool that does it all with respect to the semantics of rough sets, nor will there be if one adheres to the hard requirement of staying within the decidable fragment of FOL, let alone within the tractable zone. Some results can be obtained, however: in addition to representing most of rough sets' semantics with $\mathcal{SROIQ}$ using the TBox only, the linking to data and, moreover, ascertaining if a concept is really a rough concept can be achieved within the framework of Ontology-Based Data Access (OBDA) by exploiting the mapping layer [10]. While this, arguably, may not be perceived as a great outcome, it is possible (and the remainder of the issues can be passed on to an application layer with scientific workflows and refinements in the technologies). To demonstrate it is not merely theoretically possible to have rough concepts and vague instances in one's DL knowledge base, but that it is indeed practically possible, we take the use case about horizontal gene transfer with a hypothesized (rough) concept Promiscuous Bacterium, and demonstrate how this can be modelled more precisely in an OWL 2 DL ontology and deployed in an OBDA system using a $DL\text{-}Lite_{\mathcal{A}}$ ontology stored as an owl file so that the instances from the 17GB large HGT-DB database can be retrieved.

The remainder of the paper is structured as follows. We first introduce the basics of rough sets and discuss identification of rough concepts in Section 2. Trade-offs to include such roughness features in DLs will be discussed in Section 3. Results of the experimentation with rough concepts and with vague instances will be presented in Section 4, where we consider both the HGT ontology with the HGT-DB database and [7]'s septic patients. We close with conclusions in section 5.

2 Identifying rough concepts

To be able to have a correspondence of a rough set with a rough concept in an ontology and to represent its essential characteristics, we first outline the basics of rough sets following the standard "Pawlak rough set model" (see for a recent overview [11, 12]).

2.1 Rough sets

The Pawlak rough set model is depicted informally in Fig. 1 and formally, it is as follows. $I = (U, A)$ is called an *information system*, where U is a non-empty finite set of objects and A a finite non-empty set of attributes and such that for every $a \in A$, we have the function $a : U \mapsto V_a$ where v_a is the set of values that attribute a can have. For any subset of attributes $P \subseteq A$, one can define the equivalence relation $\text{IND}(P)$ as

$$\text{IND}(P) = \{(x, y) \in U \times U \mid \forall a \in P, a(x) = a(y)\} \tag{1}$$

$\text{IND}(P)$ generates a partition of U, which is denoted with $U/\text{IND}(P)$, or U/P for short. If $(x, y) \in \text{IND}(P)$, then x and y are indistinguishable with respect to the attributes in P, i.e, they are *p-indistinguishable*.

Given these basic notions, we can proceed to the definition of rough set. From the objects in universe U, we want to represent set X such that $X \subseteq U$ using the attribute set P where $P \subseteq A$. X may not be represented in a crisp way—the set may include and/or exclude objects which are indistinguishable on the basis of the attributes in P— but it can be approximated by using lower and upper approximation, respectively:

$$\underline{P}X = \{x \mid [x]_P \subseteq X\} \tag{2}$$
$$\overline{P}X = \{x \mid [x]_P \cap X \neq \emptyset\} \tag{3}$$

where $[x]_P$ denotes the equivalence classes of the p-indistinguishability relation. The *lower approximation* (2) is the set of objects that are *positively* classified as being members of set X, i.e., it is the union of all equivalence classes in $[x]_P$. The *upper approximation* is the set of objects that are *possibly* in X; its complement, $U - \overline{P}X$, is the *negative region* with sets of objects that are definitely not in X (i.e., $\neg X$). Then, "with every rough set we associate two *crisp* sets, called *lower* and *upper approximation*" [11], which is commonly denoted as a tuple $X = \langle \underline{X}, \overline{X} \rangle$. The difference between the lower and upper approximation, $B_P X = \overline{P}X - \underline{P}X$, is the *boundary region* of which its objects neither can be classified as to be member of X nor that they are not in X; if $B_P X = \emptyset$ then X is, in fact, a crisp set with respect to P and when $B_P X \neq \emptyset$ then X is rough w.r.t. P.
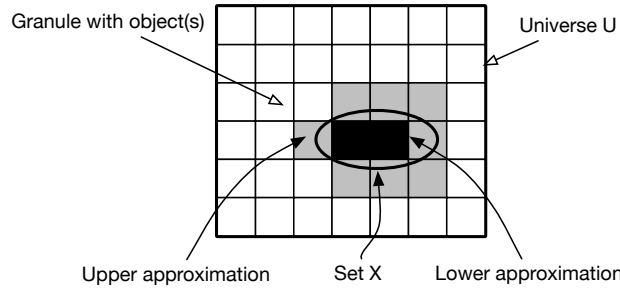
Fig. 1. A rough set and associated notions (Source: based on [11]).

The *accuracy of approximation* provides a measure of how closely the rough set is approximating the target set with respect to the attributes in P. There are several of such measures, denoted with $\alpha_P X$, for instance $\alpha_P X = \frac{|\underline{P}X|}{|\overline{P}X|}$ and $\alpha_P X = 1 - \frac{|B_P X|}{|U|}$. Clearly, if $\alpha_P X = 1$, then the boundary region $B_P X$ is empty and thus X is crisp.

Useful for subsequent sections is also the following property of approximations:

$$\underline{P}X \subseteq X \subseteq \overline{P}X \tag{4}$$

The rough set notions *reduct* and *core* can be considered to be the set of *sufficient* conditions (attributes) and the set of *necessary* conditions, respectively, to maintain the

equivalence class structure induced by P. Thus, we have $\text{CORE} \subseteq \text{RED} \subseteq P$ such that $[x]_{\text{RED}} = [x]_P$ and RED is minimal for any $a \in \text{RED}$ (i.e., $[x]_{\text{RED}-\{a\}} \neq [x]_P$), and for any reduct of P, $\text{RED}_1, \ldots, \text{RED}_n$, the core is its intersection, i.e., CORE = $\text{RED}_1 \cap \ldots \cap \text{RED}_n$. That is, those attributes that are in P but not in RED are superfluous with respect to the partitioning. On the other hand, no attribute in CORE can be removed without destroying the equivalence structure (it is possible that CORE is an empty set).

2.2 Some ontological considerations

As a first step toward rough ontologies, it would be a severe under-usage of DL knowledge bases if one only were to copy Pawlak's 'information system' essentials, because

1. In a logic-based (formal) ontology we have more constructors and possible constraints at our disposal, most notably a set of roles, $\mathcal{R}$, over objects and universal and existential quantification;
2. There is more flexibility on how to represent 'attributes' of a concept $C \in \mathcal{C}$: either with one or more roles $R \in \mathcal{R}$ (i.e., object properties in OWL) or value attributions $D \in \mathcal{D}$ (i.e., data properties in OWL), or both;
3. We need a complete and appropriate model-theoretic semantics for $\underline{C}$ and $\overline{C}$, and, as counterpart of the rough set, a rough concept, which we denote with "$\wr C$" for presentation convenience to clearly distinguish it from a crisp concept;
4. Given that attributes are used to compute $\underline{C}$ and $\overline{C}$, then those attributes must be represented in the ontology, and with $\wr C$ a tuple of the former two, then also it must have the attributes recorded in the ontology.

Concerning item 3, the semantics of the approximations is fairly straightforward, with E denoting the reflexive, symmetric and transitive indistinguishability (equivalence) relation:

$$\underline{C} = \{x \mid \forall y : (x, y) \in E \rightarrow y \in C\} \tag{5}$$
$$\overline{C} = \{x \mid \exists y : (x, y) \in E \land y \in C\} \tag{6}$$

Then there is rough sets' tuple notation, $X = \langle \underline{X}, \overline{X} \rangle$, for which we may have an analogous one for concepts, $\wr C = \langle \underline{C}, \overline{C} \rangle$. For $\wr C$, there are two issues: the notational distinction between a crisp (C) and a rough ($\wr C$) concept, and the tuple notation. Regarding the first issue, there are two ontological commitments one can take regarding the sets—either X is a special type of rough set where $\alpha = 1$ or a rough set is a special type of a crisp set because it is defined by the two crisp sets $\underline{X}$ and $\overline{X}$—and, subsequently, if a 'rough ontology' consists of only rough concepts or may contain both rough concepts and crisp concepts. Because rough sets are defined in terms of crisp sets, and, correspondingly, rough concepts in terms of a combination of two crisp concepts, this means that the crisp set and concepts are the 'primitive' ones and that we end up with a rough ontology that has both rough and crisp concepts to be able to have rough concepts properly defined in an ontology. For this reason, we maintain the, thus far, syntactic distinction between a crisp concept C and a rough concept $\wr C$. Regarding the second point, and, in fact, the semantics of $\wr C$, using a tuple notation is not ideal for discussing ontological commitments of rough sets and rough concepts and so it is useful to flatten it out. One can commit to the subsumption relation between the sets

as in (4) and their corresponding concepts as pursued by [5, 7] or take a more flexible approach that subsumes the former by introducing two binary relationships, $lapr$ and $uapr$, to relate *any* rough concept and its associated approximations, which are typed as follows:

$$\forall \phi, \psi . lapr(\phi, \psi) \rightarrow \wr C(\phi) \wedge \underline{C}(\psi) \tag{7}$$

$$\forall \phi, \psi . uapr(\phi, \psi) \rightarrow \wr C(\phi) \wedge \overline{C}(\psi) \tag{8}$$

Observe that here we are quantifying over *sets*, not objects that are member of the respective sets; i.e., we make explicit the knowledge about the three sets and how they relate, not about the instances in those sets. With these relations we can make explicit that $\wr C$ is identified by the combination of its $\underline{C}$ and $\overline{C}$, which is achieved by the following set of constraints:

$$\begin{aligned}
&\forall \phi . \wr C(\phi) \rightarrow \exists \psi . lapr(\phi, \psi), \\
&\forall \phi . \wr C(\phi) \rightarrow \exists \psi . uapr(\phi, \psi), \\
&\forall \phi, \psi, \varphi . lapr(\phi, \psi) \wedge lapr(\phi, \varphi) \rightarrow \psi = \varphi, \\
&\forall \phi, \psi, \varphi . uapr(\phi, \psi) \wedge uapr(\phi, \varphi) \rightarrow \psi = \varphi, \\
&\forall \phi_1, \phi_2, \psi_1, \psi_2 . lapr(\phi_1, \psi_1) \wedge uapr(\phi_1, \psi_2) \wedge \\
&\qquad lapr(\phi_2, \psi_1) \wedge uapr(\phi_2, \psi_2) \rightarrow \phi_1 = \phi_2.
\end{aligned} \tag{9}$$

The axioms in (9) say that for each rough concept, there must be exactly one lower approximation and one upper approximation and for each combination of lower and upper approximation, there is one rough concept, i.e., if either one of the approximations differ, we have a different rough concept.

Last, because a partitioning of the universe of objects is done by means of selecting a specific subset P of A of rough sets' information system, we have in the DL notion of ontology that the set of 'attributes' amounts to $\mathcal{R} \cup \mathcal{D}$. Moreover, one has to impose at the knowledge layer that those attributes P taken from $\mathcal{R} \cup \mathcal{D}$ must be represented in the ontology with $\wr C$ as its domain so as to represent explicitly and persistently which properties were used to obtain the rough set as extension of $\wr C$.

Overall, we thus have a more precise notion of $\wr C$ cf. the tuple notation in [5], use both $\mathcal{R}$ and $\mathcal{D}$ for the 'attributes' (properties) of the concepts (cf. $\mathcal{R}$ only in [4, 7]), include the properties of the indistinguishability/equivalence relation (cf. their omission in [6] or using the properties of the similarity relation [2]), and adhere to proper declaration of $\underline{C}$, $\overline{C}$, and $\wr C$ in that they all have the same collection of properties from $\mathcal{R} \cup \mathcal{D}$ (cf. giving the 'approximations' different sets of attributes in [7]).

3   Considerations regarding rough DL knowledge bases

The previous section introduced two essential aspects for a rough ontology language: the necessity to represent the indistinguishability relation E and declare it reflexive, symmetric, and transitive, and the identity of a rough concept by its lower and upper approximation by means of identification constraints involving DL roles. Currently, there is no DL language with corresponding complexity results that has both features.

On the one hand, one could decide to invent a new language that includes both features and that is hopefully still tractable in the light of abundant data. However, if one were to be faithful to (7-9), then a second order logic is required, which is out of scope. Alternatively, identification constraints (**id**s) have to be added in the ontology for each rough concept (perhaps guided with an outside-the-langauge ontology design pattern), hence the requirement to have the more common **id** constraint in the language. On the other hand, one can decide to push the envelope of extant languages and tools and make concessions. From a KR perspective, the former may be more interesting, but with an eye on applicability and demands from the most active user-base of ontologies—the life sciences and health care—it is worthwhile to push extant languages and its related tools as far as possible to gain better insight if development of a new language and corresponding tools are worth the effort. Give the extant languages, $\mathcal{SROIQ}(D)$ [13] suffices for representing E, but not **id** and it does not behave well in the light of large ABoxes, whereas the languages in the $DL\text{-}lite$ family [10] are well-suited to handle large ABoxes, but then we cannot represent E's relational properties and the **id** can be represented only in $DL\text{-}Lite_{\mathcal{A},id}$. Some other DL languages, such as $\mathcal{DLR}_{ifd}$ and $\mathcal{DLR}_{\mu}$, also have either one or the other feature, but not both.

For *practical* reasons, we narrow down the DL knowledge base further to the DL-based OWL species, because they are W3C standardised languages, there are ontology development tools for them, they have automated reasoners, and they are the DL of choice among the bio-ontologists. If we represent the reflexivity, symmetry and transitivity of E, then we are confined to the new OWL 2 DL, for this is the only one where one can assert all three object properties [14, 15]. For $\wr C$, there are two principal options: either define its semantics outside the language, or declare a "RoughC" in the ontology and let all rough concepts also be subsumed by it. In the latter option and considering the ontology languages and tools such as Protégé and Racer, we cannot represent the identification constraint anyway (nor the tuple notation $\wr C = \langle \underline{C}, \overline{C} \rangle$ proposed by [5]), and for the former option the applications would have to be adjusted to include a check if the rough concepts are declared correctly. Moreover, one should ask oneself what can be gained from including $\underline{C}$ and $\overline{C}$ in the ontology, besides deducing $\underline{C} \sqsubseteq C \sqsubseteq \overline{C}$ based on the declared knowledge in the TBox (thanks to (5) and (6)). Jiang and co-authors identify the specific TBox reasoning services for their $\mathcal{RDL}_{AC}$ as definitely satisfiable, possibly satisfiable, and rough subsumption [5]. However, considering rough sets' usage, it is the interplay with the actual instances that is crucial: after all, it is only based on the fact that, *given a non-empty ABox*, the boundary region is not empty that makes a concept a rough concept, and if we do not even test it against the actual instances in the knowledge base, then there is no point in bothering oneself to include a merely hypothetical rough concept in the ontology that cannot be examined either way if it really is a rough concept.

Thus, another hurdle is the data, which can be loaded into the ABox proper or stored and dealt with in secondary storage. Considering the most widely used ontology development tool Protégé, it loads the ABox in main memory, which is doable for small data sets but not for the medium to large size biological databases that easily exceed several GB. Setting aside supercomputers and the obstacle to wait a while on a query answer, this, then, forces one to take the second option of secondary storage, which, in turn

and at the time of writing, locks one into *DL-Lite* (and for the bio-ontologist, OWL 2 QL) that can represent even less of rough set's semantics (and of the subject domain) than OWL 2 DL. With the latter option, and, realistically, the Ontology-Based Data Access framework with QuOnto [10], the lack of expressiveness of the language can be counterbalanced by putting some of the subject domain semantics in the mapping layer. This is not ideal because it is not as maintainable as when it would be represented in the ontology, and it is not transparent for the domain expert who ideally should query just the ontology and not bother with the knowledge squeezed into the mapping layer, but we can get the data out of the database and have our rough concepts.

4   Experimentation with a rough ontology and vague instances

Given these trade-offs, we will demonstrate how one can have either an ontology with rough concepts represented fairly comprehensively regarding the semantics (in Experiment 1) or have it with more limited semantics but linked to the data and be able to perform the actual hypothesis testing against the data (Experiment 2) using the HGT as use case. To be fair to the latest technologies for expressive DLs, we also experimented with a more expressive ontology than the HGT ontology and then using much less data, by revisiting the definitions of septic of [7] and data of just 17 patients. Additional files (ontologies, mappings, queries, and data) are available online as supplementary material at `http://obda.inf.unibz.it/obdahgtdb/obdahgtdb.html`. The results will be discussed in Section 4.2.

4.1   Results

The background for Experiment 1 and 2 is as follows. A geneticist has an idea about what a "promiscuous bacterium" is because some bacteria transfer and receive much more genes from other bacteria than others do. It is not fully understood who they are and why this is the case, hence, the first step is to analyse the data—in casu, stored in the 17GB HGT-DB database—using properties that indicate a certain promiscuity so as to find bacteria with comparatively many anomalous (foreign) DNA in their chromosome.

Experiment 1 (Promiscuous bacteria in OWL 2 DL) We specify a first attempt for representing the promiscuous bacterium ($PromBact$) as a subtype of $Bact$erium in the HGT ontology with an additional object- and a data property, so that it must have more than 5 so-called flexible hgt-gene clusters ($FlexCl$, which are sets of adjacent or nearby genes that are horizontally transferred) and the percentage of genes on the chromosome that are predicted to be horizontally acquired, $hgtPerctg$, as > 10:

$$PromBact \equiv Bact \sqcap \exists\, hgtPerctg.real_{>10} \sqcap\, \geq 6\, hasHGTCluster.FlexCl \quad (10)$$

In addition, we can add the assertions regarding the equivalence relation (relational properties omitted for brevity) and that $PromBact$ has exactly one lower and one upper approximation, $PromBactLapr$ and $PromBactUapr$, as follows:

$$PromBact \sqsubseteq = 1 \, lapr.PromBactLapr \tag{11}$$

$$PromBact \sqsubseteq = 1 \, uapr.PromBactUapr \tag{12}$$

$$PromBactLapr \equiv \forall E.PromBact \tag{13}$$

$$PromBactUapr \equiv \exists E.PromBact \tag{14}$$

Running ahead of the data we retrieve with OBDA, $PromBact$ is indeed a rough concept, so we also have specified a refinement, $PromBact'$ to investigate if we now have included enough properties to have an empty boundary, hence a crisp concept:

$$PromBact' \equiv PromBact \sqcap \exists \, hgtPerctg.real_{>10} \sqcap$$
$$\geq 11 \, hasHGTCluster.FlexCl \sqcap nrHGTgenes.integer_{>150} \tag{15}$$

Querying or instance classification with this OWL 2 DL version and the HGT data is currently not feasible. $\diamondsuit$

Experiment 2 (Promiscuous bacteria in OBDA) As in Experiment 1, our first attempt is to represent $PromBact$ in $DL\text{-}Lite_{\mathcal{A}}$ (roughly OWL 2 QL), where we do not have existential quantification in the subclass position, no cardinality restrictions, limited object property assertions, no class equivalence, and no data property restrictions. To not have the intended meaning of $PromBact$ as in (10) all over the place, we chose to put it in the OBDA mapping layer; that is, we have $PromBact \sqsubseteq Bact$ in the $DL\text{-}Lite_{\mathcal{A}}$ ontology, and then make a mapping between $PromBact$ in the ontology and a SQL query over the relational database (for technical details about the OBDA framework used, the reader is referred to [10]). The head of the mapping is:

```
PromBact(getPromBact($abbrev,$ccount,$percentage))
```

and the body, i.e. the SQL query over the database where the WHERE clause has the set of interesting properties for $PromBact$ (which were modelled as object and data properties in the TBox in the previous experiment):

```
SELECT organisme.abbrev, ccount, organisme.percentage
   FROM ( SELECT idorganisme, COUNT(distinct cstart)
      as ccount FROM COMCLUSTG2 GROUP BY idorganisme
     ) flexcount, organisme
WHERE organisme.abbrev = flexcount.idorganisme AND
      organisme.percentage > 10 AND flexcount.ccount > 5
```

Querying the database through the ontology with a SPARQL query using the OBDA Plugin for Protégé and answered using DIG-QUONTO, 98 objects are retrieved where *Dehalococcoides CBDB1* and *Thermotoga maritima* are truly indistinguishable bacteria, i.e. they have the same values for all the selected and constrained attributes, and a few others are very close to being so, such as *Pelodictyon luteolum DSM273* and *Synechocystis PCC6803* who have both 6 clusters and 10.1% and 10.2%, respectively, (which, practically, still lie within the error-margin of genomics data and its statistics); see online material for details. Hence, $PromBact$ is actually a rough concept.

To improve the accuracy and examine if we can turn a subconcept of $PromBact$ into a crisp concept, a *new* data property—$NrOfHGTgenes$ with integer values, set to >150—is added and the second attribute set at >10 gene clusters, which thus *revises*

the assumption of what a promiscuous bacterium really is, i.e., we have $PromBact'$ in the ontology such that $PromBact' \sqsubseteq PromBact$. The head of the mapping is:

```
PromBactPrime(getPromBactPrime($abbrev,$ccount,$percentage,$hgt))
```

and the body:

```
SELECT organisme.abbrev,ccount,organisme.percentage,organisme.hgt
  FROM  ...
WHERE organisme.abbrev = flexcount.idorganisme AND
      organisme.percentage > 10 AND flexcount.ccount > 10 AND
      organisme.hgt > 150
```

The query answer has only 89 objects and this change even eliminates the boundary region, hence $PromBact'$ is a *crisp* concept with respect to the database. $\Diamond$

Experiment 3 (Revisiting septic patients) Patients may be septic or are certainly septic, according to the so-called *Bone criteria* and Bone criteria together with three out of another five criteria, respectively. For instance, the Bone criteria are (from [7]):

- *Has infection;*
- *At least two out of four criteria of the Systemic Inflammatory Response Syndrome:*
  - *temperature $> 38°C$ OR temperature $< 36°C$;*
  - *respiratory rate > 20 breaths/minute OR $PaCO_2 < 32$ mmHg;*
  - *heart rate > 90 beats/minute;*
  - *leukocyte count < 4000 mm^3 OR leukocyte count > 12000 mm^3;*
- *Organ dysfunction, hypoperfusion, or hypotension.*

The respective encodings in Protégé 4.0 and RacerPro 2.0 preview are available online as supplementary material, as well as data of 17 'patients' such that the boundary region of each concept is not empty. The experiments were carried out on a Macbook Pro with Mac OS X v 10.5.8 with 2.93 GHz Intel core 2 Duo and 4 GB memory. Protégé 4.0 with Pellet 2.0 did not work at all. Protégé 4.0 with FaCT++ works well with a few dummy concepts and a few instances, but the esoteric definitions for septic appeared to be more challenging: it crashed with an encoding including the indistinguishability relation E and (with or without E), upon saving and reopening the owl file it had reordered the braces in the definition in such a way as to change its meaning so that it does not classify all 17 individuals correctly. These observations may be due to the fact that the software used is still in the early stages. RacerPro 2.0 preview never crashed during exerimentation and did return the correct classifications within about 2 hours. While the latter is an encouraging result because it works with the real definitions and a small data set, the automated reasoning clearly does not scale to [7]'s thousands of patients. (The authors did not respond on a request for details of their experimental set-up.) $\Diamond$

4.2   Discussion

While a rough ontology such as the amended HGT ontology in OWL 2 DL can provide a better way of representing the declarative knowledge of putative and actual rough concepts, it is only with the less expressive *DL-Lite*-based OBDA system that it could be experimentally validated against the data. The ontologies and OBDA provide a means to represent the steps of successive de-vaguening during experimentation, they make

the selected properties explicit, and, if desired, one can keep both $\wr PromBact$ and $PromBact'$ in the ontologies without generating inconsistencies.

However, TBox rough subsumption and possible and definite satisfiability reasoning might be useful during engineering of rough ontologies. To improve outcomes for the expressive ontology setting, one could split up the database and import into the ABox all the data of only one organism at a time, do the instance classification, export the results, merge the results after each classification step, and then manually assess them. However, there are currently about 500 organisms in the database (which are soon to be extended to about 1000) and, ideally, this should not be done with one-off scripting. Alternatively, one may be able to design sophisticated modularization of both the ontology and the data(base) so as to execute the reasoning only on small sections of the ontology and database, in the direction of, e.g., [16, 17].

Although a rough DL knowledge base works as proof-of-concept, the procedure to carry it out is not perceived to be an ideal one. One might be able to turn into a feature the cumbersome interaction between the more precise representation of rough concepts in OWL 2 DL and the linking to data with OWL 2 QL (or a similar tractable language) by upgrading it to a named *scientific workflow*. This guides the developer to carry out in a structured, traceable, and repeatable manner the tasks to (*i*) develop a basic ontology in OWL 2 QL or $DL\text{-}Lite_{\mathcal{A}}$, (*ii*) get the database, (*iii*) set up the OBDA system, (*iv*) declare the mappings between the concepts and roles in the ontology and SQL queries over the database, (*v*) find all rough concepts with respect to the data and add them to the ontology, (*vi*) migrate this ontology to OWL 2 DL, (*vii*) add the semantics from the WHERE clause in the SQL query of the mapping layer as object and data properties in the ontology, (*viii*) add upper and lower approximations of each rough concept, (*ix*) add the equivalence relation with its properties, (*x*) add the axioms relating the approximations to the rough concepts and vice versa, and (*xi*) when the rough reasoning services are implemented, run the reasoner with the enhanced ontology. It will also be useful to go in the reverse direction in the light of updates to the database and in case the ontology was inconsistent or a had an unsatisfiable concept.

5   Conclusions

Extension of standard Description Logics knowledge bases with the essential notions of rough sets revealed both theoretical and practical challenges. Given rough sets' semantics, there is no, nor will there be, a DL that represents all essential aspects precisely, although expressive languages, such as $\mathcal{SROIQ}(D)$, come close and some tools, such as RacerPro, can handle complex rough concept descriptions with a small amount of data. On the other hand, it is the interaction with large amounts of data that makes any extension with roughness interesting and useful. This can be addressed with a tractable Ontology-Based Data Access framework by exploiting the mapping layer that links the concepts in the ontology over queries to the database. To validate the theoretical assessment, we have experimented with rough concepts and vague instances using the HGT case study and the recurring example of septic patients. The experimentation showed it is possible to have rough knowledge bases. However, more work in the direction of streamlining the rather elaborate procedure into a scientific workflow or developing im-

plementations of sophisticated ontology and data modularization, or both, is advisable in order to achieve a platform for hypothesis-driven usage of rough ontologies that will reap the greatest benefits to meet the users' requirements.

References

1. Lukasiewicz, T., Straccia, U.: Managing uncertainty and vagueness in description logics for the semantic web. Journal of Web Semantics **6**(4) (2008) 291–308
2. Bobillo, F., Straccia, U.: Supporting fuzzy rough sets in fuzzy description logics. In: Proc. of ECSQARU'09. Volume 5590 of LNCS., Springer (2009) 676–687
3. Fanizzi, N., D'Amato, C., Esposito, F., Lukasiewicz, T.: Representing uncertain concepts in rough description logics via contextual indiscernibility relations. In: Proc. of URSW'08. Volume 423 of CEUR-WS. (2008)
4. Ishizu, S., Gehrmann, A., Nagai, Y., Inukai, Y.: Rough ontology: extension of ontologies by rough sets. In Smith, M.J., Salvendy, G., eds.: Proceedings of Human Interface and the Management of Information. Volume 4557 of LNCS., Springer (2007) 456–462
5. Jiang, Y., Wang, J., Tang, S., Xiao, B.: Reasoning with rough description logics: An approximate concepts approach. Information Sciences **179** (2009) 600–612
6. Liau, C.J.: On rought terminological logics. In: Proceedings of the 4th International Workshop on Rough Sets, Fuzzy Sets and machine Discovery (RSFD'96). (1996) 47–54
7. Schlobach, S., Klein, M., Peelen, L.: Description logics with approximate definitions— precise modeling of vague concepts. In: Proc. of IJCAI'07, AAAI Press (2007) 557–562
8. Keet, C.M., Roos, M., Marshall, M.S.: A survey of requirements for automated reasoning services for bio-ontologies in OWL. In: Proc. of OWLED'07. Volume 258 of CEUR-WS. (2007) 6-7 June 2007, Innsbruck, Austria.
9. Marshall, M.S., Post, L., Roos, M., Breit, T.M.: Using semantic web tools to integrate experimental measurement data on our own terms. In: Proc. of KSinBIT'06. Volume 4277 of LNCS., Springer (2006) 679–688
10. Calvanese, D., et al.: Ontologies and databases: The DL-Lite approach. In: Semantic Technologies for Informations Systems. Volume 5689 of LNCS., Springer (2009) 255–356
11. Pawlak, Z., Skowron, A.: Rudiments of rough sets. Information Sciences **177**(1) (2007) 3–27
12. Pawlak, Z., Skowron, A.: Rough sets: Some extensions. Information Sciences **177**(1) (2007) 28–40
13. Horrocks, I., Kutz, O., Sattler, U.: The even more irresistible $\mathcal{SROIQ}$. Proc. of KR'06 (2006) 452–457
14. Motik, B., Patel-Schneider, P.F., Parsia, B.: OWL 2 web ontology language structural specification and functional-style syntax. W3c recommendation, W3C (27 Oct. 2009) http://www.w3.org/TR/owl2-syntax/.
15. Motik, B., Grau, B.C., Horrocks, I., Wu, Z., Fokoue, A., Lutz, C.: OWL 2 Web Ontology Language Profiles. W3c recommendation, W3C (27 Oct. 2009) http://www.w3.org/TR/owl2-profiles/.
16. Lutz, C., Toman, D., Wolter, F.: Conjunctive query answering in the description logic el using a relational database system. In: Proc. of IJCAI'09, AAAI Press (2009)
17. Baader, F., Bienvenu, M., Lutz, C., Wolter, F.: Query and predicate emptiness in description logics. In: Proc. of KR'10. (2010)

Towards Soundness Preserving Approximation for ABox Reasoning of OWL2

Yuan Ren, Jeff Z. Pan and Yuting Zhao

Department of Computing Science
University of Aberdeen
Aberdeen, UK

Abstract. ABox Reasoning in large scale description logic (DL) knowledge bases, e.g. ontologies, is important for the success of many semantic-enriched systems. Performance of existing approaches, such as the tableau-based approach, and the disjunctive datalog approach, is restricted by their theoretical worst case complexity bound. In this paper, we propose a soundness-preserving approximate reasoning approach to address this issue. We first approximate an ontology in DL $\mathcal{RO}$, a major fragment of OWL2-DL, to DL $\mathcal{EL}^{++}$, the underpin of OWL2-EL, plus an additional table maintaining the complementary relations between concept names. Then we can perform ABox reasoning either internally, or externally of the TBox with additional completion rules. The approximation and reasoning can be performed in PTIME. Our preliminary evaluation shows that our approach can outperform existing DL reasoners on real world and benchmark ontologies.

1 Introduction

With the fast development of the semantic web and knowledge intensive systems, the representation and reasoning over large-scale ontologies have become important topics for research community. Web Ontology Language (OWL), the de facto standard ontology language, is based on the family of Description Logics (DLs). In the last decades, many research attentions have been paid to complexity and reasoning algorithms of various dialects of the DLs such as $\mathcal{SROIQ}$, the underpinning of the OWL2-DL, and $\mathcal{EL}^{++}$, the underpinning of the OWL2-EL [9].

Most of these works focus on TBox reasoning such as deciding subsumption between concept expressions, or checking whether a particular concept is satisfiable. ABox reasoning such as deciding to which concepts a particular individual belongs is usually realised by extensions of TBox algorithms [6], or by being reduced to TBox reasoning [12]. Other works [7] reduces ontologies into disjunctive datalog to provide dedicated ABox reasoning. In either case, the reasoning complexity is high for expressive DL fragments. However, ABox can be encoded in very expressive DLs, rather large and changing frequently for which traditional solutions can not provide efficient answers.

To solve this problem, the approximation approaches have been studied and evaluated [10, 14, 3, 13, 14]. However, most of these works still replying on reasoners of expressive DLs. For example, [10] achieves efficient query answering by pre-computing the materialization of the original ontology with an OWL DL reasoner. [3, 13] are based

on a specific reasoner KAON2 [8], which supports DL $\mathcal{SHIQ}$. The applicability of these approaches are restricted by the capability of the heavy-weight reasoner and hence the reasoning complexity can not be substantially reduced.

In our early work [11] we presented an approximate reasoning approach to reduce TBox reasoning in DL $\mathcal{R}$ to that in DL $\mathcal{EL}^+$. The reasoning complexity is reduced from 2ExpTime-hard to PTime and the soundness of results is preserved. In this paper, we extend this approach to support ABox reasoning in DL $\mathcal{RO}$, i.e. DL $\mathcal{SHO}$ plus role chains. Given an $\mathcal{RO}$ ontology, we first approximate it into an $\mathcal{EL}^{++}$ ontology with a complement table (CT) maintaining the complementary relations between named concepts, then extend the $\mathcal{EL}^{++}$ reasoning with additional completion rules to entail logical consequence, for both TBox and ABox. This approach is tractable and soundness-preserving.

The rest of this paper is organised as follows: in Sec. 2 we briefly introduce the DL $\mathcal{RO}$ and $\mathcal{EL}^{++}$, and discuss the technical challenge of existing approximate reasoning approaches. In Sec. 3 we present our approach for approximate ABox reasoning, particularly, we show how the ABox approximate reasoning should be combined with the TBox approximate reasoning. In Sec. 4 we present some preliminary evaluation of our approach and Sec. 5 concludes the paper.

2 Technical Motivations

In [11] we presented an approach to approximating $\mathcal{R}$ TBox to $\mathcal{EL}^+$ TBox with an additional complement table. We note that $\mathcal{EL}^{++}$, an extension of $\mathcal{EL}^+$ that supports singletons, is also tractable. Thus it is natural to allow the using of nominals in the original ontology. This leads to the DL $\mathcal{RO}$.

In order to motivate our investigation on syntactic approximation of $\mathcal{RO}$ ontologies to $\mathcal{EL}^{++}$ ontologies, this section first briefly introduces $\mathcal{RO}$ and $\mathcal{EL}^{++}$ and then illustrates the technical challenges in their ABox reasoning and approximation.

In $\mathcal{RO}$, concepts C, D can be inductively composed with the following constructs:

$$\top \mid \bot \mid A \mid C \sqcap D \mid \exists r.C \mid \{a\} \mid \neg C$$

where $\top$ is the top concept, $\bot$ the bottom concept, A atomic concept, n an integer number, a an individual and r an atomic role. Conventionally, $C \sqcup D$ and $\forall R.C$ are used to abbreviate $\neg(\neg C \sqcap \neg D)$ and $\neg \exists R.\neg C$, respectively. Note that $\{a_1, a_2, \ldots, a_n\}$ can be regarded as abbreviation of $\{a_1\} \sqcup \{a_2\} \sqcup \ldots \sqcup \{a_n\}$. Without loss of generality, in what follows, we assume all the concepts to be in their negation normal forms (NNF)[1] and use ~C to denote the NNF of $\neg C$. We also call $\top, \bot, A, \{a\}$ *basic concepts* because they are not composed by other concepts or roles. Given a KB Σ, we use CN_Σ (RN_Σ, IN_Σ) to denote the set of basic concepts (atomic roles, individuals) in Σ.

Target language $\mathcal{EL}^{++}$ supports

$$\top \mid \bot \mid A \mid C \sqcap D \mid \exists r.C \mid \{a\}.$$

---

[1] An $\mathcal{RO}$ concept is in NNF *iff* negation is applied only to atomic concepts and singletons. NNF of a given concept can be computed in linear time[4].

Both $\mathcal{RO}$ and $\mathcal{EL}^{++}$ support concept inclusions (CIs, e.g. $C \sqsubseteq D$), role inclusions (RIs, e.g. $r \sqsubseteq s$, $r_1 \circ \ldots \circ r_n \sqsubseteq s$), class assertions (e.g. $a : C$) and role assertions (e.g. $(a, b) : r$). If $C \sqsubseteq D$ and $D \sqsubseteq C$, we write $C \equiv D$. If C is non-atomic, $C \sqsubseteq D$ is a general concept inclusion (GCI). For more details about syntax and semantics of DLs, we refer the readers to [2]. Given a set of axioms Σ (a single axiom α), its signature, denoted by $Sig(\Sigma)$ ($Sig(\alpha)$) is the set of all the concept names (including $\top$ and $\bot$), role names and individual names appearing in Σ (α).

Traditionally in expressive and very expressive DLs, ABox reasoning is performed together with the TBox by the tableau algorithm [6]. The tableau algorithm [5] constructs a tableau (as a witness of a model of the ontology) as a graph in which each node x represents an individual and is labeled with a set of concepts it must satisfy, each edge $\langle x, y \rangle$ represents a pair of individuals satisfying a role that labels the edge.

Instance checking $\Sigma \models a : C$ is reduced to knowledge base consistence for the extended knowledge base $\Sigma' = \Sigma \cup \{a : \neg C\}$ [12]. To test this, a tableau is initialised with the concept and role assertions in Σ' and is then expanded by repeatedly applying the completion rules. Similar to other reasoning services, tableau-based instance checking has to deal with the non-determinism of GCI, which results in an exponential blowup of the search space.

Reasoning with $\mathcal{EL}^{++}$ is more efficient. [1] presents a set of TBox completion rules (Table 1) [2] to compute, given a normalised $\mathcal{EL}^{++}$ TBox $\mathcal{T}$, for each $A \in CN_{\mathcal{T}}$, a subsumer set $S(A) \subseteq CN_{\mathcal{T}}$ in which for each $B \in S(A)$, $\mathcal{T} \models A \sqsubseteq B$, and for each $r \in RN_{\mathcal{T}}$, a relation set $R(r) \subseteq CN_{\mathcal{T}} \times CN_{\mathcal{T}}$ in which for each $(A, B) \in R(r)$, $\mathcal{T} \models A \sqsubseteq \exists r.B$. These sets are initialised as: for each $A \in CN_{\mathcal{T}}$, $S(A) = \{A, \top\}$ and for each $r \in RN_{\mathcal{T}}$, $R(r) = \emptyset$. Reasoning with rules **R1-R8** is tractable.

Table 1. $\mathcal{EL}^{++}$ completion rules (no datatypes)

R1	If $A \in S(X)$, $A \sqsubseteq B \in \mathcal{T}$ and $B \notin S(X)$ then $S(X) := S(X) \cup \{B\}$
R2	If $A_1, A_2, \ldots, A_n \in S(X)$, $A_1 \sqcap A_2 \sqcap \ldots \sqcap A_n \sqsubseteq B \in \mathcal{T}$ and $B \notin S(X)$ then $S(X) := S(X) \cup \{B\}$
R3	If $A \in S(X)$, $A \sqsubseteq \exists r.B \in \mathcal{T}$ and $(X, B) \notin R(r)$ then $R(r) := R(r) \cup \{(X, B)\}$
R4	If $(X, A) \in R(r)$ $A' \in S(A)$, $\exists r.A' \sqsubseteq B \in \mathcal{T}$ and $B \notin S(X)$ then $S(X) := S(X) \cup \{B\}$
R5	If $(X, A) \in R(r)$, $\bot \in S(A)$ and $\bot \notin S(X)$ then $S(X) := S(X) \cup \{\bot\}$
R6	If $\{a\} \in S(X) \cap S(A)$, $X \rightsquigarrow_R A$ and $S(A) \not\subseteq S(X)$ then $S(X) := S(X) \cup S(A)$
R7	If $(X, A) \in R(r)$, $r \sqsubseteq s \in \mathcal{T}$ and $(X, A) \notin R(s)$ then $R(s) := R(s) \cup \{(X, A)\}$
R8	If $(X, A) \in R(r_1)$, $(A, B) \in R(r_2)$, $r_1 \circ r_2 \sqsubseteq r_3 \in \mathcal{T}$, and $(X, B) \notin R(r_3)$ then $R(r_3) := R(r_3) \cup \{(X, B)\}$

---

[2] in **R6** $X \rightsquigarrow_R A$ iff there exists $C_1, \ldots, C_k \in CN_{\mathcal{T}}$ s.t. $C_1 = X$ or $C_1 = \{b\}$, $(C_j, C_{j+1}) \in R(r_j)$ for some $r_j \in RN_{\mathcal{T}}(1 \leq j \leq k)$ and $C_k = A$

When ABox $\mathcal{A}$ presents, an additional concept $C_{\mathcal{A}} := \prod_{a:C\in\mathcal{A}} \exists u.(\{a\} \sqcap C) \sqcap \prod_{(a,b):r\in\mathcal{A}} \exists u.(\{a\} \sqcap \exists r.\{b\})$, where u is a fresh role name, is introduced. To this end, instance checking $a : C$ can be reduced to subsumption checking $\{a\} \sqcap C_{\mathcal{A}} \sqsubseteq C$, which can be realised by **R1-R8**. However, this approach can not be directly applied on more expressive DLs.

To provide more scalable and efficient ABox reasoning service in expressive DLs, approximation approaches have been studied. However, most of these approaches heavily rely on existing reasoners. [14] presented approaches based on the idea of simplifying concept expressions in an ontology or a query to speed up the instance retrieval. The simplified ontology and query still needs to be processed by a heavy-weight reasoner, and the evaluation results showed that the number of subsumption tests can not always be reduced. *Semantic Approximation* [10] uses a heavy weighted reasoner to materialize the ontology and store in a database to speed up online query answering. But once the ABox changed, the entire procedure has to be performed again. [3, 13] present the SCREECH approach. It utilizes the KAON2 algorithm, which translates a $\mathcal{SHIQ}$ TBox into disjunctive datalog, and executes the rules together with a $\mathcal{SHIQ}$ ABox and a query by a datalog reasoning engine. By rewriting or eliminating all the disjunctive rules the data complexity can be reduced from coNP-complete of OWL DL to polynomial time. However this still relies on KAON2 to pre-translate the TBox. If the ontology is in a language beyond the capability of KAON2, e.g. $\mathcal{RO}$, this approach can not handle. Also, when the ABox contains complex concept expressions, this approach can not directly execute.

To sum up, tableau algorithms have difficulties to handle complex structured axioms; tractable DL algorithms can not support more expressive languages; while traditional approximation approaches still rely on existing DL reasoners. In what follows, we presented our approach which is motivated and inspired by these works, and show that it overcomes these difficulties.

3   The Approach

In this section, we first recall and extend the TBox approximation in [11] to support ontology approximation from $\mathcal{RO}$ to $\mathcal{EL}^{++}$. Then we discuss how the ABox can be reasoned internally and externally of the TBox. At the end, we discuss how the internal and external reasoning of ABox can possibly be integrated.

3.1   Approximate $\mathcal{RO}$ Ontologies to $\mathcal{EL}^{++}$

In approximation, we only consider concepts corresponding to the particular ontology in question. We use the notion *term* to refer to these "interesting" concept expressions. More precisely, a term is: (i) a concept expression in any axiom, or (ii) a singleton of any individual, or (iii) the complement of a term, or (iv) the syntactic sub-expression of a term. In order to represent terms that will be used in $\mathcal{EL}^{++}$ reasoning, we assign names to them.

Definition 1. *(Name Assignment) Given S a set of concept expressions, a name assignment fn is a function as for each $C \in S$, $fn(C) = C$ if C is a basic concept; otherwise, $fn(C)$ is a fresh name.*

Now we approximate an $\mathcal{RO}$ ontology to $\mathcal{EL}^{++}$ plus a complement table (CT). Its basic idea is to represent (non-$\mathcal{EL}^{++}$) terms with its name assignment:

Definition 2. *($\mathcal{EL}_{\mathcal{C}}^{++}$ Transformation) Given an $\mathcal{RO}$ Ontology $\mathcal{O} = (\mathcal{T}_{\mathcal{O}}, \mathcal{A}_{\mathcal{O}})$ and a name assignment fn, its $\mathcal{EL}_{\mathcal{C}}^{++}$ transformation $A_{fn,\mathcal{EL}_{\mathcal{C}}^{++}}(\mathcal{O})$ is a triple $(\mathcal{T}, \mathcal{A}, CT)$ constructed as follows:*

1. *$\mathcal{T}, \mathcal{A}$ and CT are all initialised as $\emptyset$.*
2. *for each $C \sqsubseteq D$ ($C \equiv D$) in $\mathcal{T}$, $\mathcal{T} = \mathcal{T} \cup \{fn(C) \sqsubseteq fn(D)\}$ ($\mathcal{T} = \mathcal{T} \cup \{fn(C) \equiv fn(D)\}$).*
3. *for each $\beta \in RI_{\mathcal{T}}$, add β into $\mathcal{T}$.*
4. *for each $a : C \in \mathcal{A}$, $\mathcal{A} = \mathcal{A} \cup \{a : fn(C)\}$.*
5. *for each $(a, b) : r \in \mathcal{A}$, $\mathcal{A} = \mathcal{A} \cup \{(a, b) : r\}$.*
6. *for each term C in $\mathcal{O}$, $CT = CT \cup \{(fn(C), fn(\sim C))\}$, and*
   - *(a) if C is the form $C_1 \sqcap \ldots \sqcap C_n$, then $\mathcal{T} = \mathcal{T} \cup \{fn(C) \equiv fn(C_1) \sqcap \ldots \sqcap fn(C_n)\}$,*
   - *(b) if C is the form $\exists r.D$, then $\mathcal{T} = \mathcal{T} \cup \{fn(C) \equiv \exists r.fn(D)\}$,*
   - *(c) otherwise $\mathcal{T} = \mathcal{T} \cup \{fn(C) \sqsubseteq \top\}$.*

Step 2 rewrites all the concept axioms; Step 3 preserves all the $\mathcal{EL}^{++}$ role axioms; Step 4 and 5 rewrite all the ABox axioms; Step 6 defines all the $\mathcal{EL}^{++}$ terms and constructs the complement table CT. We call this procedure an $\mathcal{EL}_{\mathcal{C}}^{++}$ *approximation*.The $\mathcal{EL}_{\mathcal{C}}^{++}$ approximation approximates an $\mathcal{RO}$ ontology into an $\mathcal{EL}^{++}$ ontology with a table maintaining the complements of all the basic concepts in linear time:

Proposition 1. *($\mathcal{EL}_{\mathcal{C}}^{++}$ Approximation) For an Ontology $\mathcal{O}$, let $A_{fn,\mathcal{EL}_{\mathcal{C}}^{++}}(\mathcal{O}) = (\mathcal{T}, \mathcal{A}, CT)$, we have: (1)$(\mathcal{T}, \mathcal{A})$ is an $\mathcal{EL}^{++}$ ontology; (2) $\mathcal{A}$ only contains basic concepts of $\mathcal{T}$; (3) for each $A \in CN_{\mathcal{T}}$, there exists $(A, B) \in CT$; (4) if $(A, B) \in CT$ then $A, B \in CN_{\mathcal{T}}$ and $(B, A) \in CT$.*

Proposition 2. *For any Ontology $\mathcal{O} = (\mathcal{T}_{\mathcal{O}}, \mathcal{A}_{\mathcal{O}})$ and $(\mathcal{T}, \mathcal{A}, CT)$ its $\mathcal{EL}_{\mathcal{C}}^{++}$ transformation, if $\mathcal{O}$ contains $n_{\mathcal{O}}$ terms, then $|\mathcal{T}| \leq n_{\mathcal{O}} + |\mathcal{T}_{\mathcal{O}}|$, $|\mathcal{A}| = |\mathcal{A}_{\mathcal{O}}|$ and $|CT| = n_{\mathcal{O}}$, where $|\mathcal{T}|(|\mathcal{A}|, |\mathcal{T}_{\mathcal{O}}|, |\mathcal{A}_{\mathcal{O}}|)$ is the number of axioms in $\mathcal{T}(\mathcal{A}, \mathcal{T}_{\mathcal{O}}, \mathcal{A}_{\mathcal{O}})$ and $|CT|$ is the number of pairs in CT.*

Given an $\mathcal{EL}_{\mathcal{C}}^{++}$ transformation $(\mathcal{T}, \mathcal{A}, CT)$, we normalise axioms of form $C \sqsubseteq D_1 \sqcap \ldots \sqcap D_n$ into $C \sqsubseteq D_1, \ldots, C \sqsubseteq D_n$, and recursively normalise role chain $r_1 \circ \ldots \circ r_n \sqsubseteq s$ with $n > 2$ into $r_1 \circ \ldots \circ r_{n-1} \sqsubseteq u$ and $u \sqsubseteq s$. This procedure can be done in linear time. In the following, we assume $\mathcal{T}$ to be always normalised. For convenience, we use a *complement function* $fc : CN_{\mathcal{T}} \mapsto CN_{\mathcal{T}}$ as: for each $A \in CN_{\mathcal{T}}$, $fc(A) = B$ such that $(A, B) \in CT$.

3.2 ABox Internalisation and Reasoning

Once we are able to approximate an $\mathcal{RO}$ ontology into $\mathcal{EL}^{++}$, it is straightforward to perform ABox reasoning by internalising the ABox into TBox. This can be done as in classical $\mathcal{EL}^{++}$ (cf. Sec. 2) by encoding the ABox as a concept. However this approach will introduce additional concept names in the normalisation phase, thus complicates the reasoning. Alternatively, we can do the following internalisation:

Definition 3. *($\mathcal{EL}_\mathcal{C}^{++}$ ABox Internalisation) Given an $\mathcal{RO}$ ontology $\mathcal{O}$, let $A_{fn,\mathcal{EL}_\mathcal{C}^{++}}(\mathcal{O}) = (\mathcal{T}', \mathcal{A}', CT')$, its $\mathcal{EL}_\mathcal{C}^{++}$ ABox internalisation $AI(A_{fn,\mathcal{EL}_\mathcal{C}^{++}}(\mathcal{O}))$ is a triple $(\mathcal{T}, \emptyset, CT)$ constructed as follows:*

1. *$\mathcal{T}$ is initialised as $\mathcal{T}'$.*
2. *$CT = CT'$.*
3. *for each $a : C \in \mathcal{A}'$, $\mathcal{T} = \mathcal{T} \cup \{\{a\} \sqsubseteq C\}$.*
4. *for each $(a, b) : r \in \mathcal{A}'$, $\mathcal{T} = \mathcal{T} \cup \{\{a\} \sqsubseteq \exists r.\{b\}\}$.*

It's easy to show that such internalisation can be constructed in linear time and the triple $(\mathcal{T}, \emptyset, CT)$ still satisfy Proposition 1. Also, $\mathcal{T}$ is normalised if $\mathcal{T}'$ normalised. To this end, we reduce ABox reasoning to TBox reasoning on $\mathcal{T}$. To utilize the complementary relations in CT, we propose additional completion rules (Table 2) to $\mathcal{EL}^{++}$.

Table 2. Complement completion rules

R9	If $A, B \in S(X)$, $A = fc(B)$ and $\bot \notin S(X)$ then $S(X) := S(X) \cup \{\bot\}$
R10	If $A \in S(B)$ and $fc(B) \notin S(fc(A))$ then $S(fc(A)) := S(fc(A)) \cup \{fc(B)\}$
R11	If $A_1 \sqcap \ldots \sqcap A_i \sqcap \ldots \sqcap A_n \sqsubseteq \bot$, $A_1, \ldots, A_{i-1}, A_{i+1}, \ldots, A_n \in S(X)$ and $fc(A_i) \notin S(X)$ then $S(X) := S(X) \cup \{fc(A_i)\}$

R9 realises axiom $A \sqcap {\sim}A \sqsubseteq \bot$. **R10** realises $A \sqsubseteq B \rightarrow {\sim}A \sqsubseteq {\sim}B$. **R11** builds up the relations between conjuncts of a conjunction, *e.g.* $A \sqcap B \sqsubseteq \bot$ implies $A \sqsubseteq {\sim}B$. The reasoning is tractable and soundness-preserving:

Theorem 1. *(Complexity) For any $\mathcal{EL}_\mathcal{C}^{++}$ internalisation $(\mathcal{T}, \emptyset, CT)$ ($\mathcal{T}$ normalised), TBox reasoning by **R1**-**R11** will terminate in polynomial time w.r.t. $|CN_\mathcal{T}| + |RN_\mathcal{T}|$.*

Theorem 2. *(Concept Subsumption Checking) Given an $\mathcal{RO}$ ontology $\mathcal{O} = (\mathcal{T}_\mathcal{O}, \mathcal{A}_\mathcal{O})$, its vocabulary $V_\mathcal{O}$ and $AI(A_{fn,\mathcal{EL}_\mathcal{C}^{++}}(\mathcal{O})) = (\mathcal{T}, \emptyset, CT)$, for any two concepts C and D constructed from $V_\mathcal{O}$, if $AI(A_{fn,\mathcal{EL}_\mathcal{C}^{++}}(\{C \sqsubseteq \top, D \sqsubseteq \top\})) = (\mathcal{T}', \emptyset, CT')$, then $\mathcal{O} \models C \sqsubseteq D$ if $fn(D) \in S(fn(C))$ can be computed by rules **R1**-**R11** on $(\mathcal{T} \cup \mathcal{T}', \emptyset, CT \cup CT')$.*

Concering ABox reasoning, this indicates that, $\mathcal{O} \models a : C$ if $fn(C) \in S(\{a\})$ can be computed. And $\mathcal{O} \models (a, b) : r$ if $fn(\exists r.\{b\}) \in S(\{a\})$ can be computed. When C is a term of $\mathcal{O}$, such computation can be performed directly on $(\mathcal{T}, \emptyset, CT)$.

3.3 TBox-irrelevant ABox Completion

The ABox internalisation absorbs the entire ABox into the TBox and reduce ABox reasoning to TBox reasoning. However, this approach has its limitation: (i) according to Theorem 1 when a large amount of individuals present, more computations are needed (individuals are converted into singletons); (ii) it yields some results useless in TBox and ABox reasoning. For example, $A \sqsubseteq \exists r.B, x : A$ will yield $(\{x\}, B) \in R(r)$ by **R3**. To optimize the performance we separate the reasoning of TBox and ABox.

We start from a simpler case, in which the approximated TBox contains no nominal. In this case, the ABox reasoning has no effect on the TBox reasoning, which can thus be pre-computed.

Given $A_{fn,\mathcal{EL}_{\mathcal{C}}^{++}}(\mathcal{O}) = (\mathcal{T}, \mathcal{A}, CT)$, after TBox reasoning of **R1-R11**, we present ABox completion rules (Table 3) to compute, for each $a \in IN_\mathcal{A}$, a class set $C(a) \subseteq CN_\mathcal{T} \cup CN_\mathcal{A}$ in which for each $A \in C(a)$, $\mathcal{T}, \mathcal{A} \models a : C$, and for each $r \in RN_\mathcal{T} \cup RN_\mathcal{A}$, a role set $RO(r) \subseteq IN_\mathcal{A} \times IN_\mathcal{A}$ in which for each $(a, b) \in RO(r)$, $\mathcal{T}, \mathcal{A} \models (a, b) : r$. These sets are initialised as: $A \in C(a)$ if $a : A \in \mathcal{A}$, $(a, b) \in RO(r)$ if $(a, b) : r \in \mathcal{A}$.

Table 3. TBox-independent $\mathcal{EL}_{\mathcal{C}}^{++}$ ABox completion rules (no datatypes)

AR1	If $A \in C(x)$, $B \in S(A)$ and $B \notin C(x)$ then $C(x) := C(x) \cup \{B\}$
AR2	If $A_1, A_2, \ldots, A_n \in C(x)$, $A_1 \sqcap A_2 \sqcap \ldots \sqcap A_n \sqsubseteq B \in \mathcal{T}$ and $B \notin C(x)$ then $C(x) := C(x) \cup \{B\}$
AR3	If $(x, y) \in RO(r)$ $A \in C(y)$, $\exists r.A \sqsubseteq B \in \mathcal{T}$ and $B \notin C(x)$ then $C(x) := C(x) \cup \{B\}$
AR4	If $(x, y) \in RO(r)$, $\bot \in C(y)$ and $\bot \notin C(x)$ then $C(x) := C(x) \cup \{\bot\}$
AR5	If $(x, y) \in RO(r)$, $r \sqsubseteq s \in \mathcal{T}$ and $(x, y) \notin RO(s)$ then $RO(s) := RO(s) \cup \{(x, y)\}$
AR6	If $(x, y) \in RO(r_1)$, $(y, z) \in RO(r_2)$, $r_1 \circ r_2 \sqsubseteq r_3 \in \mathcal{T}$, and $(x, y) \notin RO(r_3)$ then $RO(r_3) := RO(r_3) \cup \{(x, z)\}$
AR7	If $A, B \in C(x)$, $A = fc(B)$ and $\bot \notin C(x)$ then $C(x) := C(x) \cup \{\bot\}$
AR8	If $A \in C(x)$, $fc(A) \in S(fc(B))$ and $B \notin C(x)$ then $C(x) := C(x) \cup \{B\}$
AR9	If $A_1 \sqcap \ldots \sqcap A_i \sqcap \ldots \sqcap A_n \sqsubseteq \bot$, $A_1, \ldots, A_{i-1}, A_{i+1}, \ldots, A_n \in C(x)$ and $fc(A_i) \notin C(x)$ then $C(x) := C(x) \cup \{fc(A_i)\}$

It's easy to see that **AR1-AR6** are for $\mathcal{EL}^{++}$ while **AR7-AR9** are for $\mathcal{EL}_{\mathcal{C}}^{++}$ transformation. More precisely, **AR1-AR2** are analogues to **R1-R2**, **AR3-AR4** analogues to **R4-R5**, **AR5-AR6** analogues to **R7-R8**, **AR7-AR9** analogues to **R9-R11**. This indicates that similar algorithms can be applied and tractability and soundness are preserved. The fewer completion rules shall result in more efficient inference. Furthermore, **AR5-AR6** can be processed ahead of the other rules because other rules will not

generate any RO sets elements. This should further improve the efficiency and scalability. This TBox-ABox-separated approach should have the same results as the ABox-internalised approach.

3.4   Integrated ABox Approximation

ABox completion presented in Sec.3.3 has a restriction that the approximated TBox should contain no nominal. For example, $a : A, a : B, B \sqsubseteq \{b\}$, with internalisation we will infer $b : A$, which can not be computed by **AR1-AR9**. The question arises that whether the internalisation and ABox completion approach can be combined. In this section we discuss the possibility of relaxing the nominal-free restriction.

Our basic idea is to partition the approximated ABox $\mathcal{A}$ into two disjoint-union subsets $\mathcal{A}_I$ and $\mathcal{A}_E$ so that $\mathcal{A}_I$ should be internalised into the approximated TBox $\mathcal{T}$ to obtain an extended TBox $\mathcal{T}_I$ that can be classified by **R1-R11**, while $\mathcal{A}_E$ can be completed by **AR1-AR9** after TBox reasoning over $\mathcal{T}_I$. There can be different partitioning strategies. In what follows, we present a reachability-based partitioning. In general, any ABox axiom that contains concept or individual name that is directly or indirectly reachable to some nominal, should be internalised.

Definition 4. *(Nominal-reachable Signature) Let $A_{fn,\mathcal{EL}_{\mathcal{C}}^{++}}(\mathcal{O}) = (\mathcal{T}, \mathcal{A}, CT)$, its* nominal-reachable signature $Sig_{NR}(A_{fn,\mathcal{EL}_{\mathcal{C}}^{++}}(\mathcal{O}))$ ($Sig_{NR}(\mathcal{O})$ for short) is a minimal subset of $Sig(\mathcal{T}) \cup Sig(\mathcal{A})$ having the following properties:

1. *for any $a \in IN_{\mathcal{T}}$, we have $a \in Sig_{NR}(\mathcal{O})$.*
2. *for any $A \in CN_{\mathcal{T}}$, we have $A, fc(A) \in Sig_{NR}(\mathcal{O})$ if there exists $C \sqsubseteq D \in \mathcal{T}$ s.t. $A \in Sig(C)$ and $Sig(D) \cap Sig_{NR}(\mathcal{O}) \neq \emptyset$.*
3. *for any $a \in IN_{\mathcal{T}}$, we have $a \in Sig_{NR}(\mathcal{O})$ if there exists $a : A \in \mathcal{A} ((a,b) : r \in \mathcal{A})$ s.t. $A \in Sig_{NR}(\mathcal{O}) (b \in Sig_{NR}(\mathcal{O}))$.*
4. *for any $A \in CN_{\mathcal{A}}$, $A, fc(A) \in Sig_{NR}(\mathcal{O})$ if there exists $a : A \in \mathcal{A}$ s.t. $a \in Sig_{NR}(\mathcal{O})$.*

Then the ABox can be partitioned into two parts, one's signature is nominal-reachable, the other's not. The nominal-reachable part of the ABox should be internalised into the TBox:

Definition 5. *(Reachability-based Internalisation) Given an $\mathcal{RO}$ ontology $\mathcal{O}$ and $A_{fn,\mathcal{EL}_{\mathcal{C}}^{++}}(\mathcal{O}) = (\mathcal{T}', \mathcal{A}', CT')$, its* reachability-based internalisation $RbI(A_{fn,\mathcal{EL}_{\mathcal{C}}^{++}}(\mathcal{O}))$ *is a triple $(\mathcal{T}, \mathcal{A}, CT)$ constructed as follows:*

1. $CT = CT'$.
2. *let $\mathcal{A}_I = \{\alpha \in \mathcal{A}' | Sig(\alpha) \cap Sig_{NR}(\mathcal{O}) \neq \emptyset\}$, and $AI((\mathcal{T}', \mathcal{A}_I, CT)) = (\mathcal{T}_I, \emptyset, CT)$, then*
   - $\mathcal{A} = \mathcal{A}' \setminus \mathcal{A}_I$.
   - $\mathcal{T} = \mathcal{T}_I$.

For example, let $\mathcal{T}_1 = \{B \sqsubseteq \{b\}\}$ and $\mathcal{A}_2 = \{a : A, a : B\}$, then both of the ABox axioms should be internalised. Similarly, let $\mathcal{T}_2 = \{A \sqsubseteq \{a\}, \exists r.B \sqsubseteq C\}$ and $\mathcal{A}_2 = \{b : B, (a, b) : r\}$, then the entire ABox should be internalised as well so that $A \sqsubseteq C$ can be inferred. Given $RbI(A_{fn,\mathcal{EL}_\mathcal{C}^{++}}(\mathcal{O})) = (\mathcal{T}, \mathcal{A}, CT)$, the reasoning can be performed as follows:

1. classify $(\mathcal{T}, \emptyset, CT)$ by **R1-R11**.
2. extend $\mathcal{A}$ as $\mathcal{A} = \mathcal{A} \cup \{a : A | A \in S(\{a\})\} \cup \{(a, b) : r | (\{a\}, \{b\}) \in R(r)\}$.
3. reason $(\mathcal{T}, \mathcal{A}, CT)$ by **AR1-AR9**.

This integration approach of internalisation and ABox completion should have the same results as the the internalisation approach. The tractability of the reasoning is also preserved.

There could be other way of partitioning the ABox. The advantage of our proposal is that it is purely syntactic thus can be performed efficiently.

4   Evaluation

We implemented the internalisation approach (cf. Sec.3.2) and the ABox completion approach (cf. Sec.3.3) separately in our REL reasoner. In our experiments, `REL_int` system implemented the approximation (Sec.3.1) and the internalisation approach; `REL_ext` implements the approximation and the ABox completion approach. To evaluate their performance in practice, we compared with mainstream reasoners Pellet 2.0.1 and FaCT++ 1.3.0.1. All experiments were conducted in an environment of Windows XP SP3 with 2.66 GHz CPU and 1G RAM allocated to JVM 1.6.0.07.

Our test suite consists of several real world or benchmark ontologies with various size and expressivity of TBox and ABox. The VICODI [3] ontology is developed to represent the history of Europe. SEMINTEC [4] ontology is developed for semantic web mining. WINE [5] ontology is an OWL-DL show case ontology, designed to exploit the expressive power of OWL-DL. LUBM (Lehigh University Benchmark) [6] is a benchmark for OWL-Lite query answering. VICODI and SEMINTEC have relatively simple TBox but large ABox. WINE has a rather complex TBox and a moderate ABox. LUBM contains a moderately complex TBox and its ABox can be generated as large as needed. In our evaluation, we generated 1 university. Only WINE has nominals. We also converted datatype properties into object properties. As for WINE ontology, the expressivity is beyond $\mathcal{RO}$, but REL directly approximate those constructs, e.g. cardinality restrictions, inverse roles, with names.

For each ontology, we retrieve the types of all the individuals and the relations between all pairs of individuals. Each reasoner was given 10 minutes on each task. Recall of REL is calculated against the others. Thus the time shown in our evaluation includes approximation time (for REL), reasoning time, type and relation retrieval and

---

[3] http://www.vicodi.org/about.htm

[4] http://www.cs.put.poznan.pl/alawrynowicz/semintec.htm

[5] http://www.w3.org/TR/owl-guide/wine.rdf

[6] http://swat.cse.lehigh.edu/projects/lubm/

counting time. Time unit is second. `REL_ext` is tested for all the ontologies. `REL_int` is tested only for WINE ontology as the others contain no nominal. The results are presented in Table 4, in which "e/o" indicates that the reasoner exited with an error; "t/o"indicates that the reasoner failed to finish the task in 10 minutes.

Table 4. Evaluation Results

Ontology	retrieval	Pellet	FaCT++	`REL_ext`		`REL_int`	
				time	recall	time	recall
VICODI	concept	7.828	17.515	3.86	100%	-	-
	role	9.656	t/o	3.828	100%	-	-
SEMINTEC	concept	4.5	4.422	2.484	100%	-	-
	role	7.062	t/o	2.485	100%	-	-
WINE	concept	17.813	e/o	1.266	85.1%	1.641	98.2%
	role	26.9	e/o	1.266	40.0%	1.453	91.5%
LUBM$\times$1	concept	10.937	17.359	8.531	100%	-	-
	role	26.891	t/o	8.625	100%	-	-

As we can see from Tabel 4. REL is (2 times to more than 20 times) faster than all the other reasoners on all the ontologies, which indicates an improvement on the efficiency of reasoning. For simple ontologies such as VICODI, SEMINTEC and LUBM, the advantage of REL is not significant. While when the ontology has a relatively complex TBox, especially when the TBox and ABox are connected, the benefits of approximate reasoning become substantial. Note that for these two tasks, the time of REL was almost the same: because our completion rules compute the instances of all the atomic concepts and atomic roles together.

Concerning the completeness, when the ontology is simple, the recall of `REL_ext` is 100%. When the ontology TBox gets complex and contains nominals. Separate reasoning of TBox and ABox becomes not satisfying. By internalising the ABox into TBox the recall was significantly improved. It's interesting to see that the time of `REL_int` was not much longer than `REL_ext` on the WINE. That is because WINE ontology contains about 208 individuals, which is not a large number. It will be necessary to implement the integrated solution as we discussed in Sec.3.4, when the ABox goes large and TBox contains nominals.

To sum up, the evaluation showed that even naive implementations of our approach can provide efficient and rather complete ABox reasoning services. Particularly, when the ontology is complex and large, the efficiency can still be retained while the completeness is not sacrificed too much.

5 Conclusion & Future Work

In this paper, we presented an approximate reasoning approach to address the issue of ABox reasoning over ontologies of expressive DL $\mathcal{RO}$, a fragment of OWL2-DL supporting $\mathcal{ALC}$ GCIs, nominals and role chains. Our approach first approximates an

$\mathcal{RO}$ ontology to an $\mathcal{EL}^{++}$ ontology plus a complement table (CT) maintaining the complementary relations between named concepts (including $\top$ and $\bot$) and singletons. Then we presented an internalisation approach to reducing ABox reasoning into TBox reasoning, and presented additional completion rules to utilize the CT. For ontology with no nominal in TBox, we presented an optimized ABox Completion approach. We further discussed the possibility of combining the two approaches and presented one of the possible solution.

Our approximate reasoning strategy is soundness-preserving and can be realised in PTIME. Although we don't guarantee completeness, our preliminary evaluation showed that naive implementations of our approach can improve the efficiency of reasoning over real world and benchmark ontologies, while maintaining a high recall.

In the future, we would like to further improve the completeness by exploiting more reasoning patterns, to future improve the scalability by combining with relational databases, to further improve the efficiency by optimising the implementations. The lack of expressive benchmark in our evaluation also motivates us creating our own benchmarks.

References

1. F. Baader, S. Brandt, and C. Lutz. Pushing the $\mathcal{EL}$ Envelope. In *Proceedings IJCAI-05*, 2005.
2. Franz Baader, Diego Calvanese, Deborah L. McGuinness, Daniele Nardi, and Peter F. Patel-Schneider, editors. *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press, 2003.
3. Pascal Hitzler and Denny Vrandecic. Resolution-Based Approximate Reasoning for OWL DL. In *Proceedings of ISWC 2005*, 2005.
4. Bernhard Hollunder, Werner Nutt, and Manfred Schmidt-Schau. Subsumption Algorithms for Concept Description Languages. In *ECAI-90*, pages 348–353. Pitman Publishing, 1990.
5. Ian Horrocks, Ulrike Sattler, and Stephan Tobies. Practical Reasoning for Very Expressive Description Logics. *Logic Journal of the IGPL*, 8:2000, 2000.
6. Ian Horrocks, Ulrike Sattler, and Stephan Tobies. Reasoning with individuals for the description logic shiq. In *CADE-17*, 2000.
7. Ullrich Hustadt, Boris Motik, and Ulrike Sattler. Reducing SHIQ-Description Logic to Disjunctive Datalog Programs. In *KR 2004*, pages 152–162, 2004.
8. B. Motik. Practical DL Reasoning over Large ABoxes with KAON2. 2006.
9. Boris Motik, Bernardo Cuenca Grau, Ian Horrocks, Zhe Wu, Achille Fokoue, and Carsten Lutz. OWL 2 Web Ontology Language: Profiles. W3c working draft, W3C, October 2008.
10. Jeff Z. Pan and Edward Thomas. Approximating OWL-DL Ontologies. In *AAAI-2007*, pages 1434–1439, 2007.
11. Yuan Ren, Jeff Z. Pan, and Yuting Zhao. Soundness Preserving Approximation for TBox Reasoning in R. In *Description Logics 2009*, 2009.
12. A. Schaerf. Reasoning With Individuals in Concept Languages. *Data and Knowledge Engineering*, 13(2):141–176, 1994.
13. Tuvshintur Tserendorj, Sebastian Rudolph, Markus Krötzsch, and Pascal Hitzler. Approximate OWL-Reasoning with Screech. In *Proceedings of RR 08*, 2008.
14. Holger Wache, Perry Groot, and Heiner Stuckenschmidt. Scalable Instance Retrieval for the Semantic Web by Approximation. In *WISE Workshops-05*, 2005.

TBox Classification in Parallel: Design and First Evaluation

Mina Aslani and Volker Haarslev

Concordia University, Montreal, Canada
{m_aslani,haarslev}@cse.concordia.ca

Abstract. One of the most frequently used inference services of description logic reasoners classifies all named classes of OWL ontologies into a subsumption hierarchy. Due to emerging OWL ontologies from the web community consisting of up to hundreds of thousand of named classes and the increasing availability of multi-processor and multi- or many-core computers, we extend the work on parallel TBox classification and propose a new algorithm that is sound and complete and demonstrates in a first experimental evaluation a low overhead in the number of subsumption tests due to parallel execution.

1   Motivation

Due to the recent popularity of OWL ontologies in the web one can observe a trend toward the development of very large or huge OWL-DL ontologies. For instance, well known examples from the bioinformatics or medical community are UMLS, GALEN or even ontologies with more concepts. Some (versions) of the ontologies consist of more than hundreds of thousands of named concepts/classes and have become challenging even for the most advanced and optimized description logic (DL) reasoners. Although specialized DL reasoners for certain sublogics (e.g., CEL for EL++) and OWL-DL reasoners such as FaCT++, Pellet, HermiT, or RacerPro could demonstrate impressive speed enhancements due to newly designed optimization techniques, one can expect the need for parallelizing description logic inference services in the near future in order to achieve a web-like scalability. Our research is also strongly motivated by recent trends in computer hardware where processors feature multi-cores (2 to 8 cores) or many-cores (tens or even hundreds of cores). These processors promise significant speed-ups for algorithms exploiting so-called thread-level parallelism. This type of parallelism is very promising for DL reasoning algorithms that can be executed in parallel but might share common data structures (e.g., and/or parallelism in proofs, classification of TBoxes, ABox realization or query answering).

First approaches on more scalable reasoning algorithms for ABoxes (sets of declarations about individuals) were investigated with the Racer architecture [11] where novel instance retrieval algorithms were developed and analyzed, which exploit a variety of techniques such as index maintenance, dependency analysis, precompletion generation, etc. Other research focused on scalable ABox reasoning with optimization techniques to partition ABoxes into independent parts

and/or creating condensed (summary) ABoxes [8, 9, 6]. These approaches rely on the observation that the structure of particular ABoxes is often redundant and these ABoxes contain assertions not needed for ABox consistency checking or query answering.

Parallel algorithms for description logic reasoning were first explored in the FLEX system [3] where various distributed message-passing schemes for rule execution were evaluated. The only other approach on parallelizing core description logic reasoning [13] reported promising results using multi-core/processor hardware, where the parallel treatment of disjunctions and individual merging (due to number restrictions) is explored. While there exists some work on parallel DL algorithms, on parallel reasoning for first-order theorem proving (with completely different proof techniques based on resolution), and on parallel distributed RDF inferencing (e.g., [16]), parallel TBox classification has only been addressed in [1]. There has also been substantial work on reasoning through modularity and partitioning knowledge bases (e.g., [7, 5, 4]) that might be applicable to our work.

In the following we extend the work on parallel TBox classification [1] and propose a new algorithm that is sound and complete although it runs in parallel. The implemented prototype system simulates parallel TBox classification with various parameters such as number of threads, size of partitions assigned to threads, etc. First results from a preliminary evaluation look very promising and indicate a very low overhead in the number of subsumption tests due to parallel execution.

2   The New Parallel TBox Classifier

This section describes the architecture of the implemented system and its underlying *sound and complete* algorithm for parallel classification of DL ontologies. To compute the hierarchy in parallel, we developed a simulator using a multi-threaded architecture providing control parameters such as number of threads, number of concepts (also called partition size) to be inserted per thread, and strategies used to partition a given set of concepts. The simulator reads an input file containing a list of concept names to be classified and information about them. The per-concept information available in the file includes its name, parents (in the complete taxonomy), told subsumers, told disjoints, and pseudo model information. The information about parents is used to compute the set of ancestors and descendants of a concept. Told information consists of subsumers and disjoints that can be easily extracted from axioms without requiring proof procedures, e.g. the axiom $A \sqsubseteq B \sqcap \neg C$ would result in information asserting B as told subsumer of A and C as told disjoint of A. With the exception of told subsumers this information is only used for (i) emulating a tableau subsumption test, i.e., by checking whether a possible subsumer (subsumee) is in the list of ancestors (descendants) of given concept, and (ii) in order to verify the completeness of the taxonomy computed by the parallel classifier. The input information substitutes for an implemented tableaux reasoning procedure, hence

makes the parallel classifier independent of a particular DL logic or reasoner. Currently, Racer is used to generate this file for a given OWL-DL ontology after performing TBox classification.

The told subsumer information is passed to a preprocessing algorithm which creates a taxonomy skeleton based on the already known (i.e. told) subsumptions and generates a topological-order list (e.g. depth-first traversal). Using a topological sorting algorithm, the partial order can be serialized such that a total order between concept names (or sets of concept names) is defined. During classification, the concept names are processed in the order of the topological order. In our topological order list, from left to right, parent concepts precede child concepts. To manage concurrency in our system, at least two shared-memory approaches could be taken into account by using either (i) sets of local trees (so-called ParTree approach) or (ii) one global tree. In the ParTree algorithm [14] a local tree would be assigned to each thread, and after all the threads have finished the construction of their local hierarchy, the local trees need to be merged into one global tree. TBox classification through a local tree algorithm would not need any communication or synchronization between the threads. ParTree is well suited for distributed systems which do not have shared memory. The global tree approach was chosen because it implements a shared space which is accessible to different threads running in parallel and avoids the large scale overhead of ParTree on synchronizing local trees. To ensure data integrity a lock mechanism for single nodes is used. This allows a proper lock granularity and helps to increase the number of simultaneous write accesses to the subsumption hierarchy under construction.

Most TBox classification algorithms are based on two (symmetric) tasks (e.g., see [2]). The first phase (top search) determines the parents of a given concept to be inserted into the subsumption tree. It starts with the top concept ($\top$) and tries to push the given concept below the children of the current concept and repeats this process with the goal to push the given concept as much to the bottom of the subsumption tree as possible. Whenever a concept in the tree subsumes the given concept, it is pushed below this subsumer. The second phase (bottom search) determines the children of a given concept. It starts with the bottom concept ($\bot$) and tries to move the given concept above the parents of the current concept and repeats this process with the goal to move the current concept up in the tree as much as possible. Whenever a concept in the tree is subsumed by the given concept, it is moved above of this subsumee. Eventually, the given concept is correctly positioned in the current subsumption hierarchy. Both phases tag nodes of the tree ('visited', 'positive', 'negative') to prune the search and avoid visiting already processed nodes. For instance, 'positive' is used to tag nodes already known as (told) subsumers and 'negative' for already known as (told) disjoints.

The work in [2] is an example for algorithms that incrementally construct a subsumption tree and are highly optimized for sequential execution. In [10] some of these techniques were extended to better deal with huge TBox hierarchies but these algorithms are still based on a sequential execution. A recent approach
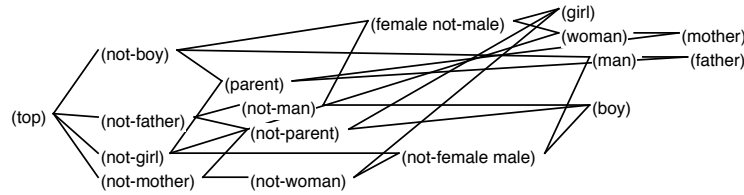
Fig. 1. Complete subsumption hierarchy for yaya-1
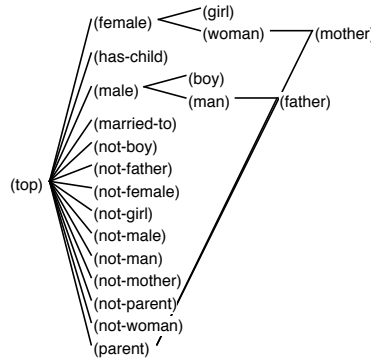
Fig. 2. Told subsumer hierarchy for yaya-1

[15] on TBox classification exploits partial information about OWL subclass relationships to reduce the number of subsumption tests and, thus, improves the algorithms presented in [2].

2.1 Example Scenario

In [1] the degree of incompleteness caused by classifying partitions of concepts in parallel was tested. For a variety of ontologies it turned out that a surprisingly few number of subsumptions were missed. This motivated the work in this paper. In the following we illustrate two scenarios which may cause that a concept is misplaced in the taxonomy due to parallel classification. We use a very small ontology named yaya-1 with 16 concepts (see Fig. 1 and 2).

For this example, we configured our system so that it runs with 4 threads and 3 number-of-tasks-per-thread. As explained previously, in parallel classification the topological sort order divides concept partitions between the threads (e.g. round-robin). For instance, in Fig. 3 a list of concepts allocated to each thread is

thread#1 ⟶ (female not-male), girl, parent
thread#2 ⟶ woman, mother, (male not-female)
thread#3 ⟶ man, boy, father
thread#4 ⟶ not-boy, not-father, not-girl
thread#1 ⟶ not-man, not-mother, not-parent, not-woman

Fig. 3. Concept assignments to each thread for classifying yaya-1

shown. The two possible scenarios that may lead to a situation where the correct place of a concept in the hierarchy is overlooked are described as follows.

Scenario I: In top search, as the new concept is pushed downward, right after the children of the current concept have been processed, at least one new child is added by another thread. In this scenario, the top search for the new concept is not aware of the recent change and this might cause missing subsumptions if there is any interaction between the new concept and the added children. The same might happen in bottom search if the bottom search for the new concept is not informed of the recent change to the list of parents of the current node.

Scenario II: Between the time that top search has been started to find the location of a new concept in the taxonomy and the time that its location has been decided, another thread has placed at least one concept into the hierarchy which the new concept has an interaction with. Again, this might cause missing subsumptions and is analogously also applicable to bottom search.

In our example (yaya-1), due to the small size of the taxonomy, scenario I was not encountered, however, scenario II occurred in our experiments because thread#1 inserted (female not-male)[1] and thread#2 added woman independently into the taxonomy and due to the parallelism each thread did not have any information regarding the latest concept insertion by other threads (see also Fig. 3). Hence, both (female not-male) and woman were initially placed under the top concept although woman should be a child of (female not-male) (see Fig. 1). This was discovered and corrected by executing lines 6-7, 16-17, and 25-36 in Algorithm 2 as shown below.

2.2 Algorithms for Parallel Classification

The procedure parallel_tbox_classification is sketched in Algorithm 1. It is called with a list of named concepts and sorts them in topological order w.r.t. to the initial taxonomy created from the already known told ancestors and descendants of each concept (using the told subsumer information). The classifier assigns in a round-robin manner partitions with a fixed size from the concept list to idle threads and activates these threads with their assigned partition using the procedure insert_partition outlined in Algorithm 2. All threads work in parallel with the goal to construct a global subsumption tree (taxonomy). They also share a global array *inserted_concepts* indexed by thread identifications. Nodes in the global tree as well as entries in the array will be locked for modification.

The procedure insert_partition inserts all concepts of a given partition into the global taxonomy. For updating a concept or its parents or children, it locks the corresponding nodes. It first performs for each concept *new* the top-search phase (starting from the top concept) and possibly repeats the top-search phase for *new* if other threads updated the list of children of its parents. Then, it sets the parents of *new* and adds *new* for each parent to its list of children. Afterwards the bottom-search phase (starting from the bottom concept) is performed.

---

[1] This notation indicates that the concepts female and not-male are synonyms for each other.

Algorithm 1: parallel_tbox_classification(*concept_list*)

---

    topological-order-list ← topological_order(*concept_list*)
    repeat
       wait until an idle thread t_i becomes available
       select a partition p_i from *topological-order-list*
       run thread t_i with insert_partition(p_i, t_i)
    until all concepts in *topological-order-list* are inserted
    compute ratio and overhead
    print statistics

Algorithm 2: insert_partition(*partition,id*)

  1: lock(*inserted_concepts*(*id*))
  2: *inserted_concepts*(*id*) ← ∅
  3: unlock(*inserted_concepts*(*id*))
  4: **for all** *new* ∈ *partition* **do**
  5:    *parents* ← top_search(*new*,⊤)
  6:    **while** ¬ consistent_in_top_search(*parents*,*new*) **do**
  7:      *parents* ← top_search(*new*,⊤)
  8:    lock(*new*)
  9:    predecessors(*new*) ← *parents*
10:    unlock(*new*)
11:    **for all** *pred* ∈ *parents* **do**
12:      lock(*pred*)
13:      successors(*pred*) ← successors(*pred*) ∪ {*new*}
14:      unlock(*pred*)
15:    *children* ← bottom_search(*new*,⊥)
16:    **while** ¬ consistent_in_bottom_search(*children*,*new*) **do**
17:      *children* ← bottom_search(*new*,⊥)
18:    lock(*new*)
19:    successors(*new*) ← *children*
20:    unlock(*new*)
21:    **for all** *succ* ∈ *children* **do**
22:      lock(*succ*)
23:      predecessors(*succ*) ← predecessors(*succ*) ∪ {*new*}
24:      unlock(*succ*)
25:    *check* ← check_if_concept_inserted(*new*, *inserted_concepts*(*id*))
26:    **if** check ≠ 0 **then**
27:      **if** check = 1 ∨ check = 3 **then**
28:        *new_predecessors* ← top_search(*new*,⊤)
29:        lock(*new*)
30:        predecessors(*new*) ← *new_predecessors*
31:        unlock(*new*)
32:      **if** check = 2 ∨ check = 3 **then**
33:        *new_successors* ← bottom_search(*new*,⊥)
34:        lock(*new*)
35:        successors(*new*) ← *new_successors*
36:        unlock(*new*)
37:    **for all** busy threads $t_i \neq id$ **do**
38:      lock(*inserted_concepts*(t_i))
39:      *inserted_concepts*(t_i) ← *inserted_concepts*(t_i) ∪ {*new*}
40:      unlock(*inserted_concepts*(t_i))

Analogously to the top-search phase the bottom search is possibly repeated and sets the children of *new* and updates the parents of the children of *new*. After finishing the top and bottom search for *new* it is checked again whether other threads updated its entry in *inserted_concepts* and the top and/or bottom search needs to be repeated. This step needs to be done only once. Finally, *new* is added to the entries in *inserted_concepts* of all other busy threads.

In order to avoid unnecessary tree traversals and tableau subsumption tests when computing the subsumption hierarchy, the parallel classifier adapted the enhanced traversal method [2], which is an algorithm that was designed for sequential execution. Algorithm 3 and 4 outline the traversal procedures for the top-search phase.

Algorithm 3: top_search(*new*,*current*)

mark(*current*,'visited')
pos-succ ← ∅
captured_successors(*new*)(*current*) ← successors(*current*)
for all $y \in$ successors(*current*) **do**
  if enhanced_top_subs(*y*,*new*) **then**
    pos-succ ← *pos-succ* ∪ {*y*}
if *pos-succ* = ∅ **then**
  return {*current*}
else
  result ← ∅
  for all $y \in$ *pos-succ* **do**
    if *y* not marked as 'visited' **then**
      result ← *result* ∪ top_search(*new*,*y*)
  return *result*

The procedure top_search outlined in Algorithm 3 recursively traverses the taxonomy top-down from a current concept and tries to push the new concept down the taxonomy as far as possible by traversing the children of the current concept. It uses an auxiliary procedure enhanced_top_subs (outlined in Algorithm 4) which itself uses an auxiliary procedure subsumes (not specified here) that implements a subsumption test.

In a symmetric manner the procedure bottom_search traverses the taxonomy bottom-up from a current concept and tries to push the new concept up the taxonomy as far as possible. It uses an auxiliary procedure enhanced_bottom_subs. Both procedures are omitted for ease of presentation.

To resolve the possible incompleteness caused by parallel classification, we utilize Algorithms 5, 6 and 7. The procedure consistent_in_bottom_search is not shown here because it mirrors consistent_in_top_search.

Algorithms 5 and 6 illustrate the solution for scenario I described in Section 2.1. As already described, in top search we start traversing from the top concept to locate the concept *new* in the taxonomy. At time *t1*, when top_search is called, we capture the children information "captured_successors" of the concept *current*; the children information is stored relative[2] to the concept *new* being

---

[2] Otherwise a different thread could overwrite captured_successors for node *current*. This is now prevented because each concept (*new*) is inserted by only one thread.

Algorithm 4: enhanced_top_subs(*current,new*)

---

if *current* marked as 'positive' **then**
　　return *true*
else if *current* marked as 'negative' **then**
　　return *false*
else if for all $z \in$ predecessors(*current*)
　　　　　　enhanced_top_subs(*z,new*)
　　　　and subsumes(*current,new*) **then**
　　mark(*current*,'positive')
　　return *true*
else
　　mark(*current*,'negative')
　　return *false*

Algorithm 5: consistent_in_top_search(*parents,new*)

---

for all *pred* $\in$ *parents* **do**
　　if successors(*pred*) $\neq$ captured_successors(*new*)(*pred*) **then**
　　　　diff_children $\leftarrow$ successors(*pred*) \ captured_successors(*new*)(*pred*)
　　　　for all *diff* $\in$ *diff_children* **do**
　　　　　　if check_interactions(*diff,new*) **then**
　　　　　　　　return *false*
return *true*

---

inserted (we use an array of arrays) and captures the successors of the concept *current* (see Algorithm 3). As soon as top_search is finished at time *t2*, and the parents of the concept *new* have been determined, we check if there has been any update on the children list of the computed parents for *new* between *t1* and *t2* (e.g., see Algorithm 5 on how this is discovered). If there is any inconsistency and also if there is a subsumption possible[3] between *new* and any concept newly added to the children list, we rerun top_search until there is no inconsistency (see line 6 in Algorithm 2).

The same process as illustrated in Algorithm 5 happens in bottom search. The only difference is that parents information is captured when bottom search starts; and when bottom search finishes, the inconsistency and interaction is checked between the parents list of the computed children for *new* and the "captured_predecessors".

Algorithms 6 and 7 describe the solution for scenario II; every time a thread inserts a concept in the taxonomy, it notifies the other threads by adding the concept name to their "inserted_concepts" list. Therefore, as soon as a thread finds the parents and children of the *new* concept by running top_search and bottom_search; it checks if there is any interaction between *new* concept and

---

[3] This is checked by subsumption_possible using pseudo model merging [12], where a sound but incomplete test for non-subsumption on cached pseudo models of named concepts and their negation is utilized.

Algorithm 6: check_interactions(*diff*,*new*)

---

    return subsumption_possible(*diff*,*new*)

Algorithm 7: check_if_concept_inserted(*new*,*inserted_concepts*)

---

    if *inserted_concepts* = ∅ **then**
      return *0*
    else
      for all *concept* ∈ *inserted_concepts* **do**
        if check_interactions(*concept*,*new*) **then**
          if check_interactions(*new*,*concept*) **then**
            return *3*
          else
            return *1*
        else if check_interactions(*new*,*concept*) **then**
          if check_interactions(*concept*,*new*) **then**
            return *3*
          else
            return *2*
      return *0*

---

the concepts located in the "inserted_concepts" list. Based on the interaction, top_search and/or bottom_search need to be repeated accordingly.

Proposition 1 (Completeness of Parallel TBox Classifier) *The proposed algorithms are complete for TBox classification.*

TBox classification based on top search and bottom search is complete in the sequential case. This means that the subsumption algorithms will find all subsumption relationships between concepts of a partition assigned to a single thread. The threads lock and unlock nodes whenever they are updating the information about a node in the global subsumption tree. Thus, we need to consider only the scenarios where two concepts C and D are inserted in parallel by different threads (e.g., thread#1 inserts concept C while thread#2 inserts concept D). In principle, if top (bottom) search pushed a new concept down (up), the information about children (parents) of a traversed node E could be incomplete because another thread might later add more nodes to the parents or children of E that were not considered when determining whether the concept being inserted subsumes or is subsumed by any of these newly added nodes. This leads to two scenarios that need to be examined for incompleteness.

W.l.o.g. we restrict our analysis to the case where a concept C is a parent of a concept D in the complete subsumption tree (CT). Let us assume that our algorithms would not determine this subsumption, i.e., in the computed (incomplete) tree (IT) the concept C is not a parent of D.

Case I: top_search incomplete for D: After D has been pushed down the tree IT as far as possible by top search (executed by thread#2) and top search has traversed the children of a concept E and E has become the parent of D, C

Table 1. Characteristics of the used test ontologies.

Ontology	DL language	No. of named concepts
Embassi-2	$\mathcal{ALCHN}$	657
Embassi-3	$\mathcal{ALCHN}$	1,121
Galen	$\mathcal{SHN}$	2,730
Galen1	$\mathcal{ALCH}$	2,730
Galen2	$\mathcal{ELH}$	3,928
FungalWeb	$\mathcal{ALCHIN(D)}$	3,603
Umls-2	$\mathcal{ALCHIN(D)}$	9,479
Tambis-2a	$\mathcal{ELH}$	10,116

is inserted by thread#1 as a new child of E. In line 6 of Algorithm 2 top_search is iteratively repeated for the concept *new* as long as consistent_in_top_search finds a discrepancy between the captured and current successors of the parents of the newly inserted concept *new*. After finishing top and bottom search, Algorithm 2 checks again in lines 27-28 whether top search needs to be repeated due to newly added nodes. If any of the newly added children of D would subsume C and become a parent of C, the repeated execution of top_search would find this subsumption. This contradicts our assumption.

Case II: bottom_search incomplete for C: After C has been pushed up the tree IT as far as possible by bottom search (executed by thread#1) and bottom search has traversed the parents of a concept E and E has become a child of C, D is inserted by thread#2 as a new parent of E. In line 16 of Algorithm 2 bottom_search is iteratively repeated for the concept *new* as long as consistent_in_bottom_search finds a discrepancy between the captured and current predecessors of the children of the newly inserted concept *new*. After finishing top and bottom search, Algorithm 2 checks again in lines 32-33 whether bottom search needs to be repeated due to newly added nodes. If C would subsume any of the newly added parents of D and it would become a child of C, the repeated execution of bottom_search would find this subsumption. This contradicts our assumption.

3   Evaluation

The Parallel TBox Classifier has been developed to speed up the classification time especially for large ontologies by utilizing parallel threads sharing the same memory. The benchmarking can be configured so that it runs various experiments over ontologies. We evaluated it with a collection of 8 mostly publicly available ontologies. Their name, size in number of named concepts, and used DL is shown in Table 1. As mentioned in the previous section, two parameters influence the parallel TBox classification, namely number of tasks/concepts per thread and number of threads; the number of tasks/concepts per thread was set to 5 and number of threads to 2 in our empirical experiments.

To better compare the performance between the sequential and parallel case, we assume that every subsumption test runs in time *t1* and in the sequential and parallel case the same amount of time is used for an executed subsumption test.

Table 2. Subsumptions tests and their ratio for the test ontologies.

	Embassi-2	Embassi-3	Galen	Galen1
Subs. Tests in sequent.	154,034	420,912	2,706,412	2,688,107
Subs. Tests in thread#1	76,267	217,324	1,363,321	1,367,302
Subs. Tests in thread#2	77,767	214,633	1,354,297	1,348,281
Worst Case Ratio	50.48%	51.63%	50.37%	50.86%
Overhead	1.64%	2.62%	0.41%	1.02%
	Galen2	**FungalWeb**	**Umls-2**	**Tambis-2a**
Subs. Tests in sequent.	5,734,976	4,996,932	87,423,341	36,555,225
Subs. Tests in thread#1	2,929,276	2,518,676	44,042,203	18,342,944
Subs. Tests in thread#2	2,893,716	2,490,329	44,025,988	18,261,532
Worst Case Ratio	51.07%	50.40%	50.37%	50.17%
Overhead	1.53%	0.24%	0.73%	0.13%

Subsumption tests can be expensive and, hence, are preferred to be avoided by optimization techniques such as pseudo model merging [12].

$$Ratio = \frac{MaxOfSubsTests}{TST_s} \quad (1) \qquad Overhead = \frac{TST_p - TST_s}{TST_s} \quad (2)$$

The ratio illustrated in Equation 1 uses TST_s, the number of times a subsumption test was computed in the sequential case, and *MaxOfSubsTests*, the maximum number of subsumption tests performed in all threads. Similarly, Equation 2 defines the overhead (where the index p refers to the parallel case).

Table 2 shows an excellent performance increase and a surprisingly small overhead when using the Parallel TBox Classifier. Using two threads the maximum number of subsumption test for all ontologies could be reduced to roughly one half compared to the sequential case. The overhead as defined in Equation 2 varies between 0.13% and 2.62%. The overhead is mostly determined by the quality of the told subsumers and disjoints information, the imposed order of traversal within a partitioning, and the division of the ordered concept list into partitions. In general, one should try to insert nodes as close as possible to their final order in the tree using a top to bottom strategy.

4   Conclusion

In this paper, we described an architecture for parallelizing well-known algorithms for TBox classification. Our work is targeted for ontologies where independent partitions cannot be easily constructed; therefore we did not use the previously mentioned modularity approaches in our system. The first experimental evaluation of our techniques shows very promising results because the overhead for ensuring completeness is surprisingly small. In our next steps we plan to extend our tests with different configurations of threads and partition sizes and a larger variety of test ontologies. We intend to feed recorded runtimes for performing single subsumption tests into our simulator in order to make the computation of the overhead more accurate. We also plan to implement and test our approach in a multi-core and multi-processor environment.

References

1. Aslani, M., Haarslev, V.: Towards parallel classifcation of TBoxes. In: Proc. of the 2008 Int. Workshop on Description Logics, Dresden, Germany, May 13-16 (2008)
2. Baader, F., Franconi, E., Hollunder, B., Nebel, B., Profitlich, H.: An empirical analysis of optimization techniques for terminological representation systems or: Making KRIS get a move on. Applied Artificial Intelligence. Special Issue on Knowledge Base Management 4(2), 109–132 (1994)
3. Bergmann, F., Quantz, J.: Parallelizing description logics. In: Proc. of 19th Ann. German Conf. on Artificial Intelligence. pp. 137–148. LNCS, Springer-Verlag (1995)
4. Cuenca Grau, B., Halaschek-Wiener, C., Kazakov, Y.: History matters: Incremental ontology reasoning using modules. In: Proc. of the 6th Int. Semantic Web Conf. (ISWC 2007), Busan, South Korea, Nov. 11-15 (2007)
5. Cuenca Grau, B., Horrocks, I., Kazakov, Y., Sattler, U.: A logical framework for modularity of ontologies. In: In Proc. of the 20th Int. Joint Conf. on Artificial Intelligence (IJCAI 2007), Busan, South Korea, Nov. 11-15. pp. 298–303 (2007)
6. Dolby, J., Fokoue, A., Kalyanpur, A., Kershenbaum, A., Ma, L., Schonberg, E., Srinivas, K.: Scalable semantic retrieval through summarization and refinement. In: 21st Conf. on Artificial Intelligence (AAAI). pp. 299–304. AAAI Press (2007)
7. Eyal, A., Mcllraith, S.: Partition-based logical reasoning for first-order and propositional theories. Artifical Intelligence 162(1-2), 49–88 (2005)
8. Fokoue, A., Kershenbaum, A., Ma, L., Schonberg, E., Srinivas, K.: The summary Abox: Cutting ontologies down to size. In: Proc. of Int. Semantic Web Conf. (ISWC). LNCS, vol. 4273, pp. 343–356. Springer-Verlag (2006)
9. Guo, Y., Heflin, J.: A scalable approach for partitioning OWL knowledge bases. In: Proc. 2nd Int. Workshop on Scalable Semantic Web Knowledge Base Systems, Athens, USA. pp. 47–60 (2006)
10. Haarslev, V., Möller, R.: High performance reasoning with very large knowledge bases: A practical case study. In: Proc. of the 17th Int. Joint Conf. on Artificial Intelligence, IJCAI-01, Aug. 4-10, Seattle, USA. pp. 161–166 (2001)
11. Haarslev, V., Möller, R.: On the scalability of description logic instance retrieval. Journal of Automated Reasoning 41(2), 99–142 (2008)
12. Haarslev, V., Möller, R., Turhan, A.Y.: Exploiting pseudo models for TBox and ABox reasoning in expressive description logics. In: Proc. of the Int. Joint Conf. on Automated Reasoning, June 18-23, 2001, Siena, Italy. pp. 61–75 (2001)
13. Liebig, T., Müller, F.: Parallelizing tableaux-based description logic reasoning. In: Proc. of 3rd Int. Workshop on Scalable Semantic Web Knowledge Base Systems (SSWS '07), Vilamoura, Portugal, Nov 27. LNCS, vol. 4806, pp. 1135–1144. Springer-Verlag (2007)
14. Shan, H., Singh, J.P.: Parallel tree building on a range of shared address space multiprocessors: Algorithms and application performance. In: 12th Int. Parallel Processing Symposium (IPPS '98), March 30 - April 3, 1998, Orlando, Florida, USA. pp. 475–484 (1998)
15. Shearer, R., Horrocks, I.: Exploiting partial information in taxonomy construction. In: Proc. of the 8th International Semantic Web Conference (ISWC 2009) (2009)
16. Urbani, J., Kotoulas, S., Oren, E., van Harmelen, F.: Scalable distributed reasoning using MapReduce. In: International Semantic Web Conference. pp. 634–649 (2009)

Part III

Posters

The Logical Difference For Fuzzy $\mathcal{EL}^+$ Ontologies

Shasha Feng[1,2,3], Dantong Ouyang[1,3], Yonggang Zhang[1,3],Haiyan Che[1,3], Jie Liu[1,3]

[1] Jilin University, Changchun 130012, P.R. China
[2] The University of Manchester, Manchester M13 9PL, UK
[3] Key Laboratory of Symbolic Computation and Knowledge Engineering of Ministry of Education, Changchun 130012, P.R. China
{fengss,oud,zhangyg,chehy,liu_jie}@jlu.edu.cn

Abstract. Ontologies undergo changes for reasons such as changes in knowledge, meeting varying application requirements. Thus, for different versions of a considered ontology, it is important to clarify the difference between them. The difference above refers to the logical difference, not syntactic one. Examples of the logical difference include the difference in taxonomy, concept subsumption difference and query difference. These has been well investigated for the lightweight description logic $\mathcal{EL}$ because of its tractability and successful application in bio-medical ontologies.

Fuzzy $\mathcal{EL}^+$ has been put forward and applied in view-based searching in Semantic portals. Thus comes the problem of comparing fuzzy ontologies of different versions and clarifying the difference. In this paper, we define the logical difference of two fuzzy $\mathcal{EL}^+$ ontologies. For fuzzy $\mathcal{EL}^+$ ontologies of different versions we investigate how to compute the difference in taxonomy and concept subsumption difference. We also explore how to compute approximation of the logical difference of two $\mathcal{EL}$ terminologies. Our work can be applied in the scenario of $\mathcal{EL}^+$ ontologies with access control if the set of all access rights is a linear order.

1   Introduction

Ontologies undergo changes for reasons such as changes in knowledge, meeting varying application requirements [1]. Thus, for different versions of a considered ontology, it is important to clarify the difference between them. The difference hereinabove refers to the logical difference of two ontologies, that is, the set of entailments implied by one ontology, but not by the other [2, 3]. For example, if a general concept inclusion (GCI) $C \sqsubseteq D$ is implied by one ontology, but not the other, then it is in the set of the logical difference, which is the difference in concept subsumption. In the above case, if C and D are simple concept names, then this kind of difference is called the difference in taxonomy. Query answering difference refers to the difference in query answering by two ontologies [3]. These has been well investigated for the lightweight description logic $\mathcal{EL}$ because of its tractability and successful application in bio-medical ontologies [2, 4, 5].

Fuzzy $\mathcal{EL}^+$ [6] has been put forward and applied in ontology alignment [7] and view-based searching in Semantic portals [8]. The fuzzy description logic allows fuzzy subsumption, and has scalable classification algorithm. Thus comes the problem of comparing fuzzy ontologies of different versions, as done in classic DL ontologies.

In this paper, we investigate how to clarify the difference between two fuzzy $\mathcal{EL}^+$ ontologies. First, we define the logical difference in fuzzy $\mathcal{EL}^+$. Then we investigate how to compute the difference in taxonomy and concept subsumption difference, and discuss how to compute approximation of the logical difference. Some of our investigation is about the following case. The second fuzzy ontology is obtained by modifying the degrees of truth of some axioms in the first fuzzy ontology. Thus, for some given entailments of interest, we want to compute the change of their degrees of truth.

As shown in [9, 10], there are some shared principles between reasoning in fuzzy $\mathcal{EL}^+$ and ontologies with access control. Thus, our results can be applied in the scenarios of $\mathcal{EL}^+$ ontologies with access control. More concretely, if the set of all access rights is a linear order, we can treat the access right of an axiom (or entailment) as its degree of truth. Thus, after modifications on some axioms' access right, the problem of computing the access right of the entailment is the same as the logical difference problem described in last paragraph.

This paper is structured as follows. In the next section, we introduce the preliminary about fuzzy $\mathcal{EL}^+$ and its reasoning methods. In section 3, we define the logical difference of two fuzzy ontologies. Then, in section 4, we investigate how to compute the difference in taxonomy, the concept subsumption difference, and approximation of the difference. At last, in section 5, we conclude the paper and discuss future work related to the logical difference.

2 Preliminary

2.1 Fuzzy $\mathcal{EL}^+$

Fuzzy set theory has been well applied in representing knowledge with vagueness [11]. Let X be a set of elements. A fuzzy subset A of X, is defined by a *membership function* $\mu_A(x)$, or simply $A(x)$, of the form $\mu_A(x) : X \rightarrow [0,1]$ [11]. This function assigns each $x \in X$ a value $n \in [0,1]$ that represents the degree of x belongs to X. Then, under this assumption, the classical set operators and logical operators are performed by mathematical functions. For example, *fuzzy complement* is a unary function of the form $c : [0,1] \rightarrow [0,1]$, *fuzzy intersection and union* are two binary functions of the form $t : [0,1] \times [0,1] \rightarrow [0,1]$, and $u : [0,1] \times [0,1] \rightarrow [0,1]$, called t-norm and t-conorm operations, respectively, and *fuzzy implication* also by a binary function $\mathcal{T} : [0,1] \times [0,1] \rightarrow [0,1]$ [11]. Certainly, the definitions of these functions have to satisfy some properties to make sense. There are difference semantics according to the choices of these functins. We only introduce Gödel semantics, which is also the base of fuzzy $\mathcal{EL}^+$ introduced hereinbelow. As for other semantics, we refer to [11]. The t-norm is $t_G(a,b) = min(a,b)$, t-conorm $u_G(a,b) = max(a,b)$, and implication $\mathcal{T}_G(a,b) = b$ if $a > b, \mathcal{T}_G(a,b) = 1$ otherwise.

Then let us illustrate fuzzy $\mathcal{EL}^+$, denoted by f_G - $\mathcal{EL}^+$ [6]. Suppose N_C is a set of concept names, and N_R a set of role names. Then, $\top$ and elements in N_C are *concept descriptions*. If C and D are concept descriptions, then $C \sqcap D$ and $\exists r.C$ are also concept descriptions, where $r \in N_R$. The f_G - $\mathcal{EL}^+$ ontology consists of finite general concept inclusions $\langle C \sqsubseteq D, n \rangle$ and role inclusions axioms (RIA) $r \sqsubseteq s$, where $n \in (0,1]$ and

$r, s \in N_R$. That is, in the f_G - $\mathcal{EL}^+$ ontology, fuzzy concept inclusions (f-GCI) are permitted, and $\langle C \sqsubseteq D, n \rangle$means that the degree of C is a subset of D is n. However, there is no fuzzy role inclusion permitted. The language is f_G - $\mathcal{EL}$ if there is no RIA in the ontology. In the later parts of the paper, we use *crisp DL ontology* to denote the ontology resulting from eliminating all the degrees of truth from the axioms.

The semantics of fuzzy DLs are defined through a *fuzzy interpretation*. A fuzzy interpretation consists of $(\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$, where $\Delta^{\mathcal{I}}$ is a non-empty set of elements, and $\cdot^{\mathcal{I}}$ is a fuzzy interpretation function, which maps,

- an individual name a to an element $a^{\mathcal{I}}$ in $\Delta^{\mathcal{I}}$,
- a concept name A to a membership function $A^{\mathcal{I}} : \Delta^{\mathcal{I}} \rightarrow [0, 1]$, and
- a role name r to a membership function $r^{\mathcal{I}} : \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}} \rightarrow [0, 1]$.

Then, the interpretation is extended as shown in Table 1.

Table 1. Syntax and Semantics of the fuzzy description logic $\mathcal{EL}^+$

Constructor	DL Syntax	Semantics
top	$\top$	$\top^{\mathcal{I}}(a) = 1$
conjunction	$C \sqcap D$	$(C \sqcap D)^{\mathcal{I}}(a) = min(C^{\mathcal{I}}(a), D^{\mathcal{I}}(a))$
existential restriction	$\exists r.C$	$(\exists r.C)^{\mathcal{I}}(a) = sup_{b \in \Delta^{\mathcal{I}}}\{min(r^{\mathcal{I}}(a, b), C^{\mathcal{I}}(b))\}$
fuzzy GCIs	$\langle C \sqsubseteq D, n \rangle$	$inf_{a \in \Delta^{\mathcal{I}}}\{\mathcal{T}_G(C^{\mathcal{I}}(a), D^{\mathcal{I}}(a))\} \geq n$
RIAs	$r_1 \circ \ldots \circ r_k \sqsubseteq s$	$[r_1^{\mathcal{I}} \circ^t \ldots \circ^t r_k^{\mathcal{I}}](a, b) \leq s^{\mathcal{I}}(a, b)$

2.2   Reasoning in Fuzzy $\mathcal{EL}^+$

The reasoning problem in fuzzy $\mathcal{EL}^+$ ontology is to decide whether $\langle \alpha, n \rangle$is implied by the ontology or not. That is, we have to determine the degree of truth n here, compared with the reasoning problem in standard $\mathcal{EL}^+$ ontology. Currently, there exist two methods as follows.

One reasoning method is to develop a calculus which deals with the degrees of truth in every inference step, as shown in the paper by Stoilos et al [6]. The method is a modification of the classification algorithm of $\mathcal{EL}^+$ ontology. For each entry in the completion process, a value n is affiliated to reflect its degree of truth (considering this entry locally). For newly generated entry by the completion rule, it will get its affiliated value $k \in (0, 1]$ according to the input entries and their affiliated values. The completion rules will be repeatedly applied until there is no new binary tuple (entry, affiliated value) generated. For the resulting entries, their degrees of truth are the biggest ones among their affiliated values. Actually, this method is a classification algorithm, that is, it will return the degree of truth of any concept name subsumption. The method is similar to the axiom pinpointing algorithm based on monotone Boolean function [12], but much simpler. That is because in deciding whether to apply a completion rule, this method employs comparison between numerial values, in contrast with logical

implication checking between formulae in pinpointing algorithm. Thus, with respect to complexity, this method sits strictly between classification algorithm and the axiom pinpointing algorithm.

From above reasoning method, we explore the property of the degree of truth of an entailment. An entailment has one or many MinAs, which is the minimal subset of the ontology implying that entailment [12]. From above inference steps, we know that the entailment has a *temporary degree of truth* from one MinA, called that *MinA's degree*, which is the minimal degree of truth of all the axioms in that MinA. The entailment's final degree of truth is the maximal among its temporary degrees of truth. Theorem 1 describes this formally.

Theorem 1. *Suppose that O is a fuzzy $\mathcal{EL}^+$ ontology, and $O \vDash \langle \alpha, n \rangle$. Obtain a classic O_{crisp} by deleting all the degrees of truth in O. Let $A_1 = \{\beta_{11}, \ldots, \beta_{1j_1}\}, \ldots, A_k = \{\beta_{k1}, \ldots, \beta_{kj_k}\}$ be all the MinAs of α in O_{crisp}. Then let $n_{11}, \ldots, n_{1j_1}, \ldots, n_{k1}, \ldots, n_{kj_k}$ be degrees of truth for $\beta_{11}, \ldots, \beta_{1j_1}, \ldots, \beta_{k1}, \ldots, \beta_{kj_k}$ in O respectively. Let $d(A_i)$ be $min_{1 \leq h \leq j_i}\{n_{ih}\}$, for $1 \leq i \leq k$. Then, we have $n = max_{1 \leq i \leq k}\{d(A_i)\}$.*

The other reasoning method treats the standard DL reasoner as an oracle in the computation of the degree of truth n of α, which is justified by the following result.

Theorem 2. *(from [10]) Suppose that O is a f_G - $\mathcal{EL}^+$ ontology. Then, $O \vDash \langle C_1 \sqsubseteq C_2, n \rangle$ if and only if $O_n \vDash C_1 \sqsubseteq C_2$, where $O_n = \{C \sqsubseteq D | \langle C \sqsubseteq D, m \rangle \in O, n \leq m\}$.*

Then, we can use binary search to compute degree of truth n [10]. This method enjoys the fast converging speed of binary search, as well as sophisticated existing $\mathcal{EL}^+$ reasoner [13].

Moreover, Theorem 2 shows that for an entailment with degree of truth n, only axioms of higher degree of truth contribute to its correctness. This property will be further used in subsection 4.2.

3   Logical Difference for Fuzzy DLs

We extend the logical difference to fuzzy DLs after listing that definition in classic DLs.

Definition 1. *(logical difference in classic DLs, from [2]) Suppose that O_1 and O_2 are two DL ontologies, and S is a signature, then their logical difference with respect to S is defined as $diff_S(O_1, O_2) = \{C \sqsubseteq D | O_1 \vDash C \sqsubseteq D, O_2 \nvDash C \sqsubseteq D$, symbols in $C \sqsubseteq D$ are from $S.\}$.*

Definition 2. *(logical difference for fuzzy DL ontologies) Suppose that O_1 and O_2 are two fuzzy DL ontologies, and S is a signature, then their logical difference is defined as $f\text{-}diff_S(O_1, O_2) = \{h\alpha, n \rangle | O_1 \vDash \langle \alpha, n \rangle$ and $O_2 \nvDash \langle \alpha, n \rangle$, symbols in α are from $S.\}$*

In the above two definitions, when the signature S is clear from context, we will drop it when mentioning the difference.

It is easy to verify that the logical difference on fuzzy DLs is the same as that on classic DLs if the underlying fuzzy DLs has only degrees of truth of 0 and 1. From

the definition, we find that an element in the logical difference consists of a normal entailment, and its degree of truth. Thus, there are two sources to the contribution of the elements. One is the difference in normal entailments, which is the logical difference of two crisp DL ontologies. The other is the difference in degrees of truth, which illustrates the following situation. Two crisp DL ontologies imply the same set of entailments, but differ on the degrees of truth of some entailments.

Since we care more about the difference caused by modifications on some degrees of truth, the crisp parts of two DL ontologies are the same. Thus, the only source of difference is the degrees of truth of the entailments.

4 Compute the Logical Difference

In this section, we will discuss how to compute the logical difference of two fuzzy DL ontologies. If modelers of ontology are interested at the subsumption relationship between two concept names, which actually is the difference in taxonomy, we recommend the method in subsection 4.1. If modelers focus on the subsumption between concept descriptions, subsection 4.2 provides guidelines. If modelers expect more information in the difference, an approximation is proposed for fuzzy $\mathcal{EL}$ terminologies in subsection 4.3. Thus, we provide a range of methods for the choice of modelers according to requirements of application.

4.1 Difference in Taxonomy

In this subsection, we focus on computing the entailments of the form $\langle A \sqsubseteq B, n \rangle$ in the difference of two fuzzy ontologies, where A and B are concept names. The method we propose in this subsection is an adaption of the method successfully applied in classic DL ontologies [14].

First, we recall some results in classic DL ontologies. In classic DL ontologies, module w.r.t a signature is a subset of an ontology which is indistinguishable with the ontology w.r.t that signature [15]. Thus, when reasoning on that signature, it is safe to use the module instead of the original ontology, which can speed up the reasoning when the module and the ontology are of different scales. For a concept name A, any entailment of the form $A \sqsubseteq C$, where C is concept description, can be implied from the locality-based module w.r.t $\{A\}$ [14].

Then, for a fuzzy $\mathcal{EL}^+$ ontology, we define its module with respect to a signature.

Definition 3. *(module for fuzzy ontology) For a fuzzy ontology O_f, its module with respect to signature S, denoted by $\mathcal{M}_{fS}$, is its subset which is indistinguishable with O with respect to S. That is, for any α which consists of symbols only from S and $n \in (0, 1]$, we always have $O_f \vDash \langle n \rangle$ if and only if $\mathcal{M}_{fS} \vDash \langle n \rangle$.*

Moreover, for a fuzzy DL ontology O_f, its locality-based module $\mathcal{M}_f$ consists of all the axioms which also appear in locality-based module when their degree of truth discarded. That is, $\mathcal{M}_f = \{ \langle \alpha, n \rangle | \alpha \in \mathcal{M}_{crisp}, \mathcal{M}_{crisp}$ is a locality-based module of $O_{crisp} = \{\beta | \langle \beta, m \rangle \in O_f\} \}$.

Theorem 3 shows that the locality-based module is a module for fuzzy $\mathcal{EL}^+$ ontology.

Theorem 3. *For a fuzzy DL ontology O_f, its locality-based module M_f (with respect to Signature S) is a module (with respect to Signature S).*

Sketch of proof. For any entailment $\langle \alpha, n \rangle$ implied by O_f, we have that α is implied by O_{crisp}, also by M_{crisp}. It means that, any mininal explanation of α is contained in M. Since M is generated under the direction of M_{crisp}, we say that any minimal explanation of $\langle \alpha, n \rangle$ is contained in M_f. Thus, $\langle \alpha, n \rangle$ is implied by M_f. That is, M_f is a module of O_f. Q.E.D.

Lemma 1. *(from [15]) For a classic DL ontology O, A is a concept name in it. Then any entailment of the form $A \sqsubseteq C$ implied by O can be implied from O's locality-based module with respect to {A}, where C is a concept description.*

Theorem 4. *For a fuzzy DL ontology O_f, A is a concept name in it. Then any entailment of the form $\langle A \sqsubseteq C, n \rangle$ implied by O_f can be implied from O_f's locality-based module with respect to {A}, where C is a concept description.*

With these results at hand, our strategy (as in [14]) is for every concept name A, we keep a record of the module with respect to A. Then for two versions of fuzzy $\mathcal{EL}^+$ ontologies, we check whether the related module is changed or not for every A. If unchanged, it means that all the subsumptionhoods between A and other concept names are unchanged. If changed, it is enough to perform the fuzzy classification algorithm only on the module, which is much smaller, to find the new related subsumptionhoods. Thus our strategy, like the corresponding method in classic DLs, is especially suitable for large scale ontologies.

If we only care about the subsumptionhood between A and a specific concept name, we might be able to avoid the fuzzy classification algorithm in some cases. We will explain this in the next subsection.

4.2   Concept Subsumption Difference

In this subsection, we focus on the entailment of the form $C \sqsubseteq D$ in the logical difference, where C and D are concept descriptions. We consider the simplest case: When the degree of truth of an axiom in the first ontology is increased, we want to know the new degree of truth of one entailment of the form $C \sqsubseteq D$. Formally, suppose that O is a fuzzy $\mathcal{EL}^+$ ontology, $\langle \alpha, n \rangle$ is an axiom in its TBox, and $\langle \beta, d \rangle$ is one of entailments of O of interest. Let O' be obtained by replacing $\langle \alpha, n \rangle$ with $\langle \alpha, m \rangle$, where $m \in (n, 1]$. Then we want to calculate the value of d' such that O' implies $\langle \beta, d' \rangle$. We discuss the problem according to the degree of truth d.

Theorem 5. *If $d < n$, then $d' = d$.*

The theorem says that, for entailment whose degree of truth is less than n, then its uncertainty degree in the new ontology is the same as that in the previous ontology.

Proof of sketch. From Theorem 1, we know that the degree of truth of α contributes to β's degree of truth, if and only if α is in one of β's MinA, and α's degree is the smallest compared with other axioms' in that MinA. Moreover, α's degree is the biggest

compared with the degrees returned by other MinAs. The statement *dn* in the precondition shows that, even α is in one of β's MinA, there exists another axiom with degree of truth $d < n$. That axiom decides the degree of β to be d. That is, the degree of α has no effection on β's degree. Thus, increase on α's degree will leave β's degree unchanged. Q.E.D.

Theorem 6. *If $n \leq d < m$, then let $O_{>d}$ be $\{\gamma \mid \langle \gamma, k \rangle \in O'$ and $k > d\}$.*

1. *if $O'_{>d} \vDash \beta$, then β's degree of truth $d' \in (d, m]$.*
2. *otherwise, β's degree of truth stays unchanged, i.e. $d' = d$.*

In this case, we can first perform some reasoning in classic $\mathcal{EL}^+$ ontologies, then we know whether β's degree of truth has to be recomputed or not. Even the degree has to be recomputed, we know the range of its new value.

Proof of sketch. In the first case, $d' > d$ can be derived from Theorem 2. If α is in one MinA, and its degree m is the smallest in that MinA, then d' possibly equals m. It is not possible that d' is bigger than m, which means that there exists a MinA A, the smallest degree in A is bigger than m. Thus, α must not be in A. Since $\langle \alpha, m \rangle$ is the only difference between two ontologies, the MinA A is also in the first ontology O. Then the degree of β in O is bigger than m. Conflict.

For the second case, we know that $O'_{\geq d} = O_{\geq d}$. From Theorem 2 we have $O_{\geq d} \vDash \beta$. Thus $O'_{\geq d} \vDash \beta$. With $O'_{>d} \nvDash \beta$ from the precondition, we conclude that β's degree of truth is d. Q.E.D.

Theorem 7. *If $d \geq m$, then $d' = d$.*

That is, for entailment whose degree of truth is not less than m, then that degree in the new ontology is the same as in the previous ontology.

Proof of Sketch. Suppose that α is in a MinA of β, and its new degree m is the smallest in that MinA. Since $d \geq m$, it means that there exists another axiom with degree of d. Thus, β's degree will stay at d. Q.E.D.

From above analysis, when the degree of truth of one axiom is increased, for an entailment of interest, we know the new range of its degree of truth before performing concrete computation. Sometimes, the new range might be enough for requirements of users, then the effort in performing concrete computation will be saved.

Similarly, we give the following results when the degree of truth of one axiom is decreased. The problem is described the same as in the beginning of this subsection. The only exception is that when $\langle \alpha, n \rangle$ is replaced by $\langle \alpha, m \rangle$, we require that $m \in [0, n)$. Thus, for $\langle \beta, d \rangle$ implied in the previous ontology O, if $d \in (0, m] \cup (n, 1]$, then β's degree of truth in the new ontology O' is still d. When the range of d is $(m, n]$, if $O'_{\geq d} \vDash \beta$, then β's degree in O' is still d. Otherwise, its degree will be in $[m, d)$.

Now, we relax the conditions in our discussion a bit. We allow that in the first ontology, several axioms have their degrees of truth increased. For an entailment of interest, to know its degree of truth in the modified ontology, we can first apply Theorem 5 and Theorem 7, to eliminate the modifications having no effect on the entailment of interest. For the modifications falling in the category of Theorem 6, we can get an estimation of the resulting degree before complete computation. Similar methods can deal with the case when some axioms have their degrees decreased, or even increased and decreased modifications co-exist.

4.3 More Difference

What we have done in the previous subsections is: first, set some entailments (either elements from taxonomy or concept subsumptions of interest), then perform reasoning to find the change of their degrees of truth. The results obtained therefore are only part of the logical difference between ontologies. In this subsection, we try to compute as much information as possible of the logical difference, instead of focusing on some pre-fixed entailments.

Before coming to our solution, we briefly go through the related work in classic $\mathcal{EL}$ [4, 2]. Given two $\mathcal{EL}$ terminologies, any entailment in their logical difference, which is of form $C \sqsubseteq D$, can be derived from either $E \sqsubseteq A$ or $B \sqsubseteq E$, where C, D, E are concept descriptions and A and B are concept names. Thus, the lists of such As and Bs form a reasonable approximation of the logical difference. There exist polynomial algorithms respectively to return the above two lists. However, this kind of approximation in $\mathcal{EL}^+$ remains an open problem due to some role inclusion axioms. Thus, our work described below is only for $\mathcal{EL}$ terminologies, not ontologies, nor $\mathcal{EL}^+$. Terminologies are special kind of ontologies, where every concept is defined (when it occurs, as the only concept, on the left side of an axiom) at most once, and does not refers to itself directly or indirectly in the definition.

Theorem 8 describes the composition of fuzzy logical difference, thus points out one way of computation.

Theorem 8. *Suppose that O and O' are two fuzzy DL ontologies, and f-diff(O, O') is the logical difference between them. Let $n \in (0, 1]$. Then f-diff$_{\geq n}(O, O') = $ diff($O_{\geq n}, O'_{\geq n}$).*

Proof of sketch. Use Theorem 2 and Definition 1 and 2. Q.E.D.

Theorem 8 points out that, the cut set of the fuzzy logical difference by n, equals to the classic logical difference between cut sets of the fuzzy ontologies. Since we can compute approximation for the classic logical difference, then the result can be used as approximation of the cut set of the fuzzy logical difference by n. Thus, with Theorem 8, we can *build* approximation of the logical difference of two fuzzy $\mathcal{EL}$ terminologies gradually.

This result can be further strengthened in the following case. Suppose that two fuzzy $\mathcal{EL}$ terminologies only differ on degrees of some axioms. Let n be the maximal value among the degrees on which two terminologies differ. Then, from Theorem 2, we know that the cut set of the fuzzy logical difference by n is empty. That is, two terminologies imply, to the degree of n, the same entailments.

5   Conclusion and Future Work

Because of the applications of fuzzy $\mathcal{EL}^+$ and its similarly to ontology with access control, we investigate the logical difference problem in fuzzy $\mathcal{EL}^+$. We define the logical difference of two fuzzy $\mathcal{EL}^+$ ontologies. We investigate strategies of computing some forms of the difference, from the simplest one of taxonomy difference to the approximation. Thus, modellers of fuzzy ontologies can choose a suitable solution according to requirements.

We discuss two interesting problems for future investigation. The first one is in computing concept subsumption difference. In subsection 4.2, our discussion is under the assumption that two fuzzy ontologies share the same set of axioms, and only differ on the degrees of truth of some axioms. If we drop this assumption, and let the two fuzzy ontologies be arbitrary, can we still find similar heuristics to avoid fuzzy classification?

The second problem is about deciding conservative extensions between two fuzzy ontologies. We can try the method described in subsection 4.3. However, it is interesting to develop algorithms which deal with degrees of truth on inference step level, instead of converting to classic DLs.

Acknowledgement

References

1. Heflin, J., Pan, Z.: A Model Theoretic Semantics for Ontology Versioning. In the Proceeding of International Conferece on Semantic Web 2002, 62-76 (2004)
2. Konev, B., Walther, D., Wolter, F.: The Logical Difference Problem for Description Logic Terminologies. In the Proceeding of IJCAR 2008. 259–274 (2008)
3. Kontchakov, R., Wolter, F., Zakharyaschev, M.: Can You Tell the Difference Between DL-Lite Ontologies?. In the Proceeding of KR 2008. 285-295 (2008)
4. Lutz, C., Wolter, F.: Conservative extensions in the lightweight description logic $\mathcal{EL}$. CADE 2007. 84–99 (2007)
5. Konev, B., Walther, D., Wolter, F.: Forgetting and Uniform Interpolation in Large-Scale Description Logic Terminologies. In the Proceeding of IJCAI 2009, 830-835 (2009)
6. Stoilos, G., Stamou, G.B., Pan, J.Z.: Classifying Fuzzy Subsumption in Fuzzy-EL+. In the Proceeding of Description Logics 2008, (2008)
7. Euzenat, P.S.J.: Ontology Matching. Springer, 2007.
8. Holi, M., Hyvonen, E.: Fuzzy view-based semantic search. In the Proceeding of Asian Semantic Web Conference 2006, (2006)
9. Peñaloza, R.: Reasoning With Weighted Ontologies. In the Proceeding of Description Logics 2009, (2009)
10. Baader, F., Knechtel, M., Peñaloza, R.: A Generic Approach for Large-Scale Ontological Reasoning in the Presence of Access Restrictions to the Ontology's Axioms. In the Proceeding of International Semantic Web Conference 2009, 49-64, (2009)
11. Klir, G.J., Yuan, B.: Fuzzy Sets and Fuzzy Logic: Theory and Application. Prentice-Hall, 1995.
12. Baader, F., Peñaloza, R., Suntisrivaraporn, B.: Pinpointing in the Description Logic EL+. In the Proceeding of KI 2007, 52-67 (2007)
13. Baader, F., Lutz, C., Suntisrivaraporn, B.: Efficient Reasoning in EL+. In the Proceeding of DL 2006, (2006)
14. Grau, B.C., Halaschek-Wiener, C., Kazakov, Y.: History Matters: Incremental Ontology Reasoning Using Modules. ISWC/ASWC 2007. 183–196 (2007)
15. Grau, B.C., Horrocks, I., Kazakov, Y., Sattler, U.: Just the right amount: extracting modules from ontologies. WWW 2007. 717–726 (2007)

Automata-Based Abduction
for Tractable Diagnosis

Thomas M. Hubauer[1,2], Steffen Lamparter[2], and Michael Pirker[2]

[1] Institute Software, Technology, and Systems (STS)
Hamburg University of Technology, Germany
[2] Autonomous Systems Group
Siemens AG, Germany

Abstract. Abductive reasoning has been recognized as a valuable complement to deductive inference for tasks such as diagnosis and integration of incomplete information despite its inherent computational complexity. This paper presents a novel, tractable abduction procedure for the lightweight description logic $\mathcal{EL}$. The proposed approach extends recent research on automata-based axiom pinpointing (which is in some sense dual to our problem) by assuming information from a predefined abducible part of the domain model if necessary, while the remainder of the domain is considered to be fixed. Our research is motivated by the need for efficient diagnostic reasoning for large-scale industrial systems where observations are partially incomplete and often sparse, but nevertheless the largest part of the domain such as physical structures is known. Technically, we introduce a novel pattern-based definition of abducibles and show how to construct a weighted automaton that commonly encodes the definite and abducible part of the domain model. We prove that its behavior provides a compact representation of all possible hypotheses explaining an observation, and is in fact computable in PTime.

1 Introduction

Abductive reasoning is a method for generating hypotheses that explain an observation based on a model of the domain, typically in the presence of incomplete data. Its non-monotonicity and explorative nature make abduction a promising candidate for the interpretation of potentially incomplete information – a task which is much harder to accomplish using established monotonic inference methods such as deduction or the more elaborate axiom pinpointing. The applications of abductive inference are diverse, ranging from text interpretation [1] to plan generation and analysis [2], and interpretation of sensor [3] or multimedia data [4]. Our research on abductive inference is motivated by industrial applications in Ambient Assisted Living and assistive diagnosis for complex technical devices. In these scenarios we found the underlying models being typically large, though not overly complex in their structure. The main consideration is therefore scalability with respect to the size of the domain model; to effectively support humans or to avoid consequential damage to machinery, information processing is subject to soft realtime constraints.

Our proposed solution to this challenge is based upon logic-based abduction which is not the only, but probably the best-studied notion of this type of inference (see [5] for a survey). In logic-based reasoning, model, observations and hypotheses are represented and manipulated using formal logics; description logics were chosen here as a representation language due to their decidability. Since logic-based abduction is known to be at least as hard as deduction [6], the underlying description logic obviously has to be polynomial for subsumption checking. As we found existential quantification to be of greater importance than universal quantification in both scenarios considered so far, we decided to base our approach on the lightweight description logic $\mathcal{EL}$. Choosing a lightweight description logic, however, does not necessarily guarantee tractability of abduction since the so-called support selection task common to all forms of goal-directed reasoning renders hypotheses generation NP-hard even for Horn-theories [7]. It was shown in [8] that this hardness result can only be alleviated if the number of hypotheses is bounded polynomially, allowing (under certain conditions) to generate a single preferred hypothesis in PTime for $\mathcal{EL}$ and $\mathcal{EL}^+$ knowledge bases [9].

The remainder of this paper is structured as follows: We first recall some basics on description logics and abduction, relating the proposed approach to existing work in this field. Sect. 3 introduces the formalism and justifies its tractability, followed by Sect. 4 where we show how it can be applied to elegantly solve a diagnosis problem. We conclude by summing up the results and giving an outlook on ongoing work.

2   Preliminaries

Description logics are a family of logic-based knowledge representation formalisms designed to ensure decidability of standard reasoning tasks. A concrete description logic is characterized by its admissible concept constructors and axiom types, typically constituting a tradeoff between expressivity and computational complexity. The $\mathcal{EL}$ family of lightweight description logics [10] was tailored specifically to tractability, resulting in a language combining PTime decidability of standard reasoning tasks with adequate expressivity for modeling e. g. the biomedical ontology SNOMED CT. Table 1 summarizes the constructs available in $\mathcal{EL}$ for defining concepts and axioms based on the sets N_{C} and N_{R} of concept names and role names, respectively. To simplify presentation we will assume for the remainder of this paper that the knowledge base $\mathcal{T}$ is in normal form, containing only general concept inclusion axioms of the form $A_1 \sqcap A_2 \sqsubseteq B$, $A_1 \sqsubseteq \exists r.B$ and $\exists r.A_1 \sqsubseteq B$, where $r \in N_{\mathrm{R}}$, $A_1, A_2, B \in N_{\mathrm{C}} \cup \{\top\}$. For the complete $\mathcal{EL}$ family, normalization of an axiom set is linear in the number of axioms both concerning the time required and the number of new axioms generated [11].

Axiom pinpointing, which provides a basis for the approach presented in this paper, can be seen to extend subsumption checking by determining sets $S \sqsubseteq \mathcal{T}$ of axioms from such that the axioms in each set provide a justification for a given subsumption $C \sqsubseteq D$ (i. e. $S \models C \sqsubseteq D$). While this non-standard

Table 1. $\mathcal{EL}$ syntax & semantics

Syntax	Semantics
$\top$	$\Delta^{\mathcal{I}}$
$C \sqcap D$	$C^{\mathcal{I}} \cap D^{\mathcal{I}}$
$\exists r.C$	$\{x \in \Delta^{\mathcal{I}} \mid \exists y \in \Delta^{\mathcal{I}} : (x, y) \in r^{\mathcal{I}} \wedge y \in C^{\mathcal{I}}\}$
$C \sqsubseteq D$	$C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$
$C \equiv D$	$C^{\mathcal{I}} = D^{\mathcal{I}}$

inference task provides useful information in case $\mathcal{T} \models C \sqsubseteq D$, it necessarily fails if $\mathcal{T} \not\models C \sqsubseteq D$. In this latter situation, abductive inference offers a solution by determining sets of hypotheses $\mathcal{H}$ compatible with $\mathcal{T}$ that justify the observation if added to the knowledge base (formally, $\mathcal{T} \cup \mathcal{H} \not\models \bot$ and $\mathcal{T} \cup \mathcal{H} \models C \sqsubseteq D$). Due to the restriction of $\mathcal{EL}$ to terminological information we focus our attention on TBox abduction, where both observations and hypotheses are represented by concept inclusion axioms.

In this respect, our work is closely related to the framework of concept abduction [12] which determines, given a knowledge base $\mathcal{T}$ and two concepts C and D, a concept H such that $\mathcal{T} \not\models C \sqcap H \equiv \bot$ and $\mathcal{T} \models C \sqcap H \sqsubseteq D$. This approach as well as the more elaborate notion of structural abduction [13] employ a tableaux-based calculus for finding a single, $\sqsubseteq$-optimal explanation. The authors do not address computational complexity; due to the underlying description logic and the tableau-based approach, we presume that their approach is at least Exp-Time-hard. Regarding ABox abduction, [4] presents an approach for $\mathcal{SHIQ}$ knowledge bases extended with non-recursive DL-safe rules. Abduction is implemented as an iterative query answering process that returns a single optimal solution subject to a quality criterion which rewards using asserted information while penalizing assumptions. The approach was successfully implemented in a media interpretation framework, its ExpTime worst-case complexity however is prohibitive in the scenario under consideration. Various aspects of abductive inference have also been studied in the context of logic programming, where resolution is most commonly employed for hypotheses generation. This integrates abductive reasoning tightly with the general setting of logic programming but also poses new questions, for example regarding the interaction of abduction with negation as failure used in most logic programming systems. The interested reader is referred to [14] for a thorough introduction to the field of abductive logic programming. [15] examines the relationships between abductive inference and filtering, a process of model selection similar to conditioning in Bayesian networks. Filtering has successfully been applied in performance-critical applications, proving that it can be implemented efficiently. Under certain conditions abduction can indeed be implemented as a process of filtering, yet in in the general case (and especially for unrestricted propositional and first-order theories) filtering is equivalent only to so-called weak abduction.

In order to obtain a tractable algorithm for abductive reasoning within description logics, we resort to recent work on automata-based axiom pinpointing for $\mathcal{EL}$ [16, 17]. The proposed method is based on encoding the model into a weighted Büchi automaton whose accepting runs (called behavior) represent all derivations of the observation from domain knowledge and abducible information, the latter of which is defined compactly using patterns. A hypothesis formula encoding this set of explanations can be determined in PTime with respect to the size of the knowledge base. The upcoming section presents the details of our approach.

3   Automata-Based Abduction for EL

We start by introducing the abductive framework this paper builds on. It differs from other approaches presented above in that both the observation we want to explain and the abducibles are general concept inclusion axioms, which is actually the only way to express relationships between domain elements in $\mathcal{EL}$ due to the absence of individuals. As mentioned before, we assume that the knowledge base $\mathcal{T}$ is in normal form.

Definition 1 (Axiom pattern; instantiation). *Let $\mathcal{T}$ be an $\mathcal{EL}$ TBox over concept names N_C and role names N_R, $\mathcal{V}^\mathrm{C}$ a set of concept variables V_i^C, and $rng : \mathcal{V}^\mathrm{C} \to \mathcal{P}(N_\mathrm{C} \cup \{\top\})$ a complete function mapping each concept variable to a set of concept names (possibly including $\top$), called its range. The range extends by subsumption to $rng^*(V_i^\mathrm{C}) = \{C \in N_\mathrm{C} \cup \{\top\} \mid \exists D \in rng(V_i^\mathrm{C}) : \mathcal{T} \models C \sqsubseteq D)\}$ (with $rng(V_i^\mathrm{C}) \subseteq rng^*(V_i^\mathrm{C})$ since $\mathcal{T} \models C \sqsubseteq C$ trivially holds). An* axiom pattern *is an axiom as defined in Table 1 (not necessarily in normal form), where concept descriptions may contain concept variables from $\mathcal{V}^\mathrm{C}$. An* instantiation *of a pattern is an axiom derived from the pattern by replacing each of its concept variables V_i^C with an element of $rng^*(V_i^\mathrm{C})$.*

Definition 2 (Abduction problem). *Let $\mathcal{T}$ be an $\mathcal{EL}$ TBox over concept names N_C and role names N_R, $A_0 \sqsubseteq B_0$ a general concept inclusion in normal form such that $A_0, B_0 \in N_\mathrm{C}$ (called the* observation*), and Pat a set of axiom patterns over $\mathcal{V}^\mathrm{C}$ whose size is bounded polynomially by the number of concept names in N_C, and rng a range function. The tuple $\mathcal{AP} = (\mathcal{T}, A_0 \sqsubseteq B_0, Pat, \mathcal{V}^\mathrm{C}, rng)$ is called an* abduction problem.

Concept patterns and range function allow for a very fine-grained definition of the parts of the domain which may be assumed. This proves valuable in large-scale applications where typically most of the domain is considered to be fixed (and assumptions most presumably contradict reality), while only certain types of axioms are likely to represent missing information. As an example, compositional (`partOf`) hierarchies of technical systems are completely known to the constructor, whereas the set of observations about such a system is much more likely to be incomplete. Furthermore, explanations are typically required to be non-trivial [5], in particular a piece of information must not be explained by

itself. This can be achieved easily here by selecting appropriate axiom patterns and concept variable ranges. As a side-effect, restricting the set of abducibles cuts the search space and the number of hypotheses generated and may therefore increase efficiency. Note that the limitation of the size of Pat in Definition 2 is required to ensure a polynomial worst-case complexity of the algorithm, yet it never posed a severe limitation for domain experts in practice.

Definition 3 (Abducible). *Given $\mathcal{AP} = (\mathcal{T}, A_0 \sqsubseteq B_0, Pat, \mathcal{V}^{\mathrm{C}}, rng)$, the set of abducibles $Abd_{\mathcal{AP}}$ contains all axioms generated by normalizing the elements of Pat and instantiating them with concept names from rng, omitting axioms already contained in $\mathcal{T}$. Let $N_{\mathrm{C}}\prime$ denote the set of concept names N_{C} extended with the new concept names introduced during normalization.*

Definition 4 (Labeling function). *Let $\mathcal{AP} = (\mathcal{T}, A_0 \sqsubseteq B_0, Pat, \mathcal{V}^{\mathrm{C}}, rng)$ be an abduction problem. Assume that each axiom ax in $\mathcal{T}$ and each abducible abd in $Abd_{\mathcal{AP}}$ is labeled with a unique propositional variable l_{ax} and l_{abd}, respectively, such that the sets of axiom labels and abducible labels are disjoint. The labeling function lab then assigns a label to each general concept inclusion gci as follows: If gci is an axiom (abducible), then $lab(gci)$ is the predefined propositional variable l_{ax} (l_{abd}). Otherwise, if gci is a tautology of the form $A \sqcap A \sqsubseteq A$ or $A \sqcap A \sqsubseteq \top$, we set $lab(gci) = \top$; in all other cases $lab(gci) = \bot$. We finally denote by $lab(\mathcal{AP})$ the set of all labels occurring in the abduction problem.*

To simplify notation we identify a propositional valuation $\mathcal{V}$ with the set of variables it assigns to be true, and let $\mathcal{A}_{|\mathcal{V}} = \{ax \in \mathcal{A} \mid lab(ax) \in \mathcal{V}\}$ denote the restriction of an axiom set $\mathcal{A}$ to the axioms made true by $\mathcal{V}$. We extend this definition to axiom problems by letting $\mathcal{AP}_{|\mathcal{V}} = (\mathcal{T} \cup Abd_{\mathcal{AP}})_{|\mathcal{V}}$.

Definition 5 (Hypotheses formula). *A* hypotheses formula *for an abduction problem $\mathcal{AP} = (\mathcal{T}, A_0 \sqsubseteq B_0, Pat, \mathcal{V}^{\mathrm{C}}, rng)$ is a monotone Boolean formula $\eta_{\mathcal{AP}}$ over $lab(\mathcal{AP})$ such that for all valuations $\mathcal{V} \subseteq lab(\mathcal{AP})$ it holds that $\mathcal{V} \models \eta_{\mathcal{AP}}$ iff $\mathcal{AP}_{|\mathcal{V}} \models A_0 \sqsubseteq B_0$.*

Abductive inference on the original knowledge base $\mathcal{T}$ can now be expressed as a pinpointing problem in the extended problem space $\mathcal{T} \cup Abd_{\mathcal{AP}}$. To this end, we define an abductive automaton employing the approach proposed in [17].

Definition 6 (Abductive automaton; behavior). *An* abductive automaton *for an abduction problem $\mathcal{AP} = (\mathcal{T}, A_0 \sqsubseteq B_0, Pat, \mathcal{V}^{\mathrm{C}}, rng)$ is a weighted Büchi automaton $\mathcal{A}_{\mathcal{AP}} = \{Q, wt, in, F\}$ over binary trees with*

- $Q = \{(A, B), (A, r, B) \mid A, B \in N_{\mathrm{C}}\prime \cup \{\top\}, r \in N_{\mathrm{R}}\}$,
- $\forall A, B, B_1, B_2 \in N_{\mathrm{C}}\prime \cup \{\top\}, \forall r \in N_{\mathrm{R}}$
  - $wt((A, B), (A, B_1), (A, B_2)) = lab(B_1 \sqcap B_2 \sqsubseteq B)$,
  - $wt((A, r, B), (A, B_1), (A, A)) = lab(B_1 \sqsubseteq \exists r.B)$,
  - $wt((A, B), (A, r, B_1), (B_1, B_2)) = lab(\exists r.B_2 \sqsubseteq B)$,
  - $wt(q_1, q_2, q_3) = \bot$ *for all other* $q_1, q_2, q_3 \in Q$,
- $in(q) = \top$ *iff* $q = (A_0, B_0)$, *otherwise* $in(q) = \bot$, *and*

– $F = \{(A, A) \mid A \in N_{\mathrm{C}}\prime \cup \{\top\}\}$,

where Q denotes the set of states, $F \subseteq Q$ the set of terminal states, in the initial distribution, and wt the transition weights of $\mathcal{A}_{\mathcal{AP}}$.

We extend the definition of wt to a complete run $\overrightarrow{r} = q_1 \cdots q_n$ as $wt(\overrightarrow{r}) = wt(q_1) \wedge \cdots \wedge wt(q_n)$, and let $succ(q)$ be the set of all successful runs of $\mathcal{A}_{\mathcal{AP}}$ starting in q. The behavior *of $\mathcal{A}_{\mathcal{AP}}$ is defined by $\bigwedge_{q \in Q}(in(q) \wedge \bigvee_{\overrightarrow{r} \in succ(q)} wt(\overrightarrow{r}))$.*

As there is exactly one state q having $in(q) \neq \bot$, namely (A_0, B_0), the behavior of $\mathcal{A}_{\mathcal{AP}}$ is the disjunction of the weights of all its successful runs starting in (A_0, B_0). Due to the specification of the transition weights, each run corresponds to a derivation of $A_0 \sqsubseteq B_0$. Intuitively, *wt* attributes triples (q_1, q_2, q_3) of states with provenance information regarding the derivation of q_1 from q_2 and q_3: Trivial derivation steps (such as $q_1 = (A, \top)$ or $q_1 = q_2 = q_3$) are labeled with the symbol $\top$ due to Definition 4; the weight of a non-trivial step is the label of an axiom / abducible such that q_1 can be deduced from q_2 and q_3 using this axiom / abducible (or $\bot$ if none exists). As an example, the definition $wt((A, B), (A, B_1), (A, B_2)) = lab(B_1 \sqcap B_2 \sqsubseteq B)$ expresses that, given $A \sqsubseteq B_1$ and $A \sqsubseteq B_2$, we can derive $A \sqsubseteq B$ if we know $B_1 \sqcap B_2 \sqsubseteq B$.

Theorem 1. *Given an abduction problem $\mathcal{AP} = (\mathcal{T}, A_0 \sqsubseteq B_0, Pat, \mathcal{V}^{\mathrm{C}}, rng)$, the behavior of the abductive automaton $\mathcal{A}_{\mathcal{AP}}$ is a hypotheses formula for the observation $A_0 \sqsubseteq B_0$.*

This result carries over from [17]. In fact, if we set $Pat = \emptyset$, the abductive automaton and hypotheses formula defined before coincide with the notions of pinpointing automaton and pinpointing formula due to the empty space of abducibles. If $Abd_{\mathcal{AP}}$ is nonempty, the automaton $\mathcal{A}_{\mathcal{AP}}$ can be interpreted as a pinpointing automaton for TBox $\mathcal{T}\prime = \mathcal{T} \cup Abd_{\mathcal{AP}}$ as noted before. Due to space limitations the reader is referred to [16, 18] for details on how to compute the behavior of such an automaton effectively. In the setting introduced above this can even be done efficiently, as the following theorem claims.

Theorem 2. *Given an abduction problem $\mathcal{AP} = (\mathcal{T}, A_0 \sqsubseteq B_0, Pat, \mathcal{V}^{\mathrm{C}}, rng)$, computing the hypotheses formula $\eta_{\mathcal{AP}}$ takes polynomial time in the size of $\mathcal{T}$.*

Proof. Given $\mathcal{AP} = (\mathcal{T}, A_0 \sqsubseteq B_0, Pat, \mathcal{V}^{\mathrm{C}}, rng)$, we denote by N_{C} and N_{R} the sets of concept and role names in $\mathcal{T}$, and by $N_{\mathrm{C}}\prime$ the extended set of concept names including the new names generated during normalization of the axiom patterns in Pat. As motivated before we can regard $\mathcal{A}_{\mathcal{AP}}$ as a pinpointing automaton for the extended problem space $\mathcal{T} \cup Abd_{\mathcal{AP}}$, whose behavior can be computed with an algorithm that is polynomial in the number of states of the automaton as shown in [17]. Following the construction given in Definition 6, $\mathcal{A}_{\mathcal{AP}}$ has $\binom{|N_{\mathrm{C}}\prime|+1}{2}$ states of type (A, B) plus $\binom{|N_{\mathrm{C}}\prime|+1}{2} * \binom{|N_{\mathrm{R}}|}{1}$ states of type (A, r, B), which is polynomial in $N_{\mathrm{C}}\prime$ and N_{R}. To complete the proof, we therefore have to show that N_{C} does not grow too fast during normalization of Pat, more concretely we require that $|N_{\mathrm{C}}\prime| = poly(|N_{\mathrm{C}}|)$ (normalization of $\mathcal{EL}$ axiom

patterns introduces no new role names at all). To this end, observe that the number of new concept names introduced by normalizing a set of axioms is linear in the number of axioms in the set [11]. Therefore, $|N_C\prime| \leq |N_C| + c*|Pat|$ for some constant c which can be chosen independently of N_C. Since the number of axiom patters is bounded polynomially by the size of N_C in Definition 2, this proves the polynomial bound on the size of $N_C\prime$ and therefore also on the size of $\mathcal{A}_{\mathcal{AP}}$. Also note that the size of the abductive automaton and thus the complexity of the proposed approach are independent of the number of concept variables used since variables cannot induce new states in $\mathcal{A}_{\mathcal{AP}}$. □

In assistive diagnosis, it is often convenient to be able to compare explanations of different, competing diagnoses (called a differential diagnosis in medicine). The abduction method proposed here naturally meets this demand, as the only part of the automaton that depends on the observation $A_0 \sqsubseteq B_0$ is the initial distribution *in*. To derive the hypotheses formula for a different observation $A_1 \sqsubseteq B_1$, the complete automaton $\mathcal{A}_{\mathcal{AP}}$ can be re-used without any modification to determine the successful runs starting in (A_1, B_1).

To conclude this section, we give an intuition of how the hypotheses formula generated by $\mathcal{A}_{\mathcal{AP}}$ can be interpreted. $\eta_{\mathcal{AP}}$ compactly encodes all possible derivations of $A_0 \sqsubseteq B_0$ w.r.t. $\mathcal{T}$ and $Abd_{\mathcal{AP}}$. An explicit representation of the set of hypotheses can be derived in a straightforward manner by transforming it $\eta_{\mathcal{AP}}$ into disjunctive normal form, each clause representing a single hypothesis. This approach is obviously not optimal since it may lead to an exponential blowup [16], a real-world system should therefore directly present, interpret and manipulate the compact representation $\eta_{\mathcal{AP}}$ whenever possible. Note that the hypotheses formula carries information on both necessary assumptions and axioms required to justify $A_0 \sqsubseteq B_0$. The proposed approach can therefore be seen to integrate and complement axiom pinpointing by allowing to infer reasons for unwanted entailments to hold as well as for expected subsumptions not to hold. This provides additional capabilities which may be useful among others for ontology debugging and refactoring. If one is only interested in determining necessary assumptions but not in their interactions with the axioms from the domain model, the approach can easily be adapted by adding only labels for abducibles to the hypotheses formula, leading to a much more compact $\eta_{\mathcal{AP}}$.

4   Industrial Scenario

This section illustrates the proposed approach by applying it to a use case in industrial diagnosis. Real-world models in this scenario typically consist of thousands of components and subcomponents, for most of which one can observe certain symptoms indicating possible failure states of the system. More often than not, the causal structure of the domain is at least partially unknown, models for diagnosis therefore have to be built on experience, relating sets of symptoms to diagnoses determined by a technician checking the system. We focus on assistive diagnosis, where sensor data and observations made by maintenance

personnel are used to interactively diagnose the system by actively requesting missing observations.

For our necessarily simplified scenario, we consider a CNC lathe with two components surveyed by sensors: the axle motor, and the oil pump of the motor cooling system. Sensors mounted at the axle motor can recognize vibrations and increased temperature, the monotored parameters for the oil pump include the current voltage. We assume that the measurements of these sensors are enough to recognize two different failure states, namely an untrue axle (characterized by vibrations and high axle motor temperature) and a power failure in the axle cooling system (defined by an overheating motor and low oil pump voltage). A system having an axle cooling failure, for example, can be represented by the following $\mathcal{EL}$ axiom:

$$\exists hasComp.(AxleMotor \sqcap \exists hasSymp.HiTemp) \sqcap$$
$$\exists hasComp.(OilPump \sqcap \exists hasSymp.LowVoltage) \sqsubseteq$$
$$\exists hasDiag.AxleCoolFail$$

Normalizing the axiom results in the normal form axioms

$$Has_{\text{HotAM}}^{\text{Comp}} \sqcap Has_{\text{DeadOP}}^{\text{Comp}} \sqsubseteq System_{\text{ACF}} \tag{1}$$

$$\exists hasComp.HotAM \sqsubseteq Has_{\text{HotAM}}^{\text{Comp}} \tag{2}$$

$$AxleMotor \sqcap Has_{\text{HiTemp}}^{\text{Symp}} \sqsubseteq HotAM \tag{3}$$

$$\exists hasSymp.HiTemp \sqsubseteq Has_{\text{HiTemp}}^{\text{Symp}} \tag{4}$$

$$\exists hasComp.DeadOP \sqsubseteq Has_{\text{DeadOP}}^{\text{Comp}} \tag{5}$$

$$OilPump \sqcap Has_{\text{LowVoltage}}^{\text{Symp}} \sqsubseteq DeadOP \tag{6}$$

$$\exists hasSymp.LowVoltage \sqsubseteq Has_{\text{LowVoltage}}^{\text{Symp}} \tag{7}$$

where $System_{\text{ACF}}$ is a new concept name defined by

$$System_{\text{ACF}} \equiv \exists hasDiag.AxleCoolFail$$

An untrue axle, the second diagnosis considered in this example, can defined and normalized analogously, leading to the following additional $\mathcal{EL}$ axioms in normal form:

$$Has_{\text{HotAM}}^{\text{Comp}} \sqcap Has_{\text{VibratAM}}^{\text{Comp}} \sqsubseteq System_{\text{UA}} \tag{8}$$

$$\exists hasComp.VibratAM \sqsubseteq Has_{\text{VibratAM}}^{\text{Comp}} \tag{9}$$

$$AxleMotor \sqcap Has_{\text{Vibrations}}^{\text{Symp}} \sqsubseteq VibratAM \tag{10}$$

$$\exists hasSymp.Vibrations \sqsubseteq Has_{\text{Vibrations}}^{\text{Symp}} \tag{11}$$

Having specified general (terminological) knowledge about the dependencies of certain symptoms and diagnoses, we now formalize the concrete system under consideration denoted by $System_{\text{Obs}}$, for which we have measured both an

increased axle temperature and low voltage in the system for pumping the oil used to cool the axle motor:

$$System_{\mathrm{Obs}} \sqsubseteq \exists hasComp.AxleMotor_{\mathrm{Obs}} \tag{12}$$

$$System_{\mathrm{Obs}} \sqsubseteq \exists hasComp.OilPump_{\mathrm{Obs}} \tag{13}$$

$$AxleMotor_{\mathrm{Obs}} \sqsubseteq AxleMotor \tag{14}$$

$$AxleMotor_{\mathrm{Obs}} \sqsubseteq \exists hasSymp.HiTemp \tag{15}$$

$$OilPump_{\mathrm{Obs}} \sqsubseteq OilPump \tag{16}$$

$$OilPump_{\mathrm{Obs}} \sqsubseteq \exists hasSymp.LowVoltage \tag{17}$$

Assume that the maintenance personnel wants to compare explanations for the diagnoses untrue axle and axle cooling failure to decide on further diagnostic or corrective steps. We then have two target states $q_0 = (System_{\mathrm{Obs}}, System_{\mathrm{ACF}})$ and $q_1 = (System_{\mathrm{Obs}}, System_{\mathrm{UA}})$ for which the hypotheses formula may be determined independently using the same abductive automaton $\mathcal{A}_{\mathcal{AP}}$ (with a modified definition of in). Regarding the space of abducibles, we regard the physical structure of the system as fixed and only allow for symptoms to be assumed. This can be done by defining $Pat = \{V_{\mathrm{Comp}} \sqsubseteq \exists hasSymp.V_{\mathrm{Symp}}\}$, where $rng(V_{\mathrm{Comp}}) = Component$ and $rng(V_{\mathrm{Symp}}) = Symptom$. The number of concept inclusions in $Abd_{\mathcal{AP}}$ is too large for an extensive listing even in this simple case, so we limit our presentation to one axiom in $Abd_{\mathcal{AP}}$ required to form a hypothesis for the diagnosis of an untrue axle:

$$AxleMotor_{\mathrm{Obs}} \sqsubseteq \exists hasSymp.Vibrations \tag{18}$$

For the same reason, we cannot present the complete automaton $\mathcal{A}_{\mathcal{AP}}$ here. Figure 1 depicts an excerpt containing one successful run for each diagnosis under consideration. These two runs actually correspond to the most natural hypotheses in terms of requiring the least number of assumptions to be made. Regular/ input/ terminal states are drawn as light/ medium/ dark rectangles, and light/ medium/ dark circles represent axiom labels, the tautology label $\top$, or the labels of abducibles, respectively. To keep the representation compact, we merge identical subtrees.

The weights of the runs from the two input nodes $(System_{\mathrm{Obs}}, System_{\mathrm{ACF}})$ and $(System_{\mathrm{Obs}}, System_{\mathrm{UA}})$ to the terminal (leaf) nodes represent two partial hypotheses formulas for the diagnoses $AxleCoolingFailure$ and $UntrueAxle$:

$$\eta_{\mathrm{ACF}}^{\mathrm{part}} = 1 \wedge (5 \wedge 13 \wedge (6 \wedge (16 \wedge \top) \wedge (7 \wedge 17)))$$
$$\wedge (2 \wedge 12 \wedge (3 \wedge (14 \wedge \top) \wedge (4 \wedge 15)))$$
$$\eta_{\mathrm{UA}}^{\mathrm{part}} = 8 \wedge (2 \wedge 12 \wedge (3 \wedge (14 \wedge \top) \wedge (4 \wedge 15)))$$
$$\wedge (9 \wedge 12 \wedge (10 \wedge (14 \wedge \top) \wedge (11 \wedge \mathbf{18})))$$

Comparing the two hypotheses, it shows that neither of them is clearly better than the other: On the one hand, an axle cooling failure is justified by the observations alone (requiring no assumptions to be made), yet it postulates faults

Fig. 1. Automaton for the diagnosis example (compacted excerpt)

in two distinct components. On the other hand, an untrue axle can be diagnosed locally for one component, it however requires the assumption of general concept inclusion axiom 18.

5   Conclusions and Future Work

We have presented a PTime procedure for TBox abduction in the lightweight description logic $\mathcal{EL}$ based on a novel reduction to axiom pinpointing, and demonstrated its applicability in an industrial diagnosis scenario. Given a knowledge base and a concept inclusion representing the observation to be explained, the procedure determines a hypotheses formula that compactly encodes all explanations with respect to a pattern-based representation of the abducible part of the domain model; the remainder of the model is considered to be fixed in accordance with our scenario. The proposed reduction of abductive inference to axiom pinpointing exploits the duality of the two tasks: whereas the latter addresses the problem of explaining why a certain unwanted subsumption is entailed by the ontology, our method determines the reason for an expected subsumption not to hold, expressed in terms of additions to the domain model necessary to actually make it hold.

We are currently working on extending the approach presented in this paper in several ways: Since role inclusion axioms and nominals are frequently used in diagnostic models, it is favorable to extend the logical expressivity as much as possible without sacrificing tractability. Additionally, including quantitative information into the model allows for weighting hypotheses and can eventually be used as a criterion for guiding hypothesis generation. Finally, extending minimality criteria for single hypotheses to sets of hypotheses compactly represented by a hypothesis formula will allow us to efficiently infer common effects (as proposed e. g. in [15]).

References

1. Hobbs, J.R., Stickel, M.E., Appelt, D.E., Martin, P.A.: Interpretation as abduction. Artificial Intelligence **63**(1-2) (1993) 69–142
2. Appelt, D.E., Pollack, M.E.: Weighted abduction for plan ascription. In: User Modeling and User-Adapted Interaction. Volume 2., Springer (1992) 1–25
3. Shanahan, M.: Perception as abduction: Turning sensor data into meaningful representation. Cognitive Science **29**(1) (2005) 103–134
4. Peraldi, I.S.E., Kaya, A., Melzer, S., Möller, R., Wessel, M.: Towards a media interpretation framework for the semantic web. In: Proceedings of the 2007 IEEE / WIC / ACM International Conference on Web Intelligence (WI 2007), Silicon Valley, California, USA, IEEE Computer Society (November 2-5 2007)
5. Paul, G.: Approaches to abductive reasoning: An overview. Artificial Intelligence Review **7**(2) (1993) 109–152
6. Eiter, T., Gottlob, G.: The complexity of logic-based abduction. Journal of the ACM **42**(1) (1995) 3–42

7. Selman, B., Levesque, H.J.: Abductive and default reasoning: A computational core. In: Proceedings of the 8th National Conference on Artificial Intelligence (AAAI 1990), Boston, Massachusetts, USA, AAAI Press / The MIT Press (July 29 - August 3 1990) 343–348

8. Eiter, T., Makino, K.: On computing all abductive explanations. In: Proceedings of the 21st National Conference on Artificial Intelligence and the 18th Innovative Applications of Artificial Intelligence Conference (AAAI 2006 + IAAI 2006), Edmonton, Alberta, Canada, AAAI Press (July 28 - August 1 2002) 62–67

9. Bienvenu, M.: Complexity of abduction in the $\mathcal{EL}$ family of lightweight description logics. In Brewka, G., Lang, J., eds.: Proceedings of the 11th International Conference on Principles of Knowledge Representation and Reasoning (KR 2008), Sydney, Australia, AAAI Press (September 16-19 2008) 220–230

10. Baader, F., Brandt, S., Lutz, C.: Pushing the $\mathcal{EL}$ envelope. In Kaelbling, L.P., Saffiotti, A., eds.: Proceedings of the 19th International Joint Conference on Artificial Intelligence (IJCAI 2005), Edinburgh, Scotland, UK, Professional Book Center (July 30 - August 5 2005) 364–369

11. Baader, F., Brandt, S., Lutz, C.: Pushing the $\mathcal{EL}$ envelope. LTCS-Report LTCS-05-01, Institute for Theoretical Computer Science, TU Dresden (2005)

12. Colucci, S., Di Noia, T., Di Sciascio, E., Donini, F.M., Mongiello, M.: Concept abduction and contraction in description logics. In Calvanese, D., De Giacomo, G., Franconi, E., eds.: Proceedings of the 16th International Workshop on Description Logics (DL 2003). Volume 81 of CEUR Workshop Proceedings., Rome, Italy, CEUR-WS.org (September 5-7 2003)

13. Di Noia, T., Di Sciascio, E., Donini, F.M.: A tableaux-based calculus for abduction in expressive description logics: Preliminary results. In Grau, B.C., Horrocks, I., Motik, B., Sattler, U., eds.: Proceedings of the DL Home 22nd International Workshop on Description Logics (DL 2009). Volume 477 of CEUR Workshop Proceedings., Oxford, UK, CEUR-WS.org (July 27-30 2009)

14. Kakas, A.C., Kowalski, R.A., Toni, F.: Abductive logic programming. Journal of Logic and Computation **2**(6) (December 1992) 719–770

15. Baral, C.: Abductive reasoning through filtering. Artificial Intelligence **120**(1) (2000) 1–28

16. Baader, F., Peñaloza, R.: Automata-based axiom pinpointing. In Armando, A., Baumgartner, P., Dowek, G., eds.: Proceedings of the 4th International Joint Conference on Automated Reasoning (IJCAR 2008). Volume 5195 of Lecture Notes in Artificial Intelligence., Sydney, Australia, Springer (August 12-15 2008) 226–241

17. Peñaloza, R.: Using tableaux and automata for pinpointing in $\mathcal{EL}$. In Goranko, V., ed.: Proceedings of the TABLEAUX 2009 Wokshop on Tableaux versus Automata as Logical Decision Methods (AutoTab 2009). (2009)

18. Baader, F., Peñaloza, R.: Automata-based axiom pinpointing. Journal of Automated Reasoning **Special Issue: IJCAR 2008** (2009)

Decidability of Description Logics with Transitive Closure of Roles in Concept and Role Inclusion Axioms

Chan Le Duc[1] and Myriam Lamolle[1]

LIASD Université Paris8 - IUT de Montreuil
140, rue de la Nouvelle France - 93100 Montreuil
{chan.leduc, myriam.lamolle}@iut.univ-paris8.fr

Abstract. This paper investigates Description Logics which allow transitive closure of roles to occur not only in concept inclusion axioms but also in role inclusion axioms. First, we propose a decision procedure for the description logic $\mathcal{SHIO}_+$, which is obtained from $\mathcal{SHIO}$ by adding transitive closure of roles. Next, we show that $\mathcal{SHIO}_+$ has the *finite model property* by providing a upper bound on the size of models of satisfiable $\mathcal{SHIO}_+$-concepts with respect to sets of concept and role inclusion axioms. Additionally, we prove that if we add number restrictions to $\mathcal{SHI}_+$ then the satisfiability problem is undecidable.

Introduction

The ontology language OWL-DL is widely used to formalize resources on the Semantic Web. This language is mainly based on the description logic $\mathcal{SHOIN}$ which is known to be decidable [1]. Although $\mathcal{SHOIN}$ is expressive and provides *transitive roles* to model transitivity of relations, we can find several applications in which *the transitive closure of roles*, that is more expressive than transitive roles, is necessary. An example in [2] describes two categories of devices as follows: (1) Devices have as their direct part a battery: Device $\sqcap \exists$hasPart.Battery, (2) Devices have at some level of decomposition a battery: Device $\sqcap \exists$hasPart$^+$.Battery. However, if we now define hasPart as a *transitive role*, the concept Device $\sqcap \exists$hasPart.Battery does not represent the devices as described above since it does not allow one to describe these categories of devices as two different sets of devices. We now consider another example in which we need to use the transitive closure of roles in role inclusion axioms.

Example 1. A process accepts a set **S** of possible states where start $\in$ **S** is an initial state. The process can reach two disjoint phases A, B $\subseteq$ **S**, considered as two sets of states. To go from a state to another one, the process has to perform an action next. Sometimes, it can execute a jump that implies a sequence of actions next.

To specify the behavior of the process as described, we might need a role name next to express the fact that a state follows another one, a nominal o for start, a role name jump for jumps, concept names A, B for the phases and the following axioms:
(1) o $\sqsubseteq \neg$A $\sqcap \neg$B, A $\sqcap$ B $\sqsubseteq \bot$, o $\sqsubseteq \forall$next$^-.\bot$
(2) $\top \sqsubseteq \exists$next.$\top$, jump $\sqsubseteq$ next$^+$
Since jumps are arbitrarily executed over **S** and they form (non-directed) cycles with next instances, we cannot use concept axioms to express them. In addition, if a transitive

role is used instead of transitive closure, we cannot express the property : an execution of jump implies a sequence of actions next. Therefore, the axiom jump $\sqsubseteq$ next$^+$ is necessary.

Such examples motivate the study of Description Logics (DL) that allow the transitive closure of roles to occur in both concept and role inclusion axioms. We introduce in this work a DL that can express the process as described in Ex.1 and propose a decision procedure for concept satisfiability problem in this DL. To the best of our knowledge, the decidability of $\mathcal{SHIO}_+$, which is obtained from $\mathcal{SHIO}$ by adding transitive closure of roles, is unknown. [3] has established a decision procedure for concept satisfiability in $\mathcal{SHI}_+$ by using neighborhoods to build completion graphs. In the literature, many decidability results in DLs can be obtained from their counterparts in modal logics [4], [5]. However, these counterparts do not take into account expressive role inclusion axioms. In particular, [5] has shown the decidability of a very expressive DL, so-called $\mathcal{CATS}$, including $\mathcal{SHIQ}$ with the transitive closure of roles but not allowing it to occur in role inclusion axioms. [5] has pointed out that the complexity of concept subsumption in $\mathcal{CATS}$ is ExpTime-complete by translating $\mathcal{CATS}$ into the logic Converse PDL in which inference problems are well studied.

Recently, there have been some works (e.g. in [6]) which have attempted to augment the expressiveness of role inclusion axioms. A decidable logic, namely $\mathcal{SROIQ}$, resulting from these efforts allows for new role constructors such as composition, disjointness and negation. In addition, [7] has introduced a DL, so-called $\mathcal{ALCQI}b_{reg}^+$, which can capture $\mathcal{SRIQ}$, and obtained the worst-case complexity (ExpTime-complete) of the satisfiability problem by using automata-based technique. $\mathcal{ALCQI}b_{reg}^+$ allows for a rich set of operators on roles by which one can simulate role inclusion axioms. However, transitive closures in role inclusion axioms are expressible neither in $\mathcal{SROIQ}$ nor in $\mathcal{ALCQI}b_{reg}^+$.

Tableaux-based algorithms for expressive DLs like $\mathcal{SHIQ}$ [8] and $\mathcal{SROIQ}$ [6] result in efficient implementations. This kind of algorithms relies on two structures, the so-called *tableau* and *completion graph*. Roughly speaking, a tableau for a concept represents a model for the concept and it is possibly infinite. A tableau translates satisfiability of all given concept and role inclusion axioms into the satisfiability of semantic constraints imposed *locally* on each individual of the tableau. This feature of tableaux will be called *local satisfiability property*. The algorithm in [9] for satisfiability in $\mathcal{ALC}_{reg}$ (including the transitive closure of roles and other role operators) introduced a method to deal with loops which can hide unsatisfiable nodes.

To check satisfiability of a concept, tableaux-based algorithms try to build a completion graph whose finiteness is ensured by a technique, the so-called *blocking technique*. It provides a termination condition and guarantees soundness and completeness. The underlying idea of the blocking mechanism is to detect "loops" which are repeated pieces of a completion graph.

The contribution of the present paper consists of (i) proving that $\mathcal{SHIO}_+$ is decidable and it has the *finite model property* by providing a upper bound on the size of models of satisfiable $\mathcal{SHIO}_+$-concepts with respect to (w.r.t.) sets of concept and role inclusion axioms, (ii) establishing a reduction of the domino problem to the concept satisfiability

problem in the logic $\mathcal{SHIN}_+$ that is obtained from $\mathcal{SHI}_+$ by adding number restrictions on *simple* roles. This reduction shows that $\mathcal{SHIN}_+$ is undecidable.

The Description Logic $\mathcal{SHIO}_+$

The logic $\mathcal{SHIO}_+$ is an extension of $\mathcal{SHIO}$ by allowing for transitive closure of roles. In this section, we present the syntax and semantics of $\mathcal{SHIO}_+$. The definitions reuse notation introduced in [8].

Definition 1. *Let $\mathbf{R}$ be a non-empty set of* role names. *We denote $\mathbf{R}_\mathsf{l} = \{P^- \mid P \in \mathbf{R}\}$, $\mathbf{R}_+ = \{Q^+ \mid Q \in \mathbf{R} \cup \mathbf{R}_\mathsf{l}\}$. The set of $\mathcal{SHIO}_+$-roles is $\mathbf{R} \cup \mathbf{R}_\mathsf{l} \cup \mathbf{R}_+$. A role* inclusion axiom *is of the form $R \sqsubseteq S$ for two $\mathcal{SHIO}_+$-roles R and S. A* role hierarchy *$\mathcal{R}$ is a finite set of role inclusion axioms.*

$*$ *An interpretation $\mathcal{I} = (\Delta^\mathcal{I}, \cdot^\mathcal{I})$ consists of a non-empty set $\Delta^\mathcal{I}$ (*domain*) and a function $\cdot^\mathcal{I}$ which maps each role name to a subset of $\Delta^\mathcal{I} \times \Delta^\mathcal{I}$ such that, for $R \in \mathbf{R}_\mathsf{l}$, $Q^+ \in \mathbf{R}_+$,*

$$R^{-\mathcal{I}} = \{\langle x, y \rangle \in (\Delta^\mathcal{I})^2 \mid \langle y, x \rangle \in R^\mathcal{I}\}, \text{ and } Q^{+\mathcal{I}} = \bigcup_{n>0} (Q^n)^\mathcal{I} \text{ with } (Q^1)^\mathcal{I} = Q^\mathcal{I},$$

$(Q^n)^\mathcal{I} = \{\langle x, y \rangle \in (\Delta^\mathcal{I})^2 \mid \exists z \in \Delta^\mathcal{I}, \langle x, z \rangle \in (Q^{n-1})^\mathcal{I}, \langle z, y \rangle \in Q^\mathcal{I}\}.$
An interpretation $\mathcal{I}$ satisfies a role hierarchy $\mathcal{R}$ if $R^\mathcal{I} \subseteq S^\mathcal{I}$ for each $R \sqsubseteq S \in \mathcal{R}$. Such an interpretation is called a model *of $\mathcal{R}$, denoted by $\mathcal{I} \models \mathcal{R}$.*

$*$ *Function Inv returns the inverse of a role as follows:*

$$\mathsf{Inv}(R) := \begin{cases} R^- & \text{if } R \in \mathbf{R}, \\ S & \text{if } R = S^- \text{ where } S \in \mathbf{R}, \\ (Q^-)^+ & \text{if } R = Q^+ \text{ where } Q \in \mathbf{R}, \\ Q^+ & \text{if } R = (Q^-)^+ \text{ where } Q \in \mathbf{R} \end{cases}$$

$*$ *A relation $\mathrel{\underline{\underline{\sqsubseteq}}}$ is defined as the transitive-reflexive closure of $\sqsubseteq$ on $\mathcal{R} \cup \{\mathsf{Inv}(R) \sqsubseteq \mathsf{Inv}(S) \mid R \sqsubseteq S \in \mathcal{R}\} \cup \{Q \sqsubseteq Q^+ \mid Q \in \mathbf{R} \cup \mathbf{R}_\mathsf{l}\}$. We denote $S \equiv R$ iff $R \mathrel{\underline{\underline{\sqsubseteq}}} S$ and $S \mathrel{\underline{\underline{\sqsubseteq}}} R$. We may abuse the notation by saying $R \mathrel{\underline{\underline{\sqsubseteq}}} S \in \mathcal{R}$.*

Notice that we introduce into role hierarchies axioms $Q \sqsubseteq Q^+$ which allows us (i) to propagate $(\forall Q^+.A)$ correctly, and (ii) to take into account the fact that $R \sqsubseteq S$ implies $R^+ \sqsubseteq S^+$.

Definition 2. *Let $\mathbf{C}' = \mathbf{C} \cup \mathbf{C}_o$ be a non-empty set of* concept names *where $\mathbf{C}$ is a set of normal concept names and $\mathbf{C}_o$ is a set of* nominals.

$*$ *The set of $\mathcal{SHIO}_+$-concepts is inductively defined as the smallest set containing all C in $\mathbf{C}'$, $\top$, $C \sqcap D$, $C \sqcup D$, $\neg C$, $\exists R.C$, $\forall R.C$ where C and D are $\mathcal{SHIO}_+$-concepts, R is an $\mathcal{SHIO}_+$-role, S is a simple role and $n \in \mathbb{N}$. We denote $\bot$ for $\neg\top$.*

$*$ *An interpretation $\mathcal{I} = (\Delta^\mathcal{I}, \cdot^\mathcal{I})$ consists of a non-empty set $\Delta^\mathcal{I}$ (*domain*) and a function $\cdot^\mathcal{I}$ which maps each concept to a subset of $\Delta^\mathcal{I}$ such that $\mathsf{card}\{o^\mathcal{I}\} = 1$ for all $o \in \mathbf{C}_o$ where $\mathsf{card}\{\cdot\}$ is denoted for the cardinality of a set $\{\cdot\}$,*
$\top^\mathcal{I} = \Delta^\mathcal{I}$, $(C \sqcap D)^\mathcal{I} = C^\mathcal{I} \cap D^\mathcal{I}$, $(C \sqcup D)^\mathcal{I} = C^\mathcal{I} \cup D^\mathcal{I}$, $(\neg C)^\mathcal{I} = \Delta^\mathcal{I} \backslash C^\mathcal{I}$,
$(\exists R.C)^\mathcal{I} = \{x \in \Delta^\mathcal{I} \mid \exists y \in \Delta^\mathcal{I}, \langle x, y \rangle \in R^\mathcal{I} \wedge y \in C^\mathcal{I}\}$,

$(\forall R.C)^{\mathcal{I}} = \{x \in \Delta^{\mathcal{I}} \mid \forall y \in \Delta^{\mathcal{I}}, \langle x, y \rangle \in R^{\mathcal{I}} \Rightarrow y \in C^{\mathcal{I}}\}$

$* C \sqsubseteq D$ *is called a* general concept inclusion (GCI) *where* C, D *are* $\mathcal{SHIO}_+$-*concepts (possibly complex), and a finite set of GCIs is called a terminology* $\mathcal{T}$*. An interpretation* $\mathcal{I}$ *satisfies a GCI* $C \sqsubseteq D$ *if* $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ *and* $\mathcal{I}$ *satisfies a terminology* $\mathcal{T}$ *if* $\mathcal{I}$ *satisfies each GCI in* $\mathcal{T}$*. Such an interpretation is called a* model *of* $\mathcal{T}$*, denoted by* $\mathcal{I} \models \mathcal{T}$*.*

$*$ *A concept* C *is called* satisfiable *w.r.t. a role hierarchy* $\mathcal{R}$ *and a terminology* $\mathcal{T}$ *iff there is some interpretation* $\mathcal{I}$ *such that* $\mathcal{I} \models \mathcal{R}$*,* $\mathcal{I} \models \mathcal{T}$ *and* $C^{\mathcal{I}} \neq \emptyset$*. Such an interpretation is called a* model *of* C *w.r.t.* $\mathcal{R}$ *and* $\mathcal{T}$*. A pair* $(\mathcal{T}, \mathcal{R})$ *is called an* $\mathcal{SHIO}_+$ *ontology and said to be consistent if there is a model of* $(\mathcal{T}, \mathcal{R})$*. A concept* D subsumes *a concept* C *w.r.t.* $\mathcal{R}$ *and* $\mathcal{T}$*, denoted by* $C \sqsubseteq D$*, if* $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ *holds in each model* $\mathcal{I}$ *of* $(\mathcal{T}, \mathcal{R})$*.*

Notice that a transitive role S can be expressed by using a role axiom $S^+ \sqsubseteq S$. Since negation is allowed in the logic $\mathcal{SHIO}_+$, unsatisfiability and subsumption w.r.t. $(\mathcal{T}, \mathcal{R})$ can be reduced each other: $C \sqsubseteq D$ iff $C \sqcap \neg D$ is unsatisfiable. In addition, we can reduce ontology consistency to concept satisfiability w.r.t. an ontology: $(\mathcal{T}, \mathcal{R})$ is consistent if $A \sqcup \neg A$ is satisfiable w.r.t. $(\mathcal{T}, \mathcal{R})$ for some concept name A.

For the ease of construction, we assume all concepts to be in *negation normal form* (NNF) i.e. negation occurs only in front of concept names. Any $\mathcal{SHIO}_+$-concept can be transformed to an equivalent one in NNF by using DeMorgan's laws and some equivalences as presented in [8]. For a concept C, we denote the nnf of C by $\mathsf{nnf}(C)$ and the nnf of $\neg C$ by $\dot{\neg} C$.

Let D be an $\mathcal{SHIO}_+$-concept in NNF. We define $\mathsf{sub}(D)$ to be the smallest set that contains all sub-concepts of D including D. For an ontology $(\mathcal{T}, \mathcal{R})$, we define the set of all sub-concepts $\mathsf{sub}(\mathcal{T}, \mathcal{R})$ as follows:

$$\mathsf{sub}(\mathcal{T}, \mathcal{R}) := \bigcup_{C \sqsubseteq D \in \mathcal{T}} \mathsf{sub}(\mathsf{nnf}(\neg C \sqcup D), \mathcal{R})$$

$$\mathsf{sub}(E, \mathcal{R}) := \mathsf{sub}(E) \cup \{\dot{\neg} C \mid \neg C \in \mathsf{sub}(E)\} \cup \{\forall S.C \mid (\forall R.C \in \mathsf{sub}(E), S \not\sqsubseteq^{*} R) \vee \\ (\dot{\neg} \forall R.C \in \mathsf{sub}(E), S \not\sqsubseteq^{*} R) \text{ and } S \text{ occurs in } \mathcal{T} \text{ or } \mathcal{R}\}$$

For the sake of simplicity, for each concept D w.r.t. $(\mathcal{T}, \mathcal{R})$ we denote $\mathsf{sub}(\mathcal{T}, \mathcal{R}, D)$ for $\mathsf{sub}(\mathcal{T}, \mathcal{R}) \cup \mathsf{sub}(D)$, and $\mathbf{R}_{(\mathcal{T}, \mathcal{R}, D)}$ for the set of roles R occurring in $\mathcal{T}, \mathcal{R}, D$ with the inverse and transitive closure of each R. If it is clear from the context we will use $\mathbf{R}$ instead of $\mathbf{R}_{(\mathcal{T}, \mathcal{R}, D)}$.

A decision procedure for $\mathcal{SHIO}_+$

In our approach, we define a sub-structure of graphs, called *neighborhood*, which consists of a node together with its neighbors. Such a neighborhood captures all semantic constraints imposed by the logic constructors of $\mathcal{SHIO}$. A graph obtained by "tiling" neighborhoods together allows us to represent in some way a model for a concept in $\mathcal{SHIO}_+$. In fact, we embed in this graph another structure, called *cyclic path*, to express transitive closure of roles. Since all expansion rules for $\mathcal{SHIO}$ can be translated into construction of neighborhoods, the algorithm presented in this paper focuses on defining cyclic paths over such a graph. In this way, the non-determinism resulting from satisfying the transitive closure of roles can be translated into the search in a space of all possible graphs obtained from tiling neighborhoods.

Neighborhood for $\mathcal{SHIO}_+$

Tableau-based algorithms, as presented in [8], use expansion rules representing tableau properties to build a completion graph. Applying expansion rules makes all nodes of a completion graph satisfy semantic constraints imposed by concept definitions in the label associated with each node. This means that *local* satisfiability in such completion graphs is sufficient to ensure *global* satisfiability. The notion of *neighborhood* introduced in Def. 3 expresses exactly the expansion rules for $\mathcal{SHIO}$, consequently, guarantees local satisfiability. Therefore, a completion graph built by a tableau-based algorithm can be considered as set of neighborhoods which are tiled together. In other terms, building a completion tree by applying expansion rules is equivalent to the search of a tiling of neighborhoods.

Definition 3 (Neighborhood). *Let D be an $\mathcal{SHIO}_+$ concept with a terminology $\mathcal{T}$ and role hierarchy $\mathcal{R}$. We denote $\mathbf{R}$ for the set of roles R occurring in D and $\mathcal{T}, \mathcal{R}$ with the inverse of each R. A* neighborhood*, denoted (v_B, N_B, l), for D w.r.t. $(\mathcal{T}, \mathcal{R})$ is formed from a* core *node v_B, a set of* neighbor *nodes N_B, edges $\langle v_B, v \rangle$ with $v \in N_B$ and a labelling function l such that $l(u) \in 2^{sub(\mathcal{T}, \mathcal{R}, D)}$ with $u \in \{v_B\} \cup N_B$ and $l\langle v_B, v \rangle \in 2^{\mathbf{R}}$ with $v \in N_B$.*

1. *A node $v \in \{v_B\} \cup N_B$ is* nominal *if there is $o \in \mathbf{C}_o$ such that $o \in l(v)$. Otherwise, v is a* non-nominal *node;*
2. *A node $v \in \{v_B\} \cup N_B$ is* valid *w.r.t. D and $(\mathcal{T}, \mathcal{R})$ iff*
    (a) *If $C \sqsubseteq D \in \mathcal{T}$ then $\mathsf{nnf}(\neg C \sqcup D) \in l(v)$, and*
    (b) *$\{A, \neg A\} \not\subseteq l(v)$ with each concept name A, and*
    (c) *If $C_1 \sqcap C_2 \in l(v)$ then $\{C_1, C_2\} \subseteq l(v)$, and*
    (d) *If $C_1 \sqcup C_2 \in l(v)$ then $\{C_1, C_2\} \cap l(v) \neq \emptyset$.*
3. *A neighborhood $B = (v_B, N_B, l)$ is* valid *iff all nodes $\{v_B\} \cup N_B$ are valid and the following conditions are satisfied:*
    (a) *If $\exists R.C \in l(v_B)$ then there is a neighbor $v \in N_B$ such that $C \in l_B(v)$ and $R \in l\langle v_B, v \rangle$;*
    (b) *For each $v \in N_B$, if $R \in l\langle v_B, v \rangle$ and $R \overset{*}{\sqsubseteq} S$ then $S \in l\langle v_B, v \rangle$;*
    (c) *For each $v \in N_B$, if $R \in l\langle v_B, v \rangle$ (resp. $R \in \mathsf{Inv}(l\langle v_B, v \rangle)$) and $\forall R.C \in l(v_B)$ (resp. $\forall R.C \in l(v)$) then $C \in l(v)$ (resp. $C \in l(v_B)$);*
    (d) *For each $v \in N_B$, if $Q^+ \in l\langle v_B, v \rangle$ (resp. $Q^+ \in \mathsf{Inv}(l\langle v_B, v \rangle)$), $Q^+ \overset{*}{\sqsubseteq} R \in \mathcal{R}$ and $\forall R.D \in l(v_B)$ (resp. $\forall \mathsf{Inv}(R).D \in l(v)$) then $\forall Q^+.D \in l(v)$ (resp. $\forall \mathsf{Inv}(Q^+).D \in l(v_B)$);*
    (e) *For each $o \in \mathbf{C}_o$, if $o \in l(u) \cap l(v)$ with $\{u, v\} \subseteq \{v_B\} \cup N_B$ then $l(u) = l(v)$;*
    (f) *There is at most one node $v \in N_B$ such that $l(v) = \mathscr{C}$ and $l\langle v_B, v \rangle = \mathscr{R}$ for each $\mathscr{C} \in 2^{sub(\mathcal{T}, \mathcal{R}, D)}, \mathscr{R} \in 2^{\mathbf{R}}$.*

We denote $\mathbb{B}_{(\mathcal{T}, \mathcal{R}, D)}$ for a set of all valid neighborhoods for D w.r.t. $(\mathcal{T}, \mathcal{R})$. When it is clear from the context we will use $\mathbb{B}$ instead of $\mathbb{B}_{(\mathcal{T}, \mathcal{R}, D)}$.

The condition 3f in Def. 3 ensures that any neighborhood has a finite number of neighbors. As mentioned, a valid neighborhood as presented in Def. 3 satisfies all concept definitions in the label associated with the core node. For this reason, neighborhoods

can be still used to tile a completion tree for $\mathcal{SHIO}_+$ without taking care of expansion rules for $\mathcal{SHIO}$. In other terms, the neighborhood notion expresses the local satisfiability property in a sufficient way for being used in a global context.

Lemma 1. *Let D be an $\mathcal{SHIO}_+$ concept with a terminology $\mathcal{T}$ and role hierarchy $\mathcal{R}$. Let $(v_B, N_B, l), (v_{B'}, N_{B'}, l)$ be two valid neighborhoods with $l(v_B) = l(v_{B'})$. If there is $v \in N_B$ such that there does not exist any $v' \in N_{B'}$ satisfying $l(v') = l(v)$ and $l\langle v_B, v \rangle = l\langle v_{B'}, v' \rangle$ then the neighborhood $(v_{B'}, N_{B'} \cup \{u\}, l)$ is valid where $l(u) = l(v)$ and $l\langle v_{B'}, u \rangle = l\langle v_B, v \rangle$.*

This lemma holds due to the facts that (i) a valid neighbor in a valid neighborhood B is also a valid neighbor in another valid neighborhood B' if the labels of two core nodes of B and B' are identical, (ii) since $\mathcal{SHIO}_+$ does not allow for number restrictions hence Def. 3 has no restriction on the number of neighbors of a core node.

Completion Tree with Cyclic Paths

As discussed in works related to tableau-based technique, the blocking technique fails in treating DLs with the transitive closure of roles. It works correctly only if the satisfiability of a node in completion tree can be decided from its neighbors and itself i.e. *local* satisfiability must be sufficient for such completion trees. However, the presence of the transitive closure of roles makes satisfiability of a node depend on further nodes which can be arbitrarily far from it. The problem becomes harder when we add the transitive closure of roles to role hierarchies. For instance, if $P \sqsubseteq Q^+, Q \sqsubseteq S^+$ are axioms in a role hierarchy then each Q-edge generated for satisfying Q^+ may lead to generate an arbitrary number of S-edges for satisfying S^+.

More precisely, satisfying the transitive closure P^+ in an edge $\langle x, y \rangle$ (i.e. $P^+ \in L\langle x, y \rangle$) is related to a set of nodes on a path rather than a node with its neighbors i.e. it imposes a semantic constraint on a set of nodes $x, x_1, \cdots, x_n, y$ such that they are connected together by P-edges. In general, satisfying the transitive closure is quite nondeterministic since the semantic constraint can lead to be applied to an *arbitrary* number of nodes. In addition, the presence of transitive closure of roles in a role hierarchy makes this difficulty worse. For instance, if $P \sqsubseteq Q^+, Q \sqsubseteq S^+$ are axioms in a role hierarchy then each Q-edge generated for satisfying Q^+ may lead to generate an arbitrary number of S-edges for satisfying S^+.

The most common way for dealing with a new logic constructor is to add a new expansion rule for satisfying the semantic constraint imposed by the new constructor. Such an expansion rule for the transitive closure of roles must: (i) find or create a set of P-edges forming a path for each occurrence of P^+ in the label of edges; (ii) deal with non-deterministic behaviours of the expansion rule resulting from the semantics of the transitive closure of roles; and (iii) enable to control the expansion of completion trees by a new blocking technique which has to take into account the fact that satisfying the transitive closure of a role may add an arbitrary number of new transitive closures to be satisfied. To avoid these difficulties, our approach does not aim to directly extend the construction of completion trees by using a new expansion rule, but to translate this construction into selecting a "good" completion tree, namely *completion tree with cyclic*

paths, from a finite set of trees without taking into account the semantic constraint imposed by the transitive closure of roles. The process of selecting a "good" completion tree is guided by finding in a completion tree (which is well built in advance) a *cyclic path* for each occurrence of the transitive closure of a role.

Summing up, a completion tree with cyclic paths will be built in two stages. The first one which yields a tree consists of tiling valid neighborhoods together such that two neighborhoods are tiled if they have *compatible* neighbors. The second stage deals with the transitive closure of roles by defining cyclic paths over the tree obtained from the first stage. Both of them are presented in Def. 4.

Definition 4 (Completion Tree with Cyclic Paths). *Let D be a $\mathcal{SHIO}_+$ concept with a terminology $\mathcal{T}$ and role hierarchy $\mathcal{R}$. Let $\mathbb{B}$ be the set of all valid neighborhoods for D w.r.t. $(\mathcal{T}, \mathcal{R})$. A tree $\mathbf{T} = (V, E, L)$ for D w.r.t. $(\mathcal{T}, \mathcal{R})$ is defined from $\mathbb{B}$ as follows.*

1. *If there is a valid neighborhood $(v_0, N_0, l) \in \mathbb{B}$ with $D \in l(v_0)$ then a root node x_0 and successors x of x_0 are added to V such that $L(x_0) = l(v_0)$, and $L(x) = l(v)$, $L\langle x_0, x\rangle = l\langle v_0, v\rangle$ for each $v \in N_0$.*
2. *For each node $x \in V$ with its predecessor x',*
   (a) *If there is an ancestor y of x such that $L(y) = L(x)$ then x is* blocked *by y. In this case, x is a leaf node;*
   (b) *Otherwise, if we find a valid neighborhood (v_B, N_B, l) from $\mathbb{B}$ such that*
      i. *$l(v_B) = L(x), l(v) = L(x'), \mathsf{Inv}(l\langle v_B, v\rangle) = L\langle x', x\rangle$ for some $v \in N_B$, and*
      ii. *if there is some nominal $o \in \mathbf{C}_o$ such that $o \in l(u) \cap L(w)$ with $u \in N_B \setminus \{v\}$, $w \in V$ then $l(u) = L(w)$*
      then we add a successor y of x for each $u \in N_B \setminus \{v\}$ such that $L(y) = l(u)$ and $L(\langle x, y\rangle) = l(\langle v_B, u\rangle)$.

We say a node x is an R-successor of $x' \in V$ if $R \in L\langle x', x\rangle$. A node x is called an R-neighbor of x' if x is an R-successor of x' or x' is a $\mathsf{Inv}(R)$-successor of x. In addition, a node x is called an R-block of x' if x blocks an R-successor of x' or x' blocks a $\mathsf{Inv}(R)$-successor of x.
$\mathbf{T} = (V, E, L)$ is called a completion tree with cyclic paths *if for each $\langle u, v\rangle \in E$ such that $Q^+ \in L\langle u, v\rangle$ and $Q \notin L\langle u, v\rangle$ there exists a* cyclic path *$\varphi = \langle x_0, \cdots, x_n\rangle$ which is formed from nodes $v_i \in V$ and satisfies the following conditions:*

- *$x_0 = u$ and x_i is not blocked for all $i \in \{0, \cdots, n\}$;*
- *There do not exist $i, j \in \{1, \cdots, n-1\}$ with $j > i$ such that $L(x_i) = L(x_j)$;*
- *$L(x_n) = L(v)$ and x_i is a Q-neighbor or Q-block of x_{i+1} for all $0 \le i \le n-1$.*

In this case, φ is called a cyclic path and denoted by $\varphi_{\langle u,v\rangle}$.

At this point we have gathered all necessary elements to introduce a decision procedure for the concept satisfiability in $\mathcal{SHIO}_+$. However, in order to provide a upper bound on the size of models of satisfiable $\mathcal{SHIO}_+$-concepts we need an extra structure, namely *reduced tableau*.

Definition 5 (Reduced Tableau). *Let* $\mathbf{T} = (V, E, L)$ *be a completion tree with cyclic paths for a* $\mathcal{SHIO}_+$*-concept* D *w.r.t.* $(\mathcal{T}, \mathcal{R})$. *An equivalence relation* $\sim$ *over* V *is defined as follows:* $x \sim y$ *iff* $L(x) = L(y)$.
Let $V/\sim := \{[x] \mid x \in V\}$ *be the set of all equivalence classes of* V *by* $\sim$. *A graph* $G = (V/\sim, E', L)$ *is called* reduced tableau *for* D *w.r.t.* $(\mathcal{T}, \mathcal{R})$ *if:*

- $L([x]) = L(x')$ *for any* $x' \in [x]$;
- $\langle [x], [y] \rangle \in E'$ *iff there are* $x' \in [x], y' \in [y]$ *such that* $\langle x', y' \rangle \in E$;
- $L(\langle [x], [y] \rangle) = \bigcup\limits_{x' \in [x], y' \in [y], \langle x', y' \rangle \in E} L(\langle x', y' \rangle) \cup \bigcup\limits_{x' \in [x], y' \in [y], \langle y', x' \rangle \in E} \mathsf{Inv}(L\langle y', x' \rangle)$
  where $\mathsf{Inv}(L\langle x, y \rangle) = \{\mathsf{Inv}(R) \mid R \in L\langle y, x \rangle\}$

A reduced tableau as defined in Def. 5 identifies nodes whose labels are the same. The following lemma states an important property of reduced tableaux.

Lemma 2. *Let* D *be a* $\mathcal{SHIO}_+$*-concept with a terminology* $\mathcal{T}$ *and role hierarchy* $\mathcal{R}$. *Let* $G = (V/\sim, E', L)$ *be a reduced tableau for* D *w.r.t.* $(\mathcal{T}, \mathcal{R})$. *We define* $\Delta^{\mathcal{I}} = V/\sim$ *and a function* $\cdot^{\mathcal{I}}$ *that maps:*

- *each concept name* A *occurring in* $D, \mathcal{T}$ *and* $\mathcal{R}$ *to* $A^{\mathcal{I}} \subseteq V/\sim$ *such that* $A^{\mathcal{I}} = \{[x] \mid A \in L([x])\}$;
- *each role name* R *occurring in* $D, \mathcal{T}$ *and* $\mathcal{R}$ *to* $R^{\mathcal{I}} \subseteq (V/\sim)^2$ *such that* $R^{\mathcal{I}} = \{\langle [x], [y] \rangle \mid R \in L\langle [x], [y] \rangle\} \cup \{\langle [y], [x] \rangle \mid \mathsf{Inv}(R) \in L\langle [x], [y] \rangle\}$

If D *has a reduced tableau* G *then* $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ *is a model of* D *w.r.t.* $(\mathcal{T}, \mathcal{R})$.

Lem. 2 affirms that a reduced tableau of a concept D can represent a model of D. The construction of reduced tableaux as presented in Def. 5 preserves not only the validity of neighborhoods but also cyclic paths. The following result is an immediate consequence of Lem. 2.

Proposition 1. *Let* D *be a* $\mathcal{SHIO}_+$*-concept with a terminology* $\mathcal{T}$ *and role hierarchy* $\mathcal{R}$. *If there is a completion tree with cyclic paths* $\mathbf{T}$ *for* D *w.r.t.* $(\mathcal{T}, \mathcal{R})$ *then* D *has a finite model whose size is bounded by an exponential function in the size of* $D, \mathcal{T}, \mathcal{R}$.

Indeed, by the construction of the reduced tableau $G = (V/\sim, E', L)$, the number of nodes of G is bounded by 2^K where $K = \mathsf{card}\{\mathsf{sub}(\mathcal{T}, \mathcal{R}, D)\}$ and K is a polynomial function in the size of $D, \mathcal{T}$ and $\mathcal{R}$.

Lemma 3. *Let* D *be a* $\mathcal{SHIO}_+$*-concept with a terminology* $\mathcal{T}$ *and role hierarchy* $\mathcal{R}$. *If* D *has a model w.r.t.* $(\mathcal{T}, \mathcal{R})$ *then there exists a completion tree with cyclic paths.*

A proof of Lem. 3 can be performed in three steps. First, we define directly valid neighborhoods from individuals of a model. Next, a completion tree can be built by tiling valid neighborhoods with help of role relationships between individuals of the model. Finally, cyclic paths are embedded into the obtained tree by devising paths from finite cycles for the transitive closure of roles in the model. Lem. 1 makes possible adding a new node to a given neighborhood as neighbor if the new node is a neighbor of a node whose label equals to that of the core node of the neighborhood.

From the construction of completion trees with cyclic paths according to Def. 4 and Lem. 2 and 3, we can devise immediately Algorithm 1 for the concept satisfiability in $\mathcal{SHIO}_+$.

Algorithm 1: Decision procedure for concept satisfiability in $\mathcal{SHIO}_+$

    Input : Concept D, terminology $\mathcal{T}$ and role hierarchy $\mathcal{R}$
    Output: IsSatisfiable(D)

    foreach *Tree* $\mathbf{T} = (V, E, L)$ *obtained from tiling valid neighborhoods* **do**
        if *For each* $\langle x, y \rangle \in E$ *with* $Q^+ \in L\langle x, y \rangle, Q \notin L\langle x, y \rangle$, $\mathbf{T}$ *has a* $\varphi_{\langle x,y \rangle}$ **then**
            └ **return** `true`;
    return `false`;

Lemma 4 (Termination). *Alg. 1 for $\mathcal{SHIO}_+$ terminates and the size of completion trees is bounded by a double exponential function in the size of inputs.*

Termination of Alg. 1 is a consequence of the following facts: (i) the number of valid neighborhoods is bounded, (ii) the size of completion trees which are tiled from valid neighborhoods is bounded by $\left(2^{m \times n}\right)^{2^{n \times (m+1)}}$ where $m = \mathsf{card}\{\mathsf{sub}(\mathcal{T}, \mathcal{R}, D)\}, n = \mathsf{card}\{\mathbf{R}\}$.
Alg. 1 is highly complex since it is not a goal-directed procedure. Such an exhaustive behavior is very different from that of tableau-based algorithms in which the construction of a completion tree is inherited from step to step. In Alg. 1, when a tree obtained from tiling neighborhoods cannot satisfy an occurrence of the transitive closure of a role (after satisfying others), the construction of tree has to restart. The following theorem is a direct consequence of Lem.3 and 4.

Theorem 1. *Alg. 1 is a decision procedure for the satisfiability of $\mathcal{SHIO}_+$-concepts w.r.t. a terminology and role hierarchy, and it runs in deterministic 3-*EXPTIME *and nondeterministic 2-*EXPTIME*.*

Thm. 1 is a consequence of the following facts: (i) the size of completion trees is bounded by a double exponential function in the size of inputs , and (ii) the number of of completion trees is bounded by a triple exponential function in the size of inputs.

Remark 1. From the construction of reduced tableaux in Def. 5, we can devise an algorithm for deciding the satisfiability in $\mathcal{SHIO}_+$ which runs in nondeterministic EXPTIME. In fact, such an algorithm can check the validity of neighborhoods and cycles for transitive closures in a graph whose size is bounded by an exponential function in the size of inputs.

Adding number restrictions to $\mathcal{SHI}_+$

The logic $\mathcal{SHIN}_+$ is obtained from $\mathcal{SHI}_+$ ($\mathcal{SHIO}_+$ without nominals) by allowing, additionally, for number restrictions as follows:

Definition 6. *Let* $\mathbf{R}, \mathbf{C}$ *be sets of role and concept names. The set of $\mathcal{SHIN}_+$-roles, role hierarchy $\mathcal{R}$ and model $\mathcal{I}$ of $\mathcal{R}$ are defined similarly to those in Def. 1.*
$\ast$ *A role R is called* simple *w.r.t. $\mathcal{R}$ iff* $(Q^+ \sqsubseteq\!\!\!\!\!\not\;\; R) \notin \mathcal{R}$ *for any* $Q^+ \in \mathbf{R}_+$.
$\ast$ *The set of $\mathcal{SHIN}_+$-concepts is inductively defined as the smallest set containing all*

$C \in \mathbf{C}$, $\top$, $C \sqcap D$, $C \sqcup D$, $\neg C$, $\exists R.C$, $\forall R.C$, $(\leq n\, S)$ *and* $(\geq n\, S)$ *where C and D are $\mathcal{SHIN}_+$-concepts, R is a $\mathcal{SHIN}_+$-role and S is a simple role. We denote $\bot$ for $\neg\top$.*
∗ *An interpretation $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ consists of a non empty set $\Delta^{\mathcal{I}}$ (*domain*) and a function $\cdot^{\mathcal{I}}$ which maps each concept name to a subset of $\Delta^{\mathcal{I}}$. In addition, the function $\cdot^{\mathcal{I}}$ satisfies the conditions for the logic constructors in $\mathcal{SHI}_+$ (as introduced in Def. 2 without nominal), and*

$(\geq n\, R)^{\mathcal{I}} = \{x \in \Delta^{\mathcal{I}} \mid \mathsf{card}\{y \in \Delta^{\mathcal{I}} \mid \langle x,y \rangle \in R^{\mathcal{I}}\} \geq n\}$,
$(\leq n\, R)^{\mathcal{I}} = \{x \in \Delta^{\mathcal{I}} \mid \mathsf{card}\{y \in \Delta^{\mathcal{I}} \mid \langle x,y \rangle \in R^{\mathcal{I}})\} \leq n\}$
∗ *Satisfiability of a $\mathcal{SHIN}_+$-concept C w.r.t. a role hierarchy $\mathcal{R}$ and a terminology $\mathcal{T}$ is defined similarly to that in Def. 2.*

A definition for neighborhoods in $\mathcal{SHIN}_+$ would be provided if we adopt that there may be two neighborhoods such that the labels of their core nodes are identical but they cannot be merged together i.e. a property being similar to Lem. 1 no longer holds for $\mathcal{SHIN}_+$. In such a situation, the local information related to the labels of the ending nodes of a path would be not sufficient to form a cycle. This prevents us from embedding cyclic paths to a completion tree in guaranteeing the soundness and completeness. Note that for the logics $\mathcal{SHI}_+$ and $\mathcal{SHIO}_+$ we can transform a reduced tableau to a tableau (e.g. as described in [8]) such that if any two nodes x, y having the same label then there is an isomorphism between the two neighborhoods (x, N_x, l) and (y, N_y, l). This means that if we know the label of a node in such a tableau it is possible to determine all nodes which are arbitrarily far from this node. This property does not hold for $\mathcal{SHIN}_+$ tableaux.

In the sequel, we show that the difficulty mentioned is insurmountable i.e. the concept satisfiability problem in $\mathcal{SHIN}_+$ is undecidable. The undecidability proof uses a reduction of the domino problem [10]. The following definition, which is taken from [8], reformulates the problem in a more precise way.

Definition 7. *A domino system $\mathbf{D} = (\mathcal{D}, \mathcal{H}, \mathcal{V})$ consists of a non-empty set of domino types $\mathcal{D} = \{D_1, \cdots, D_l\}$ and of sets of horizontally and vertically matching pairs $\mathcal{H} \subseteq \mathcal{D} \times \mathcal{D}$ and $\mathcal{V} \subseteq \mathcal{D} \times \mathcal{D}$. The problem is to determine if, for a given $\mathbf{D}$, there exists a tiling of an $\mathbb{N} \times \mathbb{N}$ grid such that each point of the grid is covered with a domino type in $\mathcal{D}$ and all horizontally and vertically adjacent pairs of domino types are in $\mathcal{H}$ and $\mathcal{V}$ respectively, i.e., a mapping $t : \mathbb{N} \times \mathbb{N} \to \mathcal{D}$ such that for all $m, n \in \mathbb{N}$, $\langle t(m,n), t(m+1,n) \rangle \in \mathcal{H}$ and $\langle t(m,n), t(m,n+1) \rangle \in \mathcal{V}$.*

The reduction of the domino problem to the satisfiability of $\mathcal{SHIN}_+$-concepts will be carried out by (i) constructing a concept, namely A, and two sets of concept and role inclusion axioms, namely $\mathcal{T}_D$ and $\mathcal{R}_D$, and (ii) showing that the domino problem is equivalent to the satisfiability of A w.r.t. $\mathcal{T}_D$ and $\mathcal{R}_D$. Axioms in Def. 8 specify a grid that represents such a domino system. Globally, given a domino set $\mathcal{D} = \{D_1, \cdots, D_l\}$, we need axioms that impose that each point of the plane is covered by exactly one $D_i^{\mathcal{I}}$ (axiom 8 in Def. 8) and ensure that each D_i is compatibly placed in the horizontal and vertical lines (axiom 9). Locally, the key idea is to use $\mathcal{SHIN}_+$ axioms 3, 5, 10, 11, 12 and 13 in Def. 8 for describing the grid as illustrated in Fig. 1.
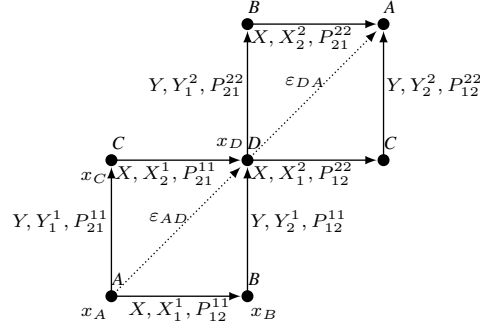
Fig. 1. How each square can be formed from a diagonal represented by an ε

Definition 8. *Let* $\mathbf{D} = (\mathcal{D}, \mathcal{H}, \mathcal{V})$ *be a domino system with* $\mathcal{D} = \{D_1, \cdots, D_l\}$. *Let* N_C *and* N_R *be sets of concept and role names such that* $N_C = \{A, B, C, D\} \cup \mathcal{D}$, $N_R = \{X_j^i \mid i, j \in \{1, 2\}\} \cup \{X, Y\} \cup \{P_{rs}^{ij} \mid i, j, r, s \in \{1, 2\}, r \neq s\} \cup \{\varepsilon_{AD}, \varepsilon_{DA}, \varepsilon_{BC}, \varepsilon_{CB}\}$.
   Role hierarchy:

1. $X_r^i \sqsubseteq P_{rs}^{ij}, Y_s^j \sqsubseteq P_{rs}^{ij}$ *for all* $i, j, r, s \in \{1, 2\}, r \neq s$,
2. $X_r^i \sqsubseteq X, Y_r^i \sqsubseteq Y$ *for all* $i, r \in \{1, 2\}$,
3. $\varepsilon_{AD} \sqsubseteq P_{12}^{11^+}, \varepsilon_{AD} \sqsubseteq P_{21}^{11^+}, \varepsilon_{DA} \sqsubseteq P_{12}^{22^+}, \varepsilon_{DA} \sqsubseteq P_{21}^{22^+}$,
4. $\varepsilon_{BC} \sqsubseteq P_{21}^{21^+}, \varepsilon_{BC} \sqsubseteq P_{12}^{21^+}, \varepsilon_{CB} \sqsubseteq P_{21}^{12^+}, \varepsilon_{CB} \sqsubseteq P_{12}^{12^+}$,
   Concept inclusion axioms:
5. $\top \sqsubseteq\, \leq 1 P_{rs}^{ij}$ *for all* $i, j, r, s \in \{1, 2\}, r \neq s$,
6. $\top \sqsubseteq\, \leq 1 X, \top \sqsubseteq\, \leq 1 Y$,
7. $\top \sqsubseteq\, \leq 1 \varepsilon_{AD}, \top \sqsubseteq\, \leq 1 \varepsilon_{DA}, \top \sqsubseteq\, \leq 1 \varepsilon_{BC}, \top \sqsubseteq\, \leq 1 \varepsilon_{CB}$,
8. $\top \sqsubseteq \bigsqcup_{1 \leq i \leq l} (D_i \sqcap (\bigsqcap_{1 \leq j \leq l, j \neq i} \neg D_j))$,
9. $D_i \sqsubseteq \forall X. \bigsqcup_{(D_i, D_j) \in \mathcal{H}} D_j \sqcap \forall Y. \bigsqcup_{(D_i, D_k) \in \mathcal{V}} D_k$ *for each* $D_i \in \mathcal{D}$,
10. $A \sqsubseteq \neg B \sqcap \neg C \sqcap \neg D \sqcap \exists X_1^1.B \sqcap \exists Y_1^1.C \sqcap \exists \varepsilon_{AD}.D \sqcap \forall P_{12}^{22}.\bot \sqcap \forall P_{21}^{22}.\bot$,
11. $B \sqsubseteq \neg A \sqcap \neg C \sqcap \neg D \sqcap \exists X_2^2.A \sqcap \exists Y_2^1.D \sqcap \exists \varepsilon_{BC}.C \sqcap \forall P_{21}^{12}.\bot \sqcap \forall P_{12}^{12}.\bot$,
12. $C \sqsubseteq \neg A \sqcap \neg B \sqcap \neg D \sqcap \exists X_2^1.D \sqcap \exists Y_2^2.A \sqcap \exists \varepsilon_{CB}.B \sqcap \forall P_{21}^{21}.\bot \sqcap \forall P_{12}^{21}.\bot$,
13. $D \sqsubseteq \neg A \sqcap \neg B \sqcap \neg C \sqcap \exists X_1^2.C \sqcap \exists Y_1^2.B \sqcap \exists \varepsilon_{DA}.A \sqcap \forall P_{12}^{11}.\bot \sqcap \forall P_{21}^{11}.\bot$.

Theorem 2 (Undecidability of $\mathcal{SHIN}_+$). *The concept A is satisfiable w.r.t. concept and role inclusion axioms in Def. 8 iff there is a compatible tiling t of the first quadrant $\mathbb{N} \times \mathbb{N}$ for a given domino system* $\mathbf{D} = (\mathcal{D}, \mathcal{H}, \mathcal{V})$.

Complete proofs of the obtained results in this work can be found in [11].

Conclusion and Discussion

We have presented in this paper a decision procedure for the logic $\mathcal{SHIO}_+$ and shown the finite model property for this logic. To do this we have introduced the neighborhood notion which is an abstraction of the local satisfiability property of tableaux enables us to encapsulate all semantic constraints imposed by the logic constructors in $\mathcal{SHIO}$, and thus to deal with transitive closure of roles independently from the other constructors. According to Rem. 1, we can devise a decision procedure for deciding the concept satisfiability in $\mathcal{SHIO}_+$ so that it runs in nondeterministic exponential time (NEXPTIME). This result with the proof of Lem. 4 implies that this procedure runs in a deterministic doubly exponential. However, the worst-case complexity of the problem remains an open question.

References

1. Tobies, S.: The complexity of reasoning with cardinality restrictions and nominals in expressive description logics. Journal of Artificial Intelligence Research **12** (2000) 199–217
2. Sattler, U.: A concept language extended with different kinds of transitive roles. In: Proceedings of the 20th German Annual Conf. on Artificial Intelligence (KI 2001). Volume 1137., Springer Verlag (2001) 199–204
3. Le Duc, C.: Decidability of $\mathcal{SHI}$ with transitive closure of roles. In: Proceedings of the European Semantic Web Conference, Springer-Verlag (2009)
4. De Giacomo, G., Lenzerini., M.: Boosting the correspondence between description logics and propositional dynamic logics. In: Proceedings of the 12th National conference on Artificial Intelligence, The MIT Press (1994) 205–212
5. De Giacomo, G., Lenzerini, M.: What's in an aggregate: Foundations for description logics with tuples and sets. In: Proceedings of the Fourteenth International Joint Conference On Intelligence Artificial 1995 (IJCAI95). (1995)
6. Horrocks, I., Kutz, O., Sattler, U.: The even more irresistible $\mathcal{SROIQ}$. In: Proceedings of the International Conference on Principles of Knowledge Representation and Reasoning, Springer-Verlag (2006)
7. Ortiz, M.: An automata-based algorithm for description logics around $\mathcal{SRIQ}$. In: Proceedings of the fourth Latin American Workshop on Non-Monotonic Reasoning 2008, CEUR-WS.org (2008)
8. Horrocks, I., Sattler, U., Tobies, S.: Practical reasoning for expressive description logics. In: Proceedings of the International Conference on Logic for Programming, Artificial Intelligence and Reasoning (LPAR 1999), Springer (1999)
9. Baader, F.: Augmenting concept languages by transitive closure of roles: An alternative to terminological cycles. In: Proceedings of the Twelfth International Joint Conference on Artificial Intelligence. (1991)
10. Berger, R.: The undecidability of the dominoe problem. In: The Memoirs of the American Mathematical Society. Number 66, American Mathematical Society, Providence Rhode Island (1966)
11. Le Duc, C., Lamolle, M.: Decidability of description logics with transitive closure of roles. In: Technical Report, http://www.iut.univ-paris8.fr/files/webfm/recherche/linc/RR200911A.pdf (2009)

Structure Preserving TBox Repair
Using Defaults

Thomas Scharrenbach[1], Rolf Grütter[1], Bettina Waldvogel[1], and Abraham
Bernstein[2]

[1] Swiss Federal Institute for Forest, Snow and Landscape Research WSL
Zürcherstrasse 111, 8903 Birmensdorf, Switzerland
{Thomas.Scharrenbach, Rolf.Gruetter, Bettina.Waldvogel}@wsl.ch
[2] University of Zurich, Department of Informatics Zurich, Switzerland
{bernstein}@ifi.uzh.ch

Abstract. Unsatisfiable concepts are a major cause for inconsistencies
in Description Logics knowledge bases. Popular methods for repairing
such concepts aim to remove or rewrite axioms to resolve the conflict by
the original logics used. Under certain conditions, however, the structure
and intention of the original axioms must be preserved in the knowledge
base. This, in turn, requires changing the underlying logics for repair. In
this paper, we show how Probabilistic Description Logics, a variant of
Reiter's default logics with Lehmann's Lexicographical Entailment, can
be used to resolve conflicts fully-automatically and receive a consistent
knowledge base from which inferences can be drawn again.

Key words: default logics, unsatisfiability, justifications, TBox repair

1   Introduction

Ontologies have become standard for knowledge representation in the Semantic
Web. While ontologies are usually expressed in Web Ontology Language (OWL)
recommended by the W3C [1], one of the underlying formalisms for reasoning
about data in the ontology is the Description Logic (DL) $\mathcal{SHOIN}(D)$, being a
decidable subset of first-order logic [2].

Knowledge may evolve over time and might lead to contradictions in the knowl-
edge base. Contradictions may as well occur when mapping two ontolgies on
each other. In the case of terminological knowledge, this causes concepts to be
inferred unsatisfiable. For example, in Figure 1, the concepts C, D and E are in-
ferred unsatisfiable. Unsatisfiable concepts, in turn, cause the whole knowledge
base to be inconsistent, if there exist assertions instantiating them.

Traditional approaches make the TBox satisfiable again by removing trouble-
causing axioms and (possibly) adding new axioms modelling the unsatisfiabil-
ity [3]. This will, anyway, lead to a loss of the information originally specified
in the ontology. However, under certain conditions, all axiomatic information

---

[3] The second case can be seen as axiom rewriting.

should be preserved as much as possible in its original form as well as intuition. We propose to use *default logics* for relaxing the axioms that cause the incoherency.

Defaults, introduced by Reiter [3] and re-interpreted by Lehmann [4], facilitate the co-existence of default rules for typical cases together with exceptions from these rules. When querying the knowledge base, more specific knowledge, i.e. the exceptions, is preferred to more general knowledge, i.e. the defaults.

Transforming subclass inclusion axioms into defaults requires an extension of traditional DL reasoning that copes with the properties that come along with defaults. Probabilistic Description Logics (PDL) [5] is currently the only approach that is able to provide $\mathcal{SHOIN}(D)$ default reasoning, yet as a special case.

We introduce the Δ-transformation for transforming DL axioms and sets of these into defaults. We can show that, under certain conditions, transforming the axioms justifying the unsatisfiability of concepts [4] in the TBox results in a consistent P-$\mathcal{SHOIN}(D)$ knowledge base which re-enables us to draw conclusions.

This work is structured as follows: After introducing preliminaries and custom notions for methods used in Section 2, we present the proposed transformation scheme in Section 3. The actual approach along with supporting examples and formal framework is given in Sections 4 and 5. In Section 6 we give an overview about related work. We conclude this paper and give an outlook to future work in Section 7.

2 Unsatisfiable Concepts and Justifications

While we introduce the unsatisfiability of concept descriptions in Description Logics (DL) and how to justify these, we will not give an introduction to Description Logics in this paper. The interested reader os referred to [2]. For the rest of this paper, we will restrict ourselves to the DL $\mathcal{SHOIN}(D)$, because the methods presented in this paper build on Probabilistic Description Logics which is currently defined for $\mathcal{SHOIN}(D)$. An extension to the current W3C recommendation $\mathcal{SROIQ}(D)$ will remain for future work.
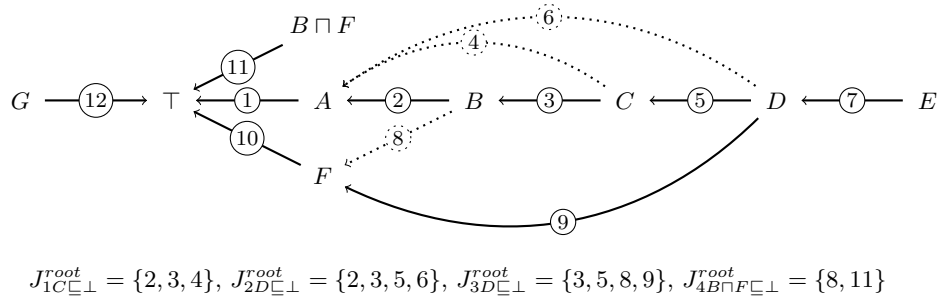
A concept description U is called unsatisfiable w.r.t. a TBox $\mathcal{T}$, iff $\mathcal{T} \models U \sqsubseteq \bot$. A justification for an entailment $\mathcal{T} \models \eta$ is the minimal set of axioms from $\mathcal{T}$ such that the entailment still holds. It is possible to compute the set of all justifications for an entailment [6] using an adapted version of Reiter's Minimum Hitting Set Tree (HST) [7] that originates from the area of Model Based Diagnosis.

Definition 1 (Justifications). *Let $\mathcal{T}$ be an TBox. $J_\eta \subseteq \mathcal{T}$ is a justification for $\mathcal{T} \models \eta$, iff $J_\eta \models \eta$ and $J' \not\models \eta$ for any $J' \subset J_\eta$.*

It turns out that the unsatisfiability of a concept description U_1 may depend on the unsatisfiability of another concept description U_2, i.e. $J_{U_1 \sqsubseteq \bot} \supseteq J_{U_2 \sqsubseteq \bot}$. In

---

[4] By concepts we consider atomic as well as concept descriptions, if not stated otherwise.

Fig. 1. Example TBox. Nodes correspond to concepts and arcs correspond to subclass inclusions. Dotted arcs represent disjoints. The axioms are numbered for referring to them in the text. Below, the root justifications for the non-purely derived unsatisfiable concepts are shown.

$$J_{1C \sqsubseteq \bot}^{root} = \{2,3,4\}, \; J_{2D \sqsubseteq \bot}^{root} = \{2,3,5,6\}, \; J_{3D \sqsubseteq \bot}^{root} = \{3,5,8,9\}, \; J_{4B \sqcap F \sqsubseteq \bot}^{root} = \{8,11\}$$

this case we say that J_{U_2} is more general. The most general justifications for the unsatisfiable concepts of an ontology are called *root unsatisfiable.* [5]

It should be noted that root justifications are sets of axioms and should not be mixed up with the notion of *root unsatisfiable*, *partially derived* and *purely derived* concepts as denoted in [6]. However, there is a correspondence, i.e. every root unsatisfiable concept description has only root justifications, every partially dervied unsatisfiable concept description has at least one root justification and every purely derived unsatisfiable concept description has no root justification.

Definition 2 (Root Justifications). *Let $\mathcal{J}$ be the set of all justifications for all unsatisfiable concept description of a TBox $\mathcal{T}$. Then $J_{U \sqsubseteq \bot}^{root} \in \mathcal{J}$ is a* root *justification for some unsatisfiable concept description U, iff for any concept description U' there is no $J_{U' \sqsubseteq \bot} \in \mathcal{J}$ such that $J_{U' \sqsubseteq \bot} \subset J_{U \sqsubseteq \bot}^{root}$.*

Root justifications allow us to resolve only the most general causes for unsatisfiablility in a Tbox, which in turn result in the satisfiability of all unsatisfiable concepts. For example in Figure 1, the partially derived unsatisfiable concept D will be inferred unsatisfiable for the same reason as is the root unsatisfiable concept C. The unsatisfiability of concept E is purely derived, since it depends on the unsatisfiability of D. We do not have to distinguish between the unsatisfiability of these concepts as long as we remove the most general causes. Please also note, that the (root) justification for the concept description $B \sqcap F$ contains its declaration. If it did not, then $J_{3D \sqsubseteq \bot}^{root} \supset J_{4B \sqcap F \sqsubseteq \bot}^{root}$ and hence the unsatisfiability of the atomic concept D would depend on the unsatisfiability of the concept description $B \sqcap F$. This is indeed not the case, and hence this dependency has to

---

[5] Please note that axioms of the form $A \sqsubseteq \top$ are only included in a justification, if A is a complex concept description, but not, if A is an atomic concept.

be seen as an artifact. Therefore the declaration $B \sqcap F$ is included in the (root) justification.

3 Resolving Unsatisfiable Concepts

$\mathcal{SHOIN}(D)$ fulfils the monotonicity assumption, i.e. adding new axioms does not invalidate existing entailments or introduce unsatisfiability. Hence, unsatisfiability cannot be resolved by just adding new axioms. Repair has to involve the removal of axioms and is therefore always a non-monotone operation.

The currently most convenient way of resolving unsatisfiability in a TBox is to remove axioms that are responsible for it. This task is often referred to as OWL-Debugging. The interested reader may find a more detailed survey of approaches to OWL-Debugging in [6].

In addition to that, attempts for semi-automatic axiom rewriting have been made [6], referred to as repair plans. Common modelling errors have been identified empirically, and according to the kind of axioms that caused unsatisfiability, repair plans are generated and proposed to an end-user that decides how to repair the unsatisfiability.

Instead of doing the repair in $\mathcal{SHOIN}(D)$, it is also possible to change the formalism for knowledge representation and/or inference. We propose that it is desirable to keep as much of the original intention as well as structure of the stated axioms as possible. Keeping the axioms' structure requires some mechanism of how to prefer some of the contradicting axioms over the others to keep possible models of the ontology consistent.

Default logic is a way of generating a model of preferece for axioms of a first-order logic knowledge base that soley relies upon the structure of the knowledge base.

3.1 Probabilistic Description Logics

Recently, a method called *probabilistic description logics* (PDL) has been proposed [5] that extends a $\mathcal{SHOIN}(D)$ knowledge base with probabilistic constraints. Such a constraint $(A|B)[l, u]$ can be viewed as assigning the $\mathcal{SHOIN}(D)$ TBox axiom $B \sqsubseteq A$ the belief interval $[l, u]$ with $0 \le l \le u \le 1$. The special case $l = u = 1$, however, corresponds to to Reiter's normal defaults [3]. For sake of readability, we omit the interval and write defaults as $(A|B)$.

As a consequence, PDL can be used as a way of modelling OWL-axioms $B \sqsubseteq A$ as a set of defaults $(A|B)$. The resulting logic is called P-$\mathcal{SHOIN}(D)$. PDL extends a classical $\mathcal{SHOIN}(D)$ TBox by a set of constraints called PBox $\mathcal{P}$. Together, both of these form a so-called PTBox PT $= (\mathcal{T}, \mathcal{P})$.

In case we restrict these constraints to defaults like above, inferences can be drawn according to Lehmann's lexicographical entailment [4]. The PTBox is partitioned into sets of defaults $P_0, \ldots, P_N$ where P_0 contains the most general defaults and P_N the most specific ones. Models are defined as in classical knowledge bases. A default $(A|B)$ falls into partition P_n iff there is a model for the

TBox and the remaining defaults [6] that satisfies $A(i)$ as well as $B(i)$ for a new individual i. We say that such a model *verifies* this default. A PTBox is consistent iff there exists a partition for the PBox.

Inferences are drawn according to the *lexicographical minimal model* for a PTBox, where models are ordered lexicographically w.r.t. the number and level of generality of defaults they violate. Models violating as few of the least specific defaults as possible have higher preference when ordering the models.

3.2   From DL to PDL: The Δ-Transformation

Using default logic for resolving an unsatisfiable concept of a TBox $\mathcal{T}$, we must transform a TBox into a PTBox and hence a subset of $\mathcal{T}$ into a set of defaults. We introduce the Δ-transformation for this transformation which changes the logics from $\mathcal{SHOIN}(D)$ to P-$\mathcal{SHOIN}(D)$.

Definition 3 (Δ-Transformation). *Let $\alpha = B \sqsubseteq A$ be a subclass inclusion axiom in a TBox $\mathcal{T}$, and $\mathcal{U}_n$ be a set of subclass inclusion axioms being a subset of $\mathcal{T}$. The Δ-transformation for $\mathcal{T}$ maps axioms from $\mathcal{T}$ to defaults and sets of axioms to a (partitioned) PBox.*

$$
\begin{array}{llll}
(i) & \Delta_{\mathcal{T}}(\alpha) & = & (A|B) \\
(ii) & \Delta_{\mathcal{T}}(\mathcal{U}_n) & = & \{\Delta_{\mathcal{T}}(\alpha)|\alpha \in \mathcal{U}_n\} \\
(iii) & \Delta_{\mathcal{T}} \underbrace{(\mathcal{U}_0, \ldots, \mathcal{U}_N)}_{\text{pairwise disjoint}} & = & (\underbrace{(\mathcal{T} \setminus \mathcal{U}_0 \cup \ldots \cup \mathcal{U}_N)}_{\text{new TBox}}, \underbrace{(\Delta_{\mathcal{T}}(\mathcal{U}_0), \ldots, \Delta_{\mathcal{T}}(\mathcal{U}_N))}_{\text{partitioned PBox}})
\end{array}
$$

Please note that the Δ-transformation is bijective, i.e. we can easily define $\Delta_{\mathcal{T}}^{-1}(A|B) = B \sqsubseteq A$.

4   Constraints on the TBox

The most obvious method for resolving unsatisfiable concepts of a TBox is to remove all the axioms from the justifications proving the unsatisfiability. However, it clearly suffices to remove only all the axioms of the *root justifications* from the TBox, since any (purely) derived unsatisfiable concept will then also become satisfiable.

Removing axioms from the TBox results in a loss of knowledge. We therefore propose not to fully remove the axioms but to keep them in a different form, i.e. as defaults. The Δ-transformtion will be applied to all axioms of the root justifications for the unsatisfiable concepts of a TBox. In this section, it is shown that this transformation results in a consistent PTBox, if the TBox fulfils certain constraints which are explained in the remainder of this section. Conflict resolving using default logic works only if the axioms justifying the conflict are on different levels of preference, like it is implied by PDL. Situations where conflicting axioms are on the same level of preference must be excluded.

---

[6] $\mathcal{T} \cup (P \setminus P_0 \cup \ldots \cup P_{n-1})$

This means that all kinds of cycles and situtations where a concept is explicitely stated to be subclass of one concept and its negation must not be allowed, since the Δ-transformation will result in an inconsistent PTBox.

4.1 Disallow Cycles, Logical and Direct Contradictions

If we allowed for cycles in the TBox, then a justification may also contain this cycle. In turn, all axioms involved in the conflict are on the same level of preference and there cannot be a verifying model for any of these axioms.

In the following, we assume every TBox and hence all justifications, to be free of cycles. Logical contradictions, i.e. concepts of the form $A \sqcap \neg A$ cannot be resolved by applying the Δ-transformation. There is no valid world w.r.t. [5] for the concept $A \sqcap \neg A$.

Corollary 1. *If one of the axioms of a TBox $\mathcal{T}$ contains a logical contradiction on the right hand side, then the Δ-transformation of $\mathcal{T}$ is inconsistent.*

Since logical contradictions do not provide any useful information, we can safely remove axioms containing $A \sqcap \neg A$ from the TBox without changing the intended semantics. In the following, we assume every TBox not to contain any logical contradiction. Default logics require contradictive information to be on different level of preference in order to provide a consistent way for inference. This mechanism is doomed to fail in cases where a contradiction is stated explicitly, i.e. some concept C is explicitly stated to be a subclass of the concepts A_1 and A_2 where $A_1 \sqcap A_2$ are a logical contradiction.

Definition 4 (Direct Contradictions).
A set of two axioms $\mathcal{DC} = \{C \sqsubseteq A_1, C \sqsubseteq A_2\}$ from a TBox $\mathcal{T}$ is called a direct contradiction $\mathcal{DC}$ for a concept $C \in \mathcal{T}$, iff $A_1 \sqcap A_2$ is a logical contradiction.

There exists some justification for $\mathcal{T} \models C \sqsubseteq \bot$ that soley consists of the axioms of the direct contradiction. Since there cannot exist a model that satisfies $A_1(i)$, $A_2(i)$ and $C(i)$ at the same time for a new individual i, the Δ-transformation of the example TBox will lead to an inconsistent PTBox. Hence default logics cannot resolve the unsatisfiability of C.

Corollary 2. *The Δ-transformation of a TBox that contains a direct contradiction results in an inconsistent PTBox.*

While logical contradictions can simply be removed from the TBox without loss of relevant information, the situation for a direct contradiction $\mathcal{DC}$ is slightly more difficult. Removing the axioms of the $\mathcal{DC}$ might lead to a loss of information. Yet there is an option how to resolve a $\mathcal{DC}$.

Considering PDL, we can simply add some new "intermediate" concept to at least one of the axioms of the direct contradiction. In particular, we replace, e.g., the axiom $C \sqsubseteq A_1 \in \mathcal{DC}$ with $C \sqsubseteq B$ and $B \sqsubseteq A_1$ where B is a new concept that does not yet occur in the TBox. The concept C is still unsatisfiable, but there is no direct contradiction anymore. We can therefore safely assume the TBox not

Fig. 2. Example TBox where C is inferred unsatisfiable due to the two direct contradictions $C \sqsubseteq A, \neg A$ and $C \sqsubseteq \forall R.A, \exists R.\neg A$ in the left figure (a).
One possibility for resolving the direct contradictions by adding a new "intermediate" concept is shown in the right figure (b).

(a)   (b)
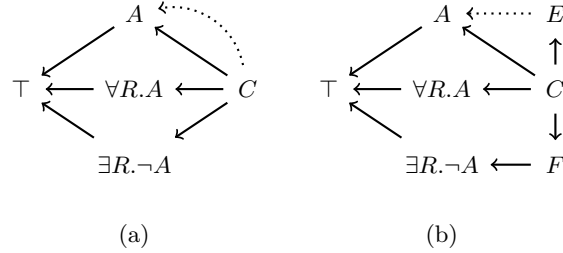
to contain any direct contradictions.

In the example in Figure 2(a) there exist the direct contradictions $\mathcal{DC}_1 = \{C \sqsubseteq A, C \sqsubseteq \neg A\}$ and $\mathcal{DC}_2 = \{C \sqsubseteq \forall R.A, C \sqsubseteq \exists R.\neg A\}$. These can be resolved, for example, by introducing the new concepts E and F in between the subconcept hierarchy of $C \sqsubseteq \neg A$ and $C \sqsubseteq \exists R.\neg A$, respectively.

5 Consistency of the Δ-Transformed TBox

After having excluded logical as well as direct contradictions and cycles, we have to show that the PTBox that results from Δ-transforming all axioms from all (root) justifications is consistent. According to [5], we have to show that the resulting PBox is a valid z-partiton. We do so soley using the structure of the justifications for the unsatisfiable concepts.

For each unsatisfiable concept U, we split the union of its justifications into two parts: one that contains unsatisfiable concepts in the axioms, Γ_U, and one that does not, Θ_U. The idea is to iteratively first transform axioms for a new partition that occur only in some Θ, but not in some Γ, since these are not in conflict with any other axiom.

Every axiom we transformed and hence removed from some Θ_U has its conflicting axioms in its correponding Γ_U set. The conflict for a Γ_{U_0} set is solved, if its Θ_{U_0} set is empty. The next partition is hence formed by all axioms in these Γ_U for which the Θ_U set is empty. We can now proceed with step one and, since the number of axioms is finite, the procedure will terminate eventually.

5.1 Splitting the Root Justifications

Every justification for an unsatisfiable concept U contains at least one axiom with U on the left-hand side of the subclass inclusion. As such, every justification can be split up into two sets of axioms: one that contains axioms with U on the left-hand-side and one that contains the rest. We call the first one the Γ-set of

Table 1. Procedure for Δ-transforming the unsatisfiability splitting for the root justifications for the non-purely derived unsatisfiable concepts in Figure 1. The last column shows the axioms that are chosen to be Δ-transformed to the partitions P_0, P_1, P_2 of the resulting PBox during the corresponding Θ- or Γ step (indicated by a bold symbol), whereas the $\mathcal{J}$-columns show the current contents of the Θ and Γ sets.

Step		$\mathcal{J}(C \sqsubseteq \bot)$	$\mathcal{J}(D \sqsubseteq \bot)$	$\mathcal{J}(B \sqcap F \sqsubseteq \bot)$	
1	$\boldsymbol{\Theta}$	2	2, 3, 8	8	$P_0 = \{\Delta_{\mathcal{T}}(\{2, 8\})\}$
	Γ	3, 4	5, 6, 9	11	- - - - - - - - - - - - - -
2	Θ	$\emptyset$	3	$\emptyset$	- - - - - - - - - - - - - -
	$\boldsymbol{\Gamma}$	3, 4	5, 6, 9	11	$P_1 = \{\Delta_{\mathcal{T}}(\{3, 4, 11\})\}$
3	$\boldsymbol{\Theta}$	$\emptyset$	$\emptyset$	$\emptyset$	- - - - - - - - - - - - - -
	Γ	$\emptyset$	5, 6, 9	$\emptyset$	- - - - - - - - - - - - - -
4	Θ	$\emptyset$	$\emptyset$	$\emptyset$	- - - - - - - - - - - - - -
	$\boldsymbol{\Gamma}$	$\emptyset$	5, 6, 9	$\emptyset$	$P_2 = \{\Delta_{\mathcal{T}}(\{5, 6, 9\})\}$
5	$\boldsymbol{\Theta}$	$\emptyset$	$\emptyset$	$\emptyset$	- - - - - - - - - - - - - -
	Γ	$\emptyset$	$\emptyset$	$\emptyset$	- - - - - - - - - - - - - -

U and the latter one its Θ-set. The splitting for the example of Figure 1 can be obtained from the first row of Table 1.

Definition 5 (Unsatisfiability Splitting).
Let $U_0, \ldots, U_N$ be the unsatisfiable concepts of a TBox $\mathcal{T}$. Let $\mathcal{J}^{root}_{U_i \sqsubseteq \bot}$ be the union of the root justifications for the unsatisfiability of the concept U_i. The unsatisfiability splitting *for $\mathcal{T}$ is defined as:*[7]
$$\mathcal{J}^{root}_{U_i \sqsubseteq \bot} = \Theta_{U_i} \oplus \Gamma_{U_i} \text{ where } \Gamma_{U_i} = \{X \sqsubseteq Y \in \mathcal{J}^{root}_{U_i \sqsubseteq \bot} | X = U_i\}$$

5.2 Obtaining the Partition by Δ-Transforming the Splitting

For an axiom of the root justifications, there exist three different possibilities where it my reside:

1. In some Θ_{U_j} but not in any Γ_{U_k}. We denote these axioms with ϑ.
2. In some Γ_{U_k} but not in any Θ_{U_j}. These axioms are denoted with γ.
3. In both some Θ_{U_j} as well as some Γ_{U_k}. In this case the axiom is denoted with η.

In our example, processing step one, axioms 2 and 8 are of the ϑ type whereas axioms $4, 5, 6, 9$ and 11 are of type γ. Axiom 3 is of type η, since it is contained in both, Γ_C and Θ_D. For preparing the proof of the induction, we first proof some auxiliary lemma stating the important properties of ϑ, γ and η axioms.

---
[7] The operator $\oplus$ denotes the union of pairwise disjoint sets.

Lemma 1 (Satisfiability of θ, γ and η axioms).

1. *If some axiom ϑ is contained only in some Θ_{U_i} but not in some Γ_{U_j}, then there exists verifying model for $\Delta_{\mathcal{T}}(\vartheta)$.*
2. *If some axiom γ is contained in some Γ_{U_i} for which the corresponding Θ_{U_i} is empty, then there exists verifying model for $\Delta_{\mathcal{T}}(\gamma)$.*
3. *If some axiom η is contained in both, some Θ_{U_i} and some Γ_{U_j}, then there cannot exist a verifying model for $\Delta_{\mathcal{T}}(\eta)$ w.r.t. these two sets.*

We explain the procedure using the example from Figure 1. We alternatively Δ-transform axioms according to 1, the so-called Θ-step, followed by the Γ-step where axioms are Δ-transformed according to 2. The single steps are visualized in Table 1.

In our example, step one, we can find a model for each ϑ axiom 2 and 8. In particular we can obviously find a model in which $A \sqcap B$ is satisfied and some model in which $B \sqcap \neg F$ is satisfied. On the other hand, all remaining axioms contain by definition some unsatisfiable concept, which denies the existence of a model for each of the remaining axioms. So even though axiom 3 is part of Θ_D we are not able to find a verifying model for it [8]. Hence, the first partition is $P_0 = \{2, 8\}$.

We proceed with the next step and have a look at the Γ sets for which the Θ set is empty. This is the case for Γ_C. We remember that Θ_{U_i} contains at least one element from each justification for $\mathcal{T} \models U_i \sqsubseteq \bot$. Hence, for each justification for $\mathcal{T} \models C \sqsubseteq \bot$ we Δ-transformed at least one axiom which means that $\mathcal{T} \setminus \Delta_{\mathcal{T}}^{-1}(P_0) \not\models C \sqsubseteq \bot$. As a consequence, we can find a verifying model for each axiom in Γ_C. On the other hand, $D \sqsubseteq \bot$ still holds, such that we cannot find a verifying model for any of the remaining axioms $5, 6$ and 9. Hence, the second partition is $P_1 = \{3, 4, 11\}$.

For the next Θ-step we find that all of the Θ sets are empty, so we proceed with the next Γ-step and find that Γ_D is the only Γ-set left. Since all conflicting axioms have already been Δ-transformed, we can for each axiom in Γ_D trivially find a verifying model which results in the next partition $P_2 = \{5, 6, 9\}$.

In step nine, there are no more axioms left that we could process. The resulting PTBox is:

$$\text{PT} = ((\underbrace{\{1, 10, 12\}}_{\mathcal{T} \setminus \Delta_{\mathcal{T}}^{-1}(\mathcal{P})}), \underbrace{(\overbrace{\Delta_{\mathcal{T}}(\{2, 8\})}^{P_0}, \overbrace{\Delta_{\mathcal{T}}(\{3, 4, 11\})}^{P_1}, \overbrace{\Delta_{\mathcal{T}}(\{5, 6, 9\})}^{P_2})}_{\mathcal{P}}))$$

Since we found a valid partition w.r.t. PDL, PT is consistent.
We now proof the parts of Lemma 1.

Proof. 1 If some axiom ϑ is contained only in some Θ_U but not in some $\Gamma_{U'}$, then ϑ has no unsatisfiable concept on the left-hand side. It also cannot have an unsatisfiable concept on the right-hand side, because then it would be purely derived. As such, we can find a model in which both the subconcept and the superconcept are satisfied.

---

[8] Indeed, axiom 3 is of type η.

Proof. 2 If some axiom $\gamma = U \sqsubseteq A$ is contained in some Γ_U for which the corresponding Θ_U is empty, there has been one axiom removed from every root justification for $\mathcal{T} \models U \sqsubseteq \bot$, i.e. the elements that had been in the now empty Θ_U and were Δ-transformed before. Hence U is not unsatisfiable anymore and A must be satisfiable for the same reasons as in the proof for 1 which proofs the existence of a model.
It should be noted that in this case, γ has been root unsatisfiable and the Δ-transformed axioms from the Θ_U were part of the root justifications.

Proof. 3 Some axiom η is contained in both, some Θ_U and some $\Gamma_{U'}$. Because $\eta \in \Theta_U$, there still exists some Γ_U correspoding to Θ_U, which means that there still exists a justification for $U \sqsubseteq \bot$. Hence, we cannot find a model for an axiom that contains an unsatisfiable concept.
It should be noted that in this case, $\eta = U \sqsubseteq A$ is part of a justification for some partially derived unsatisfiable concept.

5.3 Consistency of the Δ-Transformation of the Splitting

It remains to show that we can always find some axioms that fulfil the conditions of the Θ-step followed by a Γ step. We do this by induction. As stated before, every justification can be split into non-empty Θ_U and Γ_U. Since the number of sets of the splitting of Definition 5 is finite, there has to exist some Θ_{U_0} such that for all axioms $\vartheta \in \Theta_{U_0}$ follows $\vartheta \notin \Gamma_U$. We Δ-transform all of these axioms into the starting partition P_0 and proceed with all axioms γ of the sets Γ_{U_0} that correspond to Θ_{U_0}. By Lemma 1, part 2, these form the next partition P_1.
In the induction step we have to show that having transformed the Γ-axioms

1. either there is some Θ_{U_0} such that $\Theta_{U_0}$is disjoint to all remaining Γ_U,
2. or there is some Γ_{U_0} for which all Θ-axioms have already been Δ-transformed
3. or there are no more axioms left to transform.

Since every Γ_{U_i} refers to a Θ_{U_i}, and since the number of sets is finite, and justifications cannot be circular, for at least one Γ_{U_i} there has to exist some Θ_{U_i} that contains neither γ nor η axioms. Please note that we allow the case $\Theta_{U_i} = \emptyset$. In case Θ_{U_i} is non-empty we proceed with the Θ-step, if it is empty, we proceed with the Γ-step. The procedure terminates, if also the Γ sets are empty.

Theorem 1. *Let $\mathcal{T}$ be a TBox and $\mathcal{P} = (P_0, \ldots, P_N)$ be the partition resulting from the Δ-transformation of the unsatisfiability splitting of all root justifications for all unsatisfiable concepts in $\mathcal{T}$.*
Then the PTBox $PT = (\mathcal{T} \setminus \Delta_{\mathcal{T}}^{-1}(P_0, \ldots, P_N), \mathcal{P})$ is consistent.

5.4 Complexity of the Δ-Transformation of the Splitting

The complexity of the presented procedure is dominated by the complexity for finding justifications. This in turn depends on the complexity for consistency checking in the tableaux calculus which is - in the case of $\mathcal{SHOIN}(D)$ -

NEXPTIME-complete.

It should be noted that the presented approach does not involve any satisfiability checks in addition to checking and tracing unsatisability, which have to be performed anyway.

6   Related Work

In recent years, much progess has been made in the task to explain why a conclusion can be drawn from a DL knowledge base by soley using axioms from the knowledge base itself. Schlobach and Cornet [8] came up with minimal unsatisfiable preserving sub-TBoxes (MUPS) which can explain the reason for unsatisfiability of concepts. Kalyanpur et al. [6] introduced justification as a form of minimal explanation for any arbitrary entailment. It could be shown that computing all justifications for an entailment is feasible in the tableaux calculus [6]. In the area of ontology evolution, the main focus usually lies on resolving inconsistencies and hence changes mainly occur on instance level or rather restricted TBoxes [9]. Repair can also be done using higher-order logics like in the Ontology Repair System [10]. This, however makes changes to the ontology and cannot be applied easily to OWL ontologies.

Alternatives to do reasoning with incoherent DL knowledge bases are, for example, paraconsistent logics [11]. However, these change the notion of inference and hence their semantics much more than default logic does.

There have been made propositions of how to incorporate default knowledge in OWL-DL knowledge bases in [12] [13], and [14]. While the first two deal with applcations of Reiter's interpretation of defaults, to our knowledge, P-$\mathcal{SHOIN}(D)$ [5] is currently the only formalism providing default reasoning services w.r.t. Lehmann's lexicographical entailment for OWL DL knowledge bases for which an implementation is available [15].

7   Conclusion

We showed that default logics as introduced in [5] provide a way of re-enabling coherency for incoherent DL knowledge bases. This way, structure as well as semantics of the original axioms is kept as much as possible. The proposed approach makes use of justifications, a standard technique for computing reasons for conflicts in DL knowledge bases. Since these have to be computed anyhow for repairing the knowledge base, the presented approach does not need to perform any additional satisfiability checks.

While this paper proofs the correctness of the approach, an implementation and evaluation on real-world data has to be performed showing whether the approach is feasible. Comparisons to alternative approaches, for example, what can still be inferred from the knowledge base after the repair and what not, will also remain for future work.

References

1. McGuinness, D.L., van Harmelen, F.: OWL web ontology language overview. W3C recommendation, W3C (February 2004) http://www.w3.org/TR/2004/REC-owl-features-20040210/.
2. Baader, F., Calvanese, D., McGuinness, D.L., Nardi, D., Patel-Schneider, P.F., eds.: The Description Logic Handbook: Theory, Implementation, and Applications. 2nd edn. Cambridge University Press, Cambridge, MA, USA (August 2007)
3. Reiter, R.: A logic for default reasoning. Artificial Intelligence **13**(1-2) (1980) 81–132
4. Lehmann, D.: Another perspective on default reasoning. Ann. Math. Artif. Intell. **15** (1995) 61–82
5. Lukasiewicz, T.: Expressive probabilistic description logics. Artif. Intell. **172**(6-7) (April 2008) 852–883
6. Kalyanpur, A.: Debugging and Repair of OWL Ontologies. PhD thesis, University of Maryland, Department of Computer Science (2006)
7. Reiter, R.: A theory of diagnosis from first principles. Artif. Intell. **32**(1) (1987) 57–95
8. Schlobach, S., Cornet, R.: Non-standard reasoning services for the debugging of description logic terminologies. In: IJCAI, Morgan Kaufmann (2003) 355–362
9. Haase, P., Völker, J.: Ontology learning and reasoning - dealing with uncertainty and inconsistency. In: URSW Pt. 1. Volume 5327 of LNCS. (January 2009) 45–55
10. Bundy, A.: Where's my stuff? an ontology repair plan. In: Workshop on DIS-PROVING - Non-Theorems, Non-Validity, Non-Provability. Volume 4., CADE Inc (July 2007) 2–11
11. Ma, Y., Lin, Z., Lin, Z.: Inferring with inconsistent owl dl ontology: A multi-valued logic approach. In: EDBT Workshops. Volume 4254 of LNCS. (March 2006) 535–553
12. Baader, F., Hollunder, B.: Embedding defaults into terminological knowledge representation formalisms. In: J. Autom. Reas., Morgan Kaufmann (1995) 306–317
13. Dao-Tran, M., Eiter, T., Krennwallner, T.: Realizing default logic over description logic knowledge bases. In: ECSQARU 2009. 602–613
14. Navarro, J.L., Sanchez, J.M., Zurita, J.M.: Default reasoning in web ontology language. In: Proc. Intell. Systems and Agents (IADIS2007). (July 2007) 35–42
15. Klinov, P.: Pronto: A non-monotonic probabilistic description logic reasoner. In: ESWC. Volume 5021 of LNCS., Springer (2008) 822–826

A Compatible Approach to Temporal Description Logics

Norihiro Kamide

Waseda Institute for Advanced Study, Waseda University,
1-6-1 Nishi Waseda, Shinjuku-ku, Tokyo 169-8050, JAPAN.
`logician-kamide@aoni.waseda.jp`

1 Introduction

Temporal description logics (TDLs) have been studied by many researchers (see e.g., [1, 10] for surveys and [4, 2, 15] for recent or important results). These TDLs are, however, not compatible in the following sense: these are not embeddable into the standard (non-temporal) description logics (DLs), and hence the existing algorithms for testing satisfiability in the standard DLs are not available for these TDLs. Such a compatibility issue is important for obtaining reusable and practical algorithms for temporal reasoning in ontologies.

In this paper, two compatible TDLs, $\mathcal{XALC}$ and $\mathcal{BALC}_l$, are introduced by combining and modifying the description logic $\mathcal{ALC}$ [14] and *Prior's tomorrow tense logic* [12, 13]. $\mathcal{XALC}$ has the next-time operator, and $\mathcal{BALC}_l$ has some restricted versions of the next-time, any-time and some-time operators, in which the time domain is bounded by a positive integer l. *Semantical embedding theorems* of $\mathcal{XALC}$ and $\mathcal{BALC}_l$ into $\mathcal{ALC}$ are shown. By using these embedding theorems, the concept satisfiability problems for $\mathcal{XALC}$ and $\mathcal{BALC}_l$ are shown to be decidable. The complexities of the decision procedures for $\mathcal{XALC}$ and $\mathcal{BALC}_l$ are also shown to be the same complexity as that for $\mathcal{ALC}$. Next, tableau calculi, $\mathcal{TXALC}$ (for $\mathcal{XALC}$) and $\mathcal{TBALC}_l$ (for $\mathcal{BALC}_l$), are introduced, and *syntactical embedding theorems* of these calculi into a tableau calculus, $\mathcal{TALC}$ (for $\mathcal{ALC}$), are proved. The completeness theorems for $\mathcal{TXALC}$ and $\mathcal{TBALC}_l$ are proved by combining both the semantical and syntactical embedding theorems.

Prior's tomorrow tense logic, which is a base logic of $\mathcal{XALC}$ and $\mathcal{BALC}_l$, is regarded as the next-time fragment of *linear-time temporal logic* (LTL) [11], and hence $\mathcal{XALC}$ and $\mathcal{BALC}_l$ may also be familiar with many users of the existing LTL-based TDLs. The bounded temporal operators in $\mathcal{BALC}_l$ are, indeed, regarded as restricted versions of the corresponding LTL-operators. Although the standard temporal operators of LTL have an infinite (unbounded) time domain, i.e., the set ω of natural numbers, the bounded operators which are presented in this paper have a *bounded time domain* which is restricted by a fixed positive integer l, i.e., the set $\omega_l := \{x \in \omega \mid x \leq l\}$.

To restrict the time domain of temporal operators is not a new idea. Such an idea has been discussed [5–9]. It is known that to restrict the time domain is a technique to obtain a decidable or efficient fragment of first-order LTL [8].

Restricting the time domain implies not only some purely theoretical merits, but also some practical merits for describing temporal databases and planning specifications [6, 7], and for implementing an efficient model checking algorithm called *bounded model checking* [5]. Such practical merits are due to the fact that there are problems in computer science and artificial intelligence where only a finite fragment of the time sequence is of interest [6].

Finally in this section, other characters of $\mathcal{XALC}$ and $\mathcal{BALC}_l$ are summarized as follows: (1) the temporal operators in $\mathcal{XALC}$ and $\mathcal{BALC}_l$ are only applied to concepts and ABox assertions, (2) $\mathcal{XALC}$ and $\mathcal{BALC}_l$ are based on the assumptions of *rigid roles* and *rigid individual names*, i.e., the interpretations of atomic roles and individual names are not changed over time, and (3) $\mathcal{XALC}$ and $\mathcal{BALC}_l$ are based on the *constant domain* assumption, i.e., only one time domain is used in the logics.

2 Temporal Description Logic with Next-Time, $\mathcal{XALC}$

2.1 $\mathcal{ALC}$

The $\mathcal{ALC}$-language is constructed from atomic concepts, atomic roles, $\sqcap$ (intersection), $\sqcup$ (union), $\neg$ (classical negation or complement), $\forall R$ (universal concept quantification) and $\exists R$ (existential concept quantification). We use the letters A and A_i for atomic concepts, the letter R for atomic roles, and the letters C and D for concepts.

Definition 1 *Concepts C are defined by the following grammar:*

$$C ::= A \mid \neg C \mid C \sqcap C \mid C \sqcup C \mid \forall R.C \mid \exists R.C$$

Definition 2 *An* interpretation $\mathcal{I}$ *is a pair* $\langle \Delta^{\mathcal{I}}, \cdot^{\mathcal{I}} \rangle$ *where*

1. $\Delta^{\mathcal{I}}$ *is a non-empty set,*
2. $\cdot^{\mathcal{I}}$ *is an interpretation function which assigns to every atomic concept A a set $A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$ and to every atomic role R a binary relation $R^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$.*

The interpretation function is extended to concepts by the following inductive definitions:

1. $(\neg C)^{\mathcal{I}} := \Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}$,
2. $(C \sqcap D)^{\mathcal{I}} := C^{\mathcal{I}} \cap D^{\mathcal{I}}$,
3. $(C \sqcup D)^{\mathcal{I}} := C^{\mathcal{I}} \cup D^{\mathcal{I}}$,
4. $(\forall R.C)^{\mathcal{I}} := \{a \in \Delta^{\mathcal{I}} \mid \forall b \, [(a,b) \in R^{\mathcal{I}} \Rightarrow b \in C^{\mathcal{I}}]\}$,
5. $(\exists R.C)^{\mathcal{I}} := \{a \in \Delta^{\mathcal{I}} \mid \exists b \, [(a,b) \in R^{\mathcal{I}} \wedge b \in C^{\mathcal{I}}]\}$.

An interpretation $\mathcal{I}$ is a model *of a concept C (denoted as $\mathcal{I} \models C$) if $C^{\mathcal{I}} \neq \emptyset$. A concept C is said to be* satisfiable *in $\mathcal{ALC}$ if there exists an interpretation $\mathcal{I}$ such that $\mathcal{I} \models C$.*

The syntax of $\mathcal{ALC}$ is extended by a non-empty set N_I of individual names. We denote individual names by o, o_1, o_2, x, y and z.

Definition 3 *An* ABox *is a finite set of expressions of the form:* $C(o)$ *or* $R(o_1, o_2)$ *where* o, o_1 *and* o_2 *are in* N_I, C *is a concept, and* R *is an atomic role. An expression* $C(o)$ *or* $R(o_1, o_2)$ *is called an* ABox *statement. An interpretation* $\mathcal{I}$ *in Definition 2 is extended to apply also to individual names* o *such that* $o^{\mathcal{I}} \in \Delta^{\mathcal{I}}$. *Such an interpretation is a model of an ABox* $\mathcal{A}$ *if for every* $C(o) \in \mathcal{A}$, $o^{\mathcal{I}} \in C^{\mathcal{I}}$ *and for every* $R(o_1, o_2) \in \mathcal{A}$, $(o_1^{\mathcal{I}}, o_2^{\mathcal{I}}) \in R^{\mathcal{I}}$. *An ABox* $\mathcal{A}$ *is called* satisfiable *in* $\mathcal{ALC}$ *if it has a model.*

We adopt the following *unique name assumption*: for any $o_1, o_2 \in N_I$, if $o_1 \neq o_2$, then $o_1^{\mathcal{I}} \neq o_2^{\mathcal{I}}$.

Definition 4 *A* TBox *is a finite set of expressions of the form:* $C \sqsubseteq D$. *The elements of a TBox are called* TBox *statments. An interpretation* $\mathcal{I} := \langle \Delta^{\mathcal{I}}, \cdot^{\mathcal{I}} \rangle$ *is called a* model *of* $C \sqsubseteq D$ *if* $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$. *An interpretation* $\mathcal{I}$ *is said to be a model of a TBox* $\mathcal{T}$ *if* $\mathcal{I}$ *is a model of every element of* $\mathcal{T}$. *A TBox* $\mathcal{T}$ *is called* satisfiable *in* $\mathcal{ALC}$ *if it has a model.*

Definition 5 *A knowledge base* Σ *is a pair* $(\mathcal{T}, \mathcal{A})$ *where* $\mathcal{T}$ *is a TBox and* $\mathcal{A}$ *is an ABox. An interpretation* $\mathcal{I}$ *is a model of* Σ *if* $\mathcal{I}$ *is a model of both* $\mathcal{T}$ *and* $\mathcal{A}$. *A knowledge base* Σ *is called* satisfiable *in* $\mathcal{ALC}$ *if it has a model.*

Since the satisfiability for an ABox, a TBox or a knowledge base can be reduced to the satisfiability for a concept [3], we focus on the concept satisfiability in the following discussion.

2.2 $\mathcal{XALC}$

Similar notions and terminologies for $\mathcal{ALC}$ are also used for $\mathcal{XALC}$. The symbol ω is used to represent the set of natural numbers. The $\mathcal{XALC}$-language is constructed from the $\mathcal{ALC}$-language by adding X (next-time operator). An expression $X^n C$ is inductively defined by $X^0 C := C$ and $X^{n+1}C := XX^n C$.

Definition 6 *Concepts* C *are defined by the following grammar:*

$$C ::= A \mid \neg C \mid XC \mid C \sqcap C \mid C \sqcup C \mid \forall R.C \mid \exists R.C$$

Definition 7 *A* temporal interpretation *$\mathcal{TI}$ is a structure* $\langle \Delta^{\mathcal{TI}}, \{\cdot^{\mathcal{I}^i}\}_{i \in \omega} \rangle$ *where*

1. $\Delta^{\mathcal{TI}}$ *is a non-empty set,*
2. *each* $\cdot^{\mathcal{I}^i}$ $(i \in \omega)$ *is an interpretation function which assigns to every atomic concept* A *a set* $A^{\mathcal{I}^i} \subseteq \Delta^{\mathcal{TI}}$ *and to every atomic role* R *a binary relation* $R^{\mathcal{I}^i} \subseteq \Delta^{\mathcal{TI}} \times \Delta^{\mathcal{TI}}$,
3. *for any atomic role* R *and any* $i, j \in \omega$, $R^{\mathcal{I}^i} = R^{\mathcal{I}^j}$.

The interpretation function is extended to concepts by the following inductive definitions:

1. $(XC)^{\mathcal{I}^i} := C^{\mathcal{I}^{i+1}}$,

2. $(\neg C)^{\mathcal{I}^i} := \Delta^{\mathcal{T}\mathcal{I}} \setminus C^{\mathcal{I}^i}$,
3. $(C \sqcap D)^{\mathcal{I}^i} := C^{\mathcal{I}^i} \cap D^{\mathcal{I}^i}$,
4. $(C \sqcup D)^{\mathcal{I}^i} := C^{\mathcal{I}^i} \cup D^{\mathcal{I}^i}$,
5. $(\forall R.C)^{\mathcal{I}^i} := \{a \in \Delta^{\mathcal{T}\mathcal{I}} \mid \forall b \ [(a,b) \in R^{\mathcal{I}^i} \Rightarrow b \in C^{\mathcal{I}^i}]\}$,
6. $(\exists R.C)^{\mathcal{I}^i} := \{a \in \Delta^{\mathcal{T}\mathcal{I}} \mid \exists b \ [(a,b) \in R^{\mathcal{I}^i} \wedge b \in C^{\mathcal{I}^i}]\}$.

For any $i \in \omega$, an expression $\mathcal{I}^i \models C$ is defined as $C^{\mathcal{I}^i} \neq \emptyset$. A temporal interpretation $\mathcal{T}\mathcal{I} := \langle \Delta^{\mathcal{T}\mathcal{I}}, \{\cdot^{\mathcal{I}^i}\}_{i \in \omega} \rangle$ is a model of a concept C (denoted as $\mathcal{T}\mathcal{I} \models C$) if $\mathcal{I}^0 \models C$. A concept C is said to be satisfiable in $\mathcal{X}\mathcal{ALC}$ if there exists a temporal interpretation $\mathcal{T}\mathcal{I}$ such that $\mathcal{T}\mathcal{I} \models C$.

Definition 8 *A temporal interpretation $\mathcal{T}\mathcal{I}$ in Definition 7 is extended to apply also to individual names o such that for any $i, j \in \omega$, $o^{\mathcal{I}^i} \in \Delta^{\mathcal{T}\mathcal{I}}$ and $o^{\mathcal{I}^i} = o^{\mathcal{I}^j}$. Such a temporal interpretation is a model of an ABox $\mathcal{A}$ if for every $C(o) \in \mathcal{A}$, $o^{\mathcal{I}^0} \in C^{\mathcal{I}^0}$ and for every $R(o_1, o_2) \in \mathcal{A}$, $(o_1^{\mathcal{I}^0}, o_2^{\mathcal{I}^0}) \in R^{\mathcal{I}^0}$. Such a temporal interpretation is called a model of $C \sqsubseteq D$ if $C^{\mathcal{I}^0} \subseteq D^{\mathcal{I}^0}$. The satisfiability of ABox, a TBox or a knowledge base in $\mathcal{X}\mathcal{ALC}$ is defined in the same way as in $\mathcal{ALC}$.*

Remark that $\mathcal{X}\mathcal{ALC}$ is an extension of $\mathcal{ALC}$ since $\cdot^{\mathcal{I}^0}$ includes $\cdot^{\mathcal{I}}$. Remark also that $\mathcal{X}\mathcal{ALC}$ adopts the *constant domain assumption*, i.e., it has the single common domain $\Delta^{\mathcal{T}\mathcal{I}}$, and the *rigid role and name assumption*, i.e., it satisfies the conditions: for any atomic role R, any individual name o and any $i, j \in \omega$, we have $R^{\mathcal{I}^i} = R^{\mathcal{I}^j}$ and $o^{\mathcal{I}^i} = o^{\mathcal{I}^j}$.

3 Semantical Embedding and Decidability

Definition 9 *Let N_C be a non-empty set of atomic concepts and N_C^i be the set $\{A^i \mid A \in N_C\}$ of atomic concepts where $A^0 = A$, i.e., $N_C^0 = N_C$. [1] Let N_R be a non-empty set of atomic roles and N_I be a non-empty set of individual names. The language $\mathcal{L}^x$ of $\mathcal{X}\mathcal{ALC}$ is defined using N_C, N_R, N_I, X, $\neg, \sqcap, \sqcup, \forall R$ and $\exists R$. The language $\mathcal{L}$ of $\mathcal{ALC}$ is obtained from $\mathcal{L}^x$ by adding $\bigcup_{i \in \omega} N_C^i$ and deleting X.*
A mapping f from $\mathcal{L}^x$ to $\mathcal{L}$ is defined inductively by

1. *for any $R \in N_R$ and any $o \in N_I$, $f(R) := R$ and $f(o) := o$,*
2. *for any $A \in N_C$, $f(X^i A) := A^i \in N_C^i$, esp. $f(A) := A$,*
3. *For any $A(o) \in N_C$, $f(X^i A(o)) := A^i(f(o)) \in N_C^i$, esp. $f(A(o)) := A(f(o))$,*
4. $f(X^i \neg C) := \neg f(X^i C)$,
5. $f(X^i (C \sharp D)) := f(X^i C) \sharp f(X^i D)$ *where $\sharp \in \{\sqcap, \sqcup\}$,*
6. $f(X^i \forall R.C) := \forall f(R).f(X^i C)$,
7. $f(X^i \exists R.C) := \exists f(R).f(X^i C)$.

Lemma 10 *Let f be the mapping defined in Definition 9. For any temporal interpretation $\mathcal{T}\mathcal{I} := \langle \Delta^{\mathcal{T}\mathcal{I}}, \{\cdot^{\mathcal{I}^i}\}_{i \in \omega} \rangle$ of $\mathcal{X}\mathcal{ALC}$, we can construct an interpretation $\mathcal{I} := \langle \Delta^{\mathcal{I}}, \cdot^{\mathcal{I}} \rangle$ of $\mathcal{ALC}$ such that for any concept C in $\mathcal{L}^x$ and any $i \in \omega$,*

---

[1] A can include individual names, i.e., A can be $A(o)$ for any $o \in N_I$.

$$C^{\mathcal{I}^i} = f(\mathrm{X}^i C)^{\mathcal{I}}.$$

Proof. Let N_C be a non-empty set of atomic concepts and N_C^i be the set $\{A^i \mid A \in N_C\}$ of atomic concepts where $A^0 = A$. Let N_R and N_I be sets of atomic roles and individual names, respectively.

Suppose that $\mathcal{TI}$ is a temporal interpretation $\langle \Delta^{\mathcal{TI}}, \{\cdot^{\mathcal{I}^i}\}_{i \in \omega} \rangle$ where

1. $\Delta^{\mathcal{TI}}$ is a non-empty set,
2. each $\cdot^{\mathcal{I}^i}$ $(i \in \omega)$ is an interpretation function which assigns to every atomic concept $A \in N_C$ a set $A^{\mathcal{I}^i} \subseteq \Delta^{\mathcal{TI}}$, to every atomic role $R \in N_R$ a binary relation $R^{\mathcal{I}^i} \subseteq \Delta^{\mathcal{TI}} \times \Delta^{\mathcal{TI}}$ and to every individual name $o \in N_I$ an element $o^{\mathcal{I}^i} \in \Delta^{\mathcal{TI}}$,
3. for any $R \in N_R$, any $o \in N_I$ and any $i, j \in \omega$, $R^{\mathcal{I}^i} = R^{\mathcal{I}^j}$ and $o^{\mathcal{I}^i} = o^{\mathcal{I}^j}$.

Suppose that $\mathcal{I}$ is an interpretation $\langle \Delta^{\mathcal{I}}, \cdot^{\mathcal{I}} \rangle$ where

1. $\Delta^{\mathcal{I}}$ is a non-empty set such that $\Delta^{\mathcal{I}} = \Delta^{\mathcal{TI}}$,
2. $\cdot^{\mathcal{I}}$ is an interpretation function which assigns to every atomic concept $A \in \bigcup_{i \in \omega} N_C^i$ a set $A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$, to every atomic role $R \in N_R$ a binary relation $R^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$ and to every individual name $o \in N_I$ an element $o^{\mathcal{I}} \in \Delta^{\mathcal{I}}$,
3. for any $R \in N_R$, any $o \in N_I$ and any $i \in \omega$, $R^{\mathcal{I}} = R^{\mathcal{I}^i}$ and $o^{\mathcal{I}} = o^{\mathcal{I}^i}$.

Suppose moreover that $\mathcal{TI}$ and $\mathcal{I}$ satisfy the following condition: for any $A \in N_C$, any $o in N_I$ and any $i \in \omega$,

$$A^{\mathcal{I}^i} = (A^i)^{\mathcal{I}} \text{ and } (A(o))^{\mathcal{I}^i} = (A^i(o))^{\mathcal{I}}.$$

The lemma is then proved by induction on the complexity of C. The base step is obvious. We show some cases in the induction step below.

Case $C \equiv \neg D$: We obtain: $a \in (\neg D)^{\mathcal{I}^i}$ iff $a \in \Delta^{\mathcal{TI}} \setminus D^{\mathcal{I}^i}$ iff $a \in \Delta^{\mathcal{I}} \setminus D^{\mathcal{I}^i}$ (by the condition $\Delta^{\mathcal{TI}} = \Delta^{\mathcal{I}}$) iff $a \in \Delta^{\mathcal{I}} \setminus f(\mathrm{X}^i D)^{\mathcal{I}}$ (by induction hypothesis) iff $a \in (\neg f(\mathrm{X}^i D))^{\mathcal{I}}$ iff $a \in f(\mathrm{X}^i \neg D)^{\mathcal{I}}$ (by the definition of f).

Case $C \equiv \mathrm{X}D$: We obtain: $a \in (\mathrm{X}D)^{\mathcal{I}^i}$ iff $a \in D^{\mathcal{I}^{i+1}}$ iff $a \in f(\mathrm{X}^{i+1}D)^{\mathcal{I}}$ (by induction hypothesis) iff $a \in f(\mathrm{X}^i \mathrm{X}D)^{\mathcal{I}}$.

Case $C \equiv \forall R.D$: We obtain:

$d \in (\forall R.D)^{\mathcal{I}^i}$

iff $d \in \{a \in \Delta^{\mathcal{TI}} \mid \forall b \, [(a,b) \in R^{\mathcal{I}^i} \Rightarrow b \in D^{\mathcal{I}^i}]\}$

iff $d \in \{a \in \Delta^{\mathcal{I}} \mid \forall b \, [(a,b) \in R^{\mathcal{I}} \Rightarrow b \in D^{\mathcal{I}^i}]\}$ (by the conditions $\Delta^{\mathcal{TI}} = \Delta^{\mathcal{I}}$ and $R^{\mathcal{I}^i} = R^{\mathcal{I}}$)

iff $d \in \{a \in \Delta^{\mathcal{I}} \mid \forall b \, [(a,b) \in R^{\mathcal{I}} \Rightarrow b \in f(\mathrm{X}^i D)^{\mathcal{I}}]\}$ (by induction hypothesis)

iff $d \in (\forall R.f(\mathrm{X}^i D))^{\mathcal{I}}$

iff $d \in (\forall f(R).f(\mathrm{X}^i D))^{\mathcal{I}}$ (by the definition of f)

iff $d \in f(\mathrm{X}^i \forall R.D)^{\mathcal{I}}$ (by the definition of f).

∎

Lemma 11 *Let f be the mapping defined in Definition 9. For any temporal interpretation $\mathcal{TI} := \langle \Delta^{\mathcal{TI}}, \{\cdot^{\mathcal{I}^i}\}_{i \in \omega} \rangle$ of $\mathcal{XALC}$, we can construct an interpretation $\mathcal{I} := \langle \Delta^{\mathcal{I}}, \cdot^{\mathcal{I}} \rangle$ of $\mathcal{ALC}$ such that for any concept C in $\mathcal{L}^x$ and any $i \in \omega$,*

$\mathcal{I}^i \models C$ *iff* $\mathcal{I} \models f(\mathrm{X}^i C)$.

Proof. We obtain: $\mathcal{I}^i \models C$ iff $C^{\mathcal{I}^i} \neq \emptyset$ iff $f(\mathrm{X}^i C)^{\mathcal{I}} \neq \emptyset$ (by Lemma 10) iff $\mathcal{I} \models f(\mathrm{X}^i C)$. ∎

Lemma 12 *Let f be the mapping defined in Definition 9. For any interpretation $\mathcal{I} := \langle \Delta^{\mathcal{I}}, \cdot^{\mathcal{I}} \rangle$ of $\mathcal{ALC}$, we can construct a temporal interpretation $\mathcal{TI} := \langle \Delta^{\mathcal{TI}}, \{\cdot^{\mathcal{I}^i}\}_{i \in \omega} \rangle$ of $\mathcal{XALC}$ such that for any concept C in $\mathcal{L}^x$ and any $i \in \omega$,*

$\mathcal{I} \models f(\mathrm{X}^i C)$ *iff* $\mathcal{I}^i \models C$.

Proof. Similar to the proof of Lemma 11. ∎

Theorem 13 (Semantical embedding) *Let f be the mapping defined in Definition 9. For any concept C,*

C *is satisfiable in $\mathcal{XALC}$ iff $f(C)$ is satisfiable in $\mathcal{ALC}$.*

Proof. By Lemmas 11 and 12. ∎

Theorem 14 (Decidability) *The concept satisfiability problem for $\mathcal{XALC}$ is decidable.*

Proof. By decidability of the satisfiability problem for $\mathcal{ALC}$, for each concept C of $\mathcal{XALC}$, it is possible to decide if $f(C)$ is satisfiable in $\mathcal{ALC}$. Then, by Theorem 13, the satisfiability problem for $\mathcal{XALC}$ is decidable. ∎

The satisfiability problems of a TBox, an ABox and a knowledge base for $\mathcal{XALC}$ are also shown to be decidable.

Since f is a polynomial-time reduction, the complexities of the satisfiability problems of a TBox, an ABox and a knowledge base for $\mathcal{XALC}$ can be reduced to those for $\mathcal{ALC}$, i.e., the complexities of the problems for $\mathcal{XALC}$ are the same as those for $\mathcal{ALC}$. For example, the satisfiability problems of an acyclic TBox and a general TBox for $\mathcal{XALC}$ are PSPACE-complete and EXPTIME-complete, respectively. For the concept satisfiability problem for $\mathcal{XALC}$, the existing tableau algorithms for $\mathcal{ALC}$ are applicable by using the translation f with Theorem 13.

4 Syntactical Embedding and Completeness

From a purely theoretical or logical point of view, a sound and complete axiomatization is required for the underlying semantics. In this section, we thus give such a tableau calculus $\mathcal{TXALC}$ for $\mathcal{XALC}$.

Definition 15 *A concept is called a* negation normal form *(NNF) if the classical negation connective $\neg$ occurs only in front of atomic concepts.*

Let $C(x)$ be a concept in NNF. In order to test satisfiability of $C(x)$, the tableau algorithm starts with the ABox $\mathcal{A} = \{C(x)\}$, and applies the inference rules of a tableau calculus to the ABox until no more rules apply.

Definition 16 ($\mathcal{T}\mathcal{ALC}$) *Let $\mathcal{A}$ be an ABox that consists only of NNF-concepts. The inference rules for the tableau calculus $\mathcal{T}\mathcal{ALC}$ for $\mathcal{ALC}$ are of the form:*

$$\frac{\mathcal{A}}{\mathcal{A} \cup \{C_1(x), C_2(x)\}} \ (\sqcap)$$

where $(C_1 \sqcap C_2)(x) \in \mathcal{A}$, $C_1(x) \notin \mathcal{A}$ or $C_2(x) \notin \mathcal{A}$,

$$\frac{\mathcal{A}}{\mathcal{A} \cup \{C_1(x)\} \quad | \quad \mathcal{A} \cup \{C_2(x)\}} \ (\sqcup)$$

where $(C_1 \sqcup C_2)(x) \in \mathcal{A}$ and $[C_1(x) \notin \mathcal{A}$ and $C_2(x) \notin \mathcal{A}]$,

$$\frac{\mathcal{A}}{\mathcal{A} \cup \{C(y)\}} \ (\forall R)$$

where $(\forall R.C)(x) \in \mathcal{A}$, $R(x, y) \in \mathcal{A}$ and $C(y) \notin \mathcal{A}$,

$$\frac{\mathcal{A}}{\mathcal{A} \cup \{C(y), R(x, y)\}} \ (\exists R)$$

where $(\exists R.C)(x) \in \mathcal{A}$, there is no individual name z such that $C(z) \in \mathcal{A}$ and $R(x, z) \in \mathcal{A}$, and y is an individual name not occurring in $\mathcal{A}$.

Definition 17 *Let $\mathcal{A}$ be an ABox that consists only of NNF-concepts. Then, $\mathcal{A}$ is called* complete *if there is no more rules apply to $\mathcal{A}$. $\mathcal{A}$ is called* clash *if $\{A(x), \neg A(x)\} \subseteq \mathcal{A}$ for some atomic concept $A(x)$. A tree produced by a tableau calculus from $\mathcal{A}$ is called* complete *if all the nodes in the tree are complete. A branch of a tree produced by a tableau calculus from $\mathcal{A}$ is called* clash-free *if all its nodes are not clash.*

The following theorem is known.

Theorem 18 (Completeness) *For any $\mathcal{ALC}$-concept C in NNF, $\mathcal{T}\mathcal{ALC}$ produces a complete tree with a clash-free branch from the Abox $\{C\}$ iff C is satisfiable in $\mathcal{ALC}$.*

The way of obtaining NNFs for $\mathcal{X}\mathcal{ALC}$-concepts is almost the same as that for $\mathcal{ALC}$-concepts, except that we also use the law: $\neg XC \leftrightarrow X\neg C$, which is justified by the fact: $(\neg XC)^{\mathcal{I}^i} = (X\neg C)^{\mathcal{I}^i}$ for any $i \in \omega$.

Definition 19 ($\mathcal{T}\mathcal{X}\mathcal{ALC}$) *Let $\mathcal{A}$ be an ABox that consists only of NNF-concepts. The inference rules for the tableau calculus $\mathcal{T}\mathcal{X}\mathcal{ALC}$ for $\mathcal{X}\mathcal{ALC}$ are of the form:*

$$\frac{\mathcal{A}}{\mathcal{A} \cup \{X^i C_1(x), X^i C_2(x)\}} \ (X\sqcap)$$

where $X^i(C_1 \sqcap C_2)(x) \in \mathcal{A}$, $X^iC_1(x) \notin \mathcal{A}$ *or* $X^iC_2(x) \notin \mathcal{A}$,

$$\frac{\mathcal{A}}{\mathcal{A} \cup \{X^iC_1(x)\} \quad | \quad \mathcal{A} \cup \{X^iC_2(x)\}} \ (X\sqcup)$$

where $X^i(C_1 \sqcup C_2)(x) \in \mathcal{A}$ *and* $[X^iC_1(x) \notin \mathcal{A}$ *and* $X^iC_2(x) \notin \mathcal{A}]$,

$$\frac{\mathcal{A}}{\mathcal{A} \cup \{X^iC(y)\}} \ (X\forall R)$$

where $(X^i\forall R.C)(x) \in \mathcal{A}$, $R(x, y) \in \mathcal{A}$ *and* $X^iC(y) \notin \mathcal{A}$,

$$\frac{\mathcal{A}}{\mathcal{A} \cup \{X^iC(y), R(x, y)\}} \ (X\exists R)$$

where $(X^i\exists R.C)(x) \in \mathcal{A}$, *there is no individual name* z *such that* $X^iC(z) \in \mathcal{A}$ *and* $R(x, z) \in \mathcal{A}$, *and* y *is an individual name not occurring in* $\mathcal{A}$.

An expression $f(\mathcal{A})$ denotes the set $\{f(\alpha) \mid \alpha \in \mathcal{A}\}$.

Theorem 20 (Syntactical embedding) *Let* $\mathcal{A}$ *be an ABox that consists only of NNF-concepts in* $\mathcal{L}^x$, *and* f *be the mapping defined in Definition 9. Then:*

$\mathcal{TXALC}$ produces a complete tree with a clash-free branch from $\mathcal{A}$ *iff*
$\mathcal{TALC}$ produces a complete tree with a clash-free branch from $f(\mathcal{A})$

Proof. ($\Longrightarrow$): By induction on the complete trees T with a clash-free branch from $\mathcal{A}$ in $\mathcal{TXALC}$. ($\Longleftarrow$): By induction on the complete trees T' with a clash-free branch from $f(\mathcal{A})$ in $\mathcal{TALC}$. ∎

Theorem 21 (Completeness) *For any* $\mathcal{XALC}$-*concept* C *in NNF,* $\mathcal{TXALC}$ *produces a complete tree with a clash-free branch from the Abox* $\{C\}$ *iff* C *is satisfiable in* $\mathcal{XALC}$.

Proof. Let C be a $\mathcal{XALC}$-concept in NNF. Then, we obtain:

$\mathcal{TXALC}$ produces a complete tree with a clash-free branch from $\{C\}$
iff $\mathcal{TALC}$ produces a complete tree with a clash-free branch from $\{f(C)\}$ (by Theorem 20)
iff $f(C)$ is satisfiable in $\mathcal{ALC}$ (by Theorem 18)
iff C is satisfiable in $\mathcal{XALC}$ (by Theorem 13).

∎

5   Temporal Description Logic with Bounded-Time, $\mathcal{BALC}_l$

5.1 $\mathcal{BALC}_l$

Similar notions and terminologies for $\mathcal{XALC}$ are also used for $\mathcal{BALC}_l$. The symbol $\geq$ or $\leq$ is used to represent a linear order on ω. In the following discussion, l is fixed as a certain positive integer. The $\mathcal{BALC}_l$-language is constructed from the $\mathcal{XALC}$-language by adding G (any-time operator) and F (some-time operator). Remark that the temporal operators X, G and F used in $\mathcal{BALC}_l$ are interpreted as some l-bounded versions of the original operators.

Definition 22 Concepts C *are defined by the following grammar:*

$$C ::= A \mid \neg C \mid \mathrm{X}C \mid \mathrm{G}C \mid \mathrm{F}C \mid C \sqcap C \mid C \sqcup C \mid \forall R.C \mid \exists R.C$$

Definition 23 *A* bounded-time interpretation $\mathcal{BI}$ *is the same as a temporal structure, i.e., it is obtained from a temporal structure* $\langle \Delta^{\mathcal{TI}}, \{\cdot^{\mathcal{I}^i}\}_{i \in \omega} \rangle$ *by replacing the notation* $\Delta^{\mathcal{TI}}$ *with the notation* $\Delta^{\mathcal{BT}}$. *The interpretation function is extended to concepts by induction on concepts. The definitions of the interpretation function is obtained from the conditions in Definitions 7 and 8 by replacing* $\Delta^{\mathcal{TI}}$ *with* $\Delta^{\mathcal{BT}}$, *deleting the condition 1 in Definition 7, and adding the following conditions:*

1. *for any* $i \leq l - 1$, $(\mathrm{X}C)^{\mathcal{I}^i} := C^{\mathcal{I}^{i+1}}$,
2. *for any* $i \geq l$, $(\mathrm{X}C)^{\mathcal{I}^i} := C^{\mathcal{I}^l}$,
3. *for any* $m \in \omega$, $(\mathrm{X}C)^{\mathcal{I}^{l+m}} := C^{\mathcal{I}^l}$,
4. $(\mathrm{G}C)^{\mathcal{I}^i} := C^{\mathcal{I}^i} \cap C^{\mathcal{I}^{i+1}} \cap \cdots \cap C^{\mathcal{I}^{i+l}}$,
5. $(\mathrm{F}C)^{\mathcal{I}^i} := C^{\mathcal{I}^i} \cup C^{\mathcal{I}^{i+1}} \cup \cdots \cup C^{\mathcal{I}^{i+l}}$.

The notions of satisfiability etc. are defined in the same way as in $\mathcal{XALC}$.

Remark that the new conditions for the interpretation function in Definition 23 are intended to have the following axiom schemes:

1. for any $m \in \omega$, $\mathrm{X}^{l+m}C \leftrightarrow \mathrm{X}^l C$,
2. $\mathrm{G}C \leftrightarrow C \sqcap \mathrm{X}C \sqcap \cdots \sqcap \mathrm{X}^l C$,
3. $\mathrm{F}C \leftrightarrow C \sqcup \mathrm{X}C \sqcup \cdots \sqcup \mathrm{X}^l C$,
4. $\neg \mathrm{G}C \leftrightarrow \mathrm{F}\neg C$,
5. $\neg \mathrm{F}C \leftrightarrow \mathrm{G}\neg C$.

Remark also that the new conditions in Definition 23 are the l-bounded time versions of the following standard non-restricted conditions:

1. $(\mathrm{X}C)^{\mathcal{I}^i} := C^{\mathcal{I}^{i+1}}$,
2. $(\mathrm{G}C)^{\mathcal{I}^i} := \bigcap \{C^{\mathcal{I}^j} \mid i \leq j \in \omega\}$,
3. $(\mathrm{F}C)^{\mathcal{I}^i} := \bigcup \{C^{\mathcal{I}^j} \mid i \leq j \in \omega\}$.

These non-restricted conditions imply a standard LTL-based temporal description logic.

5.2 Semantical Embedding and Decidability

Definition 24 *The language $\mathcal{L}^b$ of $\mathcal{BALC}_l$ is obtained from the language $\mathcal{L}^x$ in Definition 9 by adding G and F. The language $\mathcal{L}$ of $\mathcal{ALC}$ is defined as the same language in Definition 9. A mapping f from $\mathcal{L}^b$ to $\mathcal{L}$ is obtained from the mapping defined in Definition 9 by adding the following conditions:*

1. *for any $m \geq l$, $f(\mathrm{X}^m\mathrm{X}C) := f(\mathrm{X}^lC)$,*
2. *$f(\mathrm{X}^i\mathrm{G}C) := f(\mathrm{X}^iC) \sqcap f(\mathrm{X}^{i+1}C) \sqcap \cdots \sqcap f(\mathrm{X}^{i+l}C)$,*
3. *$f(\mathrm{X}^i\mathrm{F}C) := f(\mathrm{X}^iC) \sqcup f(\mathrm{X}^{i+1}C) \sqcup \cdots \sqcup f(\mathrm{X}^{i+l}C)$.*

Lemma 25 *Let f be the mapping defined in Definition 24. For any bounded-time interpretation $\mathcal{BI} := \langle \Delta^{\mathcal{BI}}, \{\cdot^{\mathcal{I}^i}\}_{i\in\omega} \rangle$ of $\mathcal{BALC}_l$, we can construct an interpretation $\mathcal{I} := \langle \Delta^{\mathcal{I}}, \cdot^{\mathcal{I}} \rangle$ of $\mathcal{ALC}$ such that for any concept C in $\mathcal{L}^b$ and any $i \in \omega$,*

$$C^{\mathcal{I}^i} = f(\mathrm{X}^iC)^{\mathcal{I}}.$$

Proof. Similar to the proof of Lemma 10 by replacing $\Delta^{\mathcal{TI}}$ with $\Delta^{\mathcal{BI}}$. ∎

We then obtain the key lemmas which correspond to Lemmas 11 and 12, and hence obtain the following theorems.

Theorem 26 (Semantical embedding) *Let f be the mapping defined in Definition 24. For any concept C,*

C is satisfiable in $\mathcal{BALC}_l$ iff $f(C)$ is satisfiable in $\mathcal{ALC}$.

Theorem 27 (Decidability) *The concept satisfiability problem for $\mathcal{BALC}_l$ is decidable.*

The complexity of the decision procedure for concept satisfiability in $\mathcal{BALC}_l$ is the same as that in $\mathcal{ALC}$.

5.3 Syntactical Embedding and Completeness

The way of obtaining NNFs for $\mathcal{BALC}_l$-concepts is almost the same as that for $\mathcal{XALC}$-concepts, except that we also use the laws: $\neg\mathrm{G}C \leftrightarrow \mathrm{F}\neg C$ and $\neg\mathrm{F}C \leftrightarrow \mathrm{G}\neg C$.

Definition 28 ($\mathcal{TBALC}_l$) *Let $\mathcal{A}$ be an ABox that consists only of NNF-concepts.*
The inference rules for the tableau calculus $\mathcal{TBALC}_l$ for $\mathcal{BALC}_l$ are of the form:

$$\frac{\mathcal{A}}{\mathcal{A} \cup \{\mathrm{X}^lC(x)\}} \ (\mathrm{X})$$

where $\mathrm{X}^{l+m}C(x) \in \mathcal{A}$ for any $m \in \omega$,

$$\frac{\mathcal{A}}{\mathcal{A} \cup \{\mathrm{X}^iC(x), \mathrm{X}^{i+1}C(x), ..., \mathrm{X}^{i+l}C(x)\}} \ (\mathrm{G})$$

where $\mathrm{X}^i \mathrm{G} C(x) \in \mathcal{A}$ *and* $\mathrm{X}^{i+j} C(x) \notin \mathcal{A}$ *for some* $j \in \omega_l$,

$$\frac{\mathcal{A}}{\mathcal{A} \cup \{\mathrm{X}^i C(x)\} \quad | \quad \mathcal{A} \cup \{\mathrm{X}^{i+1} C(x)\} \quad | \quad \cdots \quad | \quad \mathcal{A} \cup \{\mathrm{X}^{i+l} C(x)\}} \text{ (F)}$$

where $\mathrm{X}^i \mathrm{F} C(x) \in \mathcal{A}$ *and* $[\mathrm{X}^i C(x) \notin \mathcal{A},\ \mathrm{X}^{i+1} C(x) \notin \mathcal{A},\ \dots,\ \text{and } \mathrm{X}^{i+l} C(x) \notin \mathcal{A}]$.

Theorem 29 (Syntactical embedding) *Let $\mathcal{A}$ be an ABox that consists only of NNF-concepts in $\mathcal{L}^b$, and f be the mapping defined in Definition 24. Then:*

> $\mathcal{TBALC}_l$ *produces a complete tree with a clash-free branch from $\mathcal{A}$ iff*
> $\mathcal{TALC}$ *produces a complete tree with a clash-free branch from $f(\mathcal{A})$*

Theorem 30 (Completeness) *For any $\mathcal{BALC}_l$-concept C in NNF, $\mathcal{TBALC}_l$ produces a complete tree with a clash-free branch from the Abox $\{C\}$ iff C is satisfiable in $\mathcal{BALC}_l$.*

6   Related Works

Some recent works concerned with TDLs are surveyed below. In [4], Baader et al. considered the case where linear-time temporal operators are allowed to occur only in front of DL axioms over $\mathcal{ALC}$ (i.e., ABox assertions and general concept inclusion axioms), but not inside of concepts descriptions. They showed that reasoning in the presence of rigid roles becomes considerably simpler in this setting. The decision procedures described in [4] were developed for the purpose of showing worst-case complexity upper bounds: with rigid roles, satisfiability is 2EXPTIME-complete, without rigid roles, the complexity decreases further to EXPTIME-complete (i.e., the same complexity as reasoning in $\mathcal{ALC}$ alone). They also considered two other ways of decreasing the complexity of satisfiability to EXPTIME. Compared with [4], our approach is mainly intended to obtain: (1) reusable TDLs, i.e., the existing $\mathcal{ALC}$-based satisfiability testing algorithms are reusable and (2) "light-weight" TDLs, i.e., the complexity of satisfiability testing is the same as that of $\mathcal{ALC}$.

In [2], Baader et al. extended the known approaches to LTL runtime verification. In this approach, they used an $\mathcal{ALC}$-based temporal description logic, $\mathcal{ALC}$-LTL, instead of the propositional LTL. They also considered the case where states may be described in an incomplete way by $\mathcal{ALC}$-ABoxs, instead of assuming that the observed system behavior provides us with complete information about the states of the system. Compared with [2], applications of our proposed logics have not yet been proposed. In particular, it is not clear if the boundedness of the time domain in $\mathcal{BALC}_l$ is really useful for ontological reasoning.

References

1. A. Artale and E. Franconi, A survey of temporal extensions of description logics, *Annals of Mathematics and Artificial Intelligence* 30, pp. 171–210, 2000.
2. F. Baader, A. Bauer and M. Lippmann, Runtime verification using a temporal description logic, Proceedings of the 7th International Symposium on Frontiers of Combining Systems (FroCoS 2009), LNCS 5749, pp. 149–164, 2009.
3. F. Baader, D. Calvanese, D. McGuinness, D. Nardi and Peter F. Patel-Schneider (Eds.), The description logic handbook: Theory, implementation and applications, Cambridge University Press, 2003.
4. F. Baader, S. Ghilardi and C. Lutz, LTL over description logic axioms, Proceedings of the 11th International Conference on Principles of Knowledge Representation and Reasoning (KR 2008), pp. 684–694, AAAI Press, 2008.
5. A. Biere, A. Cimatti, E.M. Clarke, O. Strichman and Y. Zhu, Bounded model checking, *Advances in Computers* 58, pp. 118–149, 2003.
6. S. Cerrito, M.C. Mayer and S. Prand, First order linear temporal logic over finite time structures, LNCS 1705, pp. 62–76, 1999.
7. S. Cerrito and M.C. Mayer, Bounded model search in linear temporal logic and its application to planning, LNCS 1397, pp. 124–140, 1998.
8. I. Hodkinson, F. Wolter and M. Zakharyaschev, Decidable fragments of first-order temporal logics, *Annals of Pure and Applied Logic* 106, pp. 85–134, 2000.
9. N. Kamide, Reasoning about bounded time domain: An alternative to NP-complete fragments of LTL, Proceedings of the 2nd International Conference on Agents and Artificial Intelligence (ICAART 2010), Volume 1: Artificial Intelligence, pp. 536–539, INSTICC Press, 2010.
10. C. Lutz, F. Wolter and M. Zakharyashev, Temporal description logics: A survey, Proceedings of the 15th International Symposium on Temporal Representation and Reasoning (TIME 2008), pp. 3–14, IEEE Computer Society, 2008.
11. A. Pnueli, The temporal logic of programs, Proceedings of the 18th IEEE Symposium on Foundations of Computer Science, pp. 46–57, 1977.
12. A.N. Prior, Time and modality, Oxford: Clarendon Press, 1957.
13. A.N. Prior, Past, present and future, Oxford: Clarendon Press, 1967.
14. M. Schmidt-Schauss and G. Smolka, Attributive concept descriptions with complements, *Artificial Intelligence* 48, pp. 1–26, 1991.
15. F. Wolter and M. Zakharyashev, Temporalizing description logic, In: Frontiers of Combining Systems, D. Gabbay and M. de Rijke (Eds.), pp. 379–402, Studies Press/Wiley, 1999.

Guiding Reification in OWL
through Aggregation

Paula Severi[1], José Fiadeiro[1], and David Ekserdjian[2]

[1] Department of Computer Science
[2] Department of History of Art and Film
University of Leicester, United Kingdom

Abstract. We put forward a methodological approach aimed at guiding ontology modellers in choosing which relations to reify. Our proposal is based on the notion of aggregation as used in conceptual modelling approaches for representing situations that, normally, would require non-binary relations or complex integrity constraints. The feedback received from using the method in a real-word situation is that if offers a more controlled use of reification and a closer fit between the resulting ontology and the application domain as perceived by an expert.

1 Introduction

A well-known limitation of OWL 2 (Web Ontology Language) is that only binary relations between classes can be represented [1–3]. In practice, relations of arbitrary arity are quite common and they have to be represented in OWL in an indirect way by coding them as classes[3]. In the literature of Description Logic (DL) [4], the class codifying an n-ary relation ρ is called *the reification of* ρ [4].

As any codification, reification requires extra work in addition to 'simple' modelling, which can make it quite impractical (and unintuitive), especially when performed by people who are not 'experts': extra classes, predicates, individuals and axioms [5] need to be introduced and, as the number of classes increases, ontologies can become very difficult to read and understand, mainly because this additional information (which is encoded) is not directly visible. That is, there is a mismatch between the layer of abstraction at which domain modellers work and that of the representation where information is encoded, which is particularly harmful when we want to extend and reuse ontologies.

In this paper, we propose the use of a methodological construction that has been devised many years ago in the database community, which is based on the notion of aggregation as proposed in [6]. Aggregation is an abstraction that

---

[3] Similarly for RDF (Resource Description Framework)

[4] The term *reification* can have several meanings and uses in Logic in general, and the Semantic Web in particular. In this paper, we use it as a synonym for encoding n-ary relations as classes. We do not use it to refer to the usage of RDF as a metalanguage to describe other logics, or in situations in which a statement can be assigned a URI and treated as a resource, or the use of classes as individuals.

was offered therein for increasing the "understandibility of relational models by the imposition of additional semantic structure". Although, in ontologies, the technical problems that arise are not necessarily the same as those of relational databases, the methodological issues are similar in the sense that the solution to our problem lies first of all in helping modellers to conceptualize the real world in a way that can lead to a better representation, and then offering them a mechanism for implementing these semantic structures in ontologies. By 'better' we mean a more controlled use of reification and a closer fit between the resulting ontology and the real-world domain as perceived by an expert.

Having this in mind, we start by motivating the problem using the case study that led us to investigate the representation of n-ary relationships — an ontology of 16th-century Italian altarpieces. In Section 3, we discuss a formal, set-theoretical, notion of aggregation and the way that it can be implemented in ontologies through reification. Then, in Section 4, we discuss how aggregation as a modelling abstraction can be used effectively in a number of situations that are recurrent in domains such as that of altarpieces.

2 Motivation

In order to illustrate some of the problems that may arise from the limitations of having to encode n-ary relations through reification and the method that we propose to minimize them, we use the Ontology of Altarpieces [7] — a joint project between the Departments of Computer Science and History of Art and Film at the University of Leicester. This case study is a good example of a domain in which n-ary relations arise quite naturally and frequently.

Suppose that we want to express the following knowledge as produced in natural language by an art expert:

1. *The altarpiece painted by Raphael called "Sistine Madonna"*[5] *has the figure of the Virgin on it.*
2. *The altarpiece painted by Raphael called "Sistine Madonna" has the figure of the Christ on it.*
3. *The altarpiece painted by Raphael called "The Marriage of the Virgin"*[6] *has the figure of the Virgin on it.*

The above sentences can be represented by a ternary relation $hasFigure$ between the sets $Painters$, $PictureNames$ and $Figures$.

$$hasFigure = \{(\mathsf{raphael}, \mathsf{sistine\ madonna}, \mathsf{virgin}),$$
$$(\mathsf{raphael}, \mathsf{sistine\ madonna}, \mathsf{christ}),$$
$$(\mathsf{raphael}, \mathsf{marriage\ of\ virgin}, \mathsf{virgin})\}$$

Figure 1 shows an entity relation (ER) diagram for the relationship $hasFigure$ of which the set above is an extension. This relation cannot be represented in

---

[5] See http://en.wikipedia.org/wiki/Sistine_Madonna.

[6] See http://en.wikipedia.org/wiki/The_Marriage_of_the_Virgin_(Raphael).
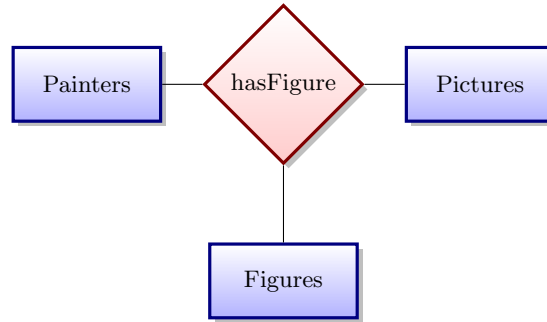
Fig. 1. ER diagram: hasFigure as a relationship of arity 3

OWL unless we code it as a class $C_{hasFigure}$ of individuals that represent the tuples — the reification of the relation [4]. For example, we create an individual r_1 that represents the tuple

$$\langle \mathsf{raphael}, \mathsf{sistine\ madonna}, \mathsf{virgin} \rangle$$

and we connect r_1 to each component in the tuple using the role names painter, picturename and figure as shown in Figure 2.
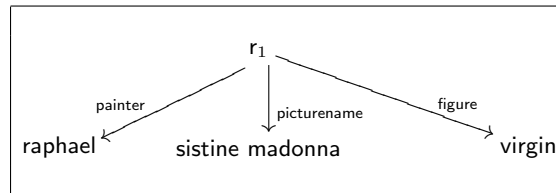
**Fig. 2.** Connecting r_1 to the components of the tuple

However, reifying *hasFigure* is not necessarily the right decision that a modeller should make. This is because Figure 1 shows the relationship *hasFigure* isolated from the rest of the ontology. A diagram that shows other relationships between these entities in a wider conceptual model of the domain of altarpieces is depicted in Figure 3 In this diagram, we can see another relationship involving *Painters* and *PictureNames* and a number of 'descriptive attributes' (functional relationships involving a data type) that apply to that relationship. Naturally, one cannot take a blind approach to the representation of these aspects of the domain and reify relations as they come: the complexity of the ontologies thus generated would be even beyond skilled computer scientists, let alone domain experts.

Suppose we want to express that *The Virgin is holding Christ in the altarpiece called "Sistine Madonna" by Raphael.* To represent the above sentence, we need a relation *holds* of arity 4 where

$$holds = \{(\mathsf{raphael}, \mathsf{sistine\ madonna}, \mathsf{virgin}, \mathsf{christ})\}$$

In the Ontology of Altarpieces we have about 20 relations of arity 3 such as *hasFigure* and more than 4000 relations such as *holds* of arity 4. It would not make sense to blindly reify all the relations of arity strictly greater than 2. Given that each relation may have an average of 1000 tuples, doing so would mean 1000 individuals for coding the tuples and 1000 x 4 pairs connecting the individuals with their components. If we consider that the details of those figures and other attributes of the altarpieces need to be represented, it is easy to see that the whole ontology would become quite unwieldy.

In other words, basic questions that a modeller needs to consider very carefully is: "Can I reduce the number of reifications in my ontology?", "Which relations are more convenient to reify?". Our answer in this paper is given in methodological terms, inspired by similar problems faced by the relational database community 30 years ago.
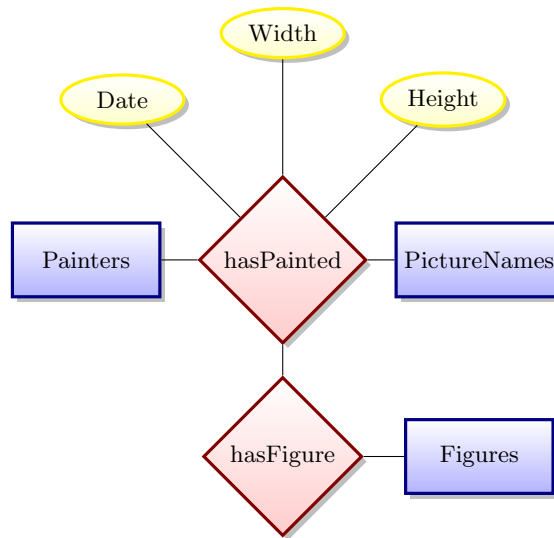
Fig. 3. ER diagram: how the hasFigure relationship interacts with other relationships

The examples presented in this paper are very simple and try to extract the main concepts behind the method. However, we have applied aggregations to more complex relations in the Ontology of Altarpieces.

3   Aggregation in Set Theory vs Reification in OWL

Aggregation as defined in [6] refers to an abstraction in which a relationship between objects is regarded as a higher-level object. The intention, as stated therein, was to adapt cartesian product structures (as proposed by T. Hoare for record structures in programming languages) to be used in the context of relational models. Although a formal definition was not given as a semantics for the abstraction, we found it useful to advance one so that, on the one hand, we can be precise about our usage of the term and, on the other hand, we can relate it to the mechanism of reification. Throughout the paper, we use the Greek alphabet for entities that we define in Set Theory.

Definition 1. *Let $\Delta_1, \Delta_2 \subseteq \Delta$ and $\rho \subseteq \Delta_1 \times \Delta_2$ be a binary relation. An aggregation of ρ is a set $\Delta_\rho \subseteq \Delta$ together with two (total) functions π_1 and π_2 (called* projections*) from Δ_ρ to Δ_1 and Δ_2, respectively, such that:*

1. *For all $r \in \Delta_\rho$, $\langle \pi_1(r), \pi_2(r) \rangle \in \rho$ — i.e., there is no 'junk' in Δ_ρ.*
2. *For all $\langle x_1, x_2 \rangle \in \rho$, there exists $r \in \Delta_\rho$ such that $\pi_1(r) = x_1$ and $\pi_2(r) = x_2$ — the aggregation covers the whole relation ρ.*
3. *For all $r_1, r_2 \in \Delta_\rho$, if $\pi_1(r_1) = \pi_1(r_2)$ and $\pi_2(r_1) = \pi_2(r_2)$ then $r_1 = r_2$ — i.e., there is no 'confusion': every tuple of the relation has a unique representation as an aggregate.*

It is trivial to prove the following result:

Proposition 1. *Δ_ρ is isomorphic to ρ.*

That is, an aggregation is indeed offering a 'faithfull' representation of the relation. We denote this isomorphism by Ψ_ρ or just Ψ where $\Psi(r) = \langle \pi_1(r), \pi_2(r) \rangle$. Its inverse defines the encoding of the relation, i.e. it assigns to each tuple in the relation ρ a unique element (aggregate) of the set Δ_ρ.

Informally, the reification of a relation ρ is a class C_ρ representing the tuples of ρ [4, 8]. This representation should be as close as possible to the relation itself in order to avoid any possible mismatch between the representation and the model that the expert has in mind. In order to be able to analyse this relationship, we have found it useful to provide a concrete definition of how we are using the notion of reification:

Definition 2. *Let $\Delta_1, \Delta_2 \subseteq \Delta$ and $\rho \subseteq \Delta_1 \times \Delta_2$ be a binary relation. A* reification *of ρ in OWL is a concept C_ρ together with two roles P_1 and P_2, called* projections*, two domains D_1 and D_2, and the following collection of axioms:*

(proj func)	$\top \sqsubseteq \; \leq 1P_1 \sqcap \; \leq 1P_2$
(proj domain)	$\exists P_1.\top \sqcap \exists P_2.\top \sqsubseteq C_\rho$
(proj range)	$\top \sqsubseteq \forall P_1.D_1 \sqcap \forall P_2.D_2$
(proj totality)	$C_\rho \sqsubseteq \exists P_1.D_1 \sqcap \exists P_2.D_2$
(unique rep)	$C_\rho \; \mathsf{hasKey}(P_1, P_2)$

These definitions can be generalized to relations of arbitrary arity. We can now define more precisely how a reification relates to the relation:

Definition 3. *Let $\rho \subseteq \Delta_1 \times \Delta_2$ be a binary relation. Given an interpretation I, we say that the reification $(C_\rho, D_1, D_2, P_1, P_2)$ is* faithful *to ρ in relation to I iff $D_1^I = \Delta_1$, $D_2^I = \Delta_2$, and (C_ρ^I, P_1^I, P_2^I) is an aggregation of ρ.*

Unfortunately, the axioms that are part of the reification are not sufficient to guarantee that it is faithfull to ρ in relation to every interpretation:

- The first four axioms state that the role names P_1 and P_2 are total functions from C_ρ to D_1 and D_2, respectively. However, a limitation of OWL is that the reasoner does not show any inconsistency if we forget to define P_1 or P_2 for some element of C_ρ (see [9]). This type of mistake could obviously avoided if OWL provided us with relations of arity n.
- The axiom (unique rep) states that two named individuals in C_ρ that have the same projections should be equal. This axiom is weaker than the third condition of Definition 1 in the sense that unicity of the representation is not enforced for *all* individuals but only on those that are explicitly *named* in the ontology. This is because the hasKey constructor of OWL-2 is a weak form of key representation (so-called "EasyKey constraints") that is valid only for individuals belonging to the Herbrand Universe [10].

Summarising, reification is not only hard work (in the sense that it requires the modeller to introduce a number of roles and axioms that are 'technical', i.e. more related to the limitations of the formalism and less specific to the domain of application) but also prone to errors. Essentially, errors may arise if the modeller forgets to enforce the properties that cannot be expressed in OWL: totality, 'no junk' or coverage.

Notice that, in the specific case of binary relations, we can add an atomic role R to the ontology and add the following axiom to the reification, which corresponds to the first condition of Definition 1 — 'no junk':

$$(R\text{-contains}) \quad (P_1)^{-1} \circ P_2 \sqsubseteq R$$

This axiom states that the relation R can be recovered from the reification C_ρ through the projections P_1 and P_2. In this case, faithfulness would require that $R^I = C_\rho^I$. The ability to work with an atomic role R also has methodological advantages as illustrated in the next section.

Also note that, in the binary case, the converse of (R-contains), which would correspond to the second condition of Definition 1, is as follows

$$(R\text{-inclusion}) \quad R \sqsubseteq (P_1)^{-1} \circ P_2$$

However, this axiom cannot be expressed in OWL in the above form [7] because the right-hand side of the inclusion is not a role name (see [3]).

---

[7] This is not a proof that the axiom cannot be expressed in the logic which would be more involved.

These shortcomings show why methodological support is necessary when using reification in OWL: one should make sure that abstraction mechanisms are available through which a modeller can keep a close fit between the representation and the domain and that these mechanisms are supported, as much as possible, by tools. The aim of the techniques put forward in the next section is precisely to overcome the gap that may exist between the perception of the relationships that exist in the domain of discourse and the use of reification to encode them in OWL.

4 Guiding the Use of Reifications in Ontologies

In this section, we put forward a methodological approach aimed at guiding the modeller in the use of reification. The method is based on the usage of the semantic primitive of aggregation as used in conceptual modelling precisely for representing situations that, normally, would require non-binary relations or complex integrity constraints [11]. We illustrate the approach with some examples that are representative of the situations that we have encountered in the altarpieces project.
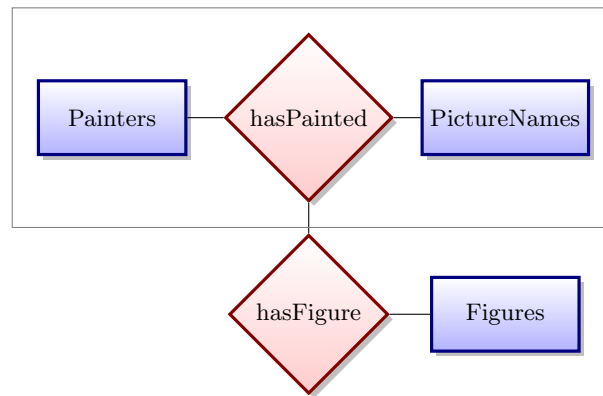

Fig. 4. ER diagram: hasPainted as an aggregate of hasFigure

4.1 Relationships amongst Relationships

A recurrent situation in database modelling is the use of aggregation in order to reduce certain ternary relationships to binary ones [11]. Using ER diagrams, the method can be explained in terms of evolving situations such as the one depicted in Figure 1 to the one depicted in Figure 4. More specifically, the method consists in identifying a binary relationship — *hasPainted* — such that the ternary

relationship — $hasFigure$ — can be expressed as a binary relationship between the aggregation of the former — $hasPainted$ — and the remaining domain — $Figures$. The aggregation of a relationship is indicated by the box that surrounds the relationship diagram. Following this method, instead of reifying the whole relation $hasFigure$, we reify $hasPainted$. Since $hasPainted$ is a binary relation, we represent it by the role hasPainted and consider the reification of $hasPainted$ as in Definition 2. For this, we introduce the class Altarpieces as the reification $C_{hasPainted}$ and the roles painter and picturename as the projections. The relation $hasFigure$ is represented as an object property whose domain is $C_{hasPainted}$ and whose range is Figures as shown in Figure 5.
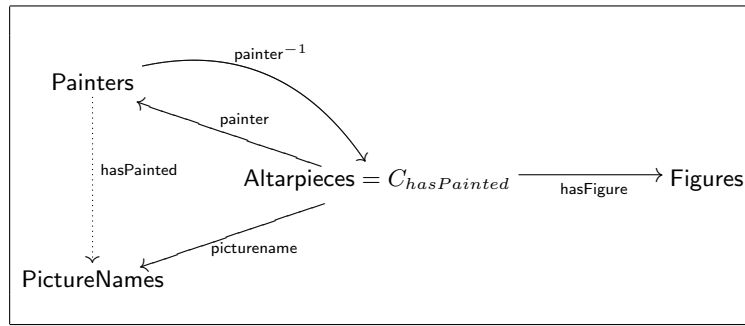
Fig. 5. Representation of hasFigure that considers hasPainted as an aggregate

We have chosen the simplest example from the ontology to illustrate the method. In this case it is clear from the use of the noun in the sentences in Section 2 that we should have chosen to model altarpieces as a class Altarpieces right from the start. The point is that, from the point of view of conceptual modelling, altarpieces are an aggregation of a relation: art experts identify altarpieces precisely through the name of the painter and the designation of the picture. Therefore, the class Altarpieces corresponds, in a natural way, to the reification of the relation $hasPainted$. In the case of other examples in our ontology, for instance the relations $holds$ and $wears$, among others, the class does not arise so naturally (indeed, they do not correspond to nouns), which explains why using reification to represent them is somewhat artificial, i.e. driven by technical, not conceptual concerns.

Definition 4. *Let* $\Delta_1, \Delta_2, \Delta_3 \subseteq \Delta$, $\rho \subseteq \Delta_1 \times \Delta_2$ *be a binary relation and* $\rho' \subseteq \Delta_1 \times \Delta_2 \times \Delta_3$ *be a ternary relation. We say that* ρ *participates in* ρ' *if the following condition is satisfied:*

- *For all* $x \in \Delta_1, y \in \Delta_2, z \in \Delta_3$, $(x,y) \in \rho$ *whenever* $(x,y,z) \in \rho'$.

If ρ participates in ρ' then ρ' "can be seen" as a binary relation between the aggregation Δ_ρ and Δ_3.

The relationship *hasPainted* 'participates' in the relationship *hasFigure* since the following constraint is satisfied:

$$\text{if } (x, y, z) \in hasFigure \text{ then } (x, y) \in hasPainted. \tag{1}$$

The above constraint is enforced in OWL by the axiom that states that the domain of hasFigure is $C_{hasPainted}$, and the axiom (hasPainted-contains) from the binary-relation extension of Definition 2 (see Figure 5).

The method that we propose for guiding reification consists in analysing which relations participate in other relations: if ρ *participates in* ρ' then, instead of reifying the whole relation ρ', we should consider reifying the participating relation ρ and represent ρ' as a role whose domain is C_ρ. If ρ participates in yet another relation, say ρ'', that relation does not need to be reified either but reuse instead the reification of ρ. Indeed, *hasPainted* participates in many relations other than *hasFigure* — e.g. *hasField*, for representing polyptych altarpieces that have many fields. All the corresponding relations can be represented in OWL as object properties whose domain is $C_{hasPainted}$ as in Figure 5.

Another important aspect of this representation (which is another reason why it is better than the reified ternary relation) is that we now have the relation *hasFigure* represented as a property hasFigure and not as a class $C_{hasFigure}$. Reifications represent properties but they cannot be used in the syntax as properties because they are actually classes. We cannot use constructors for roles on $C_{hasFigure}$ such as composition, quantification or transitive closure, which may restrict the ability of the modeller to capture important aspects of the domain. Whilst the representation of *hasFigure* as a property allows us to use the role name hasFigure in quantifications or in compositions. For instance, we can use an existential quantifier over the role hasFigure to express that all altarpieces must have some religious figure on it as follows:

$$\text{Altarpieces} \sqsubseteq \exists\text{hasFigure.Religious}$$

4.2   Descriptive attributes

Another related methodological guideline for the use of reification arises from what in [11] are called descriptive attributes. Descriptive attributes are used to record information about a relationship rather than about one of the participating entities, again using an aggregation. From a conceptual modelling point of view, they allow us to capture typical situations in which a functional dependency exists on a ternary relation as an attribute of the aggregation of a binary relation. For example, it would be intuitive to represent *height*, *width* and *date* in Figure 6 as descriptive attributes associated with the relationship *hasPainted*.

Definition 5. *Let* $\rho \subseteq \Delta_1 \times \Delta_2$ *and* $\rho' \subseteq \Delta_1 \times \Delta_2 \times \Delta_3$. *We say that* ρ' *is descriptive attribute of* ρ *if the following conditions holds:*

1. *ρ' is a function from $\Delta_1 \times \Delta_2$ into Δ_3.*
2. *ρ participates in ρ' (see Definition 4).*
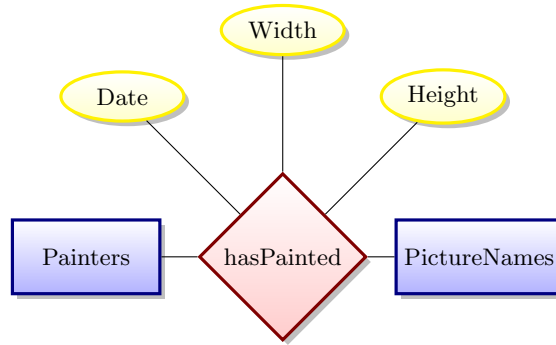
Fig. 6. ER diagram: hasPainted and descriptive attributes associated with it

This property is, indeed, satisfied by *height*:

1. There is a functional dependency between the height of the altarpiece and the pair given by the painter and the picture name. In other words, the ternary relation *height* is actually a function

$$height \in Painters \times PictureNames \rightarrow Int$$

2. *hasPainted* participates in *height*, i.e.:

$$\text{if } (x, y, z) \in height \text{ then } (x, y) \in hasPainted. \tag{2}$$

Given that descriptive attributes involve a participating relation, the methodological guidelines that we discussed in 4.1 suggest that descriptive attributes be represented as (functional) roles of the reification of the participating relation. For instance, using the reification $C_{hasPainted}$, the descriptive attribute *height* can be represented in OWL by a data type property height and two axioms

$$\top \sqsubseteq\ \leq 1.\mathsf{height}$$

$$\geq 1\mathsf{height} \sqsubseteq C_{hasPainted}$$

The constraints associated with the descriptive attribute *height* are deduced from the above two axioms and the axiom (hasPainted-contains).

5 Related Work and Concluding Remarks

The use of conceptual modelling primitives in the context of ontologies is not new. For instance, [12] and [13] show how to transform ER diagrams into Description Logic. However, this transformation does not include relationships involving relationships or descriptive attributes as illustrated in Section 4, nor does it address aggregation as a modelling abstraction. A paper that focuses specifically on

aggregation is [14]. However, the author represents aggregations using union of classes, which does not correspond in any way to their original meaning [6]. Our use of aggregation (based on cartesian products) adheres to its use in databases and explores its methodological advantages for conceptual modelling [11].

Our approach is also related with proposals that, like [15], put forward patterns for representing relations $\rho \subseteq A \times B \times C$. The third case of Pattern 1 in that note does the reification of the whole relation and the remaining cases do the reification of $B \times C$ and represent ρ as a property whose range is the reification $C_{B \times C}$. Our method is based on semantic abstractions and, therefore, goes beyond simple patterns. In fact, it deepens the study of these patterns in the sense that it guides the application of reification by the identification of relations that, like *hasPainted*, participate in other relations.

On the subject of representing non-binary relations, [16] provides a trivial extension of the syntax of OWL with n-ary properties. Decidability is not studied and the extension contemplates only properties: there are no other constructors to deal with predicates of arity n as in [4, 8]. On the other hand, OWL 2 provides the possibility of defining n-ary datatype predicates F, albeit in a restricted way [17]. We can use an n-ary predicate F in expressions of the form $\forall P_1 \ldots P_n.F$ or $\exists P_1 \ldots P_n.F$ where $P_1 \ldots P_n$ are binary data type predicates. The n-ary predicate F is actually a functional proposition defined implicitly by a formula of the form $\lambda(x_1 \ldots x_n).\mathsf{comp}(p, q)$ where $\mathsf{comp} \in \{\leq, =, \geq, <, >, \neq\}$ and p and q are linear polynomials on $x_1, \ldots, x_n$. However, OWL does not support the definition of n-ary predicates by listing the tuples as for object and datatype properties.

Our plans for future work include further study of the extensions of DL for n-ary relations [4, 8, 18]. In particular, we have in mind to investigate the formal mechanisms that, from a DL point of view, can support the constructions illustrated in Section 4. For instance, following the argument in Section 10.6.1 of [5], the class Altarpieces can be seen as a binary relation with two attributes painter and picturename or as a ternary relation with three attributes painter, picturename and height. This is possible in the case of a descriptive attribute because of the fact that there is a functional dependency. However, in the case of general ternary relations such as hasFigure, this is not possible: the class Altarpieces cannot be seen as a ternary relation with attributes painter, picturename and hasFigure. The relation hasFigure is being represented by the role name hasFigure and not by Altarpieces. This change of point of view is important when we consider aggregations of aggregations, which is another topic that we are exploring.

The examples presented in this paper are very simple and try to extract the main concepts behind the method. However, we have applied aggregations to more complex relations in the Ontology of Altarpieces. For instance, there are cases where we need to add another layer of aggregations and, therefore, consider aggregations of aggregations. This is the case of many relations such as *holds* and *wears* where the relation *hasFigure* itself needs to be reified.

References

1. W3C: OWL 2 Overview. http://www.w3.org/TR/owl2-overview/
2. Horrocks, I., Patel-Schneider, P.F., McGuinness, D.L., Welty, C.A.: OWL: a Description Logic Based Ontology Language for the Semantic Web. In Baader, F., Calvanese, D., McGuinness, D., Nardi, D., Patel-Schneider, P.F., eds.: The Description Logic Handbook. CUP (2007)
3. Horrocks, I., Kutz, O., Sattler, U.: The Even More Irresistible $\mathcal{SROIQ}$. In: Proc. of the 10th Int. Conf. on Principles of Knowledge Representation and Reasoning (KR 2006), AAAI Press (2006) 57–67
4. Calvanese, D., Giacomo, G.D.: Expressive Descriptions Logics. In: The Description Logic Handbook. CUP 193–236
5. Borgida, A., Brachman, R.J.: Conceptual Modeling with Description Logics. (2003) 349–372
6. Smith, J., Smith, D.: Database abstractions: aggregation. Communications of the ACM **20**(6) (June 1977) 405–413
7. Ekserdjian, D., P.Severi: Ontology of Altarpieces. Technical report, University of Leicester (2009)
8. Giacomo, G.D., Lenzerini, M.: Description Logics with Inverse Roles, Functional Restrictions, and N-ary Relations. In: JELIA '94, Springer (1994) 332–346
9. Motik, B., Horrocks, I., Sattler, U.: Bridging the Gap Between OWL and Relational Databases. J. of Web Semantics **7**(2) (April 2009) 74–89
10. Parsia, B., Sattler, U., Schneider, T.: Easy Keys for OWL. In: OWLED. (2008)
11. Ramakrishna, R., Gehrke, J.: Database Management Systems (3rd edition). Mc Graw Hill (2003)
12. Sattler, U., Calvanese, D., Molitor, R.: Relation with Other Formalisms. In: The Description Logic Handbook. CUP 149–192
13. Borgida, A., Lenzerini, M., Rosati, R.: Description Logic for Databases. In: The Description Logic Handbook. CUP 149–192
14. Veres, C.: Aggregation in Ontologies: Practical Implementations in OWL. In: ICWE 2005. LNCS, Springer (2005)
15. Noy, N., Rector, A.: N-ary relations. http://www.w3.org/TR/swbp-n-aryRelations/
16. Salguero, A., Delgado, C., Araque, F.: Easing the Definition of N-Ary Relations for Supporting Spatio-Temporal Models in OWL. In: EUROCAST. (2009) 271–278
17. B.Parsia, Stattler, U.: OWL 2: Data Range Extension: Linear Equations. http://www.w3.org/TR/2009/NOTE-owl2-dr-linear-20091027/
18. Giacomo, G.D., Lenzerini, M.: What's in an Aggregate: Foundations for Description Logics with Tuples and Sets. In: IJCAI (1). (1995) 801–807

Generating Referring Expressions with OWL2

Yuan Ren, Kees van Deemter, and Jeff Z. Pan

Department of Computing Science
University of Aberdeen
Aberdeen, UK

Abstract. The task of generating referring expressions, an important subtask of Natural Language Generation is to generate phrases that uniquely identify domain entities. Until recently, many GRE algorithms were developed using only simple and essentially home-made formalisms. Following the fast development of ontology-based systems, reinterpretations of GRE in terms of description logic have been studied. However, the quantifiers generated are still limited, not exceeding the works covered by existing GRE approaches. In this paper, we propose an DL-based approach to GRE that exploits the full power of OWL2 to generate referring expressions that goes beyond the expressivity of previous GRE algorithms. The potential of DL reasoning in GRE is also discussed.

1 GRE and KR: the story so far

Generation of Referring Expressions (GRE) is the subtask of Natural Language Generation (NLG) that focuses on the identification of objects in natural language. For example, Fig.1 depicts the relations between several individuals of women, dogs and cats. In such a scenario, a GRE system might identify a given object as "Dog" or, if this fails to identify the referent uniquely, "the Dog that loves a Cat", which is literally unique for $d1$. Reference has long been a key issue in theoretical linguistics and psycholinguistics, and GRE is a crucial component of almost every practical NLG system [5, 4]. Accordingly, GRE has become one of the best developed areas of NLG, with links to many other areas of Cognitive Science.
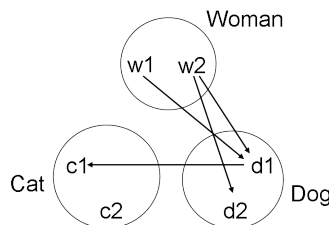
Fig. 1. An example of women, dogs and cats, in which edges from women to dogs denote $feed$ relations, from dogs to cats denote $love$ relations.

Traditional GRE algorithms are usually based on very simple, essentially home-made, forms of Knowledge Representation (KR); in many cases all properties are ex-

pressed in a simple $\langle Attribute : Value \rangle$ format, e.g $\langle Type : Dog \rangle$. This is perhaps justifiable as long as the properties expressed by these algorithms are simple one-place predicates (being a cat, being a Dog, etc.), but when logically more complex descriptions are involved, the potential advantages of "serious" KR become overwhelming. A few proposals in recent years have started to combine GRE with well-developed KR. Following on from earlier work based on labelled directed graphs [9, 14], for example, analyzed GRE as a *projection* problem in Conceptual Graphs. More recently, [1] reinterpreted GRE as a problem in Description Logic (DL), producing a formula such as $Dog \sqcap \exists love.Cat$ to refer to the (unique) Dog who loves at least one Cat. It is this last approach that forms the basis of the present paper, which aims to show that when a principled, logic based approach is chosen, it becomes possible to refer to objects which no existing approach to GRE has so far been able to refer to.

In doing so, we shall follow Areces et al (and many other researchers in GRE) in focussing on the semantic core of the GRE problem: we shall be content to generate descriptions of logical/semantic content, leaving the decision of what English (or other languages) words to use for expressing this content (e.g., 'the ancient dog', or 'the dog which is old') to later stages in the NLG pipeline. Secondly, we shall assume that all domain objects are equally salient [8]. Perhaps most importantly, we do not consider here the important matter of the naturalness or efficacy of the descriptions generated. We shall be content proposing an algorithm that produces uniquely referring expressions whenever such expressions are possible, leaving the choice of the *optimal* referring expression in each given situation for later.

In what follows we start by explaining how DL has been applied in GRE (Sec.2), pointing out the limitations of existing works. In Sec.3 we discuss which kinds of additional expressivity are required and how they can be achieved through modern DL. In Sec.4 we present an generic algorithm to compute these expressive REs. Sec.5 concludes the paper by comparing its aims and achievements with current practise in GRE.

2 DL for GRE

2.1 Description Logics

Description Logic (DLs) come in different flavours, based on decidable fragments of first-order logic. A DL-based KB represents the domain with descriptions of concepts, relations, and their instances. DLs underpin the Web Ontology Language (OWL), whose latest version of OWL2 [11] is based on DL $\mathcal{SROIQ}$ [7].

An $\mathcal{SROIQ}$ ontology Σ usually consists of a TBox $\mathcal{T}$ and an ABox $\mathcal{A}$. $\mathcal{T}$ contains a set of concept inclusion axioms such as $C \sqsubseteq D$, relation inclusion axioms such as $R \sqsubseteq S$, $R_1 \circ \ldots \circ R_n \sqsubseteq S$, and other axioms saying that a particular relation is functional, or reflexive, or symmetric, etc.; $\mathcal{A}$ contains axioms about individuals, e.g. $a : C$ (a is an instance of C), $(a, b) : R$ (a has an R relation with b).

Given a set of atomic concepts, the entire set of concepts expressible by $\mathcal{SROIQ}$ is defined recursively. Assuming that C and D are concepts, then so are $\top \mid \bot \mid A \mid \neg C \mid C \sqcap D \mid C \sqcup D \mid \exists R.C \mid \forall R.C \mid \leq nR.C \mid \geq nR.C \mid \exists R.Self \mid \{a_1, \ldots, a_n\}$, where $\top$ is the top concept, $\bot$ the bottom concept, A an atomic concept, n a non-negative

integer number, $\exists.Self$ the self-restriction, a_i individual names and R a relation which can either be an atomic relation r or the inverse of another relation (R^-).

An *interpretation* $\mathcal{I}$ is a pair $\langle \Delta^{\mathcal{I}}, \cdot^{\mathcal{I}} \rangle$ where $\Delta^{\mathcal{I}}$ is a non-empty set and $\cdot^{\mathcal{I}}$ is a function that maps atomic concept A to $A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$, atomic role r to $r^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$ and individual a to $a^I \in \Delta^{\mathcal{I}}$. The interpretation of complex concepts and axioms can be defined inductively based on their semantics, e.g. $(C \sqcap D)^{\mathcal{I}} = C^{\mathcal{I}} \cap D^{\mathcal{I}}$, etc.

$\mathcal{I}$ is a *model* of Σ, written $\mathcal{I} \models \Sigma$, iff the semantics of all the axioms in Σ are satisfied. It should be noted that one Σ can have multiple models. For example when $\mathcal{T} = \emptyset, \mathcal{A} = \{a : A \sqcup B\}$, there can be a model $\mathcal{I}_1$ s.t. $a^{\mathcal{I}_1} \in A^{\mathcal{I}_1}, a^{\mathcal{I}_1} \notin B^{\mathcal{I}_1}$, and another model $\mathcal{I}_2$ s.t. $a^{\mathcal{I}_2} \notin A^{\mathcal{I}_2}, a^{\mathcal{I}_2} \in B^{\mathcal{I}_2}$. In other word, the world is open. For details, we refer the readers to [2].

When dealing with a closed world, people usually (partially) close the ontology with a DBox $\mathcal{D}$ [12], which is syntactically similar to the $\mathcal{A}$. However, $\mathcal{D}$ contains only $a : A$ $((a, b) : r)$ where A (r) is atomic. Furthermore, every concept or relation appearing in $\mathcal{D}$ is closed. Its extension is exactly defined by the contents of $\mathcal{D}$, i.e. if A (r) appearing in $\mathcal{D}$ and $a : A \notin \mathcal{D}$ $((a, b) : r \notin \mathcal{D})$, then $a : \neg A$ $((a, b) : \neg r)$, thus is the same in all the models. The concepts and relations not appearing in $\mathcal{D}$ are still open.

DL reasoning can be exploited to infer implicit information from ontologies. For example, given $\mathcal{T} = \{Dog \sqsubseteq \exists feed^-.Woman\}$ and $\mathcal{A} = \{d1 : Dog, w1 : Woman\}$, we know that there must be some $Woman$ who feeds $d1$. When the domain is closed as $\mathcal{D} = \mathcal{A}$ we can further infer that this $Woman$ is $w1$ although there is no explicit relation between $w1$ and $d1$. The complexity of such reasoning services is normally 2NEXPTIME-complete.

2.2 Background Assumptions

When applying DL to GRE, people usually impose the following assumptions.

- Identifiers cannot be used in REs. For example, "the Woman who feeds $d1$" would be invalid, because $d1$ is an identifier. Such identifiers are typically outlawed in GRE because, in many applications, many objects do not have identifiers that readers/hearers would be familiar with: e.g. chairs, trees, or time periods seldom have commonly known identifiers.
- *Closed Domain Assumption (CWA)*: In existing works regarding DL and GRE, people assume that $\mathcal{D} = \mathcal{A}$. Furthermore, the domain is usually considered to be finite and containing individuals only visible in $\mathcal{D}$. As we will show, this "partially" close the interpretation of the atomic symbols mentioned in $\mathcal{A}$ but will still allow the rest open.
- *Unique Name Assumption (UNA)*: Different names denote different individuals. If, for example, $w1$ and $w2$ may potentially be the same woman, then we can not distinguish one from the other.

We follow these assumptions when discussing existing works and presenting our approach. We will also show how our approach can be extended to achieve more when these assumptions are not imposed. In addition, we consider the entire KB, including both $\mathcal{A}$ ($\mathcal{D}$) and $\mathcal{T}$.

It is worth mentioning that, in the syntax of $\mathcal{SROIQ}$, negation of relations are not allowed in concept expressions, e.g. you can not compose a concept $\exists \neg feed.Dog$. However, if we impose the CWA and close $feed$, then we can interpret $(\neg feed)^{\mathcal{I}} = \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}} \setminus feed^{\mathcal{I}}$. In the rest of the paper, we use this as a syntactic sugar.

2.3 Using DL for GRE

Every DL concept can be interpreted as a set. If the KB allows one to prove that this set is a singleton then the concept is a referring expression. It is this simple idea (earlier expounded by [6]) that [1] explored. In doing so, they say little about the TBox, appearing to consider only the ABox (DBox), which contains only axioms about instances of atomic concepts and relations. For example, the domain in Fig.1 can be described as

KB1: $\mathcal{T}_1 = \emptyset$, $\mathcal{A}_1 = \{w1 : Woman, w2 : Woman, d1 : Dog, d2 : Dog, c1 : Cat, c2 : Cat, (w1, d1) : feed, (w2, d1) : feed, (w2, d2) : feed, (d1, c1) : love\}$

Assuming that this represents a Closed World, the authors propose an algorithm that is able to generate descriptions by partitioning the domain.[1] More precisely, the algorithm first finds out which objects are describable through increasingly large conjunction of (possibly negative) atomic concepts, then tries to extend these conjunctions with complex concepts of the form $(\neg)\exists R.Concept$, then with concepts of the form $(\neg)\exists R1.(Concept \sqcap (\neg)\exists R2.Concept)$, and so on. At each stage, only those concepts that have been acceptable through earlier stages are used. Consider, for instance, KB1 above. Regardless of what the intended referent is, the algorithm starts partitioning the domain with atomic concept (suppose staring with Dog) in (1), then the ones in (2), then the ones in (3). Each stage makes use of all previous stages. During stage (3), for example, the object $w2$ could only be identified because $d2$ was identified in phase (2).

(1). $Dog = \{d1, d2\}$, $\neg Dog \sqcap Woman = \{w1, w2\}$,
$\neg Dog \sqcap \neg Woman = \{c1, c2\}$.
(2). $Dog \sqcap \exists love.(\neg Dog \sqcap \neg Woman) = \{d1\}$,
$Dog \sqcap \neg\exists love.(\neg Dog \sqcap \neg Woman) = \{d2\}$.
(3). $(\neg Dog \sqcap Woman) \sqcap \exists feed.(Dog \sqcap \neg\exists love.(\neg Dog \sqcap \neg Woman)) = \{w2\}$,
$(\neg Dog \sqcap Woman) \sqcap \neg\exists feed.(Dog \sqcap \neg\exists love.(\neg Dog \sqcap \neg Woman)) = \{w1\}$.

As before, we disregard the important question of the quality of the descriptions generated, other than whether they do or do not identify a given referent uniquely. Other aspects of quality depend in part on details, such as the question in which order atomic properties are combined during phase (1), and analogously during later phases.

---

[1] Areces et al. [1] consider several DL fragments (e.g., $\mathcal{ALC}$ and $\mathcal{EL}$). Which referring expressions are expressible, in their framework, depends on which DL fragment is chosen. Their analysis of the differences between fragments is perhaps the most valuable aspect of their paper. Existential quantification, however, is the only quantifier that was used, and inverse relations are not considered either.

Although this demonstrates how DL can be used in GRE, it does not extend the expressive power of GRE. This is not because of some specific lapse on the part of the authors: it seems to have escaped the GRE community as a whole that relations can enter REs in a variety of alternative ways. Also, the above algorithm considers only the ABox therefore some background information will not be used. For example, suppose we extend Fig.1 with background knowledge saying that one should care about thoes beloved by whom he/she is feeding, and an additional love relation (Fig.2).
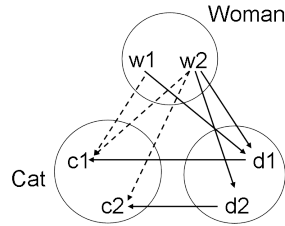
Fig. 2. An extended example of Fig.1. Edges from women to cats denote *care* relations. Dashed-edge indicates implicit relations.

Together, the domain can be described as:

KB2: $\mathcal{T}_2 = \{feed \circ love \sqsubseteq care\}$, $\mathcal{A}_2 = \mathcal{A}_1 \cup \{(d2, c2) : love\}$

The TBox axiom should allow the inference of additional facts: the facts $(w1, c2) :$ $care$, $(w2, c1) : care$, and $(w2, c2) : care$ can be inferred using ontology reasoning. Our own approach will allow this kind of reasoning to be brought to GRE. Continuing to focus on the materialised KB2, we note another limitation of existing works: if only existential quantifiers are used then some objects are unidentifiable (i.e., it is not possible to distinguish them uniquely). These objects would become identifiable if other quantifiers and inverse relations were allowed. For example,

The cat who is cared by *at least 2* women $= Cat \sqcap \geq 2feed^-.Woman =$ $\{c1\}$,
The woman feeding *only* those fed by *at least 2* women $= Woman \sqcap \forall feed. \geq$ $2.feed^-.Woman = \{w1\}$,
The woman who feeds *all* the dogs $= \{w2\}$.

It thus raises the question: which quantifiers will be appreciated and how to use DL to realise them in GRE?

3   Beyond Existential Descriptions

In this section, we show how more expressive DLs can make objects referable that were previously unreferable. Far from being a minor modification of existing works, this will amount to a substantial reformulation which will allow the DL-based approach to move beyond other GRE algorithms in its expressive power and representational efficiency.

3.1 Expressing Generalized Quantifiers in OWL2

Because the proposal in [1] uses only existential quantification, it fails to identify any individual in Fig.2. Before attempting to fill this gap, we briefly pause to ask what level of expressivity might be achievable. In doing so, we shall make use of a conceptual apparatus developed in the formal study of natural language. The most general format for REs that involve a relation R is, informally, `the N1 who R Q N2's`, where `N1` and `N2` denote sets and `Q` is a quantifier. (Thus for example `the women who feed SOME dogs`.) An expression of this form is a uniquely identifying expression if it corresponds to exactly one element in the domain. Using a slightly more formal set-theoretic notation, this means that the following set has a cardinality of 1:

$$\{y \in N1 : Qx \in N2 \mid Ryx\}$$

where Q is a *generalized quantifier* (GQ [10]). For example, if Q is the existential quantifier, while $N1$ denotes the set of women, N2 the set of dogs, and R the relation of feeding, then this says that the number of women who feed SOME dog is just one. If Q is the quantifier *Exactly three*, however, then it says that the number of women who feed exactly THREE dogs is just one. It will be convenient to write the formula above in the standard GQ format where quantifiers are seen as relations between sets of domain objects A, B. For example, using the universal quantifier as an example, instead of writing $\forall x \in A \mid x \in B$, we write $\forall(AB)$. Thus, the formula above is written

$$\{y \in N1 : Q(N2\{z : Ryz\})\}.$$

Instantiating this as before, we get $\{y \in Woman : \exists(Dog\{z : Feed\ yz\})\}$, or "women who feed a dog", where Q is $\exists$, $A = Dog$ and $B = \{z : Feed\ yz\}$ for some y.

Mathematically characterizing the class of *all* quantifiers that can be expressed in referring expressions is a complex research programme to which we do not intend to contribute directly, partly because this class includes quantifiers that are computationally problematic; for example, quantifiers such as *most* (i.e., more than 50%) and *many* (which is vague) are not first-order expressible, as is well known.

To make transparent which quantifiers are expressible in the logic that we are using, let us think of quantifiers in terms of simple quantitative constraints on the sizes of the sets $A \cap B$, $A - B$, and $B - A$, as is often done in GQ theory, asking what types of constraints can be expressed in referring expressions based on $\mathcal{SROIQ}$. The findings are illustrated in Tab.1. The table shows that OWL2 can express any of the following, plus disjunctions and conjunctions of anything it can express.

Let us call the class of quantifiers defined by the table NatGQ. To see how large and general NatGQ is, a few examples will be useful. When $n = 1$, for example, type 1 becomes $\exists R.N2$, i.e. the *existential* quantifier. When $n = 0$ type 7 becomes $\forall R.N2$, i.e. the *only* quantifier. When $n = 0$ type 6 becomes $\forall \neg R.\neg N2$, i.e. the *all* quantifier. In types 2, 4, 6 and 8, negation of relation is used in a concept expression. This is not directly supported in $\mathcal{SROIQ}$ but, as we indicated earlier, given a closed KB Σ, when relation R is closed, $\neg R$ is valid in concepts.

Together, this allows the expression of a description such as "women who feed at least one but at most 7 dogs", by conjoining a quantifier of type 1 (with $n = 1$) with one of type 5 (with $n = 7$). It even allows expression of "women who do not feed

Table 1. Expressing GQ in DL

	QAB	DL
1	$\geq nN2\{z : Ryz\}$	$y :\geq nR.N2$
2	$\geq nN2\neg\{z : Ryz\}$	$y :\geq n\neg R.N2$
3	$\geq n\neg N2\{z : Ryz\}$	$y :\geq nR.\neg N2$
4	$\geq n\neg N2\neg\{z : Ryz\}$	$y :\geq n\neg R.\neg N2$
5	$\leq nN2\{z : Ryz\}$	$y :\leq nR.N2$
6	$\leq nN2\neg\{z : Ryz\}$	$y :\leq n\neg R.N2$
7	$\leq n\neg N2\{z : Ryz\}$	$y :\leq nR.\neg N2$
8	$\leq n\neg N2\neg\{z : Ryz\}$	$y :\leq n\neg R.\neg N2$

all dogs and who feed at least one non-dog", which can be expressed as $Woman \sqcap \neg\forall\neg Feed.\neg Dog \sqcap \exists Feed.\neg Dog$. In addition to Tab.1, $\mathcal{SROIQ}$ can even represent reflexive relation such as "the dog who loves itself" by $Dog \sqcap \exists love.Self$, which used to be regarded infeasible [6].

Comparing the quantifiers that become expressible through OWL2's apparatus with classes of quantifiers studied in the theory of GQ, it is clear that OWL2 is highly expressive: it does not only include quantifiers expressible in Van Benthem's binary tree of numbers [13] – which is often regarded as sufficient – but much else besides. Wider classes of referring expressions can certainly be conceived – for example by moving into intensionally or higher-order logic – but these are not likely to have overwhelming practical utility in todays's NLG applications.

4   Generating $\mathcal{SROIQ}$-enabled REs

In this section, we present an algorithms that is able to compute the descriptions we presented in sect.3. A GRE algorithm should have the following behaviour: if an entity is distinguishable from all the others, the algorithm should find a unique description; otherwise, the algorithm should say there exists no unique description. There are two major tasks in a GRE program:

1. Finding possible descriptions: Generating syntactically valid descriptions.
2. Validating a description: Checking whether a description can be satisfied by a particular object.

These two tasks can be done simultaneously for all the domain elements, or for a target referent. In the former case, candidate descriptions are generated and then tested for each domain element. If a description holds for only one element, then it is the RE for that element. This generate & test cycle is repeated until all the REs can be found. In the later case, candidate descriptions are generated and tested for a particular target referent until a valid RE is found or such a target referent can not be distinguished from some other element. In this paper, we follow the strategy of Areces et.al work to generate REs in a simultaneous way.

Since we consider more constructs than any previous treatment of relational descriptions, the combination of them can result in an enormously large search space. To measure the complexity of these descriptions, we define their depth:

Definition 1. *(Depth) Given a description d, its depth $|d|$ is calculated as follows:*

1. $|d| = 1$ *for* $d := \top | \bot | A | \neg A$.
2. $|d \sqcap d'| = |d \sqcup d'| = max(|d|, |d'|) + 1$.
3. $|\exists r.d| = |\forall r.d| = |\leq nr.d| = |\geq nr.d| = |= nr.d| = |d| + 1$.

Syntactically different descriptions can have same semantics, e.g. $\neg \forall R.A \equiv \exists R.\neg A$. We leave aside the question which syntactic variant should be used and focus on generating one form, assuming all the concepts are in their unique *negation normal form* (NNF). A NNF has $\neg$ in front of only atomic concepts (include $\top$ and $\bot$) or nominals. The NNF of $\neg C$ is denoted by $\sim C$.

To ensure we can generate proper descriptions w.r.t. particular requirements, we present the following abstract algorithm A-1. Given an ontology Σ, we initialise the algorithm with the following sets:

1. The concept name set CN is the minimal set satisfying:
   - $\top \in CN$;
   - if A is an atomic concept in Σ, then $A \in CN$;
   - if R is an atomic role in Σ, then $\exists r.Self \in CN$;
   - if $A \in CN$, then $\sim A \in CN$;
2. The relation name set RN is the minimal set satisfying:
   - if R is an atomic relation in Σ, then $R \in RN$;
   - if $R \in RN$, then $\sim R \in RN$;
   - if $R \in RN$, then $R^- \in RN$;
3. The number set $N = \{1, 2, \ldots, n\}$ where n is the number of individuals in Σ.
4. The construct set S contains all the constructs that supported by a particular language. For $\mathcal{SROIQ}$, $S = \{\neg, \sqcap, \sqcup, \exists, \forall, \leq, \geq, =\}$. Usage of names is disallowed (cf sect.2).

Obviously, $\sim(\sim A) = A$, $\sim(\sim R) = R$, $(R^-)^- = R$, and $(\sim R)^- = \sim R^-$. Then the algorithm takes an ontology Σ as its input and output a queue D of descriptions.

Algorithm A-1: Construct-description(Σ, CN, RN, N, S)
INPUT: Σ, CN, RN, N, S
OUTPUT: Description Queue D

```
 1: D := CN
 2: for d = fetch(D) do
 3:    for each s ∈ S do
 4:       if s = ⊓ or s = ⊔ then
 5:          for each d' ∈ D do
 6:             D := Add(D, d ⊓ d'(d ⊔ d'))
 7:       if s = ∃ or s = ∀ then
 8:          for each r ∈ RN do
 9:             D := Add(D, ∃r.d(∀r.d))
10:       if s =≤ or s =≥ or s is = then
11:          for each r ∈ RN, each k ∈ N do
12:             D := Add(D, ≤ kr.d(≥ kr.d, = kr.d))
```

13: **return** D

Algorithm A-2:Add(D, e)
INPUT: D, e
OUTPUT: (Extended)Description Queue D
 1: **for** $d \in D$ **do**
 2:     **if** $|d| < |e|$ and $d \sqsubseteq_\Sigma e$ **then**
 3:         **return** D
 4:     **else if** $|d| = |e|$ and $d \sqsubset_\Sigma e$ **then**
 5:         **return** D
 6: **if** $|[[e]]|^\Sigma > 0$ **then**
 7:     $D := D \cup \{e\}$
 8: **return** D

In Step 1, D is initialised by CN. From Step 2, we recursively process elements of D one by one. We use $fetch(D)$ to retrieve the first unprocessed element of D and new elements are added to the end of D. Thus D is a first-come-first-server queue (note that processed elements are not removed from D). For each element d of D, Step 3 to 12 extend it with a construct s:

1. If s is $\sqcap$ or $\sqcup$, in Step 5 and 6, we extend d with all the elements of D and add new descriptions to D.
2. If s is $\exists$ or $\forall$, in Step 8 and 9, we extend d with all relations of RN and add new descriptions to D. In Areces et el.'s work, $\forall$ is also available when using $\neg$ and $\exists$ together, however due to their algorithm they can never generates descriptions like $\forall r.A$.
3. If s is $\leq, \geq$ or $=$, in Step 11 and 12, we extend d with all relations of RN and all numbers of N, and add new descriptions to D.
   In this step, $= kr.d \equiv \geq kr.d \sqcap \leq kr.d$, which means $=$ construct can be equivalently substituted by the combination of $\leq, \geq$ and $\sqcap$ constructs.
   Therefore, it is an modelling choice to use either $\leq, \geq$, or only $=$, or all of them. In this algorithm we present all of them from a syntactic point of view.

Because we compute only the NNF and we disallow the usage of individual identifiers, negation $\neg$ appears only in front of atomic concept names, which have all been included in CN. Thus in extension, we do not consider $s = \neg$. The ordering of choosing constructs, relations, integers and conjuncts/disjuncts is not the topic of this paper.

Obviously, at any time, D, RN, N, S are all finite, thus Step 3 to 12 terminates for a particular $d \in D$. Because Step 3 to 12 generates descriptions with incremental depth, for a particular n, there are finite $d \in D$ such that $|d| = n$. Thus, the termination of Algorithm A-1 depends on the increment of D. This is controlled by the Add procedure, which determines whether a new generated description is added into D or not.

The mechanism of Add depends on the requirements of the application. In this paper we control the addition by following a simple heuristic: more complex descriptions should have smaller extension.

In Algorithm A-2, Step 2 ensures that, when adding a new description e into D, its extension should be smaller than any existing description $d \in D$ with a smaller depth than e. Step 4 ensures that when adding a new description e into D, its extension should be no larger than any existing description $d \in D$ with same depth as e. Step 6 to 7 adds a new description when its extension is non-empty. The subsumption checking in Step 2 and 4, the instance retrieval in Step 6, must be realised by DL reasoning.

A-2 guarantees that when the complexity of descriptions increases, their extensions are getting smaller and smaller (but still non-empty). Because descriptions of a particular depth is always finite, when the domain is finite, Algorithm A-1 always terminates.

It can be shown that, our approach is an extension of the algorithm presented in Areces et al.'s work. The example in Fig.2 shows that some referring expressions generated by our algorithm cannot be generated by our predecessors; more importantly even, some objects that are not referable for them are referable for us.

It is worth stressing here that our algorithm focusses on finding uniquely referring expressions, leaving aside which of all the possible ways in which an object can be referred to is "best". For this reason, *empirical* validation of our algorithm – a very sizable enterprise in itself, which should probably be based on descriptions elicited by human speakers – is not yet in order.

Discussion Now we revisit the basic assumptions to see what can be achieved without them.

1. Using names in REs, e.g. "the husband of Marie Curie". Here "Marie Curie" servies as both the identifier of the individual and the name of its interpretation. In this case, we extend our Algorithm A-1 by including $\{Maria\_Curie\}$ in CN.
2. An open world: when the domain is not restricted to be closed, traditional GRE approaches may fail because they have always been assuming a single model with complete knowledge. In this case, interesting REs can still be found by our approach. For example, if someone is known to be the only Chinese or Japanese, we can refer to him/her as $Chinese \sqcup Japanese$ although the exact nationality is unknown.
3. Individual with multiple names. DL imposes the UNA by explicit asserting the inequality of each two individuals. Without UNA, reasoning can still infer some results, e.g. $\{Woman \sqcap Man \sqsubseteq \bot, David : Man, May : Woman\} \models David \neq May$. Thus we can refer to David as "the man" if the domain is closed.

5 Conclusion: widening the remit of GRE

This paper has shown some of the benefits that arise when the power of KR is brought to bear on an important problem in NLG, namely the generation of referring expressions (GRE). We have done this by using DL as a representation and reasoning formalism, extending previous work in GRE in two ways. In order to explain what class of referring expressions is covered by our proposal, we have related our algorithm to the theory of Generalized Quantifiers, which allowed us to formally characterize the set of quantifiers that are used by our algorithm, thereby making exact how much expressive power we have gained. Secondly, we have demonstrated the benefits of *implicit* knowledge

through inferences that exploit TBox-information, thereby allowing facts to be represented more efficiently and elegantly, and allowing GRE to tap into kinds of generic (as opposed to atomic) knowledge that it had so far left aside, except for hints in [6] and in [3].

Current work on reference is overwhelmingly characterized by an emphasis on empirical accuracy, often focussing on very simple referring expressions, which are constituted by conjunctions of 1-place relations (as in "the grey poodle", "the Swedish woman"), and asking which of these conjunctions are most likely to be used by human speakers (or sometimes, which of these would be most useful to a human hearer or reader). The present work stresses entire different concerns: we have focussed on questions of expressive power, focussing on relatively complex descriptions, asking what referring expressions are possible when relations (such as "love" or "feed") between domain objects are used. We believe that, at the present stage of work in GRE, it is of crucial importance to gain insight into questions of this kind, since this will tell us what types of reference are possible in principle. Once these questions are answered, we shall explore how the newly gained expressive power can be put to practical use.

References

 1. Carlos Areces, Alexander Koller, and Kristina Striegnitz. Referring expressions as formulas of description logic. In *Proceedings of the 5th International Natural Language Generation Conference*, Salt Fork, Ohio, 2008.
 2. Franz Baader, Diego Calvanese, Deborah L. McGuinness, Daniele Nardi, and Peter F. Patel-Schneider, editors. *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press, 2003.
 3. Madalina Croitoru and Kees van Deemter. A conceptual graph approach to the generation of referring expressions. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence*, 2007.
 4. Robert Dale. Cooking up referring expressions. In *Proceedings of the 27th annual meeting on Association for Computational Linguistics*, pages 68–75, 1989.
 5. Robert Dale and Ehud Reiter. Computational interpretations of the gricean maxims in the generation of referring expressions. *CoRR*, cmp-lg/9504020, 1995.
 6. Claire Gardent and Kristina Striegnitz. Generating bridging definite descriptions. *Computing Meaning*, 3:369–396, 2007.
 7. Ian Horrocks, Oliver Kutz, and Ulrike Sattler. The Even More Irresistible SROIQ. In *KR 2006*, 2006.
 8. Emiel Krahmer and Mari?t Theune. Efficient context-sensitive generation of descriptions in context. *Information Sharing: Givenness and Newness in Language*, pages 223–264, 2002.
 9. Emiel Krahmer, Sebastiaan van Erk, and Andr Verleg. Graph-based generation of referring expressions. *Computational Linguistics*, 29(1):53–72, 2003.
10. A Mostowski. On a generalization of quantifiers. *Fund. Math.*, 44:235–273, 1957.
11. Boris Motik, Bernardo Cuenca Grau, Ian Horrocks, Zhe Wu, Achille Fokoue, and Carsten Lutz. Owl 2 web ontology language: Profiles. W3c working draft, W3C, October 2008.
12. Inanç Seylan, Enrico Franconi, and Jos de Bruijn. Effective query rewriting with ontologies over dboxes. In *IJCAI 2009*, 2009.
13. Johan van Benthem. Essays in Logical Semantics. Reidel, 1986.
14. Kees van Deemter and Emiel Krahmer. Graphs and booleans: On the generation of referring expressions. *Computing Meaning*, 3:397–422, 2007.

A Multi-Context System Computing Modalities

Tarek R. Besold[1] and Bernhard Schiemann[2]

[1] University of Erlangen-Nuremberg,
Chair of Computer Science 8: Artificial Intelligence,
D-91058 Erlangen, Germany
`sntabeso@i8.informatik.uni-erlangen.de`
[2] `Bernhard.Schiemann@googlemail.com`

Abstract. The system description presents the conception and a prototypical implementation of a multi-context system, used for computing and implementing temporal modalities within given data without the use of modal operators. Instead, an external constraint based rule system is used for computing the corresponding temporal relations, making use of the way a multi-context system works for transporting the needed information between contexts and knowledge bases.

1   Point of Departure and Problem Statement

>   *Introducing new modalities should involve no more fuss than introducing a new predicate.* [1]

The above quotation, taken from John McCarthy's paper *"Modality, Si! Modal Logic, No!"*, addresses a point quite well-known to description logicians and people working with systems for knowledge representation, based on DL. Real world applications often demand for the use of modalities (e.g. to express statements about time), or using McCarthy again:

>   *In particular, human-level AI requires that programs be able to introduce modalities when this is appropriate, e.g. have function taking modalities as values.* [1]

Nevertheless, allowing for the unrestricted use of modal operators within a description logic framework (given the not very probable case that this may be possible), major problems concerning decidability, completeness and computability arise. Even if only some modal operators shall be incorporated into a standard DL framework (as e.g. $SHOIN(D)$, corresponding to the widely used OWL DL V1.0 language, or $SROIQ(D)$ for OWL DL V2.0), a large part of the "pleasant" properties of the original formalism gets lost.

Thus, we decided to build a working and applicable implementation of modalities (in the following we will use time as an example modality) without the use of modal operators, thereby following the basic idea lined out in McCarthy's paper. We managed to do so by using concepts from the field of multi-context systems (MCS), mainly based on the results presented in [2] and [3].

2   Attempt at a Solution

The idea underlying our solution to the stated problem is the following: Knowledge bases in DL normally contain definitions of concepts, roles and instances, stated in present tense. For formal reconstruction purposes within many different domains, e.g. cultural heritage, descriptions of instances which are situated in the past would be needed. But modal logical extensions of DL often carry – besides the problem of (un)decidability – new operators (e.g. "Until") with them. In many cases, a description not relying on the use of modal operators would be sufficient. Moreover, the modellers wouldn't have to learn how to use the new operators properly.

In order to enhance the widely used DL $SHOIN(D)$ (OWL DL V1.0) with these kind of descriptions, in a reference ontology for cultural heritage (ECRM) Allen's time relations ([4]) are added as descriptions (as already demanded by the CIDOC-CRM[3]). Having done so, the descriptions are contained, but a mechanism performing calculations on this modalities is still missing. This mechanism may be introduced by means of an external module, a constraint based rule system (CBRS).

3   Mode of Operation

Given a DL knowledge base, the functionality of our system may basically be sketched as follows:

1. Extend the concrete knowledge base: Collect time describing features within the DL descriptions and state them explicitly.
2. Transfer the collection of time describing features to a constraint based rule system (CBRS), already containing abstract rules which model Allen's time relations.
3. Generate a model by means of the CBRS and afterwards extract the newly calculated, concrete time relations from the model.
4. Inject the extracted concrete time relations into the DL knowledge base.

Thus the result of the external calculations in the CBRS may be used for the proper reasoning within the DL knowledge base. Moreover, this allows – besides of the possibility of the description of the instances in the past – for an automatic sorting of the time descriptions on a timeline given by Allen's relations.

Furthermore, the approach is very flexible, as the means for calculations and computation may be extended: The descriptions in the ontology may be expanded with more relations, given the corresponding bridge rules to the CBRS. Thereby, the limitations of the description are not caused by the restrictions

---

[3] CIDOC, the Committee on Documentation of the International Council of Museums, is a working group focusing on the documentation requirements and standards of museums, archives, and similar organizations. CIDOC has defined a *Conceptual Reference Model*, the CIDOC-CRM, which provides a formal and extensible ontology for cultural heritage information.

from the $SHOIN(D)$ language, but are given according to the configurable external logic. As this logic may be configured and extended stepwise, exactly those modalities needed for a concrete formal reconstruction may be implemented. Hence, in place of making necessary an entire extension of DL, constructed around one modal operator or another, our approach allows for the incremental construction of a TimeDL.

This extension may be transferred to other formal reconstructions wit relative ease, and can also be applied to other DLs. Only the quite simple DL S is needed when describing Allen's temporal relations. Thus re-usability within other DLs, build upon S (as e.g. $SROIQ$, corresponding to OWL DL V2.0) may easily be reached.

Further independence between the extension and the underlying knowledge representation formalism has been obtained.

4   (Very) Short Introduction to Multi-Context Systems

Here we restate the key definitions given in [2]. First, the concept of *logic*[4] is defined in terms of input-output conditions.

Definition 1. *A logic $L = (\mathbf{KB}_L, \mathbf{BS}_L, \mathbf{ACC}_L)$ is composed of the following components:*

1. $\mathbf{KB}_L$ *is the set of well-formed knowledge bases of L. We assume each element of $\mathbf{KB}_L$ is a set.*
2. $\mathbf{BS}_L$ *is the set of possible belief sets,*
3. $\mathbf{ACC}_L : \mathbf{KB}_L \mapsto 2^{\mathbf{BS}_L}$ *is a function describing the "semantics" of the logic by assigning to each element of $\mathbf{KB}_L$ a set of acceptable sets of beliefs.*

Given several logics, bridge rules are used to translate between the logics.

Definition 2. *Let $L = \{L_1, \ldots, L_n\}$ be a set of logics. An L_k-bridge rule over $L, 1 \leq k \leq n$, containing m conditions, is of the form*

$$s \leftarrow (r_1 : p_1), \ldots, (r_j : p_j), \mathbf{not}(r_{j+1} : p_{j+1}), \ldots, \mathbf{not}(r_m : p_m) \qquad (1)$$

where $j \leq m$, $1 \leq r_k \leq n$, p_k is an element of some belief set of L_{r_k} and s is a syntactically valid element of a knowledge base from $\mathbf{KB}_k$,[5] and for each $kb \in \mathbf{KB}_k : kb \cup \{s\} \in \mathbf{KB}_k$.

A configuration of logics and bridge rules comprises a multi-context system.

---

[4] As in the following, no information containing satisfiability or inference rules within the corresponding logics will be given, also the denomination "pre-logic" would be justifiable. For the sake of consistency we will keep the original naming calling it a "logic".

[5] In contrast to [2] where a similar constraint concerning the nature of s is imposed only implicitly.

Definition 3. *A multi-context system* $M = (C_1, \ldots, C_n)$ *consists of a collection of contexts* $C_i = (L_i, kb_i, br_i)$*, where* $L_i = (\mathbf{KB}_i, \mathbf{BS}_i, \mathbf{ACC}_i)$ *is a logic,* kb_i *a knowledge base (an element of* $\mathbf{KB}_i$ *), and* br_i *is a set of* L_i*-bridge rules over* $\{L_1, \ldots, L_n\}$.

A belief state is the combination of the belief sets of all contexts of the MCS.

Definition 4. *Let* $M = (C_1, \ldots, C_n)$ *be a MCS. A belief state is a sequence* $S = (S_1, \ldots, S_n)$ *such that each* S_i *is an element of* $\mathbf{BS}_i$.

We say a bridge rule r of form (1) is applicable in a belief state $S = (S_1, \ldots, S_n)$ iff for $1 \leq i \leq j : p_i \in S_{r_i}$ and for $j + 1 \leq k \leq m : p_k \notin S_{r_k}$. A belief state is an equilibrium if the consequences of all applicable bridge rules are given, hence each context has an acceptable belief set given the belief sets of the other contexts.

Definition 5. *A belief state* $S = (S_1, \ldots, S_n)$ *of* M *is an equilibrium iff, for* $1 \leq i \leq n$, *the following condition holds:*

$$S_i \in \mathbf{ACC}_i(kb_i \cup \{head(r)|r \in br_i \ applicable \ in \ S\}).^6$$

Now, we moreover introduce "bridge rule models" as a possibility to expatiate on the actual reasoning by signalizing explicitly which bridge rules within an MCS are active with respect to a given belief state (for further details vide [5]).

Definition 6. *Let* $\mathbf{Br}$ *be a set of* n *bridge rules of an MCS. A bridge rule model is an assignment* $\mathbf{Br} \mapsto \{0, 1\}^n$ *that represents for each bridge rule in* $\mathbf{Br}$ *whether it is active (i.e. the bridge rule is applied, and the element in its head is added to the respective target knowledge base) or not.*

Proposition 1. *For each equilibrium there is exactly one bridge rule model.*

Proof. Given an MCS and an equilibrium belief state. Then for every bridge rule we can decide whether or not its prerequisites are satisfied. Hence, for each position in the bridge rule model we can decide whether the value is 0 or 1.

5   MCS and Museum Data Completion and Consistency

The now presented MCS, implementing the algorithm for finding the equilibria of an MCS from [3], has been developed in cooperation with part of the team of the WissKI research project[7].

---

[6] Please note that, if r is a bridge rule of the form $a \leftarrow \ldots$, then $head(r) = a$

[7] Research project "WissKI - Wissenschaftliche KommunikationsInfrastruktur", funding provided by the German Research Council (DFG). The project is being carried out at the Chair for Computer Science 8: Artificial Intelligence of the University of Erlangen-Nuremberg, together with collaborators from the GNM Nuremberg and the ZFMK Bonn. For a presentation of the project vide [6] and http://www.wiss-ki.eu.

The WissKI project's main purpose is to extend the the current conception of wiki-style media to a medium of science communication and scientific interaction. Amongst its main goals are therefore to enable semantic content analysis by means of CIDOC-CRM (ISO 21127) (e.g. vide [7] and [8]). One of the tools under development in the project is a semantics enhanced content management system (CMS+S), in which we integrated a special form of an MCS in order to obtain additional temporal reasoning functionality for the semantic analysis.[8]

We accomplished this by using a form of inference fusion: Starting from a DL based initial representation of data, we transfer parts of it (containing temporal information) to another formalism. Then, we use smodels[9] as a reasoner in order to obtain additional results (in concrete for – if possible – establishing an ordering within the expressed time statements), and finally combine the results of the reasoner and the original data into another enhanced DL representation.

The scenario where the MCS comes into play within the CMS+S may be sketched as follows: We are given a CIDOC-CRM conform file (OWL/DL format, vide [10]) containing information from the context of master pieces of goldsmith art (vide [11]), which also contains – in the form of free text – temporal information about persons and events. An already implemented software tool preprocesses the free text for the MCS: It parses the free text parts, identifies, and then returns place names, person names and time specifications (again vide [11]). Our MCS takes the time specifications (represented as time intervals using Turtle file format[10]) as inputs, creates bridge rules for the transport of all the given statements to a representation suitable for reasoning with smodels,[11] and afterwards uses smodels and given constraints on temporal reasoning in order to create a linear ordering of the given time statements (the ordering of the intervals is established according to [4]). If a linear ordering can be established (if this cannot be done the set of time statements is inconsistent), after ordering the time statements, again bridge rules are created to transport the ordering information to a knowledge base in OWL/DL format, using the CIDOC-CRM

---

[8] The implementation of the CIDOC-CRM the WissKI project is working on is based on a logical formalism equivalent to a $SHOIN(D)$ DL. Therefore, temporal reasoning is not supported by the used logical formalism itself.

[9] An implementation of the stable model semantics for logic programs, vide e.g. [9].

[10] For details concerning the Turtle file format vide e. g. [12]. A typical Turtle triplet encountered in the implementation example contains as first string an identifier, the second string states a property of the corresponding object, and the third string explicitly states a value, related to this property, in XSD time data format: `http://wiss-ki.eu/ns/tmp/gen_e61_2_N65566` `http://www8.informatik.uni-erlangen.de/IMMD8/Services/cidoc-crm/erlangen` `-crm_090330_5_0_1_TQ.owl#has_primitiveTime` ''2009-06-22''

[11] By this not only extracting the information from the initial context and adding it to a context suitable for reasoning with smodels, but also performing a translation between the OWL/DL language and a suitable representation for lparse (vide [13]) input at the same time, as the segmentation of bridge rules in a head part and a body part – without a constraint restricting the formal languages used in both parts to be the same – allows for this kind of use.

time relations[12] to reproduce the ordering found. Afterwards, the results may be merged with the original CIDOC-CRM conform file and another reasoner may be used (e.g. RACER[13] on the given data) in two ways: to check the consistency of the enhanced knowledge base, or for data completion purposes taking into account the newly obtained time information.

The MCS just described is of special form, we call it a "linear multi-context system", as the information is passed through it in a linear way. The basic principle may be sketched as follows: The parser/tagger software tool creates a Turtle format representation of the time statements, this is kb_1 in context C_1. Then, as all information from kb_1 has to be transported to the smodels context's knowledge base $kb_2 \in C_2$, the bridge rules from kb_1 to kb_2 may automatically be created given kb_1. The reasoning part is done in kb_2 when the import has been completed. Afterwards, all obtained information concerning time relations has to be transported to the OWL/DL representation in context C_3 (containing kb_3), the bridge rules may automatically be created, completely covering all corresponding elements of kb_2.[14] When the transport to kb_3 has been completed, the work of the MCS is mainly done, the fusion of kb_3 with the original OWL/DL base in the narrower sense is not part of the MCS. Now the mode of operation of the linear MCS shall be discussed in more detail. Given the Turtle triplets – containing information concerning person names, place names and time statements – the parser/tagger returns as output of his free text analysis, the MCS has to identify and extract the time information by simple structural filtering (due to the chosen output format of the parser/tagger, a strictly syntactical discrimination of the different types of statements is possible, e.g. via the use of regular expressions). The results of this process are written to kb_1, the knowledge base of C_1. Now, the bridge rules for the transport of the time interval information from kb_1 to kb_2 (knowledge base of C_2) have to be created and added to br_2. The bridge rules only have one condition (the element of kb_1 which shall be transported): "$f(a) \leftarrow (1 : a)$", where $a \in kb_1$, $f(a) \in kb_2$ and $f(\cdot)$ a function returning as result a "translation" of a to the smodels formalism. Patterns of the bridge rules requiring generation are given in form of generic bridge rules within

---

[12] Allen's temporal relations between time intervals are implemented by CIDOC-CRM properties P114 to P120, vide [7].

[13] Vide e.g. [14].

[14] One might be tempted to think about generating all the bridge rules already initially, before starting the MCS procedure, and not dynamically during processing. Unfortunately, this would raise major problems: As no information concerning the ordering within the time statements (computed within kb_2) is ab initio available, in br_3 we would have to create a bridge rule for every possible relation between two time intervals for every tuple of time intervals that may have been transported from kb_1 to kb_2. This would yield thirteen bridge rules for every tuple, out of which only one may in fact be activated in the final equilibrium bridge rule model. Therefore, the number of bridge rule models which (according to [3]) have to be tested for representing the equilibrium of the MCS would be multiplied, substantially worsening the performance of the entire MCS.
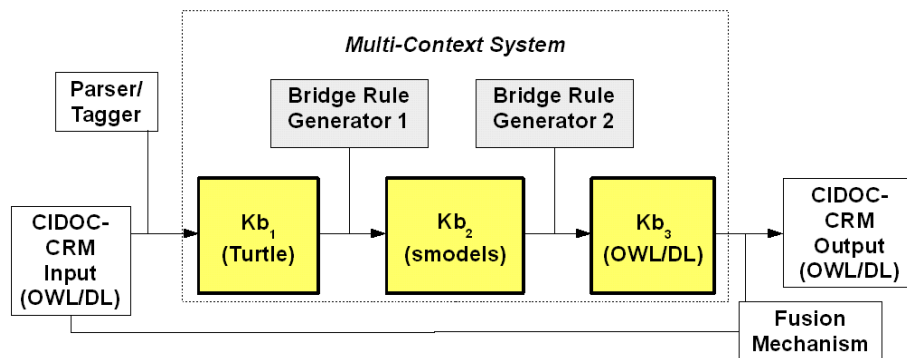
Fig. 1. *A sketch of the design of the linear MCS.*

br_2.[15] For every interval from kb_1, starting time point and ending time point are transported. Having built all bridge rules according to the generic prototypes, we may apply them all at once, as a complete transfer of knowledge from kb_1 to kb_2 shall be performed (the bridge rules have been constructed accordingly). Now, kb_2 is populated with all the temporal information concerning time intervals obtained from the free texts. But kb_2 contains constraints depicting the relations between time intervals that Allen proposed. A reasoner call over kb_2 is performed. The result is again used for bridge rule generation: For each statement indicating the relation between two time intervals, a bridge rule has to be created, transferring this statement to the third context C_3, adding the corresponding CIDOC-CRM statementin OWL/DL format to the knowledge base kb_3. Again, generic bridge rules from br_3 prototypically indicate which bridge rules have to be created.[16] When all of the mentioned bridge rules have been added to br_3, for the same reason as above they may again all be applied at a time. Finally, kb_3 – now containing all the statements concerning the ordering relations amongst the time intervals – is merged with the initial CIDOC-CRM knowledge base (which contains the free texts the parser/tagger originally used), or a single OWL file containing the interval information may be produced.

6 Used Technical Infrastructure

For the implementation of the just sketched MCS we made use of a proof-of-concept MCS Software Framework developed as part of one of the authors'

---

[15] E.g. $day(X, Y) \leftarrow (1 : day(X, Y))$, indicating that for every element from kb_1, unifyable with $day(X, Y)$ (X an identifier for the temporal entity, Y a numeric value indicating the day's date within the month) a bridge rule shall be generated mapping it to its corresponding element of kb_2.

[16] E.g. "$P114.is\_equal\_in\_time\_to(X, Y) \leftarrow 2 : is\_equal\_in\_time\_to(X, Y)$" ($X$, Y variables for identifiers of time intervals), would create for every fitting element in kb_2, a bridge rule mapping it to its corresponding P114 CIDOC-CRM statement in kb_3.

master's thesis. A detailed overview of the concrete MCS implementation is given in [5]. Therefore, in the following we only want to give an overview of the main technical characteristics of the system used:

As programming language we used SCALA[17] (e. g. vide [15]), a general purpose programming language. The advantages of SCALA, apart from allowing for both object-oriented and functional programming, are its full interoperability with native JAVA code (JAVA may be directly called from SCALA and vice versa), the full byte code compatibility - making possible the full use of existing JAVA libraries or application code - and the possibility to run SCALA programs on the widespread JAVA VM.

Moreover, as previously mentioned, we made use of the lparse/smodels combination as implementation of the stable model semantics for logic programmes, which are both freely available to the scientific community. Both, the lparse front-end and smodels itself, are at some points called as external components by the MCS framework.

For the smodels reasoning, we set up a constraint based implementation of Allen's Interval Algebra, allowing smodels to compute an Allen-like ordering within the time intervals whenever possible (vide Figure 2).

The modality computing MCS has then been integrated into the already existing WissKI software system,[18] placing it in a line with the aforementioned parser/tagger used for performing the free text analysis and production of the MCS's input Turtle triplets. Having computed the linear ordering within the time intervals, the output of the MCS is being merged with the remaining data in OWL/DL format, serving data enrichment and completion purposes.

7   Why Use an MCS – Enhanced Scenarios

Up to now, critics of this approach may question why to use an MCS for this purpose, because as seen from some angles a more or less elaborately written script might provide almost the same functionality. But the scenario just shown is only the first step of evolution of this kind when using an MCS: We conceive by far more complex systems, not being linear, but containing at least one feedback loop from C_3 to C_1, e.g. testing the fusion of C_3 and the original OWL/DL base for consistency, and modifying the knowledge base $kb_1 \in C_1$ if any inconsistency is detected. Performing a run of the MCS excluding in a systematical manner elements from kb_1 from being promoted to kb_2, we might possibly identify the causes for the inconsistency.

---

[17] We compiled the MCS framework implementation with "`Scala version 2.7.5 final (Java Hotspot(TM) Client VM, Java 1.6.0_15)`".
[18] For more details again vide e.g. http://www.wiss-ki.eu.

$$\ldots$$

$\text{month\_leq}(X, Y) : -\text{month}(X, A), \text{month}(Y, B),\ A \leq B,\ X \neq Y.$

$\text{month\_leq}(X, Y) : -\text{not month}(X, \_),\ \text{month}(Y, B),\ \text{starting\_time\_primitive}(\_, X).$

$\text{month\_leq}(Y, X) : -\text{not month}(X, \_),\ \text{month}(Y, B),\ \text{ending\_time\_primitive}(\_, X).$

$\text{month\_eq}(X, Y) : -\text{month}(X, A),\ \text{month}(Y, B),\ A = B,\ X \neq Y.$

$$\ldots$$

$\text{before}(X, Y) : -\text{year\_leq}(X, Y),\ \text{not year\_eq}(X, Y).$

$\text{before}(X, Y) : -\text{year\_eq}(X, Y),\ \text{month\_leq}(X, Y),\ \text{not month\_eq}(X, Y).$

$\text{before}(X, Y) : -\text{year\_eq}(X, Y),\ \text{month\_eq}(X, Y),\ \text{day\_leq}(X, Y),$
$\qquad \text{not day\_eq}(X, Y).$

$$\ldots$$

$\text{before}(X, Y) : -\text{year\_eq}(X, Y),\ \text{month\_eq}(X, Y),\ \text{day\_eq}(X, Y),$
$\qquad \text{hour\_eq}(X, Y),\ \text{minute\_eq}(X, Y),\ \text{second\_eq}(X, Y),$
$\qquad \text{millisecond\_leq}(X, Y),\ \text{not millisecond\_eq}(X, Y).$

$\text{time\_primitive}(Y) : -\text{starting\_time\_primitive}(X, Y),\ \text{year}(Y, Z).$

$\text{time\_primitive}(Y) : -\text{ending\_time\_primitive}(X, Y),\ \text{year}(Y, Z).$

$\text{equal}(X, Y) : -\text{not before}(X, Y),\ \text{not before}(Y, X),\ \text{time\_primitive}(X),$
$\qquad \text{time\_primitive}(Y),\ X \neq Y.$

$\text{inconsistent}(X) : -\text{starting\_time\_primitive}(X, A),\ \text{ending\_time\_primitive}(X, B),$
$\qquad \text{before}(B, A).$

$: -\text{inconsistent}(X).$

$\text{finishes}(X, Y) : -X \neq Y,\ \text{starting\_time\_primitive}(X, A),$
$\qquad \text{ending\_time\_primitive}(X, B),\ \text{starting\_time\_primitive}(Y, C),$
$\qquad \text{ending\_time\_primitive}(Y, D),\ \text{equal}(B, D),\ \text{before}(C, A),$
$\qquad \text{not inconsistent}(X),\ \text{not inconsistent}(Y).$

$\text{is\_finished\_by}(X, Y) : -X \neq Y,\ \text{starting\_time\_primitive}(X, A),$
$\qquad \text{ending\_time\_primitive}(X, B),\ \text{starting\_time\_primitive}(Y, C),$
$\qquad \text{ending\_time\_primitive}(Y, D),\ \text{equal}(B, D),\ \text{before}(A, C),$
$\qquad \text{not inconsistent}(X),\ \text{not inconsistent}(Y).$

$$\ldots$$

$\text{occurs\_during}(X, Y) : -X \neq Y,\ \text{starting\_time\_primitive}(X, A),$
$\qquad \text{ending\_time\_primitive}(X, B),\ \text{starting\_time\_primitive}(Y, C),$
$\qquad \text{ending\_time\_primitive}(Y, D),\ \text{before}(C, A),\ \text{before}(B, D),$
$\qquad \text{not inconsistent}(X),\ \text{not inconsistent}(Y).$

$$\ldots$$

Fig. 2. *Parts of a constraint base for computing Allen's time interval relations.*

Input:
$tagger\_output : List[String]$ (containing strings with data triplets in Turtle format),
$MCS = (C_1, C_2, C_3)$ ($kb_2 : List[String]$ containing the smodels constraint rules for temporal reasoning, $br_2 : List[String]$ and $br_3 : List[String]$ each containing generic bridge rules,
$kb_1 : List[String] = kb_3 : List[String] = br_1 : List[String] = \emptyset$).

Output:
$MCS\_output : List[String]$ (containing the relations between time intervals).

$br\_model : List[List[Int]] \leftarrow \{\}$
$kb\_buffer : List[String] \leftarrow \{\}$
$kb_1 \leftarrow extractTimeInformation(tagger\_output)$
for $generic\_br \in br_2$ **do**
    for $element \in kb_1$ **do**
        if $matchesPattern(element, generic\_br)$ **then**
            $br_2 \leftarrow br_2 \cup instantiateBR(element, generic\_br)$
            $br\_model(C_2) \leftarrow br\_model(C_2) \cup \{\{1\}\}$

    $removeFrom(generic\_br, br_2)$

$setAllValuesToZero(br\_model(C_1)), setAllValuesToZero(br\_model(C_3))$
$kb\_buffer \leftarrow extractKB(findEquilibria(MCS, br\_model), kb_2)$
for $generic\_br \in br_3$ **do**
    for $element \in kb\_buffer$ **do**
        if $matchesPattern(element, generic\_br)$ **then**
            $br_3 \leftarrow br_2 \cup instantiateBR(element, generic\_br)$
            $br\_model(C_3) \leftarrow br\_model(C_3) \cup \{\{1\}\}$

    $removeFrom(generic\_br, br_3)$

$setAllValuesToZero(br\_model(C_1)), setAllValuesToZero(br\_model(C_2))$
$kb\_buffer \leftarrow extractKB(findEquilibria(MCS, br\_model), kb_3)$
$MCS\_output \leftarrow addHeaderEtc(kb\_buffer)$
return MCS_output

Algorithm 1: *The linear MCS for the CMS+S.*

Example 1. Given an MCS $M = (C_1, C_2, C_3)$, where C_1 and C_3 are DL contexts, and C_2 is a temporal logic context.[19] Moreover, initially $kb_1 = \{david, goliath, abraham, lifetime(abraham, 150-200), lifetime(david, 175-225), lifetime(goliath, 205-250), is\_father\_of(abraham, david), is\_son\_of(david, goliath)\}$.[20]

Now, assuming that the "$is\_father\_of(X, Y)$" and the "$is\_son\_of(X, Y)$" relations have properly been modelled (i. e. also stating conditions on the relation between the lifetimes of father and son, e.g. that the lifetime of the son may not

---

[19] For the sake of readability and to avoid unnecessary complications, in the example we will not use the Turtle and OWL syntax, but a more intuitive and easily accessible notation.

[20] $is\_father\_of(X, Y)$ stating that X is the father of Y, and $is\_son\_of(X, Y)$ analogously stating that X is the son of Y.

begin before the lifetime of the father begins), performing a run of the MCS – analogously to the description of the linear MCS above – we would obtain an inconsistency in the belief set corresponding to C_3, as for the lifetimes of "*david*" and "*goliath*" – according to Allen's relations between time intervals – "$lifetime(david, 175 - 225)$ *overlaps* $lifetime(goliath, 205 - 250)$" would be stated. This contradicts the fact that "*david*" is declared a son of "*goliath*".

Excluding the "$is\_son\_of(david, goliath)$" statement from kb_1, we would obtain a consistent belief state, stating an equilibrium of the MCS. Thus, "$is\_son\_of(david, goliath)$" has been identified as possibly causing an inconsistency within the data and should be reviewed.[21]

A related application would be a series of MCS calls for data completion purposes, after each complete call using the newly obtained data for another run of the MCS, until no further augmentation of information may be obtained.

Another but far more sophisticated way of using this type of MCS would be an application in combination with the symbolic sub-symbolic integration proposed in [3]: A "modality + sub-symbolic MCS" could e.g. be used to combine person recognition systems (e.g. based on neural networks) at airports, train stations and street cameras with databases for train, flight and bus timetables etc., making possible tracing, tracking and verification of a person's movement pattern on a wide-area basis.

8   Future Prospects and Conclusion

To the best of our knowledge, the implemented functionality is quite innovative and enriches the CMS+S system with a very attractive feature: the possibility to also perform temporal reasoning. Normally, the underlying description logic by itself does not offer this possibility, but a special temporal extension to the DL must be used. With our approach, the original DL may stay untouched, and moreover the applied "temporal logic" must not be fixed, but may be extended or modified according to individual needs and thus becomes highly modularisable, as additional inference rules or entire logic formalisms may be added to the temporal logic context without modifying the description logic parts of the MCS.

As next step we see an examination of the extendability of the used concept to other modalities then time, e.g. place or possibility.

Moreover, the implementation of the enhanced functionality sketched in Sect. 7, offering functionality for the diagnosis of possibly implicit inconsistencies in the DL knowledge base, as well as for data completion purposes, is one of the main topics on our agenda.

---

[21] Also the lifetimes of "*david*" and "*goliath*" would be possible reasons for the inconsistency. Which proposal for the source of error to start with in the reviewing process has to be decided application specific, or even some kind of "hypothetical reasoning" handling alternate versions of the knowledge base may be performed (using the corresponding bridge rule models as identifiers and basis of the alternate possibilities of knowledge base evolution).

9   Acknowledgements

References

1. McCarthy, J.: Modality, Si! Modal Logic, No! Studia Logica **59**(1) (July 1997)
2. Brewka, G., Eiter, T.: Equilibria in Heterogeneous Nonmonotonic Multi-Context Systems. In: Proceedings of the National Conference on Artificial Intelligence. Volume 22., Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999 (2007) 385–390
3. Besold, T.R., Mandl, S.: Integrating Logical and Sub-Symbolic Contexts of Reasoning. In Filipa, J., Fred, A., Sharp, B., eds.: Proceedings of ICAART 2010 - Second International Conference on Agents and Artifcial Intelligence, INSTICC Press (2010) . For a full version of the paper vide also *http://www8.informatik.uni-erlangen.de/inf8/Publications/bridging_mcs_original.pdf*.
4. Allen, J.F.: Maintaining knowledge about temporal intervals. Commun. ACM **26**(11) (1983) 832–843
5. Besold, T.R.: Theory and Implementation of Multi-Context Systems Containing Logical and Sub-Symbolic Contexts of Reasoning. Master's thesis, Department of Mathematics & Department of Computer Science 8: Artificial Intelligence, FAU Erlangen-Nuremberg (2009) . The full thesis is available under http://www.opus.ub.uni-erlangen.de/opus/volltexte/2010/1587/.
6. Lampe, K.H., Krause, S., Hohmann, G., Schiemann, B.: Wissen vernetzt: Vom Wandel der Dokumentation in Museen der Natur- und Kulturgeschichte. KI - Künstliche Intelligenz **4/09, Special Issue on "Cultural Heritage and A.I."** (2009)
7. Crofts, N., Doerr, M., Gill, T., Stead, S., Stiff, M.: Definition of the CIDOC Conceptual Reference Model (Version 4.2.4). (March 2008)
8. Doerr, M.: The CIDOC Conceptual Reference Model: an ontological approach to semantic interoperability of metadata. AI Magazine **Vol. 24(3)** (2003) 75–92
9. Simons, P., Niemelá, I., Soininen, T.: Extending and implementing the stable model semantics. Artif. Intell. **138**(1-2) (2002) 181–234
10. Goerz, G., Oischinger, M., Schiemann, B.: An Implementation of the CIDOC Conceptual Reference Model (4.2.4) in OWL-DL. In: Proceedings of the 2008 Annual Conference of CIDOC - The Digital Curation of Cultural Heritage. (2008)
11. Goerz, G., Scholz, M.: Content Analysis of Museum Documentation with a Transdisciplinary Approach. In: Proceedings of the EACL 2009 Workshop on Language Technology and Resources for Cultural Heritage, Social Sciences, Humanities, and Education. (2009)
12. Beckett, D., Burners-Lee, T.: Turtle - Terse RDF Triple Language. http://www.w3.org/TeamSubmission/turtle/ (January 2008)
13. Syrjnen, T.: Lparse 1.0 User's Manual. (2000)
14. Haarslev, V., Möller, R.: Description of the RACER System and its Applications. In: Proceedings International Workshop on Description Logics (DL-2001), Stanford, USA, 1.-3. August. (2001) 131–141
15. Odersky, M.: Report on the Programming Language Scala. http://lampwww.epfl.ch/scala/ (2002)

An Algebraic Approach to Dynamic Epistemic Logic

Prakash Panangaden[1], Caitlin Phillips[1], Doina Precup[1], and Mehrnoosh Sadrzadeh[2]

[1] Reasoning and Learning Lab, School of Computer Science, McGill University, Montreal, Canada
[2] Computing Laboratory, Oxford University, Oxford, UK
{prakash,cphill, dprecup}@cs.mcgill.ca & mehrs@comlab.ox.ac.uk

Abstract. Dynamic epistemic logic plays a key role in reasoning about multi-agent systems. Past approaches to dynamic epistemic logic have typically been focused on actions whose primary purpose is to communicate information from one agent to another. These actions are unable to alter the valuation of any proposition within the system. In fields such as security, it is easy to imagine situations in which this sort of action would be insufficient. We expand the algebraic framework presented by M. Sadrzadeh [14] to include both communication actions and dynamic actions that change the state of the system. Furthermore, we propose a new modality that captures both epistemic and propositional changes resulting from the agents' actions.

1   Introduction

As the applications for epistemic logic and dynamic epistemic logic grow more numerous and more diverse, we are faced with the challenge of developing logics rich enough to model these applications but also flexible enough that they are not limited to one particular application. Although it is unlikely that a one-size-fits-all logic will work for every application, logics that incorporate more algebraic structure make it easier to model a variety of situations without having to be too explicit in the description of the situation. Once the underlying algebraic structure of such a logic is in place, its algebraic constructs can be interpreted in a variety of ways. For example, in the semantic web, where agents are constantly interacting with each other and exchanging information, having a formal, algebraic model for how these exchanges take place, and what is exchanged is key to reasoning about the security of a system. In this paper we focus on building a robust algebraic model, which can then be interpreted in a variety of situations.

Epistemic Logic, the branch of logic dealing with knowledge and belief, was first introduced by Hintikka in [10]. Hintikka gave a semantics for epistemic logic that is a simple variation of Kripke's semantics (see [11]), which was then extended to model multi-agent systems by using accessibility relations for each agent (see [5, 8, 7]). However, neither of these logics dealt with situations in which the agents' knowledge changes over time.

Since then, many different approaches have been proposed to formalize the dynamics of knowledge in multi-agent systems. In some cases dynamic modalities (see [9]) are used in conjunction with model restriction (see [16, 6]) to alter the structural properties of the Kripke model, and thus the agents' knowledge. In particular, a lot of focus has been put on logics in which actions take the form of public announcements of propositions [16, 13]. While a lot can be done using logics of this type, they have one major drawback: actions are necessarily idempotent, meaning that announcing the same proposition twice will have the exact same effect as announcing it once. However, the repetition of a statement may actually convey information; an example of problem of this type is the *Muddy Children Puzzle* (see [8]).

In [3, 14], the authors view actions as resources. This helps overcome the issue of idempotent actions, thus opening the door to modelling more complex scenarios. In [14], the dynamics of knowledge is represented by a proposition set and action set with more algebraic structure than in previous logics.

In this paper, we expand the work from [14] and [3] by broadening the algebraic framework they introduced to model knowledge in multi-agent systems. In particular, previous work focused predominantly on communication actions (see [14, 2, 5]). Our goal is to also model dynamic actions, which may change the state of the system (and hence, the valuations of the propositions within the system), in addition to conveying information. Such actions are important in multi-agent systems in which planning has to occur.

The paper is organized as follows. In Sec. 2.1, we review Kripke structures and how they are used to model agents' knowledge. In Sec. 2.2 we describe in detail the algebraic structures used to model multi-agent systems in [14]. In Sec. 3 we define systems with dynamic actions and present an example showing why the model proposed in [14] does not work for such systems. In Sec. 4, we extend the model presented in Sec. 2.2 to accommodate such situations, and revisit the example. Finally in Sec. 5, we conclude and discuss possible extensions of this work.

2 Preliminaries

2.1 Kripke Structures

Kripke structures are a common way of formalizing and connecting epistemic concepts in multi-agent systems. They allow for the model to keep track of the underlying state of the system, while also modelling which states (or worlds) each agent deems to be possible.

Definition 1. *A **Kripke structure** M for a set of agents $\mathcal{A}$ over a set of propositions Φ is a tuple $M = (W, V, \{R_A\}_{A \in \mathcal{A}})$ [5] where W is a set of states (also called possible worlds), R_A defines a binary relation on W (referred to as the **accessibility relation**) for each $A \in \mathcal{A}$, and $V :$ is a valuation mapping $\Phi \to \mathcal{P}(W)$.*

A proposition p is satisfied at a state (possible world) w in a Kripke structure M (written as $(M, w) \vDash p$) if and only if $w \in V(p)$. In addition, $(M, w) \vDash \top, \forall w$ and $(M, w) \nvDash \bot, \forall w$. More complex propositions (conjunctions, disjunctions, negation and implication) can be evaluated using propositional logic [5].

In Kripke structures, knowledge is modelled using the accessibility relation R_A defined over $(W \times W)$. For each agent, if two worlds are related by the agent's accessibility relation, it means the agent is unable to tell them apart. More specifically, if world w_1 is related to world w_2 by agent A's modality (denoted by $w_1 R_A w_2$), then when w_1 is the case, A thinks that w_2 is possible. In layman's terms, an agent "knows" a formula is true if the formula holds in all worlds it thinks might be possible. Formally, $(M, w) \vDash K_A \varphi$ iff $\forall w'$ s.t. $wR_A w', (M, w') \vDash \varphi$. For any model M and agent A, we typically require that the knowledge modality to satisfy the following axioms [5]:

A0 $M \vDash \varphi$, then $M \vDash K_A \varphi$ (Knowledge generalization)
 K $M \vDash (K_A \varphi \wedge K_A(\varphi \Rightarrow \psi)) \Rightarrow K_A \psi$ (Distribution)
 T $M \vDash K_A \varphi \Rightarrow \varphi$ (Truth or knowledge axiom)
 D $M \vDash K_A \varphi \Rightarrow \neg K_A \neg \varphi$ (Consistency axiom)
 4 $M \vDash K_A \varphi \Rightarrow K_A K_A \varphi$ (Positive Introspection)
 5 $M \vDash \neg K_A \varphi \Rightarrow K_A \neg K_A \varphi$ (Negative Introspection)

Depending on the applications, the axioms that the knowledge modality is required to satisfy might change. Remarkably, almost all of these axioms correspond to a single, specific property of the accessibility (R_A) relations. The properties of interest include reflexivity, symmetry, transitivity, Euclidity, and seriality. Table 1 presents this correspondence.

	Axiom	Name	Property
T	$M \vDash K_i \varphi \Rightarrow \varphi$	Truth or knowledge axiom	Reflexive
D	$M \vDash K_i \varphi \Rightarrow \neg K_i \neg \varphi$	Consistency axiom	Serial
4	$M \vDash K_i \varphi \Rightarrow K_i K_i \varphi$	Positive Introspection	Transitive
5	$M \vDash \neg K_i \varphi \Rightarrow K_i \neg K_i \varphi$	Negative Introspection	Euclidean

Table 1. Knowledge axioms and their corresponding properties

For many purposes, the correct choice is to require that R_A be an equivalence relation, meaning that R_A must be symmetric, transitive and reflexive (and consequently serial and Euclidean). In structures of this kind, two worlds are indistinguishable if and only if the agent has the same information about each world [5].

2.2 Intuitionistic Dynamic Epistemic Action Logic

Intuitionistic Dynamic Epistemic Action Logic (*IDEAL*) [14] is an algebraic approach to dynamic epistemic logic, in which states are described entirely by

the propositions they satisfy. By eliminating the state space and using the algebra of the set of propositions, a much richer structure than just a set with a binary relation becomes available, and with this added structure come new and useful properties. In order for these properties to hold though, we must first equip the set of propositions with sufficient algebraic structure. In particular, we will construct a complete algebraic lattice.

At this point, it is useful to distinguish between the concepts of formulas and propositions. To do this we define two sets: P (whose elements are denoted by lower-case letters $(p, q, ..)$) is a finite set of atomic propositions, and Φ a set of formulas built from Boolean combinations $(\wedge, \vee, \neg, ...)$ of propositions in P. The only requirement placed on Φ is that it must contain all propositions in P and be closed under conjunction. Other entailment axioms may be added as needed.

We will build the lattice on the set Φ, because its nature suggests an obvious, non-trivial notion of order: entailment. Entailment is a preorder on Φ, such that for $\varphi, \psi \in \Phi$, $\varphi \sqsubseteq \psi$ if and only if φ entails ψ. It is clear that the properties of a preorder (reflexivity and transitivity) are satisfied by $\sqsubseteq$. We also include a least and greatest element in Φ ($\bot$ and $\top$ respectively) to ensure that all pairs of elements (and thus all sets) have a least upperbound and a greatest lowerbound.

Recall that $(\Phi, \sqsubseteq)$ is only a preorder and thus may not be anti-symmetric (a requirement if we are building a lattice). To resolve this issue, we use filters to construct a complete algebraic lattice of formulas $(M, \leq)$, which preserves the natural entailment relation on the propositions but is now equipped with the structure of a complete algebraic lattice. Now we describe how actions interact with this lattice, and eventually how it relates to knowledge.

Unlike Kripke structures (where the valuation V is fixed), the sorts of systems we wish to model are those in which the actions serve two purposes: first, they modify the underlying valuation functions of the system and second, they allow agents to communicate information to one another. We will address the communication aspect later, as it must be considered from the perspective of a particular agent.

The idea that actions can modify the valuation of propositions can be expressed in terms of operators on the lattice. We define the action set Q to be a monoid (it is labelled Q rather than A to avoid confusion with agent names), where ; is concatenation. In some cases it makes sense to equip this monoid with a partial-order structure, in which case it becomes a *quantale* [14]. However, in this paper we consider the action set Q to be a monoid without any underlying order. Together, the action monoid and the lattice of formulas are referred to as a system.

Definition 2. *A **system** is a pair (M, Q) where M is a lattice and Q is a monoid acting on M [14]. Each element $q \in Q$ defines a mapping $q : M \to M$ such that :*

1. *$q(\bigvee_i m_i) = \bigvee_i (q(m_i))$, $\forall m_i \in M$ (q preserves joins)*
2. *$\epsilon(m) = m, \forall m \in M$ (the unit of the monoid is the identity operator on M)*
3. *$(q_1; q_2)(m) = q_1(q_2(m))$ (the mapping is associative over the binary operation of the monoid).*

Since each action $q \in Q$ is an join-preserving endomorphism of a complete algebraic lattice (M), it must have a right adjoint. This adjoint, denoted by $[q] : M \to M$, is defined as: $[q]m = \bigvee \{m' \in M | q(m') \leq m\}$, i.e., the join of all formulas m' which, when acted on by q, result in m being true. In other words, m' is the *weakest precondition*, or *dynamic modality* [14]. Formally, the dynamic modality adjoint i as follows:

$$\frac{q(m') \leq m}{m' \leq [q]m} \qquad (1)$$

If the action set is a quantale, it is possible to define other adjoints as well [14], but the dynamic modality is the one most frequently used in defining and reasoning about epistemic systems.

In Kripke structures (as defined in Sec. 2.1), the agents' uncertainty as to which propositions are true was reflected in the fact that agents found certain sets of states indistinguishable. In epistemic systems, we approach uncertainty in a different manner, using the complete algebraic lattice of formulas instead of accessibility relations. For example, if an agent A is unable to distinguish between the times when m and m' hold (for some $m, m' \in M$), we say that when m is true, it *appears* to the agent that $m \vee m'$ is true. This notion is formalized as an appearance map.

Definition 3. *An* **appearance map** *for an agent A is an endomorphism $f_A : M \to M$ with the following properties:*

1. *f_A is increasing: $m \leq f_A(m), \forall m \in M$.*
2. *f_A is monotone: $m_1 \leq m_2 \Rightarrow f_A(m_1) \leq f_A(m_2)$.*
3. *f_A is idempotent: $f_A(f_A(m)) = f_A(m), \forall m \in M$.*
4. *f_A is join-preserving: $f_A(m_1 \vee m_2) = f_A(m_1) \vee f_A(m_2)$.*

The appearance map f_A is defined with respect to a specific agent, A. Furthermore, properties 1-3 make f_A a closure operator. In the context of epistemic systems, these properties have very specific meanings. Property 1 says that f_A is obscuring information in some way, as $f_A(m)$ is *at most* as informative as m (recall that M is ordered by entailment). Property 2 says that f_A is order-preserving: if m_1 was more informative than m_2 $(m_1 \leq m_2)$, then when f_A is applied to both propositions, $f_A(m_1)$ will be more informative than $f_A(m_2)$. Property 3 says that no additional information can be gained (or lost) by applying f_A repeatedly. Finally, property 4, together with property 2 and the fact that f_A is an endomorphism of a complete lattice, tells us that f_A must have a right-adjoint. In the framework presented here, we define this adjoint to be knowledge and therefore denote it by K_A.

Definition 4. Knowledge *is the right-adjoint of the appearance map, and is defined as follows:*

$$\frac{f_A(m_1) \leq m_2}{m_1 \leq K_A m_2} \qquad (2)$$

Additionally, the following holds: $m \leq K_A f_A(m)$

We claim that the (f_A, K_A) adjoint pair is the connection between how the world appears to an agent and what the agent knows. Indeed, one can interpret the first equation in Def. 4 as saying: If agent A's view of m_1 entails m_2, then whenever m_1 holds in reality, it follows that agent A knows m_2. Because the properties of f_A induce analogous properties of K_A, we can capture some of the properties of knowledge (as described in Section 2.1). For example, positive introspection ($K_A m \Rightarrow K_A K_A m$) follows from the fact that $\leq$ is reflexive ($K_A m \leq K_A m$) and f_A is the idempotent left adjoint to K_A [14]. In systems where the action set is a quantale, it is possible to define an appearance map on the actions as well [14]. For the purpose of this work though, we assume that all actions are visible to all agents (the appearance map on the action set is the identity mapping). We can now modify the definition of a system to include appearance maps, which are key to modelling epistemic situations.

Definition 5. *An* **epistemic system** *is a tuple* $(M, Q, \{f_A\}_{A \in \mathcal{A}})$ *where* (M, Q) *is a system as defined in Def, 2) and* $\{f_A\}_{A \in \mathcal{A}}$ *is a set of appearance maps for each agent* $A \in \mathcal{A}$ *[14].*

Embedded within this epistemic system are two modalities in the form of Galois adjoints: the dynamic modality $(q, [q])$, and the knowledge modality (f_A, K_A). It remains to be shown how these two concepts interact. More specifically, since actions are not necessarily increasing on M, we do not know for any arbitrary $q \in Q, m \in M$ how $q(m)$ relates to m. Because of this, it is not possible to derive how $f_A(m)$ relates to $f_A(q(m))$. In [14], the following update inequality was defined to specify this connection:

$$f_A(q(m)) \leq q(f_A(m)) \tag{3}$$

Intuitively this means that for an agent, observing the execution of an action q should be at least as informative as imagining the outcome of the action.

3 Games with Fact-Changing Actions

Automata games are a concept derived from automata theory, involving one or more agents and an automaton (a finite-state transition system). This system can be deterministic or not. In the deterministic case, the system is defined by a triple $(S, Act, \{\tau_q\}_{q \in Act})$ where S is the state set and Act is the action set. The dynamics of the system is defined by mappings $\tau_a : S \to S$, such that $\tau_q(s) = t$ if and only if taking action q from state s causes a transitions to state t. τ_q does not need to be total (some actions may be disabled in certain states).

These systems can be the setting of a variety of epistemic tasks. Even when the underlying structure of the system is common knowledge, there are still epistemic tasks that can be studied, starting with the localization task (finding the current state), which can be extended to learning in which state the agent started, or "steering" the system to reach or avoid certain states. All of these tasks are useful in multi-agent systems as well, highlighting the need to study these systems from an epistemic perspective.

Given a transition system $(S, Act, \{\tau_q\}_{q \in Act})$, we begin by defining the proposition set. In the case of localization, we take the atomic proposition set P to be $S \cup \{\top, \bot\}$. From here we build our module M using filters as described before (see Sec. 2.2). For the action set, we use again the structure inherited from the transition system and define Act to be the set of atomic actions. From Act we can define a monoid $Q = Act^*$, equipped with an identity element ϵ and an composition function $;: Q \to Q$ such that for $q_1, q_2 \in Q$, $q_1; q_2 = q_1 q_2$ (the concatenation of the two action sequences).

Next we establish the manner in which the action set Q acts on the module of propositions, M. At an atomic level, this is already defined by the transition function τ. To make τ_q a total function, we simply extend it such that if the action q is not enabled at a state s, then $\tau_q(s) = \bot$. Furthermore, note that from Def. 2, the way sequences of actions (i.e. non-atomic actions) act on propositions is defined entirely by the composition of the atomic actions in the sequence. Hence, to characterize the effects of the action monoid Q, it is sufficient to define the effects of atomic actions on the proposition set.

First, we define the identity element $\epsilon \in Q$, such that $\epsilon(m) = m$, $\forall m \in M$. Then, for any atomic action $q \in Act$, we have the following:

- $q(\top) = \bigvee_{s \in S} q(s)$ (Recall that the proposition set $P = S \cup \{\top, \bot\}$).
- $q(\bot) = \bot$.
- $q(s) = \tau_q(s)$ where s is an atomic proposition ($s \in S \subset P$).
- $q(m_1 \vee m_2) = q(m_1) \vee q(m_2)$ where $m_1, m_2 \in M$.
- $q(m_1 \wedge m_2) = q(m_1) \wedge q(m_2)$ where $m_1, m_2 \in M$.

Finally, we have to define appearance maps and knowledge. For the purpose of the following example, we assume that the underlying structure of the transition system is common knowledge to all agents. Thus, by observing the actions available to it at any given time, an agent will be able to rule out certain states. To formalize this concept, we define a function $en : S \to \mathcal{P}(Act)$ defined by

$$en(s) = \{q \in Act | \tau_q(s) \neq \bot\}. \tag{4}$$

$en(s)$ gives the set of actions which are enabled at state s. Now we can use the *enabled* function to define appearance maps for atomic (state) propositions:

$$f_A(s) = \bigvee \{t \in S | en(s) = en(t)\}. \tag{5}$$

That is, when the agent is actually in the state s (and thus proposition s holds), it appears to the agent as though it might be in any state in which the enabled actions match those enabled in s. We also have that $f_A(\bot) = \bot$ (if an illegal action occurs, everyone sees it) and $f_A(\top) = \top$ (this follows from the fact that appearance maps are increasing). Since appearance maps are join preserving as well, we can define how they act on disjunctions of state propositions entirely by specifying their behaviour on atomic propositions: $f_A(s \vee t) = f_A(s) \vee f_A(t)$ for any $s, t \in S$.

As was the case with the actions, each appearance map f_A has a right adjoint $(f_A \dashv K_A)$ which models the agent's knowledge:

$$\frac{f_A(m) \leq m'}{m \leq K_A m'}. \tag{6}$$

3.1 Example

To illustrate the epistemic nature of automata games, and the way IDEAL works, we present a simple example, to which we will return throughout the rest of the paper. Consider the automaton in Fig. 1, in which there is only one agent trying to learn its location in the system. For simplicity, we omit the subscripts on the appearance maps and refer to the single agent's appearance map and knowledge operator as f and K respectively.

As depicted in Fig. 1, the environment is a simple four-state world ($S = \{s_1, s_2, s_3, s_4\}$) with two possible actions ($Act = \{a, b\}$). However, these two actions are not enabled in every state. In particular, states s_3 and s_4 are dead states, meaning they have no outgoing transition arrows. In other words, $en(s_3) = en(s_4) = \emptyset$. On the other hand, states s_1 and s_2 have both actions enabled, thus $en(s_1) = en(s_2) = \{a, b\}$. Note that although both actions are enabled in both states, they do not have identical outcomes. In particular, taking an a action from state s_1 leads to a dead state (s_4), whereas taking an a action from state s_2 leads to state s_1. There is no way to know the exact outcome of an action before it
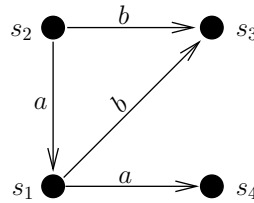
Fig. 1. The transition system for an automaton game

has been taken. This is where the appearance maps come in. Recall that we defined appearance maps for automata games as $f(s) = \bigvee \{t \in S | en(t) = en(s)\}$. Applying this definition to the example, we see that $f(s_1) = f(s_2) = s_1 \vee s_2$, and $f(s_3) = f(s_4) = s_3 \vee s_4$. Prior to taking an action, the agent can only base its knowledge on the actions available to it, and thus cannot distinguish between states s_1 and s_2 even though they behave differently under both actions.

Now let us investigate what happens *after* an a action is taken from state s_1. Recall that prior to taking the action, the agent could not know it was in state s_1 (because $f(s_1) = s_1 \vee s_2$ and applying the knowledge adjoint, $s_1 \leq K(s_1 \vee s_2) \not\leq Ks_1$). To determine exactly where in the system it is, the agent must use the

actions available to it. Since the structure of the system is common knowledge, it is easy to enumerate all possible results of an a action. In this case, if the agent believes it may be in state s_1 or s_2, it knows an a action will lead to state s_4 (if s_1 was the initial state) or state s_1 (if s_2 was the initial state). What we have just calculated is $a(f(s_1))$. However, it is only after the action has been executed that the agent will learn which of the two scenarios have occurred. If the agent truly started in state s_1, it ends up in a dead state after an a transition and ascertains (by means of the appearance map) that the current state must be s_3 or s_4 (this is the calculation for $f(a(s_1))$). To state formally what was just described:

- $a(f(s_1)) = a(s_1 \vee s_2) = a(s_1) \vee a(s_2) = \tau_a(s_1) \vee \tau_a(s_2) = s_4 \vee s_1$
- $f(a(s_1)) = f(s_3) = s_3 \vee s_4$

First and foremost, note that these elements of M are incomparable. That is, $s_4 \vee s_1 \not\leq s_3 \vee s_4$, and likewise $s_3 \vee s_4 \not\leq s_4 \vee s_1$. Hence the update inequality (Eqn. 3) is not applicable to this situation. Even more curious though, is the fact that neither of these propositions tells the whole story. Indeed, if a human were put in this position, he or she would be able to put these two pieces together and determine the exact location, ruling out s_3 since it cannot be reached on an a transition.

This sort of knowledge cannot be obtained by "imagining" the outcome of an action ($a(f(s_1))$), or by "forgetting" that the action occurred and looking only at the resulting proposition ($f(a(s_1))$). Instead, these two concepts need to be combined and actions must be remembered in some way. In order to model this situation effectively, appearance maps *must be allowed to change* as a result of the agents' actions.

4 Dynamic Appearance Maps

We would like to combine the rich algebraic structure of epistemic systems with the versatility of Kripke structures. The reason Kripke structures are able to model complex epistemic situations is that the equivalence relations upon which the agents' knowledge is based change over time. This means that even if the state of the system remains constant, the states that each agent believes possible (and hence the agent's knowledge) change as the agent observes the game.

Applying this idea to our framework means that even if the execution of an action q does not alter the valuation of a proposition m, we do not require the agent's appearance map of m to be fixed as well. In other words, the actions, in addition to acting on propositions, also alter appearance maps. This is analogous to the way equivalence relations in Kripke structures change over time but also allows us to preserve the algebraic structure of epistemic systems.

We will now introduce an extension of the epistemic system that captures the idea that agents' knowledge can change even when the underlying propositions do not. Let $\mathcal{F} : M \to M$ be the set of all possible appearance maps.

Definition 6. *An* **extended epistemic system** *for a set $\mathcal{A}$ of agents is a tuple $(M, Q, \{f_A | A \in \mathcal{A}\}, \{\hat{q} | q \in Q\})$, where $(M, Q, \{f_A | A \in \mathcal{A}\})$ is an epistemic system as defined previously and $\hat{q}$ $(q \in Q)$ defines a mapping $\mathcal{F} \to \mathcal{F}$ describing how an agent's appearance map changes after a q action is executed.*

We must now address the issue of knowledge. Because the definition of knowledge modality rests on that of the appearance map, a change in the appearance map will result in a change in knowledge. Previously it was sufficient to say that $m \leq K_A m'$, meaning that whenever m held, agent A knew m' to be true. Clearly this is no longer the case as illustrated by the example. An agent's knowledge now depends not only on the propositions that hold, but also on the actions which have occurred, and this must somehow be reflected in the knowledge modality. In order to do this, we introduce a new adjoint relationship:

$$\frac{\hat{q}(f_A)(m) \leq m'}{m \leq K_A^q m'} \tag{7}$$

First, note that $\hat{q}(f_A) \in \mathcal{F}$ is itself an appearance map and thus has a right adjoint. This adjoint, K_A^q, looks very similar to the initial knowledge modality but depends on the actions that have occurred. The next step is to define exactly *how* these actions modify the agents appearance maps. To do this, we introduce a new concept, *backward actions*, and use it to define an update inequality analogous to Eqn. 3. Backwards actions are operators acting on the module of propositions. They serve the purpose of allowing the agent to reason retroactively. Every atomic action $q \in Act$ has a corresponding backwards action, denoted by $\backslash_q$. Non-atomic actions, that is, concatenations of two or more atomic actions, also have corresponding backwards actions, which are defined inductively:

- $\backslash_\epsilon = \epsilon$ (The backwards action corresponding to the empty action sequence is the empty action sequence.)
- For any action sequence $\alpha = \beta; q$ such that $\alpha, \beta \in Q, q \in Act$, $\backslash_\alpha = \backslash_q; \backslash_\beta$ (Note that the order is of the actions is reversed: the last action in the original sequence is the first in the backward action sequence).

Backwards actions distribute over meets and joins so $(m_1 \vee m_2)\backslash_q = m_1 \backslash_q \vee m_2\backslash_q$ and $(m_1 \wedge m_2)\backslash_q = m_1\backslash_q \wedge m_2\backslash_q, \forall m_1, m_2 \in M$. Also, $\perp\backslash_q = \perp$. It remains to describe how backwards actions act on elements of the atomic proposition set P. This is where the distinction between backwards actions and normal actions becomes apparent:

Definition 7. *Backwards Actions:*

$$p\backslash_q = \begin{cases} p & \text{if } \exists m \in M, m \neq \perp \text{ s.t. } q(m) \leq p \\ \perp & \text{otherwise} \end{cases} \tag{8}$$

So for any proposition $p \in P$, $p\backslash_q \leq p$ if and only if p holds after taking a q action

Note that backwards actions are *decreasing*, that is, for all $m \in M$, $m\backslash_q \leq m$. This follows from the fact that backwards actions either leave atomic propositions unaltered or result in a contradiction ($\bot$). Using backwards actions, we define a new update inequality which defines how *normal* actions should modify appearance maps in extended epistemic systems.

Definition 8. *Update Inequality:*

$$\hat{q}(f_A)(m) \leq f_A(q(m))\backslash_q \tag{9}$$

In order to explain this update inequality, we will look at each part separately. First, recall that the left-hand side of the equation is simply the revised appearance map $\hat{q}(f)(m)$. If initially m holds, then $\hat{q}(f)(m)$ specifies how the world appears to the agent after taking a q action. Clearly, $\hat{q}(f)(m) \leq f(q(m))$, that is, after taking a q action when m held initially, the agent should have at least as much information about its environment as it would if it ignored the action itself and simply looked at the resulting proposition, $q(m)$.

However, we can say something even stronger. Recall that backwards actions are decreasing and thus, $f(q(m))\backslash_q \leq f(q(m))$. $f(q(m))\backslash_q$ can be viewed as follows: take the appearance map of the resulting proposition $q(m)$ and apply the backwards action $\backslash_q$. This allows us to eliminate any disjuncts of the formula $f(q(m))$ that are not consistent with the fact that the last action taken was an q. In other words, we are remembering the action and its effects without having to keep track of entire action sequences.

4.1 Example

We now revisit our example from Section 2.2, Fig. 1. Recall that the goal of this example was to be able to prove statements of the form: $s_1 \leq [a]Ks_4$ (if the agent is in state s_1, after an a action, it will know that it is in state s_4). We will do this by using dynamic appearance maps and their resulting knowledge modalities, which allow us to explicitly incorporate the observation of an action. Hence it suffices to show that: $s_1 \leq K^a s_4$

From here we can apply the dynamic knowledge adjoint $(\hat{a}(f), K^a)$ and see that it is enough to prove $\hat{a}(f)(s_1) \leq s_4$. However, we do not know exactly how that $\hat{a}$ action affects the appearance map, as this is not explicitly defined in the formalism. However, we know that it must respect the revised update inequality: $\hat{a}(f)(s_1) \leq f(a(s_1))\backslash_a$.

Thus, it suffices to show that $f(a(s_1))\backslash_a \leq s_4$. To this end, we evaluate the left-hand side of the equation and find that $a(s_1) = \tau_a(s_1) = s_4$. Applying the appearance map f to this result, we get that $f(s_4) = \bigvee\{s \in S | en(s) = en(s_4)\} = s_3 \vee s_4$. since s_3 and s_4 are both dead states with no actions enabled. Then we apply the backwards a-action and find that $(s_3 \vee s_4)\backslash_a = \bot \vee s_4 \leq s_4$. We have now shown that $\hat{a}(f)(s_1) \leq s_4$ by way of the dynamic knowledge adjoint and the revised update in equality. It follows then, that $s_1 \leq K^a s_4$.

5   Conclusions and Future Work

The main contribution of this work is to broaden the algebraic framework developed in [14] and [3] to include actions which are both communicative and dynamic in nature. To understand the effect these actions on agents' knowledge, we introduced the notion of dynamic appearance maps along with a new modality that captures both the epistemic and non-epistemic dynamics of the system.

There is a striking similarity between the backward actions defined in Section 4 and the model restriction process used in Public Announcement logic (see [16]). It would be interesting to further explore this similarity and see what the equivalent of the backwards action and the newly defined knowledge modality (K_A^q) are in Kripke semantics. Exploring the coalgebraic properties of backwards actions, as the authors of [14, 12] did for the knowledge modality, would also be informative.

Another possible extension is the development of a proof system. In [14], in order to prove the soundness and completeness of *IDEAL*, the author develops a sequent calculus not unlike that of propositional dynamic logic (see [9]). The sequent calculus makes it possible to formalize the axioms of the logic as rules of inference including the epistemic update. In this way, it is possible to prove the soundness and completeness of the proof system with respect to the algebraic semantics.

The most exciting extensions to this work deal with building a richer dynamic logic. There are a couple of ways in which our logic could be enriched. The first is through the introduction of new modalities. The algebraic structure of our logic provides a framework for introducing new modalities without having to rethink the entire system. This is especially important when dealing with security protocols. While several logics [4, 15] have been developed for reasoning about authentication protocols, they tend to be very specialized and often a new protocol requires a new logic. It is hoped that the algebraic structure of our logic will provide a framework in which various security protocols can be modeled effectively, with only slight alterations needed to accommodate each protocol. Another extention to this work would be to enrich the logic by equipping it with a description logic to model objects and properties [1, 17] . This combination, a strong algebraic framework for modeling change, and a strong knowledge representation for interpreting the outcomes of these actions, would provide an ideal setting in which to reason about complex multi-agent systems.

Acknowledgements

References

1. Alessandro Artale, Enrico Franconi, Milenko Mosurovi C, Frank Wolter, and Michael Zakharyaschev. The dlr us temporal description logic. In *Handbook of Time and Temporal Reasoning in Artificial Intelligence*, pages 96–105. MIT Press, 2001.

2. Alexandru Baltag and Lawrence S. Moss. Logics for epistemic programs. *Synthese*, 139(2):165–224, 2004.

3. Ru Baltag, Bob Coecke, and Mehrnoosh Sadrzadeh. Epistemic actions as resources. In *Journal of Logic and Computation*, pages 555–585, 2007.

4. Michael Burrows, Martin Abadi, and Roger Needham. A logic of authentication. *ACM Trans. Comput. Syst.*, 8(1):18–36, 1990.

5. Ronald Fagin, Joseph Y. Halpern, Yoram Moses, and Moshe Y. Vardi. *Reasoning About Knowledge*. MIT Press, 1995.

6. Jelle Gerbrandy. Dynamic epistemic logic. In *Logic, Language, and Information 2, Stanford University, CSLI Publication*. CSLI Publications, 1997.

7. Jelle Gerbrandy. Bisimulations on planet kripke. In *University of Amsterdam*, 1999.

8. Joseph Y. Halpern and Moshe Y. Vardi. Model checking vs. theorem proving: A manifesto. pages 325–334. Morgan Kaufmann, 1991.

9. David Harel, Dexter Kozen, and Jerzy Tiuryn. Dynamic logic. In *Handbook of Philosophical Logic*, pages 497–604. MIT Press, 1984.

10. Jaakko Hintikka. *Knowledge and belief : an introduction to the logic of the two notions / by Jaakko Hintikka*. Cornell University Press, Ithaca, N. Y. :, 1962.

11. S. A. Kripke. Semantical analysis of modal logic I: Normal modal propositional calculi. 9:67–96, 1963.

12. Lawrence S. Moss. Coalgebraic logic. *Annals of Pure and Applied Logic*, 96(1-3):277–317, 1999.

13. Jan Plaza. Logics of public communications. *Synthese*, 158(2):165–179, 2007.

14. Merhnoosh Sadrzadeh. *Actions And Resources In Epistemic Logic*. PhD thesis, Univerisity of Quebec a Montreal, 2005.

15. Paul Syverson and Iliano Cervesato. The logic of authentication protocols. In *Foundations of Security Analysis and Design, LNCS 2171*, pages 63–136. Springer-Verlag, 2001.

16. Hans van Ditmarsch, Wiebe van der Hoek, and Barteld Kooi. *Dynamic Epistemic Logic*. (Synthese Library). Springer, 1st edition, November 2007.

17. Frank Wolter and Michael Zakharyaschev. Modal description logics: Modalizing roles. *FUNDAMENTA INFORMATICAE*, 39:39–411, 1999.

A MapReduce Algorithm for $\mathcal{EL}^+$

Raghava Mutharaju, Frederick Maier, and Pascal Hitzler

Kno.e.sis Center, Wright State University, Dayton, Ohio

Abstract. Recently, the use of the MapReduce framework for distributed RDF Schema reasoning has shown that it is possible to compute the deductive closure of sets of over a billion RDF triples within a reasonable time span [22], and that it is also possible to carry the approach over to OWL Horst [21]. Following this lead, in this paper we provide a MapReduce algorithm for the description logic $\mathcal{EL}^+$, more precisely for the classification of $\mathcal{EL}^+$ ontologies. To do this, we first modify the algorithm usually used for $\mathcal{EL}^+$ classification. The modified algorithm can then be converted into a MapReduce algorithm along the same key ideas as used for RDF schema.

1 Introduction

The realization of Semantic Web reasoning is central to substantiating the Semantic Web vision [8]. By its very nature, automated reasoning requires a formal representation of knowledge, and in the Semantic Web at least RDF [14] and OWL [9] are two languages commonly used for this purpose. OWL, which is essentially the description logic $\mathcal{SROIQ}$, is considerably more expressive than RDF, and reasoning with it is therefore computationally more expensive. However, restricted *profiles* of OWL 2 (including OWL 2 EL, which is essentially the description logic $\mathcal{EL}^{++}$ [2]) have been developed [15], and for each of these polynomial time algorithms exist for standard inferencing tasks.

There is a large amount of data that is exposed on the Web in RDF and OWL formats. E.g., in a recent discussion of Linked Open Data [5], it is estimated that there are approximately 4.7 billion RDF triples on the Web interlinked by 142 million RDF links.[1] Reasoning with such large amounts of data is inherently difficult due to the high computational complexity of RDF and OWL reasoning. At the same time, however, it has been argued that the Linked Open Data cloud is in need of more expressive schema knowledge, knowledge of a sort expressible in OWL [10]. In order to reason with such data, scalable reasoning algorithms are essential, and parallelization of reasoning is one of the obvious routes to investigate in achieving the required scalability.

The present paper describes the first steps in an effort toward achieving that end. Specifically, we present a parallel algorithm for classifying $\mathcal{EL}^+$ ontologies using MapReduce, which is a programming model and software framework for

---

[1] Some OWL is used as well, usually for indicating that two resources should be considered equal, using `owl:sameAs`.

distributed processing of data on clusters of machines. In doing so, we follow the lead of [21, 22], where the MapReduce framework was successfully applied for computing RDF Schema closure and for reasoning with OWL Horst.

These publications are part of a recent trend in Semantic Web reasoning to explore parallelization of reasoning tasks. Some of the most notable recent developments are the use of the MapReduce framework for RDF [19, 21, 22], using distributed hash tables for RDF Schema [11], the MaRVIN peer-to-peer platform for RDF [16], and the approach in [23] for parallel computation of RDF Schema closures. However, there is relatively little work on attempting to carry these successes over to OWL reasoning, apart from some investigations into OWL Horst [18, 21], OWL RL [13], distributed resolution for $\mathcal{SHIQ}$ ontology networks [17], and some preliminary investigations [1, 6].

The remainder of the paper is structured as follows. Background information on $\mathcal{EL}^+$ and the MapReduce framework is provided in Section 2, and the new algorithm is described in Section 3. An example illustrating how the algorithm works is also given. Section 4 concludes with directions for future research.

Acknowledgements. We thank Keke Chen, Frank van Harmelen, Spyros Kotoulas and Jacopo Urbani for helpful discussions.

2 Preliminaries

2.1 The Description Logic $\mathcal{EL}^+$

Concepts in $\mathcal{EL}^+$ [3, 4, 2] are formed according to the grammar

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

where A ranges over concept names, r over role names, and C, D over (possibly complex) concepts. An $\mathcal{EL}^+$ *ontology* is a finite set of *general concept inclusions* (*GCIs*) $C \sqsubseteq D$ and *role inclusions* (*RIs*) $r_1 \circ \cdots \circ r_n \sqsubseteq r$, where C, D are concepts, n is a positive integer and $r, r_1, \ldots, r_n$ are role names.

The CEL algorithm [4] performs *classification* of an $\mathcal{EL}^+$ ontology, i.e., it computes the complete subsumption hierarchy between all concept names occurring in the ontology. Classification is one of the standard reasoning tasks. The algorithm first transforms the ontology into *normal form*, which requires that all concept and role inclusions are of one of the forms shown in the left part of Figure 1. This can be done in linear time [2]. For the remainder of the paper, we assume that input ontologies are in normal form.

The algorithm is formulated in terms of two mappings S and R, where $S(X)$ maps a class name X to a set of class names, and $R(r)$ maps each role name r to a set of class name pairs. Intuitively, $B \in S(A)$ implies $A \sqsubseteq B$, and $(A, B) \in R(r)$ implies $A \sqsubseteq \exists r.B$. For purposes of the algorithm, $\top$ is taken as a concept name. The mappings are initialized by setting $S(A) = \{A, \top\}$ for each class name A in the input ontology $\mathcal{O}$, and $R(r) = \emptyset$ for each role name in $\mathcal{O}$. The sets $S(A)$ and $R(r)$ are then extended by applying the completion rules shown in the right part of Figure 1 until no rule is applicable.

Normal Form	Completion Rule
$A_1 \sqcap \cdots \sqcap A_n \sqsubseteq B$	**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 $(X, Y) \in R(r)$, $(Y, Z) \in R(s)$, $r \circ s \sqsubseteq t \in \mathcal{O}$, $(x, Z) \notin R(t)$ then $R(t) := R(t) \cup \{(X, Z)\}$

Fig. 1. The CEL algorithm for $\mathcal{EL}^+$

The algorithm is guaranteed to terminate in polynomial time relative to the size of the input ontology, and it is also sound and complete: after termination, $B \in S(A)$ if and only if $A \sqsubseteq B$ holds, for all class names A and B.

2.2   MapReduce

MapReduce is a programming model for distributed processing of data on clusters of machines (each machine being called a *node*) [7]. The data set to be processed is divided into multiple chunks, and each chunk is assigned to an idle node. There are three different types of node, and each type has its own function.

Master: The Master node assigns chunks to Map nodes and passes the intermediate output locations to Reduce nodes. It also takes care of node failures.

Map: Map nodes accept data chunks from the Master and generate intermediate output according to a user-defined function. In its general form, the function accepts a key-value pair and returns a set of key-value pairs. The output pairs are typically written to a local disk, and the location of these is returned to the Master. The functionality of Map nodes can be represented as

$$\text{Map} : (k_1, v_1) \mapsto \text{list}(k_2, v_2).$$

Reduce: Reduce nodes are notified of the locations of intermediate output. They group values by key, and then process the values according to a user-defined Reduce function. One or more output values is produced. The general process can be represented as

$$\text{Reduce} : (k_2, \text{list}(v_2)) \mapsto \text{list}(v_3).$$

There are several prominent implementations of the MapReduce model.[2] Using them, developers need only define the Map and Reduce functions. Lower level and administrative tasks, such as allocating data to nodes and recovering from failures, are handled by general purpose components of the system.

---

[2] E.g., Hadoop (`http://hadoop.apache.org/`) is a popular Java implementation.

3 MapReduce for $\mathcal{EL}^+$

We follow the lead of [22], which describes a MapReduce algorithm for computing RDF Schema closures. However, since the completion rules from Figure 1 are structurally more complicated than the RDF Schema completion rules, we cannot straightforwardly adopt their approach. In the submitted paper [21], the authors extend their approach to OWL Horst [20], facing structurally similar problems. However, due to the specific knowledge bases they are looking at, they choose a solution which is not applicable in our case. We will return to this discussion in Section 3.2, after presenting our algorithm.

3.1 Revising the CEL algorithm

Considering the completion rules in Figure 1, it is rather straightforward to cast rules R2 and R4 into a MapReduce format, and we will see in Section 3.2 how this is done. Rules R1, R3, and R5, however, cannot be transformed directly, and so we first give an alternative formulation of the CEL algorithm. The rules of the reformulated algorithm are cast into a MapReduce format in Section 3.2.

The reformulation requires an additional function P (which stands for *Partial*) and an extension of the function R. These serve to split some of the completion rules from Figure 1 into two rules, explained shortly.

The function P maps each class name X (including $\top$) to a set of pairs (A, B), where A and B are class names (again including $\top$). Intuitively, $(A, B) \in P(X)$ implies $A \sqcap X \sqsubseteq B$. Initially, $P(X)$ is set to $\emptyset$ for each X.

R is extended to map expressions of the form $r \circ s$, for role names r and s, to pairs of class names (possibly including $\top$). The intuition remains the same, however: $(A, B) \in R(r \circ s)$ implies $A \sqsubseteq \exists (r \circ s).B$. The latter expression is not a valid $\mathcal{EL}^+$ expression, but it is semantically unproblematic. Furthermore, it causes no problems in the algorithm, since we do not formally deal with such expressions, and in particular they are not allowed in the input ontology.

We also require another normalization step: Each axiom of the form $A_1 \sqcap \cdots \sqcap A_n \sqsubseteq A$, for $n > 2$, is replaced by $n-1$ axioms $A_1 \sqcap A_2 \sqsubseteq N_1$, $N_1 \sqcap A_3 \sqsubseteq N_2$, ..., $N_{n-2} \sqcap A_n \sqsubseteq A$, where all N_i are class names not occurring anywhere else in the final knowledge base. This transformation obviously retains the original subsumption hierarchy between named classes of the original ontology.

Our revised algorithm for $\mathcal{EL}^+$ is now identical to the algorithm presented in Section 2.1, except that the completion rules from Figure 1 are replaced with those in Figure 2, and the input is now required to be in the modified normal form. The algorithm terminates if no application of any of the rules extends any of the sets $S(X)$, $R(r)$, $P(X)$, or $\mathcal{O}$.

Let us explain the rationales behind the new completion rules. The original rule R1 can be simulated by subsequent applications of R1-1 and R1-2 (note that an inclusion axiom of the form A $\sqsubseteq$ B—that is, where $n = 1$—is covered by R1-2 alone). At the same time, output produced by applying R1-1 is only used in the precondition of R1-2, and so it does not have any other effect on the outcome of the overall algorithm. Rule R2 is left untouched apart from

Normal Form	Completion Rule	Key
$A_1 \sqcap A_2 \sqsubseteq B$	**R1-1** If $A_1 \in S(X)$ and $A_1 \sqcap A_2 \sqsubseteq B \in \mathcal{O}$ then $P(X) := P(X) \cup \{(A_2, B)\}$	A_1
$(A, B) \in P(X)$	**R1-2** If $A \in S(X)$ and $((A, B) \in P(X)$ or $A \sqsubseteq B \in \mathcal{O})$ then $S(X) := S(X) \cup \{B\}$	A
$A \sqsubseteq \exists r.B$	**R2** If $A \in S(X)$ and $A \sqsubseteq \exists r.B \in \mathcal{O}$ then $R(r) := R(r) \cup \{(X, B)\}$	A
$\exists r.A \sqsubseteq B$ for A	**R3-1** If $A \in S(X)$ and $\exists r.A \sqsubseteq B \in \mathcal{O}$ then $\mathcal{O} := \mathcal{O} \cup \{\exists r.X \sqsubseteq B\}$	A
$\exists r.A \sqsubseteq B$ for r	**R3-2** If $(X, Y) \in R(r)$ and $\exists r.Y \sqsubseteq B \in \mathcal{O}$ then $S(X) := S(X) \cup \{B\}$	r (or Y)
$r \sqsubseteq s$	**R4** If $(X, Y) \in R(r)$ and $r \sqsubseteq s \in \mathcal{O}$ then $R(s) := R(s) \cup \{(X, Y)\}$	r
$r \circ s \sqsubseteq t$	**R5-1** If $(X, Z) \in R(r)$ and $(Z, Y) \in R(s)$ then $R(r \circ s) := R(r \circ s) \cup \{(X, Y)\}$	Z

Fig. 2. Revised CEL algorithm for $\mathcal{EL}^+$. The keys are used in the MapReduce algorithm. Note that in R4, r is allowed to be compound, i.e., of the form $s \circ t$.

removing the precondition $(X, B) \notin R(X)$, which is used only for termination purposes. Since we have reworded the termination condition, there is in effect no difference between the two versions of the rule. The reason for rewording is that the new termination condition is more easily cast into MapReduce format. The original rule R3 can be simulated by subsequent applications of R3-1 and R3-2. Rule R3-1 introduces new axioms into $\mathcal{O}$, but since the new axioms are logical consequences of the knowledge base, they do not affect soundness or completeness of the algorithm. Note that this also does not cause any problems with respect to termination, since there is a finite upper bound[3] on the number of possible axioms of the form $\exists r.X \sqsubseteq B$. Rule R4 is again left unchanged, apart from the fact that it now also applies to compound expressions of the form $s \circ t$. Note, however, that this extension of R4 is semantically sound, and so it does not affect the correctness of the algorithm. The original rule R5 can be simulated by subsequent application of R5-1 and R4—the latter in this case using the extended form with composed roles. The newly introduced output produced by applying R5-1 is sound, and so the correctness of the algorithm is again left unaffected.

It is straightforward to show formally that our revised algorithm is indeed sound, complete, and terminating. It is also of polynomial worst-case complexity in the size of the input ontology; this can be shown easily along the lines of argument for the algorithm presented in [3] for $\mathcal{EL}^{++}$.

---

[3] The upper bound is $k \cdot l^2$, where k is the number of role names, and l is the number of class names (including $\top$) in $\mathcal{O}$.

3.2 Parallelization using MapReduce

We now convert the completion rules of Figure 2 into MapReduce algorithms computing the closure of $\mathcal{O}$, S, R and P. Initialization is done by setting $S(A) = \{A, \top\}$ and $P(A) = \emptyset$, for each class name A (including $\top$), and $R(r) = \emptyset$ for each role name r. In the discussion below, we freely switch between viewing S, R, and P as maps and viewing them as sets, and we slightly abuse terminology by referring to all expressions of the form $A \in S(X)$, $(A, B) \in P(X)$ or $(X, Y) \in R(r)$, in addition to all elements of $\mathcal{O}$, as *axioms*.

The completion rules are applied in an iterative manner, picking one rule to apply in each iteration. The rules are interdependent and the results of previous iterations are reused in subsequent iterations. This is realized by adding the outputs of each iteration to the database where $\mathcal{O}$, S, R and P are stored.

The general strategy that is followed by all the algorithms is given in Figure 3. The set of axioms (taken from $\mathcal{O}$, S, P, and R) forms the input. This set is divided into multiple chunks, and each chunk is distributed to different computing nodes. These first act as map nodes and then as reduce nodes, thereby completing the parallel application of one of the completion rules.

To give a concrete example, consider rule R2. Each map node first identifies, in its input chunk, all axioms of the form $A \in S(X)$ and $A \sqsubseteq \exists r.B$ and then outputs them as key-value pairs $\langle A, A \in S(X) \rangle$ and $\langle A, A \sqsubseteq \exists r.B \rangle$, respectively. In the reduce phase, all pairs with key A end up in the same reduce node, which can then complete the application of R2 by adding to R.

This idea of casting completion rules into MapReduce closely follows [22]. Note, however, that the rules R1, R3, and R5 from the original CEL algorithm cannot directly be dealt with using this approach, since each of them has three preconditions which do not share a common element which could be used as a key for the reduce phase. In [21], which deals with OWL Horst, a similar problem occurs, and the authors deal with it by performing part of the operation in-memory using a central store. This is made possible because of the specific form of the problematic completion rules for OWL Horst, where one of the preconditions is always a schema axiom, and because of the specific applications the authors have in mind, where there is much less schema knowledge than facts. Note that we cannot adopt this approach for $\mathcal{EL}^+$, since a separation along similar lines would hardly be reasonable for studying classification in $\mathcal{EL}^+$. We hence choose to provide a generic algorithm, based on the revised CEL algorithm.

We refrain from giving detailed descriptions of the MapReduce algorithms corresponding to all of the completion rules in Figure 2. We do give details for rules R1-1 and R1-2 in Figures 4 and 5, however. The remaining rules are dealt with in a completely analogous manner, using the keys shown in Figure 2.

The behavior of R1-1 and R1-2 can be illustrated using the below axioms.

$$A \sqcap B \sqsubseteq C$$
$$A \sqsubseteq B$$
$$A \sqsubseteq D$$

ComputeClassificationSet()
{

1. Each node takes a subset of all axioms as input and computes additional elements
   for $\mathcal{O}$, S, R or P, depending on the completion rule which is applied.
   (a) In the Map phase, based on the rule that is applied in the current iteration,
       the set of axioms which satisfy any one of the preconditions of the rule are
       found and given as output. In the output ⟨key, value⟩ pairs, *key* is the concept
       or relationship which is common to both the preconditions of a rule (as indi-
       cated in Figure 2). The value is the corresponding axiom (which satisfies the
       precondition).
   (b) In the Reduce phase, all axioms belonging to the same key are collected from
       different nodes and conclusions of the completion rule are computed according
       to the completion rule, taking all valid combinations of axioms into account.
2. All outputs are stored in the database, unless they are already contained in it.
3. Call ComputeClassificationSet() again until no selection of a rule results in any
   additions to the database.

}

Fig. 3. General strategy followed by MapReduce algorithms for each completion rule.

It readily follows that A is a subclass of both C and D, and one can obtain this
result using R1-1 and R1-2 alone. When the algorithm is initialized, $S(X) = \{X, \top\}$ and $P(X) = \emptyset$ for each class name X. When R1-1 is applied (we may
suppose that both axioms are given to a single node), the map function generates
the key-value pair $\langle A, A \sqcap B \sqsubseteq C \rangle$. Other pairs are produced as well (specifically,
$\langle X, S(X) \rangle$ and $\langle \top, S(X) \rangle$, for each name X). This intermediate output is used
in the reduce phase, which in this example produces the following result: For
key A, $v_1 = A \in S(A)$ and $v_2 = A \sqcap B \sqsubseteq C$ together cause (B, C) to be added
to $P(A)$. This new axiom is added to the set of axioms already present, and all
axioms—old and new—are used in the next map-reduce step.

The map phase of R1-2 then yields the following key-value pairs:

$$\{\langle A, A \in S(A) \rangle, \langle B, (B, C) \in P(A) \rangle, \langle A, A \sqsubseteq B \rangle, \langle A, A \sqsubseteq D \rangle\}$$

In the reduce phase, since $A \in S(A)$ and $A \sqsubseteq B$ are both associated with
key A, B is added to $S(A)$. Analogously, D is added to $S(A)$. These are the only
significant changes made during the iteration. Applying R1-2 again, however,
causes C to be added to $S(A)$ as well. Specifically, when the map function is
invoked, since B is now an element of $S(A)$, the pair $\langle B, B \in S(A) \rangle$ will be
generated, as will $\langle B, (B, C) \in P(A) \rangle$. In the reduce phase, since both tuples are
now indexed by the same key (namely, B), they can be used in conjunction. It
is this that allows C to be added to $S(A)$.

```
map(key, value)
    // key: line number (not relevant)
    // value: an axiom
{
    if(value ==  A ∈ S(X))
        emit(⟨A,  A ∈ S(X)⟩);
    else if(value ==  A₁ ⊓ A₂ ⊑ B)
        emit(⟨A₁,  A₁ ⊓ A₂ ⊑ B⟩);
}
reduce(key, iterator values)
    // key: A concept name (e.g. A)
    // values: axioms corresponding to a rule precondition
{
    for each  v₁ in values
        for each  v₂ in values
        {
            if(v₁ ==  A₁ ∈ S(X) and  v₂ ==  A₁ ⊓ A₂ ⊑ B)
                emit((A₂, B) ∈ P(X));
        }
}
```

Fig. 4. MapReduce algorithm for R1-1. The input of the map function is an axiom, taken from either the ontology $\mathcal{O}$, or else one generated from the sets S, P, or R. Key-value pairs are generated, which are used in the reduce phase. The reduce function accepts a key and a list of values. Every possible combination of values is examined to determine whether R1-1 is applicable. A list of axioms is produced.

4 Conclusion and Future Work

Due to the ever increasing amount of data on the Web, there is a need for parallelizable approaches to reasoning algorithms. Following the lead of existing work on scalable implementations of RDF Schema closure, we have in this paper provided a MapReduce algorithm for the classification of $\mathcal{EL}^+$ ontologies. This approach, we believe, is scalable and will reduce the time needed to compute classification over large ontologies.

Our next step is to implement this algorithm using the Hadoop framework and the cloud computing infrastructure available at Wright State University. The experiences reported in [21, 22] on using MapReduce for RDF Schema indicate that optimizations, in particular concerning the choice of which completion rule is applied next, will be crucial for the performance. We intend to use the master node to monitor the outputs of previous MapReduce steps, and to use this output to decide which completion rule to apply next.

The results in [21, 22] also indicate that the MapReduce approach requires rather large datasets to show a pay-off in terms of performance, which in our case may require the generation of artificial datasets for initial experiments.

```
map(key, value)
    // key: line number (not relevant)
    // value: an axiom
{
    if(value == A ∈ S(X))
        emit(⟨A,  A ∈ S(X)⟩);
    else if(value == (A,B) ∈ P(X))
        emit(⟨A,  (A,B) ∈ P(X)⟩);
    else if(value == A ⊑ B)
        emit(⟨A,  A ⊑ B⟩);
}
reduce(key, iterator values)
    // key: A concept (e.g., A)
    // values: axioms corresponding to a rule precondition
{
    for each v₁ in values
        for each v₂ in values
        {
            if(v₁ == A ∈ S(X))
            {
                if(v₂ == (A,B) ∈ P(X) or v₂ == A ⊑ B)
                    emit(B ∈ S(X));
            }
        }
}
```

Fig. 5. MapReduce algorithm for R1-2

We furthermore consider this line of work on $\mathcal{EL}^+$ to be only the starting point for investigations into more expressive languages, such as $\mathcal{EL}^{++}$, ELP [12], or even OWL 2 DL.

References

1. Mina Aslani and Volker Haarslev. Towards Parallel Classification of TBoxes. In Franz Baader, Carsten Lutz, and Boris Motik, editors, *Proceedings of the 21st International Workshop on Description Logics (DL2008), Dresden, Germany, May 13-16, 2008*, volume 353 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2008. Available from http://ceur-ws.org/Vol-353/AslaniHaarslev.pdf.
2. Franz Baader, Sebastian Brandt, and Carsten Lutz. Pushing the EL envelope. In *Proceedings of the 19th International Joint Conference on Artificial Intelligence (IJCAI-05)*, Edinburgh, UK, 2005. Morgan-Kaufmann Publishers.
3. Franz Baader, Carsten Lutz, and Boontawee Suntisrivaraporn. CEL—A Polynomial-time Reasoner for Life Science Ontologies. In U. Furbach and N. Shankar, editors, *Proceedings of the 3rd International Joint Conference on Automated Reasoning (IJCAR'06), Seattle, WA, USA, August 17-20, 2006*, volume 4130 of *Lecture Notes in Artificial Intelligence*, pages 287–291. Springer-Verlag, 2006.

4. Franz Baader, Carsten Lutz, and Boontawee Suntisrivaraporn. Efficient reasoning in $\mathcal{EL}^+$. In *Proceedings of the 2006 International Workshop on Description Logics (DL2006)*, volume 189 of *CEUR Workshop Proceedings*, 2006.

5. Christian Bizer, Tom Heath, and Tim Berners-Lee. Linked data – the story so far. *International Journal on Semantic Web and Information Systems*, 5(3):1–22, 2009.

6. Jürgen Bock. Parallel Computation Techniques for Ontology Reasoning. In Amit P. Sheth et al., editors, *Proceedings of the 7th International Semantic Web Conference, ISWC 2008, Karlsruhe, Germany, October 26-30, 2008*, volume 5318 of *Lecture Notes in Computer Science*, pages 901–906. Springer, 2008.

7. Jeffrey Dean and Sanjay Ghemawat. MapReduce: Simplified data processing on large clusters. In *Proceedings of the 6th Symposium on Operating System Design and Implementation (OSDI 2004), December 6-8, 2004, San Francisco, California, USA*, pages 137–150. USENIX Association, 2004.

8. Pascal Hitzler. Towards reasoning pragmatics. In Krzysztof Janowicz, Martin Raubal, and Sergei Levashkin, editors, *GeoSpatial Semantics, Third International Conference, GeoS 2009, Mexico City, Mexico, December 3–4, 2009. Proceedings*, Lecture Notes in Computer Science, pages 9–25. Springer, 2009.

9. Pascal Hitzler, Markus Krötzsch, Bijan Parsia, Peter F. Patel-Schneider, and Sebastian Rudolph, editors. *OWL 2 Web Ontology Language: Primer*. W3C Recommendation 27 October 2009, 2009. Available from `http://www.w3.org/TR/owl2-primer/`.

10. Prateek Jain, Pascal Hitzler, Peter Z. Yeh, Kunal Verma, and Amit P. Sheth. Linked Data is Merely More Data. In Dan Brickley, Vinay K. Chaudhri, Harry Halpin, and Deborah McGuinness, editors, *Linked Data Meets Artificial Intelligence*, pages 82–86. AAAI Press, Menlo Park, CA, 2010.

11. Zoi Kaoudi, Iris Miliaraki, and Manolis Koubarakis. RDFS Reasoning and Query Answering on Top of DHTs. In Amit P. Sheth et al., editors, *Proceedings of the 7th International Semantic Web Conference, ISWC 2008, Karlsruhe, Germany, October 26-30, 2008*, volume 5318 of *Lecture Notes in Computer Science*, pages 499–516. Springer, 2008.

12. Markus Krötzsch, Sebastian Rudolph, and Pascal Hitzler. ELP: Tractable Rules for OWL 2. In Amit P. Sheth et al., editors, *Proceedings of the 7th International Semantic Web Conference, ISWC 2008, Karlsruhe, Germany, October 26-30, 2008*, volume 5318 of *Lecture Notes in Computer Science*, pages 649–664. Springer, 2008.

13. Gergely Lukácsy and Péter Szeredi. Scalable Web Reasoning Using Logic Programming Techniques. In Axel Polleres and Terrance Swift, editors, *Proceedings of the Third International Conference on Web Reasoning and Rule Systems, RR 2009, Chantilly, VA, USA, October 25-26, 2009*, volume 5837 of *Lecture Notes in Computer Science*, pages 102–117. Springer, 2009.

14. Frank Manola and Eric Miller, editors. *Resource Description Framework (RDF). Primer*. W3C Recommendation, 10 February 2004. Available at `http://www.w3.org/TR/rdf-primer/`.

15. Boris Motik, Bernardo Cuenca Grau, Ian Horrocks, Zhe Wu, Achille Fokoue, and Carsten Lutz, editors. *OWL 2 Web Ontology Language: Profiles*. W3C Recommendation, 27 October 2009. Available at `http://www.w3.org/TR/owl2-profiles/`.

16. Eyal Oren, Spyros Kotoulas, George Anadiotis, Ronny Siebes, Annette ten Teije, and Frank van Harmelen. Marvin: Distributed reasoning over large-scale Semantic Web data. *Web Semantics: Science, Services and Agents on the World Wide Web*, 7(4):305–316, 2009.

17. Anne Schlicht and Heiner Stuckenschmidt. Distributed Resolution for Expressive Ontology Networks. In Axel Polleres and Terrance Swift, editors, *Proceedings of the Third International Conference on Web Reasoning and Rule Systems, RR 2009, Chantilly, VA, USA, October 25-26, 2009*, volume 5837 of *Lecture Notes in Computer Science*, pages 87–101. Springer, 2009.
18. Ramakrishna Soma and Viktor K. Prasanna. Parallel inferencing for OWL knowledge bases. In *2008 International Conference on Parallel Processing, ICPP 2008, September 8-12, 2008, Portland, Oregon, USA*, pages 75–82. IEEE Computer Society, 2008.
19. Radhika Sridhar, Padmashree Ravindra, and Kemafor Anyanwu. RAPID: Enabling Scalable Ad-Hoc Analytics on the Semantic Web. In Abraham Bernstein et al., editors, *Proceedings of the 8th International Semantic Web Conference, ISWC 2009, Chantilly, VA, USA, October 25-29, 2009*, volume 5823 of *Lecture Notes in Computer Science*, pages 715–730. Springer, 2009.
20. Herman J. ter Horst. Completeness, decidability and complexity of entailment for RDF Schema and a semantic extension involving the OWL vocabulary. *Journal of Web Semantics*, 3(2–3):79–115, 2005.
21. Jacopo Urbani, Spyros Kotoulas, Jason Maassen, Frank van Harmelen, and Henri Bal. OWL reasoning with WebPIE: calculating the closure of 100 billion triples. In *Proceedings of the 8th Extended Semantic Web Conference (ESWC2010), Heraklion, Greece, May 30–June 3, 2010*. Springer, 2010.
22. Jacopo Urbani, Spyros Kotoulas, Eyal Oren, and Frank van Harmelen. Scalable Distributed Reasoning Using MapReduce. In Abraham Bernstein et al., editors, *Proceedings of the 8th International Semantic Web Conference, ISWC 2009, Chantilly, VA, USA, October 25-29, 2009*, volume 5823 of *Lecture Notes in Computer Science*, pages 634–649. Springer, 2009.
23. Jesse Weaver and James A. Hendler. Parallel materialization of the finite RDFS closure for hundreds of millions of triples. In Abraham Bernstein et al., editors, *The Semantic Web – ISWC 2009, 8th International Semantic Web Conference, ISWC 2009, Chantilly, VA, USA, October 25-29, 2009*, volume 5823 of *Lecture Notes in Computer Science*, pages 682–697. Springer, 2009.

Distance-based Measures of Inconsistency and Incoherency for Description Logics

Yue Ma[1] and Pascal Hitzler[2]

[1]Institute LIPN, Université Paris-Nord (LIPN - UMR 7030), France
[2]Kno.e.sis Center, Wright State University, Dayton, Ohio
yue.ma@lipn.univ-paris13.fr, pascal.hitzler@wright.edu

Abstract. Inconsistency and incoherency are two sorts of erroneous information in a DL ontology which have been widely discussed in ontology-based applications. For example, they have been used to detect modeling errors during ontology construction. To provide more informative metrics which can tell the differences between inconsistent ontologies and between incoherent terminologies, there has been some work on measuring inconsistency of an ontology and on measuring incoherency of a terminology. However, most of them merely focus either on measuring inconsistency or on measuring incoherency and no clear ideas of how to extend them to allow for the other. In this paper, we propose a novel approach to measure DL ontologies, named distance-based measures. It has the merits that both inconsistency and incoherency can be measured in a unified framework. Moreover, only classical DL interpretations are used such that there is no restriction on the DL languages used.

1 Introduction

Real ontology applications on the Semantic Web will often involve imperfect ontological information [1]. This is reflected as inconsistency or incoherency in the underlying description logic knowledge bases [2, 3]. Inconsistency indicates that there are some logical contradictions such that the ontology becomes trivial because any conclusion follows from it. Incoherency suggests ontology engineering mistakes because some concepts are named but never can be instantiated. Detecting inconsistency and incoherency have been shown important for ontology-based applications [4].

Inconsistency and incoherency are two kinds of relevant but different information about an ontology. Different approaches have been proposed in the literature to deal with them. For conquering the triviality of inconsistent ontologies, there are approaches that circumvent the inconsistency problem by applying non-standard reasoning methods to obtain meaningful answers, such as by paraconsistent semantics or by selecting consistent sub-ontologies [5, 6]. For incoherent ontologies, ontology debugging tools [7–9] and revision operators are studied to resolve modeling errors which lead to incoherencies [10, 11].

Besides directly handling inconsistencies or incoherencies, the measuring of inconsistency and incoherency has been proposed as a promising service to provide some context information which can be used for ontology applications [12–14]. The existing methods around this issue fall into one of the following categories: One is syntax-based

measurement [13] which calculates the percentage of axioms involved in inconsistencies; The other is the semantics-based method [12, 14] which computes the percentage of assertion atoms involved in inconsistencies under some paraconsistent models. Unlike the existing work, in this paper, we propose a new approach, named *distance-based measures*. It is based on classical DL interpretations with no need to refer to any paraconsistent semantics such that it can be used with any DL language. Note that, distance measures have been widely studied in the field of belief revision and belief merging, and also for reasoning under inconsistencies. Inspired by, but different from those works, this paper proposes a way to define inconsistency and incoherency degrees by employing distance measures.

The idea of our approach is to consider the distance between a DL ontology and its *preferred interpretations*, the most relevant classical interpretations, which shows how far it is away from being consistent/coherent. Based on such a distance, we propose the *inconsistency (resp. incoherency) deviation degree* of a DL theory. For example, the inconsistency (resp. incoherency) deviation degree of a consistent ontology (terminology) is 0, which intuitively means that it has no deviation from being consistent (resp. coherent). On the contrary, a DL ontology has 1 as its inconsistency (resp. incoherent) deviation degree if and only if all of its axioms are unsatisfiable (resp. all atomic classes are incoherent), which intuitively indicates that it is *fully* inconsistent (resp. incoherent). The definition of distance is based on the extension of distance-based semantics for propositional logic [15, 16]. Our work essentially differers from [11] in that [11] studies a model-based revision for terminologies but not for measuring incoherency which is our goal in this paper.

This paper is organized as follows. We first provide some basic notions of description logics and distance and aggregation functions in Section 2. Our measures of inconsistency and incoherency are then discussed in detail in Section 3, in which distance-based *inconsistency/incoherency deviation degrees* are defined first; And then the application of such measures for ordering inconsistent ontologies and terminologies is given; Finally the comparison of aggregation functions for better measures is discussed. We wrap up the work in Section 4 with some further perspectives.

2   Preliminaries

We assume that the reader is familiar with basic syntax and semantics of description logics, as introduced, e.g., in [2, 3]. For notation, CN is the set of atomic concepts (concept names), RN is the set of roles (role names), and IN is the set of individuals. It is safe to read this paper under the assumption that we're working with $\mathcal{ALC}$, but the approach will work for any description logic. We will refer to interpretations under the standard semantics as *classical* or *DL* interpretations. An ontology is called satisfiable (unsatisfiable) iff there exists (does not exist) such a model. We denote with $\mathcal{CM}(O)$ the set of classical models of O.

We say that a DL ontology (resp. a TBox or ABox axiom α) is inconsistent iff $\mathcal{CM}(O) = \emptyset$ (resp. $\mathcal{CM}(\alpha) = \emptyset$). A named concept C in a TBox T is unsatisfiable iff $C^I = \emptyset$ for each model I of T. A TBox is incoherent iff there exists an unsatisfiable named concept in T.

We now review basic definitions about distance which will be used in our work to define distance-based inconsistency and incoherency measures.

Definition 1 *A total function $d : U \times U \mapsto \mathbb{R}^+ \cup \{0\}$ is called a* distance *(or* metric *[17]) on U if it satisfies: (1) $\forall u, v \in U, d(u, v) = d(v, u)$; (2) $\forall u, v \in U, d(u, v) = 0$ iff $u = v$; (3) $\forall u, v, w \in U, d(u, v) + d(v, w) \geq d(u, w)$.*

Definition 2 *A numeric aggregation function f is a total function that accepts a multiset of real numbers and returns a real number satisfying: (a) f is non-decreasing in the values of its argument, that is, $f(\{x_1, ...x_i, ..., x_n\}) \leq f(\{x_1, ..., x_i', ..., x_n\})$ iff $x_i \leq x_i'$ where $i \in [1, n]$. (b) $f(\{x_1, ..., x_n\}) = 0$ if and only if $x_1 = ... = x_n = 0$, and (c) $\forall x \in \mathbb{R}, f(\{x\}) = x$.*

We will consider the following aggregation functions in this paper:

– The maximum aggregation function f: $f(\{x_1, ..., x_n\}) = \max_i x_i$;
– The summation aggregation function f: $f(\{x_1, ..., x_n\}) = \sum_i x_i$;
– The $\frac{k}{m}$-voting aggregation function f:

$$f(\{x_1, ..., x_n\}) = \begin{cases} 0 & \text{if } Zero(\{x_1, ..., x_n\}) = n; \\ \frac{1}{2} & \text{if } \lceil \frac{k}{m}n \rceil \leq Zero(\{x_1, ..., x_n\}) < n; \\ 1, & \text{otherwise,} \end{cases}$$

where $Zero(\{x_1, ..., x_n\})$ is the number of zeros in $\{x_1, ..., x_n\}$. Additionally, we use $|S|$ to stand for the cardinality of any set S.

3 Distance-based Measures

During our work on measuring DL ontologies or TBoxes, we obey the following principles:

– *Normalization Principle*: The measure should be a value in $[0, 1]$, where 0 represents a consistent ontology and 1 means a totally inconsistent ontology.
– *Variation Principle*: The possible values under the measurement should be as various as possible such that it can better distinguish between different ontologies according to their degree under this measure.
– *Applicability Principle*: This measure should be useable for measuring both the inconsistency of DL ontology and the incoherency of a DL TBox.

The normalization principle is defined for comparing different ontologies/TBoxes without having to worry about differences in their sizes, in the number of ontological entries, etc. The second principle says that finer granularity is better, since a binary measure is of limited usefulness. By the applicability principle, we enable our method to estimate both inconsistency and incoherency degrees. In fact, this is not a trivial requirement. For example, it seems there is no clear idea how to extend the existing paraconsistent semantics based inconsistency measurements [12, 14] to measure incoherency because incoherent TBoxes do not suffer from the lack of classical models which is just what

paraconsistent semantics is made for. Similarly, the incoherency measure [13] is difficult to be extended to measure inconsistency. The reason is that, unlike the available number of unsatisfiable concepts in an incoherent TBox, the lack of classical models leads to no obvious way to count the number of "inconsistencies" in an inconsistent ontologies.

To achieve such a measure, in this section, we propose a distance-based measuring framework. Before any technical details, we first summarize the underlying ideas.

Let $\mathcal{O}$ be a set of ontologies. A distance function $\lambda : \mathcal{S} \times \mathcal{O} \mapsto \mathbb{R}^+ \cup \{0\}$ is defined as a map from a classical interpretation $I \in \mathcal{S}$ and an ontology $O \in \mathcal{O}$ to a nonnegative real value, where $\mathcal{S}$ is the interpretation space which varies under different measuring tasks. In the following, we will see that the choice of interpretation space $\mathcal{S}$ is different for measuring inconsistency and measuring incoherency. Simply speaking, for measuring inconsistency, we consider the interpretation space S containing all DL interpretations; But for measuring incoherency, $\mathcal{S}$ should merely contain DL interpretations which do not interpret any atomic concept or role as an empty set. Then $\lambda(\cdot, \cdot)$ will be used to select the most relevant DL interpretations to measure inconsistency or incoherency.

3.1 Measuring Inconsistency

In a logical system, interpretations or models are used to represent the semantics. The underlying idea of our measures is that calculating the distance between interpretations is a way to estimate the deviation between two meanings. To this end, next we propose some ways to define the distance between two DL interpretations. The first is the simplest way called drastic distance:

Definition 3 (Drastic Distance) *Let $I_1 = (\Delta, \cdot^{I_1})$ and $I_2 = (\Delta, \cdot^{I_2})$ be two DL interpretations. The drastic distance[1] between I_1 and I_2, denoted $d_D(I_1, I_2)$, is defined as follows:*

$$d_D(I_1, I_2) = \begin{cases} 0 & \textit{if } I_1 = I_2; \\ 1 & \textit{otherwise}. \end{cases}$$

That is, the drastic distance of two interpretations is 0 if they are the same, and 1 otherwise. Different from the Hamming distance given below, for a given ontology with finite numbers of concept and role names, an advantage of the drastic distance $d_D(\cdot, \cdot)$ is in that it always yields a finite value even for infinite domains.

However, the drastic distance is very coarse. A more finer-grained distance is the Hamming Distance, as follows.

Definition 4 (Hamming Distance) *Let $I_1 = (\Delta, \cdot^{I_1})$ and $I_2 = (\Delta \cup \Delta', \cdot^{I_2})$ be two DL interpretations. The Hamming distance between I_1 and I_2 for inconsistency, denoted $d_H(I_1, I_2)$, is defined as follows:*

$$\begin{aligned} d_H(I_1, I_2) = &\ |\{A(a) : A(a)^{I_1} \neq A(a)^{I_2}, A \in CN, a^{I_2} \in \Delta\}| + \\ &\ |\{R(a, b) : R(a, b)^{I_1} \neq R(a, b)^{I_2}, R \in RN, a^{I_2}, b^{I_2} \in \Delta\}| \\ &\ + |CN||\Delta'| + |RN||\Delta'|^2. \end{aligned}$$

---

[1] more commonly known as the *discrete metric*

That is, the Hamming distance of two interpretations for inconsistency is the cardinality of the set of concept and role assertions which are interpreted differently on their common domain Δ plus the number of atomic grounded concept and role assertions. In this way, two interpretations of different sizes of domains are comparable.

Note that $d_H(I_1, I_2)$ can be $+\infty$ if Δ is infinite even if $|CN|$ and $|RN|$ are finite. To avoid this, we only consider finite interpretations whenever talking about the Hamming distance. This is reasonable in practical cases because only finite numbers of individuals can be represented or would be used. It is reasonable also in that if an ontology is inconsistent (resp. a TBox is incoherent), then it is inconsistent (resp. incoherent) w.r.t. finite domains.

In the rest of this paper, when the study is independent on the concrete form of distance functions between two interpretations I_1 and I_2, $d(I_1, I_2)$ is used to refer to either sort of distances, Hamming distance or drastic distance.

Based on distance defined between two classical interpretations, we can define the distance between an interpretation and a TBox or ABox axiom. We will see later that this step is necessary because the set of classical models is empty for an inconsistent ontology.

Definition 5 *Let $I = (\Delta, \cdot^I)$ be a DL interpretation and α be a TBox or ABox axiom. The distance between I and α, denoted $d(I, \alpha)$, is defined as follows:[2]*

$$d(I, \alpha) = \begin{cases} \min_{J \in \mathcal{CM}(\alpha)} d(I, J), & \text{if } \mathcal{CM}(\alpha) \neq \emptyset \\ \tau, & \text{otherwise} \end{cases}$$

where τ is a given as follows:

$$\tau = \begin{cases} |CN||\Delta| + |RN||\Delta|^2 + 1, & \text{if } d(I, J) \text{ is the Hamming distance,} \\ 2, & \text{if } d(I, J) \text{ is the drastic distance.} \end{cases}$$

That is, if α is consistent, then $d(I, \alpha)$ equals the minimal distance between I and the models of α; Otherwise, it equals the given value τ which is strictly larger than any distance between two interpretations. This means that an interpretation is further away from an unsatisfiable axiom than from any satisfiable one. In this way, we will see that compared to satisfiable axioms, an unsatisfiable axiom deviates from being consistent to a larger degree, which is intuitively plausible. The next example further illustrates this intuition.

Example 1 *Let $\alpha = A \sqsubseteq B \sqcap \neg B$, $\alpha' = A \sqcup \neg A \sqsubseteq B \sqcap \neg B$, and $I = (\{a\}, \cdot^I)$ with $A^I = \{a\}, B^I = \{a\}$. We have $d(I, \alpha) = 1$ because $I \notin \mathcal{CM}(\alpha)$ and there is $I' \in \mathcal{CM}(\alpha)$ with $A^I = \emptyset, B^I = \{a\}$ and $d_H(I, I') = 1$. Moreover, $d_H(I, \alpha') = 2 \times 1 + 1 = 3$ since α' is unsatisfiable and $\tau = 3$ with $CN = \{A, B\}$ and $RN = \emptyset$. That is, the unsatisfiable ontology α' deviates from consistency further than α does.*

Given a numeric aggregation function, we can define a distance between an ontology and a classical interpretation as follows:

---

[2] The overloaded notation should not cause any difficulties.

Definition 6 *Given a distance function d and a numeric aggregation function f, let I be a DL interpretation and $O = \{\alpha_1, ..., \alpha_n\}$ be an ontology, where α_i is a TBox or ABox axiom. The distance between I and O, written $\lambda_{d,f}(I, O)$, is defined as follows:*

$$\lambda_{d,f}(I, O) = f(\{d(I, \alpha_1), ..., d(I, \alpha_n)\}).$$

The distance defined above is syntax sensitive which falls into a category of inconsistency measuring approaches that can be useful in some applications as argued in [18].

Definition 7 *(Interpretation ordering w.r.t. distance) Let I_1 and I_2 be two DL interpretations. We say that I_1 is closer to a DL ontology O than I_2 (w.r.t a distance function d and an aggregation function f), written $I \leq_{d,f}^{O} J$, if and only if $\lambda_{d,f}(I, O) \leq \lambda_{d,f}(J, O)$.*

As usual, $I_1 <_{d,f}^{O} I_2$ denotes $I_1 \leq_{d,f}^{O} I_2$ and $I_2 \not\leq_{d,f}^{O} I_1$, and $I_1 \equiv_{d,f}^{O} I_2$ denotes $I_1 \leq_{d,f}^{O} I_2$ and $I_2 \leq_{d,f}^{O} I_1$.

The next definition captures the intuition of our distance-based inconsistency measurement such that the most relevant interpretations of an ontology are those $\lambda_{d,f}$-closest to the ontology.

Definition 8 (Preferred Consistent Interpretation) *The set of* preferred interpretations *of a DL ontology O with respect to a distance function d and an aggregation function f, written $\mathcal{PI}_{d,f}(O)$, is defined as follows:*

$$\mathcal{PI}_{d,f}(O) = \{I : \text{ for any classical interpretation } J, I \leq_{d,f}^{O} J\}.$$

That is, a preferred interpretation has minimal distance to O. When O is consistent, the following proposition holds by noting that $d(I, O) = 0$ iff $I \in \mathcal{CM}(O)$.

Proposition 1 *For any consistent ontology O, $\mathcal{PI}_{d,f}(O) = \mathcal{CM}(O)$.*

The distance between an ontology and its preferred interpretations reflects the distance of the ontology from being consistent. In other words, it represents to what extent it deviates from being consistent. Intuitively, the larger the distance is, the more inconsistent the ontology is. For consistent ontologies, the distance is 0 which says that there is no deviation from being consistent. We normalize this distance in the following definition.

Definition 9 (Inconsistency Deviation Degree) *Given a distance function d and a numeric aggregation function f, the* Inconsistency Deviation Degree *of a DL ontology O, written $IDD_{d,f}(O)$, is defined by:*

$$IDD_{d,f}(O) = \frac{\lambda_{d,f}(I, O)}{\max f(\{x_1, \cdots, x_n\})} \; ,$$

where $I \in \mathcal{PI}_{d,f}(O)$ and $\max f(\{x_1, \cdots, x_n\})$ is given below:

$$\max f(\{x_1, \cdots, x_n\}) = \begin{cases} n\tau, & \text{if } f \text{ is the summation aggregation function,} \\ 1, & \text{if } f \text{ is the voting aggregation function,} \\ \tau, & \text{if } f \text{ is the maximum aggregation function,} \end{cases}$$

where $\tau = 2$ for the drastic distance, and $\tau = |CN||\Delta_0| + |RN||\Delta_0|^2 + 1$ with $|\Delta_0| = \min_{I \in \mathcal{PI}_{d,f}(O)} |\Delta^I|$ for the Hamming distance.

Note that the minimal domain size of preferred models is used as the denominator for normalization in Definition 9. This suffices to make sure that $IDD_{d,f}(O) \in [0,1]$ because all the preferred models have the same distance from O.

Example 2 *(Example 1 contd.) Let $O = \{A \sqsubseteq B \sqcap \neg B, A \sqcup \neg A \sqsubseteq B \sqcap \neg B, A(a)\}$. We have $IDD_{d,f}(O) = \frac{f(0,3,1)}{\max_{x_i \in [0,\tau]} f(x_1,x_2,x_3)} = \frac{4}{9}$ if d is the Hamming distance and f is the summation function by noting that a preferred interpretation of O with the minimal domain size is $I = (\{a\}, \cdot^I)$ with $A^I = \emptyset, B^I = \{a\}$.*

By Proposition 1, the following corollary holds obviously.

Corollary 2 *For an ontology O, we have $IDD_{d,f}(O) \in [0,1]$. Moreover, O is consistent if and only if $IDD_{d,f}(O) = 0$ for any distance function d and aggregation function f.*

3.2 Incoherency Deviation Degree

For description logics, *incoherency* reveals the occurrence of unsatisfiable concepts w.r.t a TBox, that is, it is TBox-relevant but ABox-independent. In this section, we study the distance-based metric for measuring incoherency of a TBox.

Different from the case of measuring inconsistency, to measure incoherency, we put the atomic differences between two interpretations on concept and role names and ignore individual assertions because only a TBox is considered, which leads to a different Hamming Distance given below.

Definition 10 (Hamming Distance) *Let $I_1 = (\Delta, \cdot^{I_1})$ and $I_2 = (\Delta \cup \Delta', \cdot^{I_2})$ be two DL interpretations. The Hamming distance between I_1 and I_2 for incoherency, denoted $\widetilde{d}_H(I_1, I_2)$, is defined as follows:*

$$\widetilde{d}_H(I_1, I_2) = |\{A \in CN : A^{I_1} \neq A^{I_2} \cap \Delta\}| + |\{R \in RN : R^{I_1} \neq R^{I_2} \cap \Delta^2\}|.$$

That is, the Hamming distance of two interpretations for incoherency is the cardinality of the set of concept and role names which are interpreted differently. Unlike the Hamming distance in the case of inconsistency, $\widetilde{d}_H(I_1, I_2)$ is always finite even if Δ is infinite. So when measuring incoherency, we have no need to restrict to finite domains.

The following example shows that the Hamming distances defined for inconsistency $d_H(\cdot, \cdot)$ and for incoherency $\widetilde{d}_H(\cdot, \cdot)$ can have distinct values.

Example 3 *Consider two DL interpretations $I = (\Delta^I, \cdot^I)$ and $I' = (\Delta^{I'}, \cdot^{I'})$ defined as follows: $\Delta^I = \{a,b,c\}, A^I = \{a\}, B^I = \{b,c\}, C^I = \{c\}; \Delta^{I'} = \{a,b,c\}, A^{I'} = \{a\}, B^{I'} = \{b\}, C^{I'} = \{a,b,c\}$. We have $d_H(I,I') = |\{B^I(c), C^I(a), C^I(b)\}| = 3$, whilst $\widetilde{d}_H(I,I') = |\{B,C\}| = 2$.*

For the drastic distance, it remains the same for inconsistency and incoherence. In the rest of this paper, we use $\widetilde{d}(I_1, I_2)$ to refer to either sort of distances whenever there is no necessity to make a distinction. Similarly to the case of measuring inconsistency, we can define the distance between an interpretation I and a TBox axiom.

Definition 11 *Let $I = (\Delta, \cdot^I)$ be a DL interpretation and tt be a TBox axiom. Denote by $\mathcal{CM}(tt)$ the set of classical models of tt, that is, $\mathcal{CM}(tt) = \{I : I \models_2 tt\}$. The distance between I and tt, denoted $d(I, tt)$, is defined as follows:*

$$\widetilde{d}(I, tt) = \begin{cases} \min_{J \in \mathcal{CM}(tt)} \widetilde{d}(I, J), & \textit{if } \mathcal{CM}(tt) \neq \emptyset \\ \widetilde{\tau}, & \textit{otherwise} \end{cases}$$

where $\widetilde{\tau}$ is a given real value which depends on the value range of $\widetilde{d}(I, J)$:

$$\widetilde{\tau} = \begin{cases} |CN| + |RN| + 1, & \textit{if } \widetilde{d}(I, J) \textit{ is the Hamming distance;} \\ 2, & \textit{if } \widetilde{d}(I, J) \textit{ is the drastic distance.} \end{cases}$$

Definition 12 *Given a distance function d and a numeric aggregation function f, let I be a DL interpretation and $T = \{t_1, ..., t_n\}$ be a TBox. The distance between I and T, written $\lambda_{d,f}(I, T)$, is defined as follows:*

$$\widetilde{\lambda}_{\widetilde{d}, f}(I, O) = f(\{\widetilde{d}(I, t_1), ..., \widetilde{d}(I, t_n)\}).$$

For any two DL interpretations I_1 and I_2, we say that I_1 is closer to a TBox T than I_2 (w.r.t. a distance function $\widetilde{d}$ and an aggregation function f), written $I \leq^T_{\widetilde{d}, f} J$, if and only if $\widetilde{\lambda}_{\widetilde{d}, f}(I, T) \leq \widetilde{\lambda}_{\widetilde{d}, f}(J, T)$.

Next we turn to define *preferred interpretations* which capture the intuition of our distance-based incoherency measurement that the most relevant interpretations of a TBox are those $\widetilde{\lambda}_{\widetilde{d}, f}$-closest to the TBox. Note that one of the essential differences to measuring inconsistency is in that the interpretation space, the set of candidate preferred interpretations, consists of interpretations which interpret no concept to the empty set. For ease of notation, denote such an interpretation space by $\mathcal{S} = \{I : \forall A \in CN, A^I \neq \emptyset\}$.

Definition 13 (Preferred Coherent Interpretation) *The set of* preferred interpretations *of a TBox T w.r.t. a distance function $\widetilde{d}$ and an aggregation function f, written $\widetilde{\mathcal{PI}}_{\widetilde{d}, f}(T)$, is defined as $\widetilde{\mathcal{PI}}_{\widetilde{d}, f}(T) = \{I \in \mathcal{S} : \forall J \in \mathcal{S}, I \leq^T_{\widetilde{d}, f} J\}$.*

Example 4 *Let $T = \{A \sqsubseteq B \sqcap D, D \sqsubseteq C, A \sqsubseteq \neg B, D \sqsubseteq \neg C\}$. We know that A, D are two unsatisfiable concepts with respect to T. Consider two interpretations $I = (\Delta^I, \cdot^I)$ and $I' = (\Delta^{I'}, \cdot^{I'})$ with $\Delta^I = \Delta^{I'} = \{a, b, c\}$, $A^I = \{a\}, B^I = \{a\}, C^I = \{a, b, c\}, D^I = \{a, c\}$, and $A^{I'} = \emptyset, B^{I'} = \{a\}, C^{I'} = \{a, b, c\}, D^{I'} = \{c\}$. We have $\widetilde{\lambda}_{d,f}(I', T) \leq \widetilde{\lambda}_{d,f}(I, T)$. However, we have $I \in \widetilde{\mathcal{PI}}_{\widetilde{d}, f}(T)$, but $I' \notin \widetilde{\mathcal{PI}}_{\widetilde{d}, f}(T)$ because it assigns A to the empty set. Another preferred model of T can be $J = (\{a\}, \cdot^I)$ with $A^I = B^I = C^I = D^I = \{a\}$. By a careful computation, we obtain $\widetilde{\lambda}_{\widetilde{d}, f}(I, T) = \widetilde{\lambda}_{\widetilde{d}, f}(J, T) = f(0, 0, 1, 1)$.*

Proposition 3 *For any coherent TBox T, we have $\widetilde{\mathcal{PI}}(T) = \mathcal{CM}(T)$, where $\mathcal{CM}(T)$ is the set of classical models of T. For an incoherent TBox T, $\widetilde{\mathcal{PI}}(T) \cap \mathcal{CM}(T) = \emptyset$.*

Similarly to the definition of inconsistency deviation degree, we can define the *incoherency deviation degree* of a TBox which measures to what extent it deviates from being coherent.

Definition 14 (Incoherency Deviation Degree) *Given a distance function $\widetilde{d}$ and a numeric aggregation function f, the Inconsistency Deviation Degree of a TBox T, written $\widetilde{IDD}_{\widetilde{d},f}(T)$, is defined as follows:*

$$\widetilde{IDD}_{\widetilde{d},f}(T) = \frac{\widetilde{\lambda}(I,T)}{\max f(\{x_1,\cdots,x_n\})}, \tag{1}$$

where $I \in \widetilde{\mathcal{PI}}_{\widetilde{d},f}(T)$ and $\max f(\{x_1,\cdots,x_n\})$ is given in Definition 9 by replacing τ by $\widetilde{\tau}$.

Example 5 *(Example 4 contd.) For T, we have known that $I \in \widetilde{\mathcal{PI}}_{\widetilde{d},f}(T)$, by which we have $\widetilde{IDD}_{\widetilde{d},f}(T) = \frac{\lambda_{\widetilde{d},f}(I,T)}{\max f(\{x_1,\cdots,x_n\})} = \frac{f(0,0,1,1)}{5\times 4} = \frac{1}{10}$ when $\widetilde{d}$ is the drastic distance and f is the summation function, where $|CN| = \{A,B,C,D\} = 4, \widetilde{\tau} = 5$.*

Example 6 *Let $T_1 = \{C_i \sqsubseteq \bot : i \in [1,n]\}$ and $T_2 = \{C_i \sqsubseteq C_{i+1}\} \cap \{C_n \sqsubseteq \bot\}$. Suppose $I = (\Delta^I, \cdot^I)$ with $\Delta^I = \{a\}$ and $C_i^I = \{a\}$; We have $I \in \widetilde{\mathcal{PI}}_{\widetilde{d},f}(T_1)$ and $I \in \widetilde{\mathcal{PI}}_{\widetilde{d},f}(T_2)$. We have $\widetilde{\lambda}(I,T_1) = f(\widetilde{\tau},...,\widetilde{\tau})$ and $\widetilde{\lambda}(I,T_2) = f(\underbrace{0,...,0}_{n-1},\widetilde{\tau})$ such that $\widetilde{\lambda}(I,T_1) > \widetilde{\lambda}(I,T_2)$. This meets the intuition that T_1 contains more incoherence "resources" (unsatisfiable concepts) than T_2 does.*

Corollary 4 *For any TBox T, $\widetilde{IDD}_{\widetilde{d},f}(T) \in [0,1]$. Moreover, T is coherent if and only if $\widetilde{IDD}_{\widetilde{d},f}(T) = 0$ for any distance function $\widetilde{d}$ and aggregation function f.*

3.3 Inconsistency and Incoherency Ordering

An application of measuring inconsistency or incoherency is to order inconsistent ontologies and incoherent terminologies to assist ontology engineering. In this section, we provide a distance-based inconsistency and incoherency ordering.

Definition 15 (Distance-based Inconsistency/Incoherence Ordering) *Given two ontologies $O = \{\alpha_1,...,\alpha_n\}$ and $O' = \{\alpha'_1,...,\alpha'_m\}$ (resp. TBoxes $T = \{t_1,...,t_n\}$ and $T' = \{t'_1,...,t'_m\}$), w.l.o.g, assume $m \leq n$. We saz that O is less inconsistent than O' (resp. T is less incoherent than T') w.r.t. ς, written $O \leq_{Inconsist} O'$ (resp. $T_1 \leq_{Inconher} T'$), iff there exist preferred consistent interpretations I of O (resp. T) and I' of O' (resp. T') such that $IDD_{d,f}(O) \leq IDD_{d,f}(O')$ (resp. $\widetilde{IDD}_{\widetilde{d},f}(T) \leq \widetilde{IDD}_{\widetilde{d},f}(T')$).*

Example 7 *(Example 2 contd.) Let $O' = \{A \sqsubseteq B \sqcap \neg B, A(a)\}$. We have $IDD_{d,f}(O') = \frac{f(0,1,0)}{\max_{x_i \in [0,\tau]} f(x_1,x_2,x_3)} = \frac{1}{9}$ if d is the Hamming distance and f is the summation function by noting that a preferred interpretation of O is $I' = (\{a\}, \cdot^{I'})$ with $A^{I'} = \emptyset, B^{I'} = \{a\}$. So $IDD_{d,f}(O') <_{Inconsist} IDD_{d,f}(O)$.*

Example 8 *(Example 4 contd.) Let $T' = \{A \sqsubseteq B \sqcap C, B \sqsubseteq \neg C, A \sqsubseteq D\}$. We know that A, B are unsatisfiable concepts with respect to T. Consider $J = (\Delta^J, \cdot^J)$ with $\Delta^J = \{a, b, c\}$ and $A^J = \{a\}, B^J = \{a, b\}, A^J = \{a, b, c\}, D^J = \{c\}$. We have $\lambda_{H,f}(J, T') = f(0, 0, 1, 0) = 1$ for drastic distance function and summation aggregation function. Since T' is incoherent, there is no $J' \in \mathcal{S}$ such that $\lambda_{H,f}(J', T') < \lambda_{H,f}(J, T')$. Therefore, $J \in \mathcal{PI}(T')$. By noting that $\lambda_{H,f}(J, T') < \lambda_{H,f}(I, T) = f(0, 0, 1, 1)$, we have that T' is less incoherent than T.*

3.4 Comparison of Aggregation Functions

Above, we have given a framework for defining the inconsistency deviation degree and the incoherency deviation degree based on some given distance function and aggregation function. In this section, by the following example, we make a comparison of aggregation functions discussed in this paper. The conclusion is that the summation aggregation function is better for distinguishing ontologies (resp. terminologies) in terms of their different inconsistency (resp. incoherency) degrees.

Example 9 *Let $O = \{A \sqsubseteq B \sqcap \neg B, A(a)\}$ and $O' = \{A(a), \neg A(a), B(a), \neg B(a), C(a), \neg C(a)\}$. Consider $\Delta = \{a\}$ and two interpretations I with $A^I = \emptyset, B^I = \{a\}, C^I = \emptyset$ and I' with $A^{I'} = \emptyset, B^{I'} = \{a\}, C^{I'} = \{a\}$. We have $I \in \mathcal{PI}(O)$ and $I' \in \mathcal{PI}(O')$. Moreover, $\lambda_{\{H,D\},f}(I, O) = f(\{0, 1, 0, 0, 0, 0\}), d_{\{H,D\},f}(I', O') = f(\{0, 1, 1, 0, 1, 0\})$. Then the following hold.*

- *If f is the maximum function, then $\lambda_{\{H,D\},f}(I, O) = \lambda_{\{H,D\},f}(I', O') = 1$;*
- *If f is the voting function, then $\lambda_{\{H,D\},f}(I, O) = 0, \lambda_{\{H,D\},f}(I', O') = \frac{1}{2}$ if $\frac{k}{m} \geq 0.5$, otherwise, $\lambda_{\{H,D\},f}(I', O') = 0$;*
- *If f is the summation function, then $\lambda_{\{H,D\},f}(I, O) = 1, \lambda_{\{H,D\},f}(I', O') = 3$.*

That is, O has the same inconsistency as O' under the maximum function and the voting function (with $\frac{k}{m} \geq 0.5$ in this example). But with the summation function, we obtain that O is less inconsistent than O' which coincides with the intuition.

From this example, we can see that, compared to the maximum function and the voting function, the summation function allows for a larger range of distinctive values of the distance between an ontology and its preferred interpretations such that it better satisfies the variation principle than the other two aggregation functions.

4   Conclusion and Future Work

We studied a distance-based framework to define inconsistency measures and incoherency measures which can be used for ranking inconsistent ontologies and incoherent terminologies. We showed that such measures met the *normalization, variation, applicability* principles. In the future, we intend to study other distance functions like Hausdorff distance and other aggregation functions such as the averaging function. More importantly, we intend to develop algorithms for computing our distance-based measures and investigate them in practice.

References

1. Jain, P., Hitzler, P., Yeh, P.Z., Verma, K., Sheth, A.P.: Linked Data is Merely More Data. In Brickley, D., Chaudhri, V.K., harry halpin, McGuinness, D., eds.: Linked Data Meets Artificial Intelligence, AAAI Press, Menlo Park, CA (2010) 82–86
2. Baader, F., Calvanese, D., McGuinness, D., Nardi, D., Patel-Schneider, P., eds.: The Description Logic Handbook: Theory, Implementation, and Applications. Cambridge University Press (2003)
3. Hitzler, P., Krötzsch, M., Rudolph, S.: Foundations of Semantic Web Technologies, Textbooks in Computing. Chapman and Hall / CRC Press (2009)
4. Horrocks, I.: Ontologies and the Semantic Web. Communications of the ACM **51** (2008) 58–67
5. Huang, Z., van Harmelen, F., ten Teije, A.: Reasoning with inconsistent ontologies. In: Proc. of 19th International Joint Conference on Artificial Intelligence(IJCAI'05). Morgan Kaufmann (2005) 254–259
6. Ma, Y., Hitzler, P., Lin, Z.: Algorithms for paraconsistent reasoning with OWL. In Franconi, E., Kifer, M., May, W., eds.: Proceedings of the 4th International Semantic Web Conference(ESWC'07). Volume 4519 of Lecture Notes in Computer Science., Springer (2007) 399–413
7. Schlobach, S.: Diagnosing terminologies. In: Proc. of AAAI'05. (2005) 670–675
8. Parsia, B., Sirin, E., Kalyanpur, A.: Debugging OWL ontologies. In: Proc. of WWW'05. (2005) 633–640
9. Schlobach, S., Huang, Z., Cornet, R., van Harmelen, F.: Debugging incoherent terminologies. J. Autom. Reasoning **39** (2007) 317–349
10. Halaschek-Wiener, C., Katz, Y.: Belief base revision for expressive description logics. In Grau, B.C., Hitzler, P., Shankey, C., Wallace, E., eds.: OWLED. Volume 216 of CEUR Workshop Proceedings., CEUR-WS.org (2006)
11. Qi, G., Du, J.: Model-based revision operators for terminologies in description logics. In Boutilier, C., ed.: Proceedings of the 21st International Joint Conference on Artificial Intelligence (IJCAI'07). (2009) 891–897
12. Ma, Y., Qi, G., Hitzler, P., Lin, Z.: Measuring inconsistency for description logics based on paraconsistent semantics. In Mellouli, K., ed.: Proceedings of the 9th European Conference on Symbolic and Quantitative Approaches to Reasoning with Uncertainty. Volume 4724 of Lecture Notes in Computer Science., Springer (2007) 30–41
13. Qi, G., Hunter, A.: Measuring incoherence in description logic-based ontologies. In Aberer, K., Choi, K.S., Noy, N.F., Allemang, D., Lee, K.I., Nixon, L.J.B., Golbeck, J., Mika, P., Maynard, D., Mizoguchi, R., Schreiber, G., Cudré-Mauroux, P., eds.: ISWC/ASWC. Volume 4825 of Lecture Notes in Computer Science., Springer (2007) 381–394
14. Zhou, L., Huang, H., Qi, G., Ma, Y., Huang, Z., Qu, Y.: Measuring inconsistency degrees of *DL-Lite* ontologies. In: Proceedings of 2009 IEEE/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology (WI/IAT-09). (2009) 349–356
15. Arieli, O.: Distance-based paraconsistent logics. J. Approx. Reasoning **48** (2008) 766–783
16. Arieli, O., Zamansky, A.: Non-deterministic distance semantics for handling incomplete and inconsistent data. In Sossai, C., Chemello, G., eds.: ECSQARU. Volume 5590 of Lecture Notes in Computer Science., Springer (2009) 793–804
17. Willard, S.: General Topology. Addison Wesley (2970)
18. Hunter, A., Konieczny, S.: Approaches to measuring inconsistent information. In Bertossi, L.E., Hunter, A., Schaub, T., eds.: Inconsistency Tolerance. Volume 3300 of Lecture Notes in Computer Science., Springer (2005) 191–236

Logic for Modeling Product Structure

Henson Graves

Lockheed Martin Aeronautics Company
Fort Worth Texas, USA
henson.graves@lmco.com

Abstract. *A fragment of type theory with OWL class constructions for types and binary properties is used to formalize SysML Structural Block Diagram models. A structural SysML block diagram model is a model that does not have behavior in the sense that values of properties do not change. A structural model may include properties, variables and operations. Individuals are a special case of operators with no arguments. Type theory is chosen as the target semantic formalism as SysML constructions correspond closely to type theory term constructions. The type theoretic semantics defined in terms of introduction and elimination fits well with the informal SysML semantics. An abstract version of a structural model, called an Abstract Block Diagram (ABD), is introduced. An ABD is a theory closed under specific type, property, and operator constructions with additional axioms. The ABD corresponding to the SysML model contains axioms in the form of equations for types, and properties, and operators. This formalism captures the syntactic constructons of the SysML models and the type theoretic semantics appears to be in accord with the informal semantics, as documented in the OMG specification. ABD theory gives an explicit mechanism for introducing instances for types defined by property restrictions. This construction is useful for parts decompositions. ABD theory constructions have a limited kind of property union used to construct parts decompositions. An ABD determines a Description Logic (DL) closed under union, intersection, and existential type constructions and property constructions restricted by typing relations. The ABD constructions are useful in identifying potential extensions for SysML and may be useful, as well, for adding operator terms to Description Logic.*

Keywords: Description Logic, Ontology, OWL, Product Model, SysML, Type Theory, UML.

1 Introduction

The systems engineering language SysML [3] is a natural starting point for developing a formal logic for product modeling. SysML is sufficiently expressive to represent complex product structure such as occurs in aircraft and automobiles [4], has a graphical syntax, engineers can use it, and it has good commercial tool support. There is no other language in this category. The SysML graphical syntax uses several kinds of diagrams which are all views of a single SysML model. Providing SysML

with a formal semantics allows engineers to work with the tools they use today and apply formal reasoning to the results.

A fragment of type theory with OWL class constructions for types and binary properties is used to formalize SysML Structural Block Diagram models. A structural SysML block diagram model is a model that does not have behavior in the sense that values of variables do not change. A structural model may include variables and operations. Type theory is chosen as the target semantic formalism as SysML constructions correspond closely to type theory term constructions. The semantics of a type theory, presented in terms of introduction and elimination rules, is close to the informal semantics of SysML. An abstract version of a structural model, called an Abstract Block Diagram (ABD), is introduced. An ABD is a theory closed under specific type, property, and operator constructions with additional axioms. The ABD corresponding to the SysML model contains axioms defined in terms of equations for type, and properties, and operators.

The constructions in an ABD are needed for modeling product structural properties such as parts decompositions and (horizontal) relationships between product components. In an ABD each property P is typed with a domain and range type. ABD properties are closed under composition, inverse, restriction, and union, provided the typing conditions are met. The use of typed properties enables an exposition of typed "parts" properties [19,20,21]. The type theory semantics provides an explication and semantics for the "dot" notation to provide a fully qualified name for a part whose existence is guaranteed by an existential restriction. For example for a typed property P(A,B) and an individual a:A, the ABD contains a term a.P which has type B. Operators while not fully exploited in static models are never the less useful in that they allow value properties to be defined on a type algebraically. For example, the weight can be define as the sum as the weight of the product's components.

OWL2 [16] has been used as a formalism for capturing the structural part of a block diagram model [8,9,10]. The correspondence between SysML and OWL constructions is well known [16]. While SysML does not have all of the type constructions found in OWL2, these constructions are needed to capture the semantics of a SysML model. SysML properties are typed with a domain and range types; they can be represented as OWL2 properties with axioms which express that the domain and range properties. Information regarding subtype and equality relations between types implicit in SysML models is translated into OWL2 axioms [10]. A structural SysML model without operators and variables can be translated into an OWL KB in a semantics preserving way. However, operators are not included in OWL2. The translation into OWL KB requires explicit axioms for the domain and range classes for each property in the SysML model.

However, a type theory with type constructions closely modeled on OWL2 class constructions and typed property constructions provides SysML with semantics which is not currently supplied by OWL2. The ABD formalism simplifies the characterization of those SysML models which represent structural SysML models. ABD constructions include typed operators and variables as they are used in SysML. The ABD formalism suggests possible extensions for SysML block diagram models and a method for adding operators with variables to Description Logics.

2 Abstract Block Diagrams

An Abstract Block Diagram (ABD) consists of a finite set of basic type, property, and operator symbols with constructions for each kind of term. Property and operator terms symbols have type signatures. The syntax of an ABD theory is characterized by a recursive definition, in which the constructors that can be used to form terms are stated. The ABD types are closed under intersection, union, finite enumeration types, existential restriction, and have type constants for top and bottom types. A property symbol P has a type signature P(A, B) where A and B are types. The ABD properties are closed under property composition, inverse, restriction, and union, provided the typing conditions are met. An operator symbol has a signature $f(A1,…,An):B$ where $A1,…,An$, B are types; operators may have variables are closed under composition, and variable substitution. Types are closed under products. An operator f() with no arguments is identified with an individual. Constructors for forming products, tuples and case properties are used.

Type, property, and operator terms have rules for determining which equations are valid. The operator and property terms are recursively constructed the signature using term constructors introduced below. Formulas within an ABD theory are constructed from equation, subtype, subproperty, and operator type relations. An ABD may have as axioms equations and typing assertions.

Each inference rule is depicted as a fraction; the inputs to the rule are listed in the numerator, and the output in the denominator. The inputs to the rules may be terms or other theorems. Inference rules state that equality between terms and types is reflexive and transitive. Rules of inference allow one to substitute new terms for the free variables in a theorem and allow one to substitute new types for the type variables in a theorem. The inference rules provide mechanisms for defining new constants and new types. The usual presentation of introduction rules within logic uses a line to separate an antecedent condition above a line and the term introduced below the line. In the following A, B, C, and D are types and P, Q, and R are properties, and lower case letters are operators.

2.1 Type constructions

ABD types and individuals have a tuple construction. For any types $A1,…,An$ and any individuals $a1,…,an$ with ai: Ai, the expression $(A1,…,An)$ is a type and $(a1,…an)$ is an individual with $(a1,…an):(A1,…An)$. We write proji for the projection constructors defined for products which project an individual t with $t:(A1,…An)$ onto the ith coordinate. ABD types have a finite union construction with case constructors. ABD types are closed under finite enumeration types which will be written as $\{a1,…,an\}$ including integers. For a finite sequence of types $A1,…An$, the expression $Union(Ai,0,..n)$ is a type. We write case for the properties which have typing $case(Union(Ai,0,..n), \{0,…,n\})$. For a property P with P(A,B) and C subtype B, then (P some C) is a type with (P some C)(A,C). The rules for existential types and numeric restriction types are accompanied by an individual introduction rule which uses a "dot" notation. The expression a.P is an individual with a.P:C. ABD provides a "dot" notation and semantics for ``fully qualified names''. The "dot" constructor is

used to introduce terms dependent on an individual. When P(A,B) the abbreviation (P exactly 1) is used for (P exactly 1 A). The inference rule for (P exactly 1) introduces for an individual a with a:A an individual a.P1. Similarly the notation a.Pk is used for (P exactly k).

Inference Rules		
Thing	a: A ------------ a : Thing	
NoThing	a: NoThing --------------- a: A	
Enumeration	**a1:A, …,an:A** ------------------------------------- **{a1,…an} Subtype A,ai:**	**b:{a1,…an}** ----------------------- **b = ai:A, for some i**
A and B	**a:A, a:B** ---------------- **a: (A and B)**	**a: (A and B)** ----------------- **a:A, a:B**
A Subtype B is defined as	**(A and B) = A**	
Union(Ai, 0,…n)	**a: A** ------------------------ **a: Union(Ai, 1,…n)**	**a: Union(Ai, 1,…n),** **<a,i>: case, for some i** ------------------------------- **a: Ai**
case	---------------------------------- **case:(Union(Ai,1,…n), {0,…,n})**	
tuple	**t:(A1,…,An)** -- **Proji(t):Ai, t = (proj1(t),…,projn(t)**	**a1 :A1,...,an:An** --------------------------- **(a1,…,an):(A1,…,An)**
P some C	**P(A,B) C subtype B,** **a:A, c:C, (a,b):P** ----------------------- **a:(P some C)**	**P(A,B) C subtype B,** **a:(P some C)** ----------------------- **a.P1:B, (a, a.P1):P**
P exactly k C	**P(A,B) C subtype B,** **a:A, c:C, (a, c):P** --------------------- **a:(P exactly k C)**	**P(A,B) C subtype B,** **a:(P exactly k)** ----------------------- **a.Pk:B, (a, a.P):P**
A disjoint B is defined as	**A intersection B = NoThing**	

2.2 Property constructions

ABD properties are closed under composition, inverse, restriction, and union, provided the typing conditions are met. Properties are used to represent parts properties as well as properties such as the property that an engine in a vehicle drives the front wheels of the vehicle that the engine is part of.

Property Instances	t:P(A,B) --- first(t):A and second(t):B, t = (first(t),second(t)):P.	
Composition	P(A,B), Q(B,C) ---------------------- QoP(A,C)	(a,b):P, (b,c):Q ---------------------- (a,c): QoP
Inverse	t:P(A,B) ----------------- (second(t),first(t)):P*	
Intersection	P(A,B), Q(C,D) ------------------------------------ (P and Q)(A and C, B and D)	(a,b):P and (a,b):Q ----------------------- (a,b):(P and Q)
subproperty is defined as	(P and Q) = P	
Property Restriction	P(A,B), A1 sub A, B1 sub B ------------------------------- P|A1,B1(A1,B1)	t:P, first(t):Ai, second(t):Bi -------------------------------- t :P|A1,B1 and conversely
Property Union	Pi(A,Bi), i:{0,…n} -------------------------------- Union(Pi)(A,Union(Bi), i:{0,…n})	t:Pi -------------------------------- t: Union(Pi, i:{0,…n})

Note that any ABD type defined by restriction properties which are unions or restriction properties does not introduce any new types. For example, consider a restriction class (P some C) where P = Union(P1,P2) is a union property. Then

          a:Union(P1,P2) some C

and

          a: (P1 some C) or a:(P2 some C)

so

          a: (P1 some C) union (P2 some C).

2.3 Parts decomposition structure

The informal concept of a parts decomposition structure is made precise using a collection of typed properties called a decomposition structure. In the informal concept a decomposition structure of a product is specified by a product design. The design specifies a root class in a parts decomposition and what parts are necessary to have a product. In this concept specific part instances may be replaced by other instances of the required type. An individual instance of the root type has a parts decomposition which determines the type of specific parts and may determine the number of parts provided the parts properties specify exact cardinality. The existence of an instance of the root depends on the existence of the parts in its decomposition. However, individual parts may be replaced, and so extensionality does not in general hold for a parts decomposition. The individuals in a parts decomposition are irrefliexive and antisymmetric. Product identity is generally defined in terms of a

unique identification number. The concept of a detailed design is that any two individuals of root type, i.e., a1:Root, a2:Root have the same parts decomposition.

A parts decomposition structure for an ABD is defined as a family of (typed) properties $\mathscr{P}$ for which the signature $\mathscr{S}$ of the ABD and the family $\mathscr{P}$ is an irreflexive, antisymmetric, acyclic, connected graph. The types that occur in the typing of the P in $\mathscr{P}$ are assumed to be disjoint. If P(A,B) is in $\mathscr{P}$ there is no P1 in $\mathscr{P}$ with P1(A,A) and no P1(B,A). Since there is exactly one P(A,B) for each pair of types A and B in the signature of the ABD, we can, by abuse of notation, use the same symbol for each P in the family $\mathscr{P}$. If the S and P form a tree with root V, then the ABD is called a design. The concept of parts decomposition structures can be used to characterize the ABD theories which describe designs in the sense that they have a well defined parts decomposition. An ABD may have multiple parts decomposition structures. We use a decimal index notation P(i…j) for these compositions obtained by starting at the root. For a design ABD with a parts decomposition $\mathscr{S}$ and $\mathscr{P}$ we add the axioms that

Root Subtype (P(1) some B) and (P(1.1) some C1) and ….

A number of questions about parts decompositions while not expressible within an ABD theory can be answered regarding a parts decomposition structure. For a design ABD with root V the parts for a design instance v of V are represented as an enumeration class

{v, v.P(1),…,v.P(i….j),…}

where the P(i….j) are the compositions of are properties in the parts decomposition structure with V as the root. The parts decomposition is a tree, the cardinality of the enumeration set is the number of parts in an implementation. Since each part has a typing, the number of distinct types used by the decomposition can be determined as can the number of occurrences of a given type. For an arbitrary property the collection of individuals reachable from a given individual can be determined. A detailed design is a design ABD which does not use any subclass axioms between the basic types and all of the parts properties have an exact numeric restriction. All of the parts decompositions for a detailed design have the same graph structure.

The union construction can be used to define the property which is the union of composition of parts properties within a parts decomposition whose starting point is the root. Two parts decompositions of a root can be made disjoint by adding an axiom that insures that if an individual is in two decompositions then the instances of the root are equal. The restriction construction can be used to start with a property such as Drives within a vehicle ABD and define restrictions such as Drives|Vehicle,Engine. For example, a vehicle ABD may want to represent a drive property for an engine that represents both driving wheels and driving a generator. Both drivesFrontWheel and hasFrontWheel are subproperties of Drive.

2.4 Variables and Operators

Having operators and variables is useful even in the absence of an ABD having behavior is useful. It enables a type to describe that its instances have properties such as weight without having to bind the variables. The model theoretic semantics of an operator term with variables is defined is defined as a function defined on product domains. An operator symbol f has a type signature f(A1,…,An):B where A1,…,An, B are types. An operator symbol a():B is written a:B. Operator terms are constructed using the constructions in the table below.

Operator Expression Constructions	Syntax	
hasOperator	hasOperator(A,B)	Parts property structure used to introduce operators and associate them with an individual
hasVariable	hasVariable(A,B)	Parts property structure used to introduce variables and associate them with an individual
Variable declaration	x1: C1,…xn : Cn	The symbols x1,…xn are variables and xi is said to have type Ci
Operator Declaration	f(x1: C1,…xn : Cn):B	The symbols x1,…xn are variables and xi is said to have type Ci
Composition	f(g1,…,gn)	Where f has arity n
Substitution	t[t1/x1,…,tn/xn]	Replaces distinct variables with operator expressions
Tuple	(t1,…,tn)	Where t1,…tn are operator expressions

In an ABD theory variables (SysML value properties) and operators are all introduced using the same dot construction as is used for parts properties. An operator is always declared as belonging to an individual a. The association of variables to an individual provides a state description for the individual. For example, the state of a vehicle, *v1,* is a list of attributes (variables) of *v1* and of its components. The value of the vehicle state is a substitution or binding of values to variables. Substitution of values for variables provides the foundation for the concept of ``evaluating" the state of an individual. Usual rules for substitution of variables by terms, equality, and typing statement hold. We will write *a.x* for the term a.hasVariable1. For two instances a1 and a2 of A, note that a.x and a2.x are distinct variables. The sequence (x1, x2, f1,f2) of parts of an instance a of A is represented by

        (a.x, a.y, a.f)

which is equivalent to

    a.(x:X, y:Y, f(x:X):Y)

using the rules for tuples.

4 The correspondence between an ABD and a SysML Block Diagram

The graphical syntax of a SysML block diagram model identifies a collection of basic symbols sorted into types, properties, and operations, called the signature of the block diagram. The graphical representation of a SysML block diagram model uses rectangles for blocks (types). A directed line between P between two blocks, A and B is a property with domain A and range B. SysML uses properties to represent connections between blocks. A SysML structural model can be used to construct an ABD. The ABD starts from the signature of types, properties, and operators in the structural model. SysML employs several "parts" properties that satisfy the properties of a parts structure as defined above.
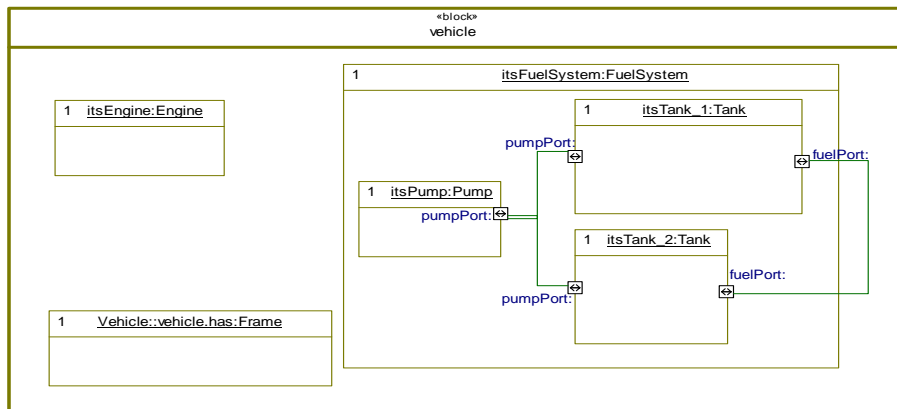

Fig. 1 The Internal Block Diagram illustrates a simplified specification for a Vehicle. The diagram uses blocks: Vehicle, Engine, Frame, FuelSystem, Pump, Tank, and properties itsEngine, itsFrame, itsFuelSystem, itsPump, itsTank1, and itsTank2. This diagram also has flowports with associations between them. The ports are distinct parts of the blocks that they are attached to. The diagram specifies that a vehicle has exactly 1 FuelSystem and a FuelFsystem has exactly two tanks. The tanks are connected through ports with a specific connection. The itsTank1 and itsTank2 are properties that specify that the fuel system has two tanks.

In an internal block diagram, such as Figure 1, the number and kind of parts of a block are described by the parts decomposition structure. itsFuelSystem is one of the part properties in this structure. The range type of itsFuelsystem is the type Fuelsystem, and the number of instances that satisfy itsFuelsystem for a vehicle instance is 1. Multiple occurrences of a rectangle with a given block are used to specify multiple occurrences for a part of an individual instance of the enclosing block. In Figure 1 the block containment relationships are defined using part properties. In the header of a block on the diagram such as itsEngine:Engine, itsEngine is a property with domain Vehicle and range Engine.

The semantics of the vehicle system block diagram model is defined in terms of its parts and their connections. The informal notion of a vehicle implementation is a parts decomposition and a description of how the parts fit together. In Figure 1 a vehicle

instance v has a parts decomposition consisting of an engine, a frame, a fuel system, where the fuel system has three parts.

If a SysML structural model has an arrow P connecting A to B then the ABD has a property P(A,B) with a numeric restriction k. While SysML does not have a restriction type construction the ABD restriction constructions is used to capture the meaning of the property within the context of a SysML model. The meaning of a block diagram arrow P drawn from A to B with a restriction of 1 on B is that any instance of A has only one P connection to B. this can be represented using the axiom

A SubType (P exactly 1)

In this approach block diagrams translate directly into subtype relations. The axiom that A is a subtype of the restriction (P exactly k) captures the meaning that for any a: A then there are k instances of B implied by the property with (a,bi):P, for i:{1,..,k}. The restriction construction can be used to represent user defined relations. The explicit interpretation of ABD type and properties captures the informal, semantics of a SysML structural block diagram model.

5 The Correspondence between an ABD, Description Logics, and axiomatizations of higher order logic in FOL

ABD type constructions correspond directly to DL constructions while not all DL class constructions are used. The property constructions contain standard DL property constructions which are restricted by property typing rules. However, the use of ABD property construction in defining types does not add any new types to the DL. As with Description Logics in general not all of the constructions are logical constructions in FOL. Some constructors are related to logical constructors in first-order logic (FOL) such as intersection or conjunction of concepts, union or disjunction of concepts, negation or complement of concepts, universal restriction and existential restriction. Other constructors have no corresponding construction in FOL including restrictions on roles for example, inverse, transitivity and functionality. However, the full set of constructions can be axiomatized within a multi-sorted FOL where the sorts are type of the ABD theory. The FOL axiomatization of ABD is similar to a FOL axiomatization of Cartesian closed category.

For each type A and each Property P of the ABD the FOL generated by the ABD contains a unary predicate and a binary predicate. By abuse of notation we use the same symbol for both the type and predicate. The context of use will make the usage clear.

A	For any a. A(a)
P	For any a,b. P(a,b)
a:A	There exists x. A(x)
a:(A and B)	(A(a) and B(a))
A = B	For any a. A(a) implies B(a) and for any a. B(a) implies A(a)
a:(P exactly 1)	For any a. there exists a unique b with P(a.b) and B(b)
t:P(A,B)	P(first(t),second(t)) and A(first(t)) and B(second(t))

In the correspondence all judgments derivable from the ABD axioms are provable in the generated FOL theory. Conversely, any theorem provable in the FOL using the logical axioms is a derivable judgment in the ABD. However, the ABD theory provides an explicit term constructions for restriction types rather than just an existence statements. A model for an ABD is defined in the same way as a model for the FOL theory using the axioms for the term constructions. Any non-empty model of a design ABD will contain an implementation of the root in the sense that any model will contain a parts decomposition of the root.

6 Conclusions

The ABD term constructions represent constructions needed for structural modeling of products. These constructions are used within SysML block diagrams. An ABD is an abstraction of SysML structural block diagrams. An ABD can be generated from the signature of the SysML model. ABD has both a type theoretic semantics defined in terms of the inference rules that appear to be in accord with the informal SysML semantics. A model theoretic semantics can be defined as well. The importance of establishing a logical formalism for SysML is that the reasoning required in engineering tasks for design and analysis can be formalized. Automated reasoning techniques can be employed or at least arguments can be automatically checked. The translation of a restricted SysML Block Diagram model into an ABD which preserves the intended semantics is a first step toward integrating product development with formal reasoning. The result provides a formal semantics for a fragment of SysML. Conversely, SysML graphical syntax can be used to develop ABDs.

ABDs can be used to represent parts and connectivity structure. An ABD with a root class under a part decomposition structure has a parts decomposition graph. The parts decomposition structure does not depend on restricting the models as is done in the Description Graph extension to OWL2. An ABD can be used to answer questions such as what parts two designs have in common, and what kinds of transformations and parts replacements to an implementation produce a valid implementation. For product development one would like to characterize ABDs (SysML models) all of whose models have some predetermined similarities. The realization of these goals requires a much richer logical system than has been presented here. However, in order to assess the impact of changing a part in a design on the properties of the design, one needs to be able to define value properties using variables which are recomputed as the value of the variables change. For example, one wants to define the total weight of a product as the sum of the weights of its components and have the total weight change as the parts are changed. An extended version of the type theory presented is a candidate formalism for SysML. Further, an ABD can be viewed as a DL which is sufficient for representing product structure. The next step in the development of the ABD formalism is to add behavior in the form of state charts.

References

1. Bock, C., *UML Composition Model*, Journal of Object Technology, vol. 3, no. 10, November-December 2004, pp 47-73.
2. Barendregt, H., *Handbook of Logic in Computer Science*, volume 2, Oxford University Press, 1992.
3. Friedenthal, S., Moore, A., and Steiner. F., OMG Systems Modeling Language (OMG SysML™) Tutorial, INCOSE Intl. Symp, 2006.
4. Graves, H., Guest, S., Vermette, J., and Bijan, Y., *Air Vehicle Model-Based Design and Simulation Pilot*, Spring 2009 Simulation Interoperability Workshop (SIW)
5. Graves, H. *Design refinement and requirements satisfaction*, Proceedings of 9th NASA-ESA Workshop on Product Data Exchange, 2007.
6. Graves, H., *Ontology engineering for product development*, Proceedings of the Third OWL Experiences and Directions Workshop, 2007. Available at www.webont.org/owled/2007/PapersPDF/submission_3.pdf
7. Graves, H., Horrocks, I., *Application of OWL 1.1 to Systems Engineering*, OWL Experiences and Directions April Workshop, 2008.
8. Graves, H., *Representing Product Designs Using a Description Graph Extension to OWL 2*. OWL Experiences and Directions October Workshop, 2008.
9. Graves, H., Leal, D., *Using OWL 2 For Product Modeling*, Proceedings of 11th NASA-ESA Workshop on Product Data Exchange, 2009.
10. Graves, H. *Integrating SysML and OWL*. OWL Experiences and Directions October Workshop, 2009.
11. Haley, T., Friedenthal, S., *Assessing the application of SysML to systems of systems simulations*, Proceedings of the Spring Simulation Interoperability Workshop, September, 2008.
12. Harrison, J. Formal Verification at **Intel**, presentation June 2002.
13. Howard, W., To H.B. Curry: *Essays on Combinatory Logic, Lambda Calculus and Formalism*. Academic Press, 1980.
14. Martin-Lof, P., constructive mathematics and computer programming. *Logic, Methodology and Philosophy of Science*, 1982.
15. OMG Systems Modeling Language (OMG SysML™), V1.1, November 2008.
16. OMG Formal Ontology Definition Metamodel
17. OWL 2 Web Ontology Language, W3C Working Draft 11 June 2009.
18. Sattler, U., A concept language for an engineering application with part-whole relations. Proceedings of the International Workshop on Description Logics, 1995.
19. Artale, A., Franconi, E., Guarino, N., Pazzi, L., Part-Whole Relations in Object-Centered Systems: An Overview, Data and Knowledge Engineering 20, North-Holland, Elsevier, 1996.
20. Barbier, F., Henderson-Sellers, B., Parc-Lacayrelle, A., Bruel, J., Formalization of the Whole-Part Relationship in the Unified Modeling Language, IEEE Transactions on Software Engineering, Vol. 29. No. 5, May 2003.

KOSIMap: Use of Description Logic Reasoning to Align Heterogeneous Ontologies

Quentin Reul[1] and Jeff Z. Pan[2]

[1] VUB STARLab, Vrije Universiteit Brussel, Pleinlaan 2, 1050 Brussels, Belgium
[2] University of Aberdeen, Aberdeen AB24 3FX, UK
Quentin.Reul@vub.ac.be; jeff.z.pan@abdn.ac.uk

Abstract. Semantic interoperability is essential on the Semantic Web to enable different information systems to exchange data. Such interoperability can be achieved by identifying similar information in heterogeneous ontologies. In this paper, we describe the Knowledge Organisation System Implicit Mapping (KOSIMap) framework, which differs from existing ontology mapping approaches by using description logic reasoning (i) to extract implicit information as background knowledge for every entity, and (ii) to remove inappropriate mappings from an alignment. The results of our evaluation show that the use of Description Logic in the ontology matching task increases coverage.

1 Introduction

Semantic interoperability enables distributed information systems to exchange data, knowledge, or resources based on common terminologies. These terminologies are often expressed in the form of ontologies as they provide a explicit and server-stored conceptualization of a domain based on well-defined semantics. However, the development of OWL ontologies relies on knowledge engineers to interpret data from domain experts. As a result, two knowledge engineers may interpret the same data differently. This leads to heterogeneity, such as differences in naming and conceptualization, that hinders interoperability among distributed information systems.

Semantic interoperability is essential on the Semantic Web to both provide and create services, and perform complex tasks without prior knowledge of available resources or how to acquire them. Ontology mapping has been recognised as a viable solution for this problem. Given two ontologies $\mathcal{O}_1$ and $\mathcal{O}_2$, the task of mapping one ontology to another is that of finding an entity (i.e. classes, properties, and instances) in $\mathcal{O}_1$ that matches an entity in $\mathcal{O}_2$ based on their intended meaning. Although mappings can be derived manually, this process is time consuming and error prone especially as the size and complexity of ontologies increase. Therefore, it is necessary to develop methods to (semi-)automatically discover similar entities in heterogeneous ontologies. Several surveys reviewing ontology matching techniques, and tools have been carried over the years [4, 3]. These surveys show that most successful approaches combine different lexical and structural similarity measures to cover lexical descriptions as well as

the descriptive information provided by semantic relations. For instance, Ctx-Match [2] (and its successor SMatch [7]) creates logical formulae by mapping classes in two ontologies to synsets in WordNet [6]. These logical formulae are then processed by a SAT solver to extract semantic mappings between classes in two ontologies. Alternatively, the OLA (OWL-Lite Alignment) framework [5] measures the similarity between two entities in OWL-Lite ontologies based on their features (e.g. labels, super-classes, properties). However, these approaches disregard the role of description logic reasoning to extract implicit information (i.e. logical consequences) about entities as a source of background knowledge.

In this paper, we describe the Knowledge Organisation System Implicit Mapping (KOSIMap) framework, which differs from existing approaches by using description logic reasoning (i) to extract implicit information as background knowledge for every entity, and (ii) to remove inappropriate mappings from an alignment. Note that this paper differs from [13] by describing the KOSIMap framework in detail rather than reporting the results of the Ontology Alignment Evaluation Initiative (OAEI) 2009 campaign. The rest of the paper is organised as follows. In section 2, we review some related approaches on ontology mapping. Section 3 presents how KOSIMap extracts mappings between entities from two ontologies. The results of the evaluation testing the core assumptions of KOSIMap are reported in section 4, while the final section discusses our results and outlines future work.

2   Related Work

OLA [5] relies on OWL-Lite constructs (e.g. `rdfs:label`, `rdfs:subClassOf`) to map entities from two ontologies. More specifically, it first calculates a local similarity score by combining the measure of different constructs through a weighted sum. For instance, the similarity between two classes aggregates the score for their names, superclasses, and properties. OLA then propagates the local similarity score to its neighbours. This iterative process ends when the approximated similarity score does not increase between two iterations. Similarly, Janowicz [9] computes the similarity between two classes based on the overlap of their respective $\mathcal{ALCNR}$ descriptions in the domain of GIScience. In SIM-DL, a similarity value of 1 indicates that compared concept descriptions are equal whereas 0 implies total dissimilarity. For example, the similarity between two classes is computed by the Jaccard similarity coefficient applied to their sets of subclasses, while the similarity between two restrictions $\exists r.C$ and $\exists s.D$ is based on the similarity between the involved roles (i.e. r and s) and fillers (C and D). Note that none of these approaches use description logic reasoning to extract implicit information about entities.

Some approaches have also added a debugging component to improve the quality of the mappings. For example, ASMOV [10] first computes a pre-alignment from the matrix that results from the similarity calculation by adding mappings that are maximal within a threshold ζ. This pre-alignment is then subjected to semantic validation, which detects inappropriate mappings and their causes

based on asserted axioms in the two ontologies. Alternatively, Meilicke et al. [12] provide non-standard reasoning based on Distributed Description Logic (DDL) to support the revision of mappings. In this case, mappings are encoded as bridge rules (e.g. $\mathcal{O}_1{:}\boldsymbol{A} \xrightarrow{\equiv} \mathcal{O}_2{:}\boldsymbol{B}$) and DDL reasoning is applied on these bridge rules to determine inconsistent mappings.

3 KOSIMap

In this section, we present the KOSIMap framework that aligns entities in ontology $\mathcal{O}_1$ and ontology $\mathcal{O}_2$ based on the application of description logic reasoning. KOSIMap first extracts logical consequences embedded in both ontologies using a DL reasoner (§3.1). Next, KOSIMap computes three different types of similarities for every pair of entities (§3.2). We then build a matrix storing the combined values from which a pre-aligment is extracted (§3.3). Finally, we remove inappropriate mappings from the pre-alignment. Note that KOSIMap performs each step consecutively. The source ontology, denoted $\mathcal{O}_s$, is described in Example 1 and is based on the Pizza tutorial [8].

Example 1. Suppose we have an ontology $\mathcal{O}_s$ defined by the following OWL statements:

```
Namespace(pizzaA: <http://www.owl-ontologies.com/pizza#>)
Ontology (
  Class(pizzaA:Pizza_Topping partial)
  Class(pizzaA:Cheese_Topping partial pizzaA:Pizza_Topping)
  Class(pizzaA:Mozzarella_Topping partial intersectionOf(pizzaA:Cheese_Topping
      restriction(pizzaA:hasSpicyness someValuesFrom(pizzaA:Mild))))
  Class(pizzaA:Pepperoni_Topping partial intersectionOf(pizzaA:Pizza_Topping
      restriction(pizzaA:hasSpicyness someValuesFrom(pizzaA:Medium))))
  Class(pizzaA:Tomato_Topping partial intersectionOf(pizzaA:Pizza_Topping
      restriction(pizzaA:hasSpicyness someValuesFrom(pizzaA:Mild))))
  Class(pizzaA:Pizza partial restriction(pizzaA:hasBase someValuesFrom(pizzaA:Pizza_Base)))
  Class(pizzaA:Americana_Pizzas partial intersectionOf(pizzaA:Pizza
      restriction(pizzaA:hasTopping someValuesFrom(pizzaA:Mozzarella_Topping))
      restriction(pizzaA:hasTopping someValuesFrom(pizzaA:Tomato_Topping))
      restriction(pizzaA:hasTopping someValuesFrom(pizzaA:Pepperoni_Topping))
      restriction(pizzaA:hasTopping allValuesFrom(
          unionOf(pizzaA:Mozzarella_Topping pizzaA:Tomato_Topping
            pizzaA:Pepperoni_Topping)))))
  Class(pizzaA:Cheesy_Pizza complete intersectionOf(pizzaA:Pizza
      restriction(pizzaA:hasTopping someValuesFrom(pizzaA:Cheese_Topping))))
  ObjectProperty(pizzaA:hasBase)
  SubPropertyOf(pizzaA:hasBase pizzaA:hasIngredient)
  ObjectProperty(pizzaA:hasTopping domain(pizzaA:Pizza) range(pizzaA:Pizza_Topping))
  SubPropertyOf(pizzaA:hasTopping pizzaA:hasIngredient)
  ObjectProperty(pizzaA:topped inverseOf(pizzaA:hasTopping))
)
```

3.1 Pre-Processing

We first process lexical descriptions of each entity (e.g. labels and names) using Natural Language Processing (NLP) techniques. We apply three types of NLP techniques. Firstly, we remove characters, such as '.', '_', '-' and ' ', from the string. Secondly, We used the Pling-Stemmer from [16] to stem plural words.

Finally, we ensure that every string is in lower case. For example, the name of **Americana_Pizzas** is converted to "*americanapizza*".

Entities are not only defined by lexical descriptions, but also by the semantics provided by the axioms in the ontology. For example, the subsumption relation links two classes according to the genus/species classification. In KOSIMap, we extract implicit information about every class and object property from the asserted axioms in the ontologies using a DL reasoner (e.g. FaCT++ [18]). The set of all named classes occurring in an ontology $\mathcal{O}$ is denoted by $\mathrm{CN}_{\mathcal{O}}$, while $\mathrm{RN}_{\mathcal{O}}$ refers to the set of all named object properties in $\mathcal{O}$.

Classes. The set of subsumers of a class $\boldsymbol{A} \in \mathrm{CN}_{\mathcal{O}}$, denoted by $\mathrm{S}_c(\boldsymbol{A})$, contains every (implicit and explicit) super-classes and equivalent classes of $\boldsymbol{A}$ following the classification of the ontology $\mathcal{O}$ by a DL reasoner. In Example 1, $\mathrm{S}_c(\boldsymbol{Americana\_Pizzas}) = \{\boldsymbol{Americana\_Pizzas}, \boldsymbol{Cheesy\_Pizza}, \boldsymbol{Pizza}\}$, whereas the set of explicit parents is $\{\boldsymbol{Americana\_Pizzas}, \boldsymbol{Pizza}\}$.

Table 1. Rules to extract properties associated with classes.

PR1 If $\geq 1\ r \sqsubseteq \boldsymbol{C} \in \mathcal{O}$, $r \in \mathrm{RN}_{\mathcal{O}}$, $\boldsymbol{C} \in \mathrm{CN}_{\mathcal{O}}$, & $r \notin \mathrm{P}_c(\boldsymbol{C})$ then $\mathrm{P}_c(\boldsymbol{C}) := \mathrm{P}_c(\boldsymbol{C}) \bigcup \{r\}$
PR2 if $\boldsymbol{C} \sqsubseteq \exists r.\boldsymbol{X} \in \mathcal{O}$, $r \in \mathrm{RN}_{\mathcal{O}}$, $\boldsymbol{C} \in \mathrm{CN}_{\mathcal{O}}$, & $r \notin \mathrm{P}_c(\boldsymbol{C})$ then $\mathrm{P}_c(\boldsymbol{C}) := \mathrm{P}_c(\boldsymbol{C}) \bigcup \{r\}$
PR3 if $\boldsymbol{C} \sqsubseteq \forall r.\boldsymbol{X} \in \mathcal{O}$, $r \in \mathrm{RN}_{\mathcal{O}}$, $\boldsymbol{C} \in \mathrm{CN}_{\mathcal{O}}$, & $r \notin \mathrm{P}_c(\boldsymbol{C})$ then $\mathrm{P}_c(\boldsymbol{C}) := \mathrm{P}_c(\boldsymbol{C}) \bigcup \{r\}$
PR4 if $\boldsymbol{C} \sqsubseteq\ = 1r.\boldsymbol{X} \in \mathcal{O}$, $r \in \mathrm{RN}_{\mathcal{O}}$, $\boldsymbol{C} \in \mathrm{CN}_{\mathcal{O}}$, & $r \notin \mathrm{P}_c(\boldsymbol{C})$ then $\mathrm{P}_c(\boldsymbol{C}) := \mathrm{P}_c(\boldsymbol{C}) \bigcup \{r\}$
PR5 if $\boldsymbol{C} \sqsubseteq\ \geq 1r.\boldsymbol{X} \in \mathcal{O}$, $r \in \mathrm{RN}_{\mathcal{O}}$, $\boldsymbol{C} \in \mathrm{CN}_{\mathcal{O}}$, & $r \notin \mathrm{P}_c(\boldsymbol{C})$ then $\mathrm{P}_c(\boldsymbol{C}) := \mathrm{P}_c(\boldsymbol{C}) \bigcup \{r\}$
PR6 if $\boldsymbol{C} \sqsubseteq\ \leq 1r.\boldsymbol{X} \in \mathcal{O}$, $r \in \mathrm{RN}_{\mathcal{O}}$, $\boldsymbol{C} \in \mathrm{CN}_{\mathcal{O}}$, & $r \notin \mathrm{P}_c(\boldsymbol{C})$ then $\mathrm{P}_c(\boldsymbol{C}) := \mathrm{P}_c(\boldsymbol{C}) \bigcup \{r\}$

Classes are also described in terms of properties, which provides information about the characteristics of a class. Most existing approaches only consider properties that are explicitly associated with classes. In more expressive ontology languages, such as OWL DL, class characteristics can be embedded in the axioms or be inherited from their subsumers. As a result, we have devised several rules to extract the properties associated with a class (Table 1). The first rule (**PR1**) states that the domain of a named property is a property of that class. In Example 1, this rule would infer that *hasTopping* is a property of **Pizza**. Rules **PR2** to **PR6** process every general concept inclusion in the ontology of the form $\boldsymbol{A} \sqsubseteq \mathrm{Rest}(r).\boldsymbol{B}$, where $\mathrm{Rest}(r)$ is a restriction (e.g. `someValuesFrom`, `allValuesFrom`, and `minCardinality`). Based on these types of general concept inclusion, we are able to extract the implicit object properties associated with $\boldsymbol{A}$. In Example 1, the application of rule **PR2** infers that *hasBase* is a property of **Pizza**.

Object Properties. The `rdfs:subPropertyOf` construct defines the property hierarchy by stating that a property is a subproperty of another. For example, ontology $\mathcal{O}_s$ states that *hasTopping* is a sub-property of *hasIngredient*. Thus, a

reasoner can deduce that if an individual is related to another by the *hasTopping*, then it is also related to the other by the *hasIngredient* property. The set of super-properties of a property $r \in \mathrm{RN}_\mathcal{O}$, denoted $\mathrm{S}_p(r)$, includes all the super-properties of r.

Table 2. Extension rules for binary relation

ER1 If $(\boldsymbol{X}, \boldsymbol{Y}) \in \mathrm{R}(r)$, $r \equiv r^- \in \mathcal{O}$, & $(\boldsymbol{Y}, \boldsymbol{X}) \notin \mathrm{R}(r)$
then $\mathrm{R}(r) := \mathrm{R}(r) \bigcup \{(\boldsymbol{Y}, \boldsymbol{X})\}$
ER2 If $(\boldsymbol{X}, \boldsymbol{Y}) \in \mathrm{R}(r)$, $r \equiv r_\circ^- \in \mathcal{O}$, & $(\boldsymbol{Y}, \boldsymbol{X}) \notin \mathrm{R}(r_\circ^-)$
then $\mathrm{R}(r_\circ^-) := \mathrm{R}(r_\circ^-) \bigcup \{(\boldsymbol{Y}, \boldsymbol{X})\}$
ER3 If $(\boldsymbol{X}, \boldsymbol{Y}) \in \mathrm{R}(r)$, $r \sqsubseteq s \in \mathcal{O}$, & $(\boldsymbol{X}, \boldsymbol{Y}) \notin \mathrm{R}(s)$
then $\mathrm{R}(s) := \mathrm{R}(s) \bigcup \{(\boldsymbol{X}, \boldsymbol{Y})\}$
ER4 If $(\boldsymbol{X}, \boldsymbol{Y}) \in \mathrm{R}(r)$, $(\boldsymbol{Y}, \boldsymbol{Z}) \in \mathrm{R}(s)$, $r \circ s \sqsubseteq t \in \mathcal{O}$, & $(\boldsymbol{X}, \boldsymbol{Z}) \notin \mathrm{R}(t)$
then $\mathrm{R}(t) := \mathrm{R}(t) \bigcup \{(\boldsymbol{X}, \boldsymbol{Z})\}$

The set of binary relation of an object property r, denoted $\mathrm{R}(r)$, is a collection of ordered pairs of elements on $\mathrm{CN}_\mathcal{O}$. More specifically, the set $\mathrm{R}(r)$ is a subset of the Cartesian product $\mathrm{CN}_\mathcal{O} \times \mathrm{CN}_\mathcal{O}$. For instance, the statement $(\boldsymbol{A}, \boldsymbol{B}) \in \mathrm{R}(r)$ is read as $\boldsymbol{A}$ is r-related to $\boldsymbol{B}$, and $\boldsymbol{A}$ and $\boldsymbol{B}$ is called the domain and the range of r respectively. In Example 1, $\mathrm{R}(\textit{hasTopping})$ contains the binary relation $(\boldsymbol{Pizza}, \boldsymbol{Pizza\_Topping})$. In $\mathcal{EL}+$ [1], we can not rely on OWL semantics to express the domain and range of an object property r (i.e. $\geq 1\ r \sqsubseteq \boldsymbol{C}$ and $\top \sqsubseteq \forall r.\boldsymbol{C}$ respectively) as number and universal restrictions are not allowed. In this case, the binary relation $(\boldsymbol{A}, \boldsymbol{B})$ is contained in $\mathrm{R}(r)$ if and only if the axiom $\boldsymbol{A} \sqsubseteq \exists r.\boldsymbol{B}$ is found in the ontology. In Example 1, $\mathrm{R}(\textit{hasBase})$ contains $(\boldsymbol{Pizza}, \boldsymbol{Pizza\_Base})$. We have extended standard reasoning with four rules to extract implicit binary relations (Table 2). The first rule covers symmetric object properties and adds the inverse binary relation to the set if it has not already been added. The second rule is similar to the first rule but deals with inverse object properties. In Example 1, $\mathrm{R}(\textit{topped})$ contains $(\boldsymbol{Pizza\_Topping}, \boldsymbol{Pizza})$ as *topped* is the inverse property of *hasTopping* and that $(\boldsymbol{Pizza}, \boldsymbol{Pizza\_Topping}) \in \mathrm{R}(\textit{hasTopping})$. The last two rules were proposed by [1] and cover role hierarchies and property chain axioms, such as transitive object properties.

3.2 Similarity Generator

The similarity generator computes three kinds of similarities; namely *label similarity*, *property-based similarity*, and *class-based similarity*. We describe each measure in more detail by calculating the similarity between entities in ontology $\mathcal{O}_s$ and entities in $\mathcal{O}_t$ (Example 2). Finally, we describe how individual scores are combined to provide an aggregated value for each pair of entities.

Example 2. Suppose we have an ontology $\mathcal{O}_t$ defined by the following OWL statements:

```
Namespace(p: <http://www.owl-ontologies.com/pizzaB#>)
Ontology (
  Class(pizzaB:Topping partial)
```

```
Class(pizzaB:CheesyTopping partial pizzaB:Topping)
Class(pizzaB:Mozzarella partial intersectionOf(pizzaB:CheesyTopping
    restriction(pizzaB:hasSpicyness someValuesFrom(pizzaB:Mild))))
Class(pizzaB:Tomatoes partial intersectionOf(pizzaB:Topping
    restriction(pizzaB:hasSpicyness someValuesFrom(pizzaB:Mild))))
Class(pizzaB:JalapenoPeppers partial intersectionOf(pizzaB:Topping
    restriction(pizzaB:hasSpicyness someValuesFrom(pizzaB:Hot))))
Class(pizzaB:Pepperoni partial intersectionOf(pizzaB:Topping
    restriction(pizzaB:hasSpicyness someValuesFrom(pizzaB:Medium))))
Class(pizzaB:AmericanHot partial intersectionOf(pizzaB:Pizza
    restriction(pizzaB:hasGarnish someValuesFrom(pizzaB:Mozzarella))
    restriction(pizzaB:hasGarnish someValuesFrom(pizzaB:Tomatoes))
    restriction(pizzaB:hasGarnish someValuesFrom(pizzaB:JalapenoPeppers))
    restriction(pizzaB:hasGarnish someValuesFrom(pizzaB:Pepperoni))))
Class(pizzaB:PizzaWithCheese complete intersectionOf(pizzaB:Pizza
    restriction(pizzaB:hasGarnish someValuesFrom(pizzaB:CheesyTopping))))
ObjectProperty(pizzaB:hasBase domain(pizzaB:Pizza) range(pizzaB:Base))
SubPropertyOf(pizzaB:hasBase pizzaB:hasIngredient)
ObjectProperty(pizzaB:hasGarnish)
SubPropertyOf(pizzaB:hasGarnish pizzaB:hasIngredient)
)
```

Label Similarity. The most basic feature of entities is their labels, which are defined through the `rdfs:label` construct. Labels are human identifiers (i.e. words) expressed in a vocabulary usually shared by experts in the same domain. Therefore, we assume that equivalent classes are likely to be modelled using similar labels (or names). In KOSIMap, we support several string similarity (e.g. Q-Gram [17] or SMOA [15]) to calculate the label similarity for each pair of entities. For example, the label similarity between **Americana_Pizzas** in $\mathcal{O}_s$ and **AmericanHot** in $\mathcal{O}_t$ based on the Q-Gram similarity (i.e. Q-Gram("*americanapizza*", "*americanhot*")) is 0.571.

Degree of Commonality Coefficient. The property-based and class-based similarity is calculated based on the *degree of commonality coefficient* (Definition 1). The DoCCoeff between two sets S_s and S_t is defined as the sum of the maximum similarity for each element in source set (i.e. S_s). Note that the coefficient returns an **asymmetric** measure to reflect the coverage of the first set by the second. This follows the observations made by [19], who argues that the similarity between sets of complex objects is directional and asymmetric.

Definition 1 (Degree of Commonality Coefficient). *Given two sets S_s and S_t, the degree of commonality coefficient between them, denoted $DoCCoeff(S_s, S_t)$ is defined as:*

$$DoCCoeff(S_s, S_t) = \frac{1}{\max(|S_s|, |S_t|)} \sum_{e_i \in S_s} \max_{e_j \in S_t} sim(e_i, e_j) \qquad (1)$$

where S_s is the source set, S_t is the target set, and $sim(e_i, e_j)$ computes the similarity between pair of elements in the two sets.

Property-based similarity computes the similarity between two entities based on the set of properties associated with them. KOSIMap first calculates the overlap between set of super-properties for each pair of properties. Let's compute the property-based similarity between *hasTopping* in $\mathcal{O}_s$ and *hasGarnish* in $\mathcal{O}_t$. In this case, the sets of super-properties for each property is:

- $\text{S}_p(\textit{hasTopping}) = \{\textit{hasIngredient}\}$
- $\text{S}_p(\textit{hasGarnish}) = \{\textit{hasIngredient}\}$

We then calculate the DoCCoeff based on the label similarity (i.e. Q-Gram). In this case, the property-based similarity is 1. Secondly, we calculate the overlap between two classes based on their respective sets of inherited properties, which are generated based in the rules in Table 1. Let's now consider the similarity between **Americana_Pizzas** in $\mathcal{O}_s$ and **AmericanHot** in $\mathcal{O}_t$. In this case, the set of inherited properties for each class is:

- $\text{P}_c(\textbf{\textit{Americana\_Pizzas}}) = \{\textit{hasBase}, \textit{hasTopping}\}$
- $\text{P}_c(\textbf{\textit{AmericanHot}}) = \{\textit{hasBase}, \textit{hasGarnish}\}$

We then calculate the DoCCoeff between the two sets. In this case, the similarity between two elements (i.e. object properties) is computed based on the property-based similarity between their sets of super-properties. Thus, the property-based similarity between **Americana_Pizzas** and **AmericanHot** is 1 as $\texttt{sim}(\textit{hasBase}, \textit{hasBase}) = 1$ and $\texttt{sim}(\textit{hasTopping}, \textit{hasGarnish}) = 1$.

Class-based similarity computes the similarity between two entities based on the set of classes associated with them. KOSIMap first computes the overlap between two classes based in their sets of subsumers. Let's compute the class-based similarity between **Americana_Pizzas** and **AmericanHot**. In this case, the set of subsumers for each class is:

- $\text{S}_c(\textbf{\textit{Americana\_Pizzas}}) = \{\textbf{\textit{Americana\_Pizzas}}, \textbf{\textit{Cheesy\_Pizza}}, \textbf{\textit{Pizza}}\}$.
- $\text{S}_c(\textbf{\textit{AmericanHot}}) = \{\textbf{\textit{AmericanHot}}, \textbf{\textit{PizzaWithCheese}}, \textbf{\textit{Pizza}}\}$.

We then calculate the DoCCoeff between the two sets based on the label similarity (i.e. Q-Gram). In this case, the class-based similarity between **Americana_Pizzas** and **AmericanHot** is 0.706. Secondly, we calculate the class-based similarity between pairs of object properties based on their set of binary relations. Let's now consider the overlap between *hasTopping* in $\mathcal{O}_s$ and *hasGarnish* in $\mathcal{O}_t$. In this case, the set of binary relations for each object property is:

- $\text{R}(\textit{hasTopping}) = \{(\textbf{\textit{Pizza}}, \textbf{\textit{Pizza\_Topping}}), (\textbf{\textit{Americana\_Pizza}}, \textbf{\textit{Tomato\_Topping}}),$
  $(\textbf{\textit{Americana\_Pizza}}, \textbf{\textit{Mozarella\_Topping}}), (\textbf{\textit{Americana\_Pizza}}, \textbf{\textit{Pepperoni\_Topping}})\}$.
- $\text{R}(\textit{hasGarnish}) = \{(\textbf{\textit{AmericanHot}}, \textbf{\textit{Pepperoni}}), (\textbf{\textit{AmericanHot}}, \textbf{\textit{JalapenoPeppers}}),$
  $(\textbf{\textit{AmericanHot}}, \textbf{\textit{Mozzarella}}), (\textbf{\textit{AmericanHot}}, \textbf{\textit{Tomatoes}})\}$

We then calculate the DoCCoeff between the two sets. As we are dealing with binary relations, the DoCCoeff between two sets of binary relations combines the similarity between the first element of two binary relations with the similarity between their second elements. In this case, the similarity between two elements (i.e. classes) is computed based on the class-based similarity calculated between two classes. For example, the degree of commonality coefficient between (**Pizza**, **Pizza_Topping**) and (**AmericanHot**, **Pepperoni**) is 0.333 as $\texttt{sim}(\textbf{\textit{Pizza}}, \textbf{\textit{AmericanHot}}) = 0.333$ and $\texttt{sim}(\textbf{\textit{Pizza\_Topping}}, \textbf{\textit{Pepperoni}}) = 0.333$. The class-based similarity between *hasTopping* and *hasGranish* is 0.593.

Similarity Aggregation combines the score of the above three types of similarity to obtain a more complete measure of similarity. The combined score for each pair of entities is then stored into a *similarity* matrix (Definition 2), where each entity in the source ontology corresponds to a row and each entity in the target ontology corresponds to a column.

Definition 2 (Similarity Matrix). *A similarity matrix, denoted* $\texttt{SIM}_{st}$, *is a matrix with dimension n*m, where n and m are the number of entities in the source and target ontology respectively. The entries $r_{st} \in [0,1]$ denotes the similarity between e_s and e_t, where e_s is an entity in the source ontology and e_t is an entity in the target ontology.*

In KOSIMap, the *global similarity* (i.e. sim_g) is computed through a linear function that balances the impact of each measure by giving it a weight w_k and is defined as:

$$sim_g(e_1, e_2) = \sum_{k=0}^{n} w_k * sim_k(e_1, e_2) \tag{2}$$

where n is the number of similarity measures considered and $w_k \in [0,1]$. Suppose we assign a weight of 0.3 to the label similarity, 0.2 to the property-based similarity and 0.5 to the class-based similarity, then the global similarity between **Americana_Pizzas** in $\mathcal{O}_s$ and **AmericanHot** in $\mathcal{O}_t$ is $0.3 * 0.571 + 0.2 * 1 + 0.5 * 0.706 = 0.724$. The global similarity for each pair of entities is then stored into a *similarity* matrix.

3.3   Mapping Extraction and Refinement

The goal of the final step is to extract a set of mappings from the similarity matrix. This is normally achieved by discarding all candidate mappings below a threshold ζ. However, this method may return multiple mappings for each entity in the source ontology.

As a result, we propose a two-step approach to extract mappings, where an entity in the source ontology is associated with at most one entity in the source ontology. The approach first extracts a pre-alignment (i.e. $\texttt{A}_{pre}$) from the similarity matrix. A mapping $\langle e_s, e_t, \equiv, r_{st} \rangle$ is added to $\texttt{A}_{pre}$ if r_{st} is the highest value for e_s and if it is bigger than a threshold ζ. Note that if two elements e_t^1 and e_t^2 have a similarity value such that $sim_g(e_s, e_t^1) = sim_g(e_s, e_t^2)$, then both $\langle e_s, e_t^1, \equiv, sim_g(e_s, e_t^1) \rangle$ and $\langle e_s, e_t^2, \equiv, sim_g(e_s, e_t^2) \rangle$ are added to A_{pre}.

This pre-alignment is then passed through a refinement process, which eliminates inappropriate mappings. In KOSIMap, we identify two types of inappropriate mappings, namely redundant and inconsistent mappings. Redundant mappings are encountered when mappings in a pre-alignment $\texttt{A}_{pre}$ can be inferred from existing mappings, while inconsistent mappings occur when a class in the source ontology is mapped to several classes in the target ontology that are defined as disjoint. [14] argues that direct siblings (i.e. entities having the same parent) are disjoint unless it introduces conflicts. As KOSIMap assumes that

the local ontologies are consistent, we consider direct siblings as disjoint entities. Table 3 shows the final set of mappings between $\mathcal{O}_s$ and $\mathcal{O}_t$ with a threshold $\zeta = 0.2$. This approach differs from ASMOV [10] in that it checks whether the information inferred by the mappings can be proven by both the explicit and implicit knowledge available in the local ontologies.

Table 3. The mapping resulting from the alignment between the two ontologies.

Entity1	Relation	Entity2	Strength
$\mathcal{O}_s$:***Americana_Pizza***	=	$\mathcal{O}_t$:***AmericanHot***	0.724
$\mathcal{O}_s$:***Cheese_Topping***	=	$\mathcal{O}_t$:***CheesyTopping***	0.567
$\mathcal{O}_s$:***Cheesy_Pizza***	=	$\mathcal{O}_t$:***PizzaWithCheese***	0.75
$\mathcal{O}_s$:***Medium***	=	$\mathcal{O}_t$:***Medium***	0.8
$\mathcal{O}_s$:***Mild***	=	$\mathcal{O}_t$:***Mild***	0.8
$\mathcal{O}_s$:***Mozarella_Topping***	=	$\mathcal{O}_t$:***Mozzarella***	0.487
$\mathcal{O}_s$:***Pepperoni_Topping***	=	$\mathcal{O}_t$:***Pepperoni***	0.533
$\mathcal{O}_s$:***Pizza***	=	$\mathcal{O}_t$:***Pizza***	1.0
$\mathcal{O}_s$:***Pizza_Topping***	=	$\mathcal{O}_t$:***Topping***	0.533
$\mathcal{O}_s$:*hasBase*	=	$\mathcal{O}_t$:*hasBase*	0.861
$\mathcal{O}_s$:*hasIngredient*	=	$\mathcal{O}_t$:*hasIngredient*	0.626
$\mathcal{O}_s$:*hasSpicyness*	=	$\mathcal{O}_t$:*hasSpicyness*	0.61
$\mathcal{O}_s$:*hasTopping*	=	$\mathcal{O}_t$:*hasGarnish*	0.534

4   Evaluation

In this section, we assess the impact of Description Logic reasoning on the computation of the similarity between two entities and on the extraction of appropriate mappings.

4.1   Method

This evaluation is carried out on a subset of the OAEI Conference track[3]. Note that we only consider the ontologies for which a reference alignment is provided (i.e. EKAW, SOFSEM, SIGKDD, IASTED, CMT, ConfOf and EDAS). The advantage of the conference track is that it includes ontologies that share the same domain of discourse (i.e. conference organisation) and that are rich in various types of axioms. The evaluation consists of two experiments:

1. **Explicit vs. Implicit Hierarchy:** This experiment compares the role of description logic in determining the class hierarchy. The first alignment is computed by applying the class-based similarity to the set of super-classes for each class in two ontologies. The set of super-classes for a class **A** is obtained by traversing the asserted hierarchy from the class itself to the root node of the ontology. The second alignment is obtained by applying the class-based similarity to the set of subsumers of each class (§3.1). Note that the reference alignment only includes mapping between classes.

---

[3] http://nb.vse.cz/~svabo/oaei2009/

2. **Disjointness vs. Siblings:** The alignment extraction method relies on disjointness to determine inconsistent mappings. This approach has been extended to consider direct siblings as disjoint entities. This evaluation focuses on the impact of siblings on the alignment extraction process. The first alignment is computed by only considering the explicit disjointness, while the second alignment is obtained by considering direct siblings. Note that the direct sibling are computed based on the classified ontology. In both cases, the weights for label similarity, property-based similarity, and class-based similarity are set to 0.4, 0.1 and 0.5 respectively. Note that because we are focusing on the extraction step the similarity measure does not have an impact on the results.

4.2 Results

The first experiment compares two methods to obtain the class hierarchy. The first method relies on the explicit class hierarchy, while the second method uses DL reasoning to extract set of subsumers. We applied the class-based similarity on the respective sets to compute 21 pairs of alignments. Generally, the highest f-measure is achieved at the same threshold for the implicit hierarchy as for the explicit hierarchy. The implicit hierarchy achieves better f-measure in 15 cases. In 6 of these 15 case, the recall achieved by the two methods is the same, but the implicit hierarchy yields a better precision. This suggests that the use of the implicit hierarchy (as background knowledge) improves the coverage of the ontology mapping task.

Table 4 shows the harmonic mean (H-Mean) f-measure score of each approach across different thresholds for the 21 tests in the conference test case. We can see that the use of the implicit hierarchy consistently yields better results than the use of the explicit hierarchy. Thus, this further suggests that the use of the implicit hierarchy improves the coverage of the ontology mapping task.

Table 4. H-Mean f-measure at different threshold for the Explicit vs. Implicit Hierarchy experiment.

Threshold	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
Implicit Hierarchy	.34	.37	.40	.44	.45	.47	.43	.42	.38
Explicit Hierarchy	.29	.32	.35	.36	.40	.44	.42	.40	.37

The second experiment to evaluate the impact of using asserted disjointness (i.e. explicitly stated in axioms) compared to using implicit disjointness. The implicit disjointness is obtained by considering direct siblings as disjoint entities based on the classified ontology. In KOSIMap, the disjointness is only used during the alignment extraction process, and thus does not have an impact when calculating the similarity between two entities. In 15 out of the 21 tests, the use of direct siblings achieves the same results as those achieved by using the explicit disjointness. This can be explained by the fact that the set of direct siblings is identical to the set of disjoint entities for the entities being mapped.

Table 5 shows the harmonic mean f-measure score of each approach across different thresholds for the 21 tests in the conference test case. We can see that the use of direct siblings consistently yields better results than the use of disjointness. We have also performed this experiment by combining both approaches and have found that this approach always achieves the best result.

Table 5. H-Mean f-measure at different threshold for the Disjointness vs. Siblings.

Threshold	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
Disjoint	.26	.33	.43	.52	.57	.57	.42	.34	0
Siblings	.27	.34	.45	.54	.59	.58	.43	.34	0

5 Conclusion

In this paper, we presented the KOSIMap framework, which uses Description Logic reasoning (i) to extract implicit information (i.e. logical consequences) about every entity, and (ii) to remove inappropriate mappings from an alignment. The framework first extracts logical consequences embedded in both ontologies using an OWL DL reasoner. Next, KOSIMap computes three different types of similarities for every pair of entities. We then build a matrix storing the aggregated values for every pair of entities from which mappings are extracted. Finally, we remove inappropriate mappings from the set of all possible mappings. Note that each step is performed in an ordered and consecutive manner.

The results of our evaluation showed that the use of the implicit hierarchy consistently yields better overall f-measure on the OAEI conference track. We also observed that the use of the implicit hierarchy improves the coverage. Secondly, we checked whether the use of direct siblings during the alignment extraction process had a negative impact on the coverage. The overall f-measure showed that the use of direct siblings consistently yields better results than the use of disjointness during the alignment extraction process.

Although these results are encouraging, we realise that the approach can be further improved. For example, the pre-alignment process phase could be improved by iteratively considering another entity in the target ontology when a mapping has been removed during the mapping refinement phase. Moreover, the refinement process could be improved by not only considering local logical inconsistencies, but by also considering logical inconsistencies in the distributed ontologies. For example, Meilicke et al. [12] provide non-standard reasoning based on DDL to support the mapping revision process.

6 Acknowledgements

References

1. F. Baader, S. Brandt, and C. Lutz. Pushing the $\mathcal{EL}$ envelope. In *Proc. 19th International Joint Conference on Artificial Intelligence (IJCAI-05)*, 2005.
2. P. Bouquet, L. Serafini, and S. Zanobini. Semantic coordination: a new approach and an application. In *Proc. 2nd International Semantic Web Conference (ISWC'03)*, 2003.
3. S. Castano, A. Ferrara, S. Montanelli, G. N. Hess, and S. Bruno. State of the art on ontology coordination and matching. Technical report, BOEMIE, March 2007.
4. J. Euzenat and P. Shvaiko. *Ontology Matching*. Springer-Verlag, Berlin, 2007.
5. J. Euzenat and P. Valtchev. An integrative proximity measure for ontology alignment. In *Proc. ISWC-2003 Workshop on Semantic Information Integration*, 2003.
6. C. Fellbaum. *WordNet: An Electronic Lexical Database*. MIT Press, 1998.
7. F. Giunchiglia and P. Shvaiko. Semantic matching. *The Knowledge Engineering Review Journal (KER)*, 18(3):265–280, 2003.
8. M. Horridge, H. Knublauch, A. Rector, R. Stevens, and C. Wroe. A practical guide to building OWL ontologies using the protégé-OWL plugin and CO-ODE tools. Technical report, University of Manchester, 2004.
9. K. Janowicz. Sim-DL: Towards a semantic similarity measurement theory for the description logic $\mathcal{ALCNR}$ in geographic information retrieval. In *On the Move to Meaningful Internet Systems*, 2006.
10. Y. R. Jean-Mary, E. P. Shironoshita, and M. R. Kabuka. Ontology matching with semantic verification. *Web Semantics: Science, Services and Agents on the World Wide*, 2009.
11. H. W. Kuhn. The Hungarian Method for the assignment problem. *Naval Research Logistics Quarterly*, 2:83–97, 1955.
12. C. Meilicke, H. Stuckenschmidt, and A. Tamilin. Reasoning support for mapping revision. In *Proc. 23rd Conference on Artificial Intelligence (AAAI-08)*, 2008.
13. Q. Reul and J. Pan. KOSIMap: Ontology Alignments Results for OAEI 2009. In *Proc. 4th International Workshop on Ontology Matching (OM-2009)*, 2009.
14. S. Schlobach. Debugging and semantic clarification by pinpointing. In *Proc. 2nd European Semantic Web Conference (ESWC05)*, pages 226–240, 2005.
15. G. Stoilos, G. Stamou, and S. Kollias. A string metric for ontology alignment. In *Proc. 4th International Semantic Web Conference (ISWC 2005)*, 2005.
16. F. M. Suchanek, G. Ifrim, and G. Weikum. LEILA: Learning to extract information by linguistic analysis. In *Proc. 2nd Workshop on Ontology Population (OLP2)*, 2006.
17. E. Sutinen and J. Tarhio. On using Q-Gram locations in approximate string matching. In *Proc. 3rd European Symposium on Algorithms (ESA 95)*, 1995.
18. D. Tsarkov and I. Horrocks. FaCT++ description logic reasoner: System description. In *Proc. IJCAR 2006*, 2006.
19. A. Tversky. Features of similarity. *Psychological Review*, 84(4):327–352, July 1977.

Index