Did I Damage my Ontology?

A Case for Conservative Extensions in Description Logic

Silvio Ghilardi[1], **Carsten Lutz**[2] **and Frank Wolter**[3]

[1]Dept. of Computer Science
University of Milan, Italy
ghilardi@dsi.unimi.it

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

[3]Dept. of Computer Science
University of Liverpool, UK
frank@csc.liv.ac.uk

Abstract

In computer science, ontologies are dynamic entities: to adapt them to new and evolving applications, it is necessary to frequently perform modifications such as the extension with new axioms and merging with other ontologies. We argue that, after performing such modifications, it is important to know whether the resulting ontology is a conservative extension of the original one. If this is not the case, then there may be unexpected consequences when using the modified ontology in place of the original one in applications. In this paper, we propose and investigate new reasoning problems based on the notion of conservative extension, assuming that ontologies are formulated as TBoxes in the description logic $\mathcal{ALC}$. We show that the fundamental such reasoning problems are decidable and 2EXPTIME-complete. Additionally, we perform a finer-grained analysis that distinguishes between the size of the original ontology and the size of the additional axioms. In particular, we show that there are algorithms whose runtime is 'only' exponential in the size of the original ontology, but double exponential in the size of the added axioms. If the size of the new axioms is small compared to the size of the ontology, these algorithms are thus not significantly more complex than the standard reasoning services implemented in modern description logic reasoners. If the extension of an ontology is not conservative, our algorithm is capable of computing a concept that witnesses non-conservativeness. We show that the computed concepts are of (worst-case) minimal size.

Introduction

Ontologies are used to define the terminology of application domains. As modern applications require ontologies of large scale and complex structure whose design and maintenance is a challenging task, there is a need to support ontology developers by automated reasoning tools. This need explains the success of description logics (DLs) as an underlying formalism for ontologies: powerful and theoretically well-founded reasoning systems for DLs are readily available. In particular, DL reasoners such as Racer, Fact and Pellet offer reasoning services such as deciding satisfiability and subsumption of concepts w.r.t. an ontology. In other words, these tools are capable of answering queries like "Can the concept C have any instances in models of the ontology $\mathcal{T}$?" (satisfiability of C) and "Is the concept D more general than the concept C in models of the ontology $\mathcal{T}$?" (subsumption $C \sqsubseteq D$).

In many application domains of DL ontologies such as the semantic web, medical informatics, and bio-informatics, ontologies are not static (Baader, Horrocks, & Sattler 2005; Serafini, Stuckenschmidt, & Wache 2005). The necessity to maintain, refine, customize, and integrate ontologies frequently requires revisions, extensions, and mergings with other ontologies. When such operations are performed on well-established and tested ontologies, it is of prime importance for the ontology developer to have control of the consequences of her modifications. In particular, it frequently happens that unintended consequences are introduced, and such consequences can be far from obvious. There appears to be consensus that the reasoning services provided by current DL reasoners do not provide sufficient support to verify whether a modified ontology behaves as intended. In this paper, we address this shortcoming and propose novel DL reasoning services that support the user in understanding the consequences of *extending* and *merging* existing ontologies.

To illustrate what kind of reasoning is desirable in the context of ontology extension and merging, we discuss two example scenarios. First, suppose that an ontology developer maintains a well-tested ontology $\mathcal{T}$ that formalizes an application domain. Assume that she wants to extend $\mathcal{T}$ with a number of additional axioms that describe the terminology of a part of the domain that was not yet covered by $\mathcal{T}$. Moreover, the developer wants to use the extended ontology in an application that requires computing subsumptions between concepts, and for which she has previously used $\mathcal{T}$. To avoid unexpected results in such applications when using the extended ontology, the existing part of $\mathcal{T}$ should not be compromised by the new axioms. In particular, the extended ontology should not entail new subsumptions between concepts that are formulated in the signature of the old ontology, where the term "signature" refers to the concept and role names used. An automated reasoning tool should support the ontology developer in checking whether any such additional subsumptions have been generated by the extension.

Second, assume that there are two well-established ontologies $\mathcal{T}_1$ and $\mathcal{T}_2$ that describe different and largely independent aspects of an application domain, but nevertheless

have an overlap in signature. To use $\mathcal{T}_1$ and $\mathcal{T}_2$ together in the same application, one would like to merge them by simply taking their union. Similarly to the case of ontology extensions, it is then important to know whether the merging operation compromises the component ontologies: are there any subsumptions in the signature of $\mathcal{T}_1$ entailed by $\mathcal{T}_1 \cup \mathcal{T}_2$ that are not entailed by $\mathcal{T}_1$ alone, and likewise for $\mathcal{T}_2$. Intuitively, the entailment of such subsumptions means that there may be unexpected results when using the merged ontology in place of the component ontologies.

The reasoning problems suggested by these two examples can be conveniently formalized using the notion of a conservative extension, which has been widely studied and applied in mathematical logic and in the philosophy of science. In software engineering, conservative extensions have been suggested to define the notion of a refinement of a specification (Turski & Maibaum 1987). In a similar spirit and following (Antoniou & Kehagias 2000), we use them in the context of ontologies. Formally, an ontology $\mathcal{T}'$ is a *conservative extension* of an ontology $\mathcal{T}$ iff every subsumption in the signature of $\mathcal{T}$ entailed by $\mathcal{T}'$ is already entailed by $\mathcal{T}$ (equivalently, iff every concept in the signature of $\mathcal{T}$ that is unsatisfiable w.r.t. $\mathcal{T}'$ is already unsatisfiable w.r.t. $\mathcal{T}$). We use conservative extensions to phrase two fundamental reasoning problems for ontology extension and merging:

1. Is the extension $\mathcal{T}'$ of the ontology $\mathcal{T}$ a conservative extension of $\mathcal{T}$?
2. Is the merged ontology $\mathcal{T}_1 \cup \mathcal{T}_2$ a conservative extension of its parts?

While these reasoning tasks already provide the ontology designer with relevant information, we can be even more informative: assume that the given extension of the ontology turns out *not* to be a conservative extension of the original one. This may or may not be intended, and it is up to the developer to decide whether the new ontology faithfully represents the domain under consideration. To support her in this decision, it is useful to compute *witness concepts*, i.e., examples of concepts that were satisfiable before, but not after the extension. Thus, we obtain another reasoning service:

3. If $\mathcal{T} \subseteq \mathcal{T}'$, and $\mathcal{T}'$ is not a conservative extension of $\mathcal{T}$, compute witness concepts that are as small as possible.

The purpose of this paper is *to provide a first formal analysis of reasoning problems based on conservative extensions in the basic propositionally closed description logic $\mathcal{ALC}$ (Schmidt-Schauß & Smolka 1991)*. More precisely, we show that Problems 1 and 2 are decidable and that witness concepts from Problem 3 are computable, we pinpoint the exact computational complexity of Problems 1 and 2, and we determine tight bounds on the (worst-case) length of witness concepts. In this way, we lay a foundation for the creation of automated reasoning tools that support ontology developers during ontology extension and merging.

We also consider a further refinement of our reasoning problems that is motivated by the observation that non-conservative extensions and mergings of ontologies are sometimes intended and completely acceptable. Suppose the original ontology $\mathcal{T}$ consists of a core ontology (e.g., an upper-level ontology (Guarino 1998)) that by no means should be corrupted, and of other parts for which a non-conservative extension is acceptable or even intended. In this case, the required reasoning problem is the following:

4. Given a set Γ of concept names and roles, and ontologies $\mathcal{T}$ and $\mathcal{T}'$ such that $\mathcal{T} \subseteq \mathcal{T}'$, is there a concept in the signature Γ that is unsatisfiable w.r.t. $\mathcal{T}'$, but not w.r.t. $\mathcal{T}$? If so, provide a witness concept.

This problem clearly subsumes Problems 1 to 3. Again, we prove decidability and establish tight bounds on the length of witness concepts. Concerning the computational complexity, we show that Problems 1 and 2 are easier than Problem 4.

Preliminaries

We introduce the description logic $\mathcal{ALC}$ and define a notion of conservative extensions in the context of DLs. Let $\mathsf{N_C}$ and $\mathsf{N_R}$ be disjoint and countably infinite sets of *concept names* and *role names*. Then, *$\mathcal{ALC}$ concepts* are formed according to the following syntax rule:

$$C, D \longrightarrow \top \mid A \mid \neg C \mid C \sqcap D \mid \exists r.C$$

where A ranges over concept names and r ranges over role names. The concept constructors $\bot$, $\sqcup$, and $\forall r.C$ are defined as abbreviations: $\bot$ stands for $\neg\top$, $C \sqcup D$ for $\neg(\neg C \sqcap \neg D)$ and $\forall r.C$ abbreviates $\neg\exists r.\neg C$.

The representation of ontologies in description logics is based on TBoxes, and we will from now on use these two terms interchangeably. Formally, a *TBox* is a finite set of concept implications $C \sqsubseteq D$. For the sake of completeness, we briefly illustrate the use of TBoxes as an ontology formalism: the following TBox provides an (extremely simplified) ontology of web services, where $C \doteq D$ is an abbreviation for the two concept implications $C \sqsubseteq D$ and $D \sqsubseteq C$:

$$\mathsf{webservice} \sqsubseteq \exists \mathsf{provider}.\top \sqcap \exists \mathsf{input}.\top \sqcap \exists \mathsf{output}.\top$$
$$\mathsf{amazonservice} \doteq \mathsf{webservice} \sqcap \exists \mathsf{provider}.\mathsf{amazon}$$

The upper line describes necessary conditions for being a web service: every web services has to have at least one provider, input, and output. The lower line provides both necessary and sufficient conditions for being a web service of Amazon: a web service is an Amazon web service if and only if its provider is Amazon. Note, however, that concept implications in TBoxes are not required to have atomic left-hand sides.

The semantics of $\mathcal{ALC}$ concepts is defined set-theoretically in terms of interpretations. An *interpretation* is a pair $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$, where $\Delta^{\mathcal{I}}$ is a non-empty set (the *domain*) and $\cdot^{\mathcal{I}}$ is the *interpretation function*, assigning to each concept name A a set $A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$, and to each role name r a binary relation $r^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$. The interpretation function is inductively extended to concepts as follows:

$$\begin{aligned} \top^{\mathcal{I}} &:= \Delta^{\mathcal{I}} \\ (\neg C)^{\mathcal{I}} &:= \Delta^{\mathcal{I}} \setminus C^{\mathcal{I}} \\ (C \sqcap D)^{\mathcal{I}} &:= C^{\mathcal{I}} \cap D^{\mathcal{I}} \\ (\exists r.C)^{\mathcal{I}} &:= \{d \in \Delta^{\mathcal{I}} \mid \exists e.(d,e) \in r^{\mathcal{I}} \wedge e \in C^{\mathcal{I}}\} \end{aligned}$$

An interpretation $\mathcal{I}$ *satisfies* an implication $C \sqsubseteq D$ if $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$, and $\mathcal{I}$ is a *model* of a TBox $\mathcal{T}$ if it satisfies all implications in $\mathcal{T}$. A concept C is *satisfiable* relative to a TBox $\mathcal{T}$ if there exists a model $\mathcal{I}$ of $\mathcal{T}$ such that $C^{\mathcal{I}} \neq \emptyset$. A concept C is *subsumed* by a concept D relative to a TBox $\mathcal{T}$ (written $\mathcal{T} \models C \sqsubseteq D$) if every model $\mathcal{I}$ of $\mathcal{T}$ satisfies the implication $C \sqsubseteq D$.

We now introduce conservative extensions of TBoxes. A finite set $\Gamma \subseteq \mathsf{N_C} \cup \mathsf{N_R}$ of concept and role names is called a *signature*. The *signature* $\mathsf{sig}(\mathcal{T})$ *of a TBox* $\mathcal{T}$ is the set of concept and role names occurring in $\mathcal{T}$. Given a signature Γ, we use $\mathcal{ALC}(\Gamma)$ to denote the set of $\mathcal{ALC}$-concepts using only concept and role names from Γ.

Definition 1. Let $\mathcal{T}_1$ and $\mathcal{T}_2$ be TBoxes and let $\Gamma \subseteq \mathsf{sig}(\mathcal{T}_1)$. Then

1. $\mathcal{T}_1 \cup \mathcal{T}_2$ is *a conservative extension of* $\mathcal{T}_1$ *w.r.t.* Γ if, for all $C_1, C_2 \in \mathcal{ALC}(\Gamma)$, we have $\mathcal{T}_1 \models C_1 \sqsubseteq C_2$ iff $\mathcal{T}_1 \cup \mathcal{T}_2 \models C_1 \sqsubseteq C_2$.
2. $\mathcal{T}_1 \cup \mathcal{T}_2$ is *a conservative extension of* $\mathcal{T}_1$ if $\mathcal{T}_1 \cup \mathcal{T}_2$ is a conservative extension of $\mathcal{T}_1$ w.r.t. $\mathsf{sig}(\mathcal{T}_1)$.

It is well-known that subsumption can be reduced to satisfiability and vice versa: $\mathcal{T} \models C \sqsubseteq D$ holds iff $C \sqcap \neg D$ is unsatisfiable relative to $\mathcal{T}$ and C is unsatisfiable relative to $\mathcal{T}$ iff $\mathcal{T} \models C \sqsubseteq \bot$. Therefore, $\mathcal{T}_1 \cup \mathcal{T}_2$ is conservative extension of $\mathcal{T}_1$ w.r.t. Γ if and only if each concept $C \in \mathcal{ALC}(\Gamma)$ that is satisfiable relative to $\mathcal{T}_1$ is satisfiable relative to $\mathcal{T}_1 \cup \mathcal{T}_2$:

The two kinds of conservative extensions introduced in Definition 1 give rise to two reasoning problems:

- *Deciding conservative extensions* means, given TBoxes $\mathcal{T}_1$ and $\mathcal{T}_2$, to decide whether $\mathcal{T}_1 \cup \mathcal{T}_2$ is a conservative extension of $\mathcal{T}_1$
- *Deciding relativized conservative extensions* means, given TBoxes $\mathcal{T}_1$, $\mathcal{T}_2$ and a signature $\Gamma \subseteq \mathsf{sig}(\mathcal{T}_1)$, to decide whether $\mathcal{T}_1 \cup \mathcal{T}_2$ is a conservative extension of $\mathcal{T}_1$ w.r.t. Γ.

As discussed in the introduction, it is likely that users ask for evidence if $\mathcal{T}_1 \cup \mathcal{T}_2$ turns out *not* to be a conservative extension of $\mathcal{T}_1$. Such evidence can be provided by witness concepts: a concept $C \in \mathcal{ALC}(\Gamma)$ is called a *witness concept* for $(\mathcal{T}_1, \mathcal{T}_2, \Gamma)$ if C is satisfiable relative to $\mathcal{T}_1$, but not relative to $\mathcal{T}_1 \cup \mathcal{T}_2$. A *witness concept* for $(\mathcal{T}_1, \mathcal{T}_2)$ is simply defined as a witness concept for $(\mathcal{T}_1, \mathcal{T}_2, \mathsf{sig}(\mathcal{T}_1))$.

To illustrate the use of conservative extensions for controlling the consequences of ontology extensions, we give a simple example. The following ontology $\mathcal{T}_1$ describes web services on a general level (like the OWL-S ontology (Martin *et al.* 2004)). For simplicity, our $\mathcal{T}_1$ consists only of a single concept implication:

$$\mathsf{webservice} \sqsubseteq \exists \mathsf{provider}.\top \sqcap \exists \mathsf{input}.\top \sqcap \exists \mathsf{output}.\top.$$

Suppose that an ontology developer wants to refine the TBox $\mathcal{T}_1$ for a special class of web services that take an integer and an array as input. This can be done by taking the union of $\mathcal{T}_1$ and the following TBox $\mathcal{T}_2$:

$$\begin{array}{rcl} \mathsf{mywebservice} & \sqsubseteq & \mathsf{webservice} \sqcap \exists \mathsf{input.integer} \\ & & \sqcap \exists \mathsf{input.array} \\ \mathsf{integer} & \sqsubseteq & \neg \mathsf{array} \end{array}$$

It is not difficult to check that $\mathcal{T}_1 \cup \mathcal{T}_2$ is a conservative extension of both $\mathcal{T}_1$ and $\mathcal{T}_2$. Therefore, the developer can be sure that, in every application that uses only the concept name web_service and the role names provider, input, and output, she can safely replace the TBox $\mathcal{T}_1$ by its refinement $\mathcal{T}_1 \cup \mathcal{T}_2$. In a next step, the developer further extends $\mathcal{T}_1 \cup \mathcal{T}_2$ with the following TBox $\mathcal{T}_3$:

$$\begin{array}{rcl} \mathsf{mywebservice} & \sqsubseteq & \exists \mathsf{input.user\_id} \\ \mathsf{user\_id} & \sqsubseteq & \neg \mathsf{array} \\ \mathsf{user\_id} & \sqsubseteq & \neg \mathsf{integer} \end{array}$$

Then, $\mathcal{T}_1 \cup \mathcal{T}_2 \cup \mathcal{T}_3$ is not a conservative extension of $\mathcal{T}_1 \cup \mathcal{T}_2$, as is shown by the witness concept

$$\mathsf{mywebservice} \sqcap \forall \mathsf{input}.(\mathsf{integer} \sqcup \mathsf{array}).$$

By inspecting the witness concept, the developer has to decide whether or not non-conservativeness of the extension was intended. In any case, non-conservativeness indicates that applications using the TBox $\mathcal{T}_1 \cup \mathcal{T}_2$ have to be inspected for compatibility with the new TBox $\mathcal{T}_1 \cup \mathcal{T}_2 \cup \mathcal{T}_3$.

In passing, we note that $\mathcal{T}_1 \cup \mathcal{T}_2$ is *not* a conservative extension of $\mathcal{T}_1$ if witness concepts are allowed to be formulated in $\mathcal{ALCN}$, the extension of $\mathcal{ALC}$ with number restrictions (Baader & Sattler 1999; Baader *et al.* 2003a): in this case, a witness concept is $\mathsf{webservice} \sqcap (\leqslant 1\ \mathsf{input})$. This shows that being a conservative extension strongly depends on the DL under consideration. Some more discussion of this issue is provided in the conclusion.

It is easily seen that the standard reasoning tasks satisfiability and (non-)subsumption can be polynomially reduced to deciding conservative extensions: first, subsumption can be polynomially reduced to satisfiability. Now assume that we want to check whether C is satisfiable relative to $\mathcal{T}$. W.l.o.g., we may assume that $C \in \mathcal{ALC}(\mathsf{sig}(\mathcal{T}))$. Then, C is satisfiable relative to $\mathcal{T}$ iff $\mathcal{T} \cup \{\neg C \doteq \top\}$ is not a conservative extension of $\mathcal{T}$. It follows that deciding conservative extensions is at least as hard as deciding subsumption, i.e., EXPTIME-hard in the case of $\mathcal{ALC}$ (Baader *et al.* 2003a; Schild 1991).

It is interesting to note that the reduction of satisfiability to conservative extensions can be reversed if the added concept implications do not introduce new concept or role names. Assume that we want to check whether $\mathcal{T}_1 \cup \mathcal{T}_2$ is a conservative extension of $\mathcal{T}_1$, where $\mathsf{sig}(\mathcal{T}_2) \subseteq \mathsf{sig}(\mathcal{T}_1)$. This can be done by first converting $\mathcal{T}_2$ into the form $\{\top \doteq C\}$, and then deciding whether $\neg C$ is unsatisfiable w.r.t. $\mathcal{T}_1$. As implied by the complexity results obtained in the following section, such a (polynomial) reduction is not possible if $\mathcal{T}_2$ introduces additional vocabulary.

Results

The purpose of this section is to summarize the main results of this paper. Some hints to the proofs are then given in the next two sections. We start with a basic result about the complexity of deciding conservative extensions and an upper bound on the length of witness concepts. In the following, we denote by $|C|$ the length of a concept C. Similarly, the *size* $|\mathcal{T}|$ of a TBox $\mathcal{T}$ is defined as $\sum_{C \sqsubseteq D \in \mathcal{T}} (|C| + |D|)$.

Theorem 2. *It is 2*ExpTime*-complete to decide relativized and non-relativized conservative extensions. In both cases, if input TBoxes are $\mathcal{T}_1$ and $\mathcal{T}_2$ and the extension is not conservative, then there exists a witness concept C of length at most 3-exponential in $|\mathcal{T}_1 \cup \mathcal{T}_2|$ that can be computed in time polynomial in $|C|$.*

As we will see later, the upper bound on the length of witness concepts given in Theorem 2 is tight. We now refine Theorem 2 by distinguishing between the size of the TBoxes $\mathcal{T}_1$ and $\mathcal{T}_2$. Such a more fine-grained analysis is rewarding if the sizes of $\mathcal{T}_1$ and $\mathcal{T}_2$ can be expected to differ substantially. This will usually be the case if an existing TBox is extended with a set of new concept implications: then, $|\mathcal{T}_2|$ is small compared to $|\mathcal{T}_1|$. In contrast, when merging two existing TBoxes, then no obvious assumption can be made concerning the relative size of $\mathcal{T}_1$ and $\mathcal{T}_2$.

It turns out that, when discriminating the size of $\mathcal{T}_1$ and $\mathcal{T}_2$, a difference in computational complexity emerges between deciding non-relativized and relativized conservative extensions. We first consider the former and show that there exists a decision procedure that is only exponential in the size of $\mathcal{T}_1$, but double exponential in the size of $\mathcal{T}_2$. Clearly, we cannot expect a better bound in the size of $\mathcal{T}_1$: it follows from the reduction of satisfiability given in the previous section (and the fact that already the satisfiability of concept *names* relative to TBoxes is ExpTime-hard) that deciding conservative extensions is ExpTime-hard even if $\mathcal{T}_2$ is assumed to be constant. We also provide a refined upper bound for the length of witness concepts by proving that, if $\mathcal{T}_1 \cup \mathcal{T}_2$ is not a conservative extension of $\mathcal{T}_1$, then one can compute witness concepts of size 'only' 2-exponential in the size of $\mathcal{T}_1$, but 3-exponential in the size of $\mathcal{T}_2$. A matching lower bound on the size of witness concepts is established as well.

Theorem 3.

(i) There exists a deterministic algorithm for deciding conservative extensions whose runtime is bounded by $2^{p(|\mathcal{T}_1|)\times 2^{p(|\mathcal{T}_2|)}}$, with p a polynomial.

(ii) For all TBoxes $\mathcal{T}_1$ and $\mathcal{T}_2$, if $\mathcal{T}_1 \cup \mathcal{T}_2$ is not a conservative extension of $\mathcal{T}_1$, then there exists a witness concept of length at most $2^{2^{|\mathcal{T}_1|\times 2^{|\mathcal{T}_2|}}}$.

(iii) There exist families of TBoxes $(\mathcal{T}_n)_{n>0}$ and $(\mathcal{T}'_n)_{n>0}$, such that, for all $n > 0$,
- *$\mathcal{T}_n \cup \mathcal{T}'_n$ is not a conservative extension of $\mathcal{T}_n$,*
- *$|\mathcal{T}_n| \in \mathcal{O}(n^2)$, $|\mathcal{T}'_n| \in \mathcal{O}(n^2)$, and*
- *every witness concept for $(\mathcal{T}_n, \mathcal{T}'_n)$ is of length at least $2^{(2^n \times 2^{2^n})-1}$.*

We now refine our analysis of relativized conservative extensions. In contrast to the previous case, we can prove that there exists no decision procedure that is only single exponential in the size of $\mathcal{T}_1$: even for a fixed TBox $\mathcal{T}_2$, the complexity of deciding whether $\mathcal{T}_1 \cup \mathcal{T}_2$ is a conservative extension of $\mathcal{T}_1$ w.r.t. Γ is 2ExpTime-hard. The proof is based on the following reduction.

Lemma 4. *Let $\mathcal{T}_1$ and $\mathcal{T}_2$ be TBoxes, $\Gamma = \mathsf{sig}(\mathcal{T}_1)$, and B a concept name not used in $\mathcal{T}_1$ and $\mathcal{T}_2$. Denote by $\mathcal{T}_2^B$ the TBox resulting from $\mathcal{T}_2$ by replacing every implication $C_1 \sqsubseteq C_2 \in \mathcal{T}_2$ with $C_1 \sqcap B \sqsubseteq C_2$. Let $\mathcal{T}_3 = \{B = \top\}$. Then the following are equivalent, for every $C \in \mathcal{ALC}(\Gamma)$:*

- *C is satisfiable relative to $\mathcal{T}_1$ but not relative to $\mathcal{T}_1 \cup \mathcal{T}_2$;*
- *C is satisfiable relative to $\mathcal{T}_1 \cup \mathcal{T}_2^B$ but not relative to $\mathcal{T}_1 \cup \mathcal{T}_2^B \cup \mathcal{T}_3$.*

It follows that $\mathcal{T}_1 \cup \mathcal{T}_2$ is a conservative extension of $\mathcal{T}_1$ iff $\mathcal{T}_1 \cup \mathcal{T}_2^B \cup \mathcal{T}_3$ is a conservative extension of $\mathcal{T}_1 \cup \mathcal{T}_2^B$ w.r.t. Γ.

Together with Theorem 3, Lemma 4 allows us to establish the desired 2ExpTime lower bound. A matching upper bound is obtained from Theorem 2. Using Theorem 3 and Lemma 4, we can also establish a lower bound on the size of witness concepts that is *triple* exponential in the size of $\mathcal{T}_1$. This should be contrasted with the upper bound on witness concepts given in Theorem 3 for the case of non-relativized conservative extensions, which is only double exponential in $\mathcal{T}_1$.

Theorem 5. *Let $\mathcal{T}'$ be a TBox of the form $\{B = \top\}$, with B a concept name. Then*

*(i) it is 2*ExpTime*-complete to decide, given a TBox $\mathcal{T}_1$ and a signature $\Gamma \subseteq \mathsf{sig}(\mathcal{T}_1)$, whether $\mathcal{T}_1 \cup \mathcal{T}'$ is a conservative extension of $\mathcal{T}_1$ w.r.t. Γ.*

(ii) there exist families of TBoxes $(\mathcal{T}_n)_{n>0}$ and signatures $(\Gamma_n)_{n>0}$, such that, for all $n > 0$,
- *$\mathcal{T}_n \cup \mathcal{T}'$ is not a conservative extension of $\mathcal{T}_n$ w.r.t. Γ_n,*
- *$|\mathcal{T}_n| \in \mathcal{O}(n^2)$, and*
- *every witness concept for $(\mathcal{T}_n, \mathcal{T}', \Gamma_n)$ is of length at least $2^{(2^n \cdot 2^{2^n})-1}$.*

In what follows we are going to present sketches of the proofs of these results. Full proofs can be found in the accompanying technical report (Ghilardi, Lutz, & Wolter 2006b).

Upper bounds for witness concepts

We prove the upper bound on the size of witness concepts for relativized conservative extensions given in Theorem 2 and the refined bound on the size of witness concepts for non-relativized conservative extensions given in Part (ii) of Theorem 3. Note that the latter implies the upper bound for *non*-relativized conservative extensions stated in Theorem 2.

Conceptually simple algorithms that decide the satisfiability of DL concepts relative to TBoxes in (worst-case) optimal time are often based on the notion of *types*, i.e., sets of formulas that satisfy some basic Boolean closure conditions. Types are also frequently used in modal and temporal logic (Vardi & Wolper 1986; Pratt 1979). Building on this tradition, the procedures for constructing witness concepts and deciding conservative extensions are also type-based.

We first consider the case of non-relativized conservative extensions. For a TBox $\mathcal{T}$, we denote by $cl(\mathcal{T})$ the closure under single negation of the set of subconcepts of concepts that occur in $\mathcal{T}$. A *$\mathcal{T}$-type* is a subset of $cl(\mathcal{T})$ such that

- $C_1 \sqcap C_2 \in t$ iff $C_1 \in t$ and $C_2 \in t$, for all $C_1 \sqcap C_2 \in cl(\mathcal{T})$;

- $\neg C \in t$ iff $C \notin t$, for all $\neg C \in cl(\mathcal{T})$.

Observe that, given an interpretation $\mathcal{I}$ and $d \in \Delta^{\mathcal{I}}$, the set

$$t^{\mathcal{T}}_{\mathcal{I}}(d) = \{C \in cl(T) \mid d \in C^{\mathcal{I}}\}$$

is a $\mathcal{T}$-type. In what follows we will not always distinguish between the type t and the conjunction $\bigsqcap_{D \in t} D$ over all members of t.

To get a first idea of witness concepts, suppose that there exists a $\mathcal{T}_1$-type t that

- is satisfiable relative to $\mathcal{T}_1$ and
- cannot be extended (by adding concepts) to a $\mathcal{T}_1 \cup \mathcal{T}_2$-type that is satisfiable relative to $\mathcal{T}_1 \cup \mathcal{T}_2$.

Then $\mathcal{T}_1 \cup \mathcal{T}_2$ is not a conservative extension of $\mathcal{T}_1$ and the conjunction over all concepts in t is a witness concept. Unfortunately, the existence of such a type is only a necessary condition for non-conservativeness, but not a sufficient one: in general, witness concepts can be more complicated. To nevertheless obtain witness concepts that are semantically transparent, we now generalize the notion of a $\mathcal{T}$-type to the notion of a $\mathcal{T}$-tree. Similar notions have been developed in (Visser 1996; Ghilardi 1995), see also the section on related work.

Definition 6 ($\mathcal{T}$-tree). A *$\mathcal{T}$-tree* $\mathfrak{T} = (W, <, L)$ is a finite intransitive tree $(W, <)$ such that each node $w \in W$ is labelled by a $\mathcal{T}$-type $L(w)$, each edge (w, w') is labelled by a role name $L(w, w')$ occurring in $\mathcal{T}$, and the following hold:

- for each $w \in W$, $L(w)$ is satisfiable relative to $\mathcal{T}$;
- for each non-leaf w and concept $\exists r.C \in cl(\mathcal{T})$, we have $\exists r.C \in L(w)$ iff there exists a successor w' of w such that $L(w, w') = r$ and $C \in L(w')$.

A $\mathcal{T}$-tree $\mathfrak{T} = (W, <, L)$ is called *singleton-tree* if W has exactly one element.

There is a close connection between $\mathcal{T}$-trees and concepts formulated in the signature $\mathsf{sig}(\mathcal{T})$, as explicated by the following definition.

Definition 7 (Diagram). For each $\mathcal{T}$-tree $\mathfrak{T} = (W, <, L)$ we define the *diagram* $\mathsf{diag}(\mathfrak{T}) \in \mathcal{ALC}(\mathsf{sig}(\mathcal{T}))$ inductively as follows:

- if w is a leaf of $\mathfrak{T}$, then $\mathsf{diag}(\mathfrak{T}) = \bigsqcap L(w)$.
- If w is a non-leaf with successors $\{w_0, \ldots, w_{n-1}\}$ and $\mathfrak{T}_i$ is the subtree of $\mathfrak{T}$ generated by w_i ($i < n$), then $\mathsf{diag}(\mathfrak{T})$ is

$$\bigsqcap L(w) \sqcap \bigsqcap_{r \in \mathsf{sig}(\mathcal{T})} \Big(\bigsqcap_{\{i<n \mid L(w,w_i)=r\}} \exists r.\mathsf{diag}(\mathfrak{T}_i) \sqcap \forall r. \bigsqcup_{\{i<n \mid L(w,w_i)=r\}} \mathsf{diag}(\mathfrak{T}_i) \Big).$$

Clearly, the diagram of a singleton-$\mathcal{T}$-tree is (the conjunction of all elements of) a $\mathcal{T}$-type.

Our aim is to prove the following lemma, which clearly implies the upper bounds for the length of witness concepts in Part (ii) of Theorem 3.

Lemma 8. *$\mathcal{T}_1 \cup \mathcal{T}_2$ is not a conservative extension of $\mathcal{T}_1$ iff there exists a $\mathcal{T}_1$-tree $\mathfrak{T}$ whose outdegree is bounded by $|\mathcal{T}_1| + |\mathcal{T}_2| 2^{|\mathcal{T}_2|}$ and whose depth is bounded by $2^{|\mathcal{T}_1|} \times 2^{2^{|\mathcal{T}_2|}}$ such that $\mathsf{diag}(\mathfrak{T})$ is satisfiable relative to $\mathcal{T}_1$, but not relative to $\mathcal{T}_1 \cup \mathcal{T}_2$.*

Clearly, the "if" direction of the lemma is trivial. To present the idea behind the proof of the "only if" direction, we introduce the following notion.

Definition 9 (Realizable pair). A pair (t, U) consisting of a $\mathcal{T}_1$-type t and a set of $\mathcal{T}_2$-types U is *realized by* a $\mathcal{T}_1$-tree $\mathfrak{T}$ with root w if

- $L(w) = t$,
- $\mathsf{diag}(\mathfrak{T})$ is satisfiable relative to $\mathcal{T}_1$, and
- U consists of precisely those $\mathcal{T}_2$-types s for which $s \sqcap \mathsf{diag}(\mathfrak{T})$ is satisfiable relative to $\mathcal{T}_1 \cup \mathcal{T}_2$.

A pair (t, U) is *realizable* if there is a $\mathcal{T}_1$-tree realizing it.

Intuitively, if (t, U) is a pair realizable by a $\mathcal{T}_1$-tree $\mathfrak{T}$, then (t, U) describes everything we need to know about $\mathsf{diag}(\mathfrak{T})$ as a potential witness concept: the $\mathcal{T}_1$-type t that instances of $\mathsf{diag}(\mathfrak{T})$ have in models of $\mathcal{T}_1$ as well as the set U of all possible $\mathcal{T}_2$-types that instances of $\mathsf{diag}(\mathfrak{T})$ may have in models of $\mathcal{T}_1 \cup \mathcal{T}_2$. Note that $\mathsf{diag}(\mathfrak{T})$ is formulated in the signature of $\mathcal{T}_1$, and thus there may be more than one such $\mathcal{T}_2$-type. Indeed, it is not hard to see that $\mathsf{diag}(\mathfrak{T})$ is a witness concept iff $U = \emptyset$.

We now give the proof of the "only if" direction of Lemma 8. Note that every $\mathcal{T}$-tree realizes at most a single pair (t, U). We use this fact in what follows.

Proof. (i) Suppose that $\mathcal{T}_1 \cup \mathcal{T}_2$ is not a conservative extension of $\mathcal{T}_1$. Then take a witness concept C for $(\mathcal{T}_1, \mathcal{T}_2)$ and a model $\mathcal{I}$ of C and $\mathcal{T}_1$, unravel $\mathcal{I}$ into a tree model $\mathcal{I}'$, and read of a $\mathcal{T}_1$-tree from $\mathcal{I}'$. Since $\mathcal{T}_1$-trees have to be finite, we stop the reading-off process at the *role depth* of C which is defined as the nesting depth of existential restrictions in C. In a second step, the obtained $\mathcal{T}_1$-tree $\mathfrak{T}$ is modified as follows in order to satisfy the constraints formulated in the lemma:

- To obtain a tree of depth bounded by $2^{|\mathcal{T}_1|} \times 2^{2^{|\mathcal{T}_2|}}$, we exhaustively perform the following operation: if there are nodes v and v' with associated subtrees $\mathfrak{T}_v$ and $\mathfrak{T}_{v'}$ such that $v <^* v'$ and $\mathfrak{T}_v$ realizes the same pair as $\mathfrak{T}_{v'}$, then replace $\mathfrak{T}_v$ with $\mathfrak{T}_{v'}$. It is possible to show that this operation preserves the pair realized by the whole tree. Clearly, the depth of the resulting tree is bounded by the number of realizable pairs which is bounded by $2^{|\mathcal{T}_1|} \times 2^{2^{|\mathcal{T}_2|}}$.
- To obtain a tree whose outdegree is bounded by $|\mathcal{T}_1| + |\mathcal{T}_2| 2^{|\mathcal{T}_2|}$, we exhaustively perform the following operation: if there are nodes v and v' with associated subtrees $\mathfrak{T}_v$ and $\mathfrak{T}_{v'}$ such that v' is a direct successor of v and $\mathfrak{T}_v$ realizes the same pair as the tree obtained from $\mathfrak{T}_v$ by dropping the subtree $\mathfrak{T}_{v'}$, then drop this subtree.

❑

When constructing witness concepts for *relativized* conservative extensions, we follow the same strategy. However, most notions have to be slightly extended because we have the additional parameter Γ that describes the signature of witness concepts.

A $(\Gamma, \mathcal{T})$-*type* is defined analogously to $\mathcal{T}$-types, but contains only concepts from $cl(\mathcal{T}) \cap \mathcal{ALC}(\Gamma)$. In analogy to what was done above, the simplest form of witness concepts are induced by $(\Gamma, \mathcal{T}_1)$-types t that

- can be extended to a $\mathcal{T}_1$-type that is satisfiable relative to $\mathcal{T}_1$ and
- cannot be extended to a $\mathcal{T}_1 \cup \mathcal{T}_2$-type that is satisfiable relative to $\mathcal{T}_1 \cup \mathcal{T}_2$.

Generalizing this basic case, we define $(\Gamma, \mathcal{T})$-trees in analogy with $\mathcal{T}$-trees, the only differences being that nodes are labelled with $(\Gamma, \mathcal{T})$-types and that edge labels must be from Γ. Diagrams for such trees are defined in the obvious way. The following lemma implies the upper bound on the size of witness concepts stated in Theorem 2.

Lemma 10. *$\mathcal{T}_1 \cup \mathcal{T}_2$ is not a conservative extension of $\mathcal{T}_1$ w.r.t. Γ iff there exists a $(\Gamma, \mathcal{T}_1)$-tree $\mathfrak{T}$ whose outdegree is bounded by $2|\mathcal{T}_1 \cup \mathcal{T}_2|2^{|\mathcal{T}_1 \cup \mathcal{T}_2|}$ and whose depth is bounded by $2^{2\times 2^{|\mathcal{T}_1 \cup \mathcal{T}_2|}}$ such that $\mathsf{diag}(\mathfrak{T})$ is satisfiable relative to $\mathcal{T}_1$, but not relative to $\mathcal{T}_1 \cup \mathcal{T}_2$.*

The proof of Lemma 10 is analogous to that of Lemma 8. The main difference is that we replace realizable pairs by realizable triples.

Definition 11 (Realizable triple). A triple (t, U_1, U_2), where t is a $(\Gamma, \mathcal{T}_1)$-type, U_1 is a non-empty set of $\mathcal{T}_1$-types, and U_2 is a set of $\mathcal{T}_1 \cup \mathcal{T}_2$-types, is *realized* by a $(\Gamma, \mathcal{T}_1)$-tree $\mathfrak{T}$ with root w if

- $L(w) = t$;
- U_1 is the set of $\mathcal{T}_1$-types s such that $s \sqcap \mathsf{diag}(\mathfrak{T})$ is satisfiable relative to $\mathcal{T}_1$;
- U_2 is the set of $\mathcal{T}_1 \cup \mathcal{T}_2$-types s such that $s \sqcap \mathsf{diag}(\mathfrak{T})$ is satisfiable relative to $\mathcal{T}_1 \cup \mathcal{T}_2$;

Since we are now interested in witness concepts $\mathsf{diag}(\mathfrak{T})$ formulated in the signature Γ, there is no unique $\mathcal{T}_1$-type that instances of $\mathsf{diag}(\mathfrak{T})$ have in models of $\mathcal{T}_1$. Instead, we only have a unique Γ-type t that instances of $\mathsf{diag}(\mathfrak{T})$ have in all models, and a *set* of $\mathcal{T}_1$-types U_1 that instances of $\mathsf{diag}(\mathfrak{T})$ may have in models of $\mathcal{T}_1$. Observe that demanding U_1 to be non-empty corresponds to demanding satisfiability of $\mathsf{diag}(\mathfrak{T})$ w.r.t. $\mathcal{T}_1$. The component U_2 of realizable triples plays the same role as the component U in realizable pairs. To prove Lemma 8, we can now proceed as for Lemma 10, exchanging realizable pairs by realizable triples. Since the number of realizable triples is double exponential in the size of $\mathcal{T}_1$ (whereas the number of realizable pairs is only single exponential), we obtain a tree $\mathfrak{T}$ of depth double exponential in $|\mathcal{T}_1|$.

The decision procedures

We develop algorithms for deciding relativized and non-relativized conservative extensions. In this way, we prove the upper complexity bounds for deciding relativized and non-relativized conservative extensions given in Theorem 2, and the refined upper complexity bound for non-relativized conservative extensions in Part (i) of Theorem 3. We start with the non-relativized case.

Suppose TBoxes $\mathcal{T}_1$ and $\mathcal{T}_2$ are given.

1. Determine the set $\mathcal{R}_0$ of pairs that are realizable by singleton $\mathcal{T}_1$-trees. If $\mathcal{R}_0$ is empty, then $\mathcal{T}_1 \cup \mathcal{T}_2$ is a conservative extension of $\mathcal{T}_1$. Otherwise,
2. if $\mathcal{R}_0$ contains a pair (t, U) such that $U = \emptyset$, then $\mathcal{T}_1 \cup \mathcal{T}_2$ is not a conservative extension of $\mathcal{T}_1$. Otherwise
3. generate the sequence $\mathcal{R}_1, \mathcal{R}_2, \ldots$ of sets of pairs such that
$$\mathcal{R}_{i+1} = \mathcal{R}_i \cup \mathcal{R}'_i,$$
where $\mathcal{R}'_i$ is the set of realizable pairs that can be obtained in one step from a subset of $\mathcal{R}_i$ that has at most $|\mathcal{T}_1| + |\mathcal{T}_2|2^{|\mathcal{T}_2|}$ members. Continue until $\mathcal{R}_i = \mathcal{R}_i \cup \mathcal{R}'_i$ or there exists a pair $(t, U) \in \mathcal{R}'_i$ such that $U = \emptyset$. In the latter case, $\mathcal{T}_1 \cup \mathcal{T}_2$ is not a conservative extension of $\mathcal{T}_1$. Otherwise, it is.

Figure 1: Deciding conservative extensions.

Deciding conservative extensions

The decision procedure rests on the notion of a diagram and of a realizable pair. More precisely, the algorithm generates all realizable pairs for the input TBoxes $\mathcal{T}_1$ and $\mathcal{T}_2$ and returns "$\mathcal{T}_1 \cup \mathcal{T}_2$ is not a conservative extension of $\mathcal{T}_1$" if one of the realizable pairs is of the form (t, U) with $U = \emptyset$. If no such pair is found, the algorithm returns "$\mathcal{T}_1 \cup \mathcal{T}_2$ is a conservative extension of $\mathcal{T}_1$". To construct the set of all realizable pairs, we start with pairs realizable in singleton $\mathcal{T}_1$-trees and then add pairs realized by more complex trees in a step-wise manner. Each step is defined as follows.

Definition 12 (One step). We say that a pair (t, U) can be *obtained in one step* from a set of pairs $\mathcal{R}$ if there exists a $\mathcal{T}_1$-tree $\mathfrak{T} = (W, <, L)$ with root w that realizes (t, U) such that, for each $w' \in W$ with $w < w'$, the pair realized by the subtree of $\mathfrak{T}$ generated by w' is included in $\mathcal{R}$.

Now, the decision procedure is given in Figure 1. It is not difficult to show that, if it terminates in Step 1, then $\mathcal{T}_1$ has no models (see below). If this does not happen, the algorithm looks for a realizable pair (t, U) with $U = \emptyset$ because, as has already been noted, the diagram of a $\mathcal{T}_1$-tree $\mathfrak{T}$ realizing such a pair is a witness concept.

Below, we sketch proofs of the correctness of the algorithm and of the fact that the algorithm requires time at most exponential in the size of $\mathcal{T}_1$ and at most 2-exponential in the size of $\mathcal{T}_2$.

Correctness. If the algorithm terminates returning "$\mathcal{T}_1 \cup \mathcal{T}_2$ is not a conservative extension of $\mathcal{T}_1$" (which may happen in Step 2 or 3), then there exists a realizable pair (t, U) with $U = \emptyset$. It follows immediately from the definition of realiz-

able pairs that, in this case, $\mathcal{T}_1 \cup \mathcal{T}_2$ is indeed not a conservative extension of $\mathcal{T}_1$.

Conversely, suppose that $\mathcal{T}_1 \cup \mathcal{T}_2$ is not a conservative extension of $\mathcal{T}_1$. By Lemma 8, we find a $\mathcal{T}_1$-tree $\mathfrak{T}$ such that $\mathsf{diag}(\mathfrak{T})$ is satisfiable relative to $\mathcal{T}_1$ but not relative to $\mathcal{T}_1 \cup \mathcal{T}_2$ and such that the outdegree of $\mathfrak{T}$ does not exceed $|\mathcal{T}_1| + |\mathcal{T}_1|2^{|\mathcal{T}_2|}$. Since $\mathsf{diag}(\mathfrak{T})$ is satisfiable relative to $\mathcal{T}_1$, there is a pair (t, U) realized by $\mathfrak{T}$. Since $\mathsf{diag}(\mathfrak{T})$ is not satisfiable relative to $\mathcal{T}_1 \cup \mathcal{T}_2$, $U = \emptyset$. To show that the algorithm returns "$\mathcal{T}_1 \cup \mathcal{T}_2$ is not a conservative extension of $\mathcal{T}_1$", we have to show that

(i) it does not terminate in Step 1 and

(ii) $(t, U) \in \mathcal{R}_i$ for some $i \geq 0$.

To prove (i), take a model $\mathcal{I}$ of $\mathcal{T}_1$ (which clearly exists) and a $d \in \Delta^{\mathcal{I}}$. Then define the pair $(t^{\mathcal{T}_1}_{\mathcal{I}}(d), U')$, where U' is the set of all $\mathcal{T}_2$ types t' such that $t' \sqcap t^{\mathcal{T}_1}_{\mathcal{I}}(d)$ is satisfiable w.r.t. $\mathcal{T}_1 \cup \mathcal{T}_2$. It is easy to see that this pair is realizable by a singleton $\mathcal{T}_1$-tree, and thus the algorithm does not terminate in Step 1.

To establish (ii), we can show by induction on i that, for every node w in $\mathfrak{T}$ on level $\mathsf{depth}(\mathfrak{T}) - i$, the pair (t_w, U_w) realized by the subtree of $\mathfrak{T}$ generated by w is in $\mathcal{R}_i$. Therefore, for $n = 2^{|\mathcal{T}_1|} \times 2^{2^{|\mathcal{T}_2|}}$ we know that $\mathcal{R}_n$ contains (t, U). Thus, the algorithm confirms that $\mathcal{T}_1 \cup \mathcal{T}_2$ is not a conservative extension of $\mathcal{T}_1$.

Complexity. We show that, if the computation of the step from $\mathcal{R}_i$ to $\mathcal{R}_{i+1}$ in Step 3 of the algorithm is realized in a suitable way, then the algorithm requires time at most exponential in $\mathcal{T}_1$ and at most 2-exponential in $\mathcal{T}_2$. First, Step 1 can be implemented by enumerating all $\mathcal{T}_1$-types t, checking whether t is satisfiable relative to $\mathcal{T}_1$, and if this is the case computing the corresponding U as the set of all $\mathcal{T}_2$-types s such that $t \sqcap s$ are satisfiable relative to $\mathcal{T}_1 \cup \mathcal{T}_2$. The pairs (t, U) obtained in this way are precisely the pairs realizable in a singleton-$\mathcal{T}_1$-tree. It follows that Steps 1 and 2 of the algorithm stay within the required time bounds. Second, suppose that $\mathcal{R}_i$ has been computed. Then its size is bounded by the number of pairs (t, U), i.e. by $2^{|\mathcal{T}_1|} \times 2^{2^{|\mathcal{T}_2|}}$. Thus, the number of subsets $\mathcal{R}$ of $\mathcal{R}_i$ of cardinality at most $|\mathcal{T}_1| + |\mathcal{T}_2|2^{|\mathcal{T}_2|}$ is single exponential in $|\mathcal{T}_1|$ and double exponential in $|\mathcal{T}_2|$. Consider such a set of pairs $\mathcal{R}$ and a pair (t, U). We have to show that it can be decided in time exponential in $|\mathcal{T}_1|$ and double exponential in $|\mathcal{T}_2|$ whether (t, U) can be obtained in one step from $\mathcal{R}$. To this end, it can be shown that the latter is the case iff for each $r \in \mathsf{sig}(\mathcal{T}_1)$ there exists a successor set $\mathsf{suc}_r \subseteq 2^{\mathcal{R}}$ such that

- for all $\exists r.D \in cl(\mathcal{T}_1)$, we have $\exists r.D \in t$ iff there exists $(t', U') \in \mathsf{suc}_r$ and $D \in t'$ and
- for all $\mathcal{T}_2$-types s, we have $s \in U$ iff the following conditions hold:
  - if $\exists r.C \in s$, then there exists a $(t', U') \in \mathsf{suc}_r$ and an $s' \in U'$ such that $C \in s'$ and, for all concepts $\neg\exists r.D \in s$, $D \notin s'$.
  - for every $(t', U') \in \mathsf{suc}_r$, there exists an $s' \in U'$ such that $\{\neg D \mid \neg\exists r.D \in s\} \subseteq s'$.
  - $s \cup t$ is satisfiable relative to $\mathcal{T}_1 \cup \mathcal{T}_2$.

To check whether (t, U) can be obtained in one step from $\mathcal{R}$, we can thus enumerate all the possible sets suc_r (separately for each role $r \in \mathsf{sig}(\mathcal{T}_1)$), and check the listed conditions. In this way, Step 3 can be executed within the required time bounds. Observe that suc_r is called "successor set" since a $\mathcal{T}_1$- tree that realizes (t, U) can be constructed by taking $\mathcal{T}_1$-trees that realize sets in suc_r and adding a new root. However, a single successor for each element of suc_r may not be sufficient, see (Ghilardi, Lutz, & Wolter 2006b) for details.

Suppose TBoxes $\mathcal{T}_1$ and $\mathcal{T}_2$, and a signature $\Gamma \subseteq \mathsf{sig}(\mathcal{T}_1)$ are given.

1. Determine the set $\mathcal{R}_0$ of triples that are realized by singleton $(\Gamma, \mathcal{T}_1)$-trees. If $\mathcal{R}_0 = \emptyset$, then $\mathcal{T}_1 \cup \mathcal{T}_2$ is a conservative extension of $\mathcal{T}_1$ w.r.t. Γ. Otherwise,
2. if $\mathcal{R}_0$ contains a triple (t, U_1, U_2) such that $U_2 = \emptyset$, then $\mathcal{T}_1 \cup \mathcal{T}_2$ is not a conservative extension of $\mathcal{T}_1$ w.r.t. Γ. Otherwise,
3. generate the sequence $\mathcal{R}_1, \mathcal{R}_2, \ldots$ of sets of triples such that
$$\mathcal{R}_{i+1} = \mathcal{R}_i \cup \mathcal{R}'_i,$$
where $\mathcal{R}'_i$ is the set of realizable triples that can be obtained in one step from a subset of $\mathcal{R}_i$ that has cardinality at most $2|\mathcal{T}_1 \cup \mathcal{T}_2|2^{|\mathcal{T}_1 \cup \mathcal{T}_2|}$. Continue until $\mathcal{R}_i = \mathcal{R}_i \cup \mathcal{R}'_i$ or there exists a triple $(t, U_1, U_2) \in \mathcal{R}'_i$ such that $U_2 = \emptyset$. In the latter case, $\mathcal{T}_1 \cup \mathcal{T}_2$ is not a conservative extension of $\mathcal{T}_1$ w.r.t. Γ. Otherwise, it is.

Figure 2: Deciding relativized conservative extensions.

Deciding relativized conservative extensions

The decision procedure for relativized conservative extensions is quite similar to the procedure for non-relativized ones. As in the case of upper bounds on the size of witness concepts, the main change is that we have to move from realizable pairs to realizable triples. In analogy to Definition 12, we then say that a triple (t, U_1, U_2) can be *obtained in one step* from a set of triples $\mathcal{R}$ if there exists a $(\Gamma, \mathcal{T}_1)$-tree $\mathfrak{T}$ that realizes (t, U_1, U_2) such that, for each $w' \in W$ with $w < w'$, the triple realized by the subtree of $\mathfrak{T}$ generated by w' is included in $\mathcal{R}$.

The modified decision procedure is given in Figure 2. Correctness and termination within the required time bound (i.e., double exponential in the size of both $\mathcal{T}_1$ and $\mathcal{T}_2$) can be shown similar to what was done in the previous section. When adapting the complexity proof, the components U_1 and U_2 of realizable triples are treated in precisely the same way as the component U of realizable pairs is treated in the original proofs. Since the decision procedure given in the previous section is double exponential in $\mathcal{T}_2$ and the main modification was to replace U (which is related to $\mathcal{T}_2$) with U_1 (related to $\mathcal{T}_1$) and U_2 (related to $\mathcal{T}_2$), it should not be surprising that we obtain an algorithm that is double exponential w.r.t. both $|\mathcal{T}_1|$ and $|\mathcal{T}_2|$.

Lower Bounds on Witness Concepts

We prove the lower bound on the size of witness concepts stated in Point (iii) of Theorem 3. To do this, we start with a 2-exponential lower bound and then improve this bound to a 3-exponential one. There are two reasons for this incremental approach. The first one is didactic: the TBoxes used for the 2-exponential lower bound are smaller and more intuitive than the TBoxes underlying the 3-exponential bound, and the latter can be seen as a refinement of the former. The second reason is that, as will be discussed in more detail later, the 3-exponential bound does not properly subsume the 2-exponential one.

2-Exponential Witness Concepts

We are going to prove the following

Theorem 13. *There exists a TBox $\mathcal{T}$ and a family of TBoxes $(\mathcal{T}'_n)_{n>0}$, such that, for all $n > 0$,*

(i) $\mathcal{T} \cup \mathcal{T}'_n$ is not a conservative extension of $\mathcal{T}$,

(ii) $|\mathcal{T}'_n| \in \mathcal{O}(n^2)$, and

(iii) every witness concept for $(\mathcal{T}, \mathcal{T}'_n)$ is of length $\geq 2^{2^n-1}$.

To sketch the proof of Theorem 13, we first need a bit of notation. Let r and s be role names. For an interpretation $\mathcal{I}$, a sequence $w = w_0, \ldots, w_{k-1} \in \{r, s\}^*$, and elements $x, y \in \Delta^{\mathcal{I}}$, we say that y is *w-reachable* from x if there exist elements $x_0, \ldots, x_k \in \Delta^{\mathcal{I}}$ such that $x_0 = x$, $x_k = y$, and $(x_i, x_{i+1}) \in w_i^{\mathcal{I}}$ for all $i < k$. Now, the TBoxes $\mathcal{T}$ and $\mathcal{T}'_n$ from Theorem 13 can be described as follows:

- apart from a technical trick to be described below, the TBox $\mathcal{T}$ is used only to fix the signature of witness concepts: they may use concept names A and B, and role names r and s.
- in the signature $\{A, B, r, s\}$, the TBox $\mathcal{T}'_n$ expresses that its models $\mathcal{I}$ are not *n-violating*, where an interpretation $\mathcal{I}$ is n-violating if there exist elements $x_w \in \Delta^{\mathcal{I}}$, for all $w \in \{r, s\}^*$ of length smaller than 2^n, such that the following are true:

  (a) $x_\varepsilon \in A^{\mathcal{I}}$;

  (b) $(x_w, x_{w'}) \in \sigma^{\mathcal{I}}$ if $w' = w \cdot \sigma$, for all $\sigma \in \{r, s\}$;

  (c) $x_w \notin B^{\mathcal{I}}$ if w is of length $2^n - 1$.

Thus, a witness concept C for $\mathcal{T}$ and $\mathcal{T}'_n$ has to enforce that all its models are n-violating. Intuitively, this means that the concept must enforce an instance of A that is the root of a binary tree of depth 2^n such that left children are connected with the role r, right children are connected with the role s, and all leafs are instances of $\neg B$. The fact that the number of nodes in such a tree is 2-exponential in n can be used to show that the size of witness concepts for $\mathcal{T}$ and $\mathcal{T}'_n$ is at least 2-exponential in n as well.

The precise formulation of the TBoxes $\mathcal{T}$ and $\mathcal{T}'_n$ can be found in Figure 3. In $\mathcal{T}'_n$, we use a binary counter C for counting modulo 2^n. The counter is based on the concept names $C_0, \ldots, C_{n-1}$ representing the bits of C. The expression $\forall r.(C{+}{+})$ is an abbreviation for the usual concept stating that the value of C is incremented when going to r-successors. Note that $\mathcal{T}'_n$ enforces more in the signature $\{A, B, r, s\}$ than what was stated above. For example, there can be no model of $\mathcal{T}'_n$ in which an instance of A has an r-successor that is also an instance of A. However, concepts such as $A \sqcap \exists r.A$ should not qualify as a witness concept. This explains Line (1) of $\mathcal{T}$: by enforcing that A holds only in points with no r- and s-predecessors, we guarantee that, already in models of $\mathcal{T}$, there are no two instances x and y of A such that y is w-reachable from x for some $w \in \{r, s\}^*$.

$$
\begin{array}{lrcll}
\mathcal{T}: & \top & \sqsubseteq & \forall r.\neg A \sqcap \forall s.\neg A & (1)\\
 & B & \sqsubseteq & B & (2)\\
 & & & & \\
\mathcal{T}'_n: & A & \sqsubseteq & A' \sqcap (C = 0) & (3)\\
 & A' \sqcap (C < 2^n - 1) & \sqsubseteq & \forall r.(C{+}{+}) \sqcap \forall s.(C{+}{+}) & (4)\\
 & A' \sqcap (C < 2^n - 1) & \sqsubseteq & \forall r.A' \sqcup \forall s.A' & (5)\\
 & A' \sqcap (C = 2^n - 1) & \sqsubseteq & B & (6)
\end{array}
$$

Figure 3: The TBoxes $\mathcal{T}$ and $\mathcal{T}'_n$.

It is possible to show that $\mathcal{T}$ and $\mathcal{T}'_n$ indeed behave as described above. This is used in the proof of the following lemma.

Lemma 14.

1. *A concept C that is satisfiable w.r.t. $\mathcal{T}$ and such that all models of C and $\mathcal{T}$ are n-violating is a witness concept for $\mathcal{T}$ and $\mathcal{T}'_n$;*
2. *if a concept C is a witness concept for $\mathcal{T}$ and $\mathcal{T}'_n$, then all tree models of C and $\mathcal{T}$ are n-violating.*

By Point 1 of Lemma 14, the concept C_{2^n} defined in the following is a witness concept for $\mathcal{T}$ and $\mathcal{T}'_n$. This shows that $\mathcal{T} \cup \mathcal{T}'_n$ is indeed not a conservative extension of $\mathcal{T}$:

$$D_0 := \neg B \quad D_{i+1} := \exists r.D_i \sqcap \exists s.D_i \quad C_i := A \sqcap D_i$$

Note that the concepts D_i are introduced only as abbreviations, and not as concept names in a TBox. It is not hard to see that the length of C_{2^n} is $\mathcal{O}(2^{2^n})$. To establish Theorem 13, it remains to prove that each witness concept for $\mathcal{T}$ and $\mathcal{T}'_n$ is of length at least 2^{2^n-1}. By Line (1) of Figure 3, if A is satisfied in a tree model, then it is satisfied in the root. Thus, Point 2 of Corollary 14 implies that every witness concept for $\mathcal{T}$ and $\mathcal{T}'_n$ satisfies the preconditions of the following lemma, which yields the required lower bound on the size of witness concepts.

Lemma 15. *Let P be the set of all sequences $w \in \{r, s\}^*$ of length $2^n - 1$, and let C be a concept such that the following holds:*

(i) C is satisfiable w.r.t. $\mathcal{T}$;

(ii) for all tree models $\mathcal{I}$ of C and $\mathcal{T}$ with root $r \in \Delta^{\mathcal{I}}$, we have $r \in \left(A \sqcap \bigsqcap_{w \in P} \exists w.\top\right)^{\mathcal{I}}$.

Then C is of length at least 2^{2^n-1}.

In the lemma, $\exists w.C$ with $w = w_1 \cdots w_n$ abbreviates $\exists w_1.\exists w_2. \cdots \exists w_n.C$, and $\exists \varepsilon.C$ is just C.

3-Exponential Witness Concepts

We now improve the 2-exponential lower bound to the 3-exponential one stated in Part(iii) of Theorem 3. Note that, in contrast to Theorem 13, the size of the unprimed TBox grows with n instead of being fixed. Therefore, Theorem 3 (iii) does not imply Theorem 13. We leave it as an open problem whether Theorem 3 (iii) can be strengthened such that the unprimed TBox is fixed.

The idea for constructing $\mathcal{T}_n$ and $\mathcal{T}'_n$ is very similar to what was done in the previous section, only that witness concepts must now enforce binary trees of depth *double* exponential in n. To achieve this increase of the depth, we need a counter X that counts modulo 2^{2^n}. Obviously, such a counter cannot be realized using one concept name for every bit as this would result in a TBox of size exponential in n. Therefore, we use a different form of counting: a single value of X is described using an *X-sequence*, i.e., a sequence $x_1, \dots, x_{2^n} \in \Delta^{\mathcal{I}}$ such that $(x_i, x_{i+1}) \in r^{\mathcal{I}} \cup s^{\mathcal{I}}$ for all $i < 2^n$. Each element x_i represents the truth value of one bit of the counter value via the concept name X.[1] To describe the position of the bit represented by an element x_i, we use a binary counter C that is based on concept names $C_0, \dots, C_{n-1}$ and counts from 0 to $2^n - 1$ along each X-sequence, thus assigning bit positions to elements.

The TBoxes $\mathcal{T}_n$ and $\mathcal{T}'_n$ are given in Figure 4, where we use $\forall(r \cup s).C$ as an abbreviation for $\forall r.C \sqcap \forall s.C$. We first discuss $\mathcal{T}_n$. Lines (7) and (8) are identical to Lines (1) and (2) of the TBox $\mathcal{T}$ from the previous section, and Line (9) ensures that the counter C counts correctly. Lines (10) to (12) guarantee that X-sequences starting at instances of A encode the value null, and that all elements of such sequences are marked with the concept name F (for "first counter value"). The purpose of Lines (13) to (15) is to ensure that, if an element x is part of an X-sequence, then $x \in Z^{\mathcal{I}}$ if and only if the X-sequence has a null-value for some bit strictly lower than that represented by x.

The TBox $\mathcal{T}'_n$ uses a third counter D that counts modulo 2^n using concept names $D_0, \dots, D_{n-1}$. This counter is used to ensure that X counts correctly in models of witness concepts. Before we go into detail, we give a semantic characterization of the interplay between $\mathcal{T}_n$ and $\mathcal{T}'_n$. Let $\mathcal{I}$ be an interpretation. For each $w = w_1 \cdots w_k \in \{r, s\}^*$, we call a sequence $x_1, \dots, x_{k+1} \in \Delta^{\mathcal{I}}$ a *w-path starting at x* if $x = x_1$ and, for $1 \leq i \leq k$, $(x_i, x_{i+1}) \in w_i^{\mathcal{I}}$. If w is not important, we simply speak of a *path*. For $n \geq 0$, an interpretation $\mathcal{I}$ is called *strongly n-violating* iff there exists an $x \in A^{\mathcal{I}}$ such that the following two properties are satisfied, where $m := 2^n \cdot 2^{2^n}$.

(P1) for all paths $x_1, \dots, x_k$ in $\mathcal{I}$ starting at x of length at most m, the X values of $x_1, \dots, x_k$ describe the first k bits of the consecutive values of a 2^n-bit counter that counts from 0 to $2^{2^n} - 1$ (each value represented with lowest bit first);

(P2) there exist elements $x_w \in \Delta^{\mathcal{I}}$, for all $w \in \{r, s\}^*$ of length at most $m - 1$, such that the following are true:

(a) $x_\varepsilon \in A^{\mathcal{I}}$;

(b) $(x_w, x_{w'}) \in \sigma^{\mathcal{I}}$ if $w' = w \cdot \sigma$ for all $\sigma \in \{r, s\}$;

(c) $x_w \notin B^{\mathcal{I}}$ if w is of length $m - 1$.

Intuitively, Property (P1) guarantees that the counter X counts correctly, and (P2) is the double-exponential version of what was called "n-violating" in the previous section. In (P2), we use the bound $m - 1$ since the counter X has 2^{2^n} possible values, each value of X is represented by an X-sequence of length $2^n - 1$, and any two consecutive X-sequences are connected by an additional r- or s-edge.

To ensure that witness concepts enforce models that are strongly n-violating, $\mathcal{T}'_n$ has to express that each instance of A violates either (P1) or (P2). Let x be such an instance. The purpose of Line (16) is to decide whether (P1) or (P2) is violated at x. If the latter is chosen, the violation is enforced via Lines (28) and (29) similar to what was done in the previous section. If (P1) is chosen, Lines (17) to (27) ensure that there is a path $x_1, \dots, x_k$ in $\mathcal{I}$ starting at x of length at most m such that the X-values along this path do *not* describe the first k bits of the consecutive values of a 2^n-bit counter counting from 0 to $2^{2^n} - 1$. Due to Lines (9) to (12) of $\mathcal{T}_n$, we know that the first 2^n X-values along $x_1, \dots, x_k$ are null as required. Hence, a failure of counting on the path $x_1, \dots, x_k$ can only be due to a failure of incrementing the counter X.

Lines (17) and (18) mark the place where incrementation fails using the concept name M_0. More precisely, the domain element marked with M_0 is a bit such that the corresponding bit in the consecutive X-sequence violates incrementation. There are two ways in which this may happen: first, there may be no 0-bit lower than the bit marked with M_0, but the corresponding bit in the following X-sequence is not toggled. Second, there may be a 0-bit lower than the bit marked with M_0, but the corresponding bit in the following X-sequence is toggled. These two cases are distinguished by Lines (21) and (22). In the first case, the case of X is stored in NX. In the second case, the toggled value of X is stored in NX. The counter D is reset in Line (19) and incremented in Line (22) to identify the corresponding bit in the following X-sequence. Through lines (23) and (24), the value of NX is passed on all the way to this bit. Finally, Lines (16) and (26) ensure that the X-value of the corresponding bit coincides with NX.

Lemma 16.

1. *A concept C that is satisfiable w.r.t. $\mathcal{T}_n$ and such that all models of C and $\mathcal{T}$ are strongly n-violating is a witness concept for $\mathcal{T}_n$ and $\mathcal{T}'_n$;*
2. *If a concept C is a witness concept for $\mathcal{T}_n$ and $\mathcal{T}'_n$, then all tree models of C and $\mathcal{T}$ are strongly n-violating.*

We now define a witness concept for $\mathcal{T}_n$ and $\mathcal{T}'_n$, thus showing that $\mathcal{T}_n \cup \mathcal{T}'_n$ is indeed not a conservative extension of $\mathcal{T}_n$. For $1 \leq i \leq m$, let b_i denote the i-th bit of a 2^n-bit counter counting from 0 to $2^{2^n} - 1$. Then we set:

$$\begin{array}{rcl} D_0 & := & \neg B \\ D_{i+1} & := & \exists r.D_i \sqcap \exists s.D_i \end{array}$$

[1] We deliberately confuse the name of the counter and the name of the concept name representing the truth value of bits.

$$\top \sqsubseteq \forall r.\neg A \sqcap \forall s.\neg A \quad (7)$$
$$B \sqsubseteq B \quad (8)$$
$$\top \sqsubseteq \forall (r \cup s).(C{+}{+}) \quad (9)$$
$$A \sqsubseteq (C = 0) \sqcap F \quad (10)$$
$$F \sqcap (C < 2^n) \sqsubseteq \forall (r \cup s).F \quad (11)$$
$$F \sqsubseteq \neg X \quad (12)$$
$$(C = 0) \sqsubseteq \neg Z \quad (13)$$
$$\neg X \sqcup Z \sqsubseteq \forall r.(\neg(C = 0) \rightarrow Z) \quad (14)$$
$$X \sqcap \neg Z \sqsubseteq \forall r.(\neg(C = 0) \rightarrow \neg Z) \quad (15)$$

$$A \sqsubseteq P_1 \sqcup P_2 \quad (16)$$
$$P_1 \sqsubseteq M_0 \sqcup \exists r.P_1 \sqcup \exists s.P_1 \quad (17)$$
$$P_1 \sqcap (C = 2^n - 1) \sqsubseteq Z \sqcup \neg X \quad (18)$$
$$M_0 \sqsubseteq M \sqcap (D = 0) \quad (19)$$
$$M_0 \sqcap \neg Z \sqsubseteq NX \leftrightarrow X \quad (20)$$
$$M_0 \sqcap Z \sqsubseteq NX \leftrightarrow \neg X \quad (21)$$
$$M \sqcap (D < 2^n) \sqsubseteq \forall (r \cup s).(D{+}{+}) \quad (22)$$
$$M \sqcap (D < 2^n) \sqcap NX \sqsubseteq \exists (r \cup s).(M \sqcap NX) \quad (23)$$
$$M \sqcap (D < 2^n) \sqcap \neg NX \sqsubseteq \exists (r \cup s).(M \sqcap \neg NX) \quad (24)$$
$$M \sqcap (D = 2^n - 1) \sqcap NX \sqsubseteq X \quad (25)$$
$$M \sqcap (D = 2^n - 1) \sqcap \neg NX \sqsubseteq \neg X \quad (26)$$
$$M \sqcap (D < 2^n - 1) \sqcap (C = 2^n - 1) \sqsubseteq Z \sqcup \neg X \quad (27)$$
$$P_2 \sqcap (\neg(C = 2^n - 1) \sqcup Z \sqcup \neg X) \sqsubseteq \forall r.P_2 \sqcup \forall s.P_2 \quad (28)$$
$$P_2 \sqcap (C = 2^n - 1) \sqcap \neg Z \sqcap X \sqsubseteq B \quad (29)$$

Figure 4: The TBoxes $\mathcal{T}_n$ (left) and $\mathcal{T}'_n$ (right).

$$C_i \quad := \quad A \sqcap D_i \sqcap \prod_{w \in \{r,s\}^i, 1 \leq i \leq n, b_i = 0} \forall w.\neg X \quad \sqcap \prod_{w \in \{r,s\}^i, 1 \leq i \leq n, b_i = 1} \forall w.X$$

It can be checked that the length of C_n is $\mathcal{O}(2^{2^n \cdot 2^{2^n}})$. To establish Part (iii) of Theorem 3, it remains to prove that each witness concept for $\mathcal{T}_n$ and $\mathcal{T}'_n$ is of length at least 2^{m-1}. This can be done by establishing an analogue of Lemma 15.

Lower Complexity Bounds

Theorem 2 states that, given two TBoxes $\mathcal{T}_1$ and $\mathcal{T}_2$, it is 2ExpTime-hard to decide whether $\mathcal{T}_1 \cup \mathcal{T}_2$ is a conservative extension of $\mathcal{T}_1$. To prove this, we can build on the construction of triple exponential witness concepts in the previous section to reduce the word problem of exponentially space-bounded Alternating Turing Machines (ATMs). Given such an ATM M and a word w, we can refine the TBoxes given in Figure 4 into TBoxes $\mathcal{T}_1$ and $\mathcal{T}_2$ such that such that models of witness concepts encode successful computations of M on w. Then, $\mathcal{T}_1 \cup \mathcal{T}_2$ is a conservative extension of $\mathcal{T}_1$ iff M accepts w. For reasons of space limitation, we refer to (Ghilardi, Lutz, & Wolter 2006b) for details.

Related work

The problem to decide for two TBoxes $\mathcal{T}_1$ and $\mathcal{T}_2$ whether $\mathcal{T}_1 \cup \mathcal{T}_2$ is a conservative extension of $\mathcal{T}_1$ is closely related to the notion of a *uniform interpolant*. This, in turn, is an extension of the standard Craig interpolants requiring that the interpolant is uniform for all possible formulas in the antecedent (Pitts 1992). As uniform interpolation appears to be the most important notion related to the algorithmic problem we are concerned with in this paper and results on uniform interpolation might be useful for future research on conservative extensions in DLs, we briefly discuss the connection.

Definition 17 (Uniform interpolants for TBoxes). Given a TBox $\mathcal{T}$ and a signature $\Gamma \subseteq \mathsf{sig}(\mathcal{T})$, we call a TBox $\mathcal{T}_\Gamma$ over $\mathsf{sig}(\mathcal{T}) - \Gamma$ a *uniform interpolant of $\mathcal{T}$ with respect to* Γ if the following conditions hold:

- $\mathcal{T} \models \mathcal{T}_\Gamma$;
- for every implication $C \sqsubseteq D$ such that no symbol from Γ occurs in C, D, we have that $\mathcal{T} \models C \sqsubseteq D$ implies $\mathcal{T}_\Gamma \models C \sqsubseteq D$.

It is not difficult to see that $\mathcal{T}_1 \cup \mathcal{T}_2$ is a conservative extension of $\mathcal{T}_1$ if and only if $\mathcal{T}_1$ is a uniform interpolant for $\mathcal{T}_1 \cup \mathcal{T}_2$ with respect to $\mathsf{sig}(\mathcal{T}_2) - \mathsf{sig}(\mathcal{T}_1)$. Thus, if it is the case that, for every TBox $\mathcal{T}$ and signature Γ, there exists a uniform interpolant $\mathcal{T}_\Gamma$ of $\mathcal{T}$ w.r.t. Γ (and it is computable), then we have a procedure deciding whether $\mathcal{T}_1 \cup \mathcal{T}_2$ is a conservative extension of $\mathcal{T}_1$: compute the uniform interpolant and check whether it is logically equivalent to $\mathcal{T}_1$. The most important logics known to have uniform interpolation are intuitionistic logic, the Gödel-Loeb logic, Grzegorczyk-logic, the μ-calculus, and the modal logic K (Visser 1996; Ghilardi 1995; D'Agostino & Lenzi 2005; Pitts 1992). On the other hand, classical first-order logic, modal logic S4 and dynamic logic PDL do not have uniform interpolation (Ghilardi & Zawadowski 2002; 1995).

Unfortunately, we show in (Ghilardi, Lutz, & Wolter 2006b) that uniform interpolants for TBoxes need not exist. More precisely, if $\mathcal{T}$ is the TBox

$$\begin{array}{lll} \top \sqsubseteq \forall r.\neg A & \qquad & A \sqsubseteq B \\ B \sqsubseteq \exists r.C, & & C \sqsubseteq \exists r.B \\ B \sqsubseteq E, & & C \sqsubseteq \neg E. \end{array}$$

and $\Gamma = \{B, C\}$, then there exists no uniform interpolant of $\mathcal{T}$ w.r.t. Γ. Thus, the uniform interpolation approach to conservative extensions fails for $\mathcal{ALC}$ with TBoxes.

Recently, (Marx, Conradie, & ten Cate 2006) have established an exponential time procedure for computing uniform

interpolants for $\mathcal{ALC}$-concepts without reference to TBoxes (or, in modal logic terms, the local consequence relation of modal logic K). The paper also establishes an exponential upper bound for the size of uniform interpolants. This result is used in (Ghilardi, Lutz, & Wolter 2006a) to show that the following decision problem is co-NEXPTIME-complete: given two $\mathcal{ALC}$-concepts C_1 and C_2, is $C_1 \sqcap C_2$ a conservative extension of C_1? In other words, does the following hold for every concept D in the signature of C_1: if the subsumption relation $C_1 \sqcap C_2 \sqsubseteq D$ is valid, then the subsumption relation $C_1 \sqsubseteq D$ is valid.

Outlook

We have proposed several reasoning problems that are suitable for providing automated reasoning support when deciding whether a given extension of an ontology is well-behaved. Still, substantial research remains to be carried out to achieve feasibility of this approach in practice. First, one should try to refine the worst-case optimal algorithms presented in this paper into more practical algorithms that can be implemented and tested on real-world ontologies. Second, the complexity analysis should be extended from $\mathcal{ALC}$ to the more expressive DLs currently supported by DL reasoners such as $\mathcal{SHIQ}$. We believe that, as long at the DL under consideration has the tree-model property, modifications of the techniques introduced in this paper can form the basis of such a complexity analysis.

Additionally, the results presented in this paper suggest to search for more pragmatic reasoning problems that are similar to the ones proposed here, but computationally less complex. For example, a developer might be interested in having a conservative extension not for all concepts over a given signature, but only for concepts of a certain form (e.g., positive concepts, existential concepts, and universal concepts).

In a similar spirit, when considering DLs with number restrictions, the user might want to achieve a conservative extension regarding concepts not containing number restrictions and, at the same time, intend to obtain a non-conservative extension when concepts containing qualified number restrictions are involved. To see that there is a considerable difference between the two cases, we refer back to the example about web services given in this paper.

Acknowledgements

The second author was supported by the EU funded IST-2005-7603 FET Project Thinking Ontologies (TONES). The third author was partially supported by UK EPSRC grant no. GR/S63182/01.

References

Antoniou, G., and Kehagias, K. 2000. A note on the refinement of ontologies. *Int. J. of Int. Systems* 15:623–632.

Baader, F., and Sattler, U. 1999. Expressive number restrictions in description logics. *J. of Logic and Comp.* 9(3).

Baader, F.; Lutz, C.; Sturm, H.; and Wolter, F. 2002. Fusions of description logics and abstract description systems. *J. of Artificial Intelligence Research* 16:1–58.

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

Baader, F.; Horrocks, I.; and Sattler, U. 2005. Description logics as ontology languages for the semantic web. In *Mechanizing Mathematical Reasoning: Essays in Honor of Jörg Siekmann on the Occasion of His 60th Birthday*, number 2605 in LNAI, 228–248. Springer Verlag.

D'Agostino, G., and Lenzi, G. 2005. An axiomatization of bisimulation quantifiers via the μ-calculus. *Theoret. Comput. Sci.* 338(1-3):64–95.

Ghilardi, S., and Zawadowski, M. 1995. Undefinability of propositional quantifiers in the modal system S4. *Studia Logica* 55(2):259–271.

Ghilardi, S., and Zawadowski, M. 2002. *Sheaves, games, and model completions*, volume 14 of *Trends in Logic—Studia Logica Library*. Kluwer.

Ghilardi, S.; Lutz, C.; and Wolter, F. 2006a. Conservative extensions in modal logic. Manuscript.

Ghilardi, S.; Lutz, C.; and Wolter, F. 2006b. Did I damage my ontology? A case for conservative extensions in description logics. Available at http://www.csc.liv.ac.uk/~frank/publ/publ.html.

Ghilardi, S. 1995. An algebraic theory of normal forms. *Ann. Pure Appl. Logic* 71(3):189–245.

Guarino, N. 1998. Formal ontologies and information systems. In *Proc. of FOIS'1998*, 3–15. IOS Press.

Martin, D.; Paolucci, M.; McIlraith, S.; Burstein, M.; McDermott, D.; McGuinness, D.; Parsia, B.; Payne, T.; Sabou, M.; Solanki, M.; Srinivasan, N.; and Sycara, K. 2004. Bringing semantics to web services: The OWL-S approach. In *Proc. of SWSWPC 2004*.

Marx, M.; Conradie, W.; and ten Cate, B. 2006. Definitorially complete description logics. In *Proc. of KR'06*.

Pitts, A. M. 1992. On an interpretation of second-order quantification in first-order intuitionistic propositional logic. *J. Symbolic Logic* 57(1):33–52.

Pratt, V. 1979. Models of program logic. In *Proc. of FOCS'79*, 115–122.

Schild, K. D. 1991. A correspondence theory for terminological logics: Preliminary report. In *Proc. of IJCAI'91*, 466–471. Morgan Kaufmann.

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

Serafini, L.; Stuckenschmidt, H.; and Wache, H. 2005. A formal investigation of mapping languages for terminological knowledge. In *Proc. of IJCAI'05*.

Turski, W., and Maibaum, T. 1987. *The Specification of Computer Programs*. Addison-Wesley.

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

Visser, A. 1996. Uniform interpolation and layered bisimulation. In *Gödel '96 (Brno, 1996)*, volume 6 of *Lecture Notes in Logic*. Springer-Verlag. 139–164.