Conservative Extensions in Expressive Description Logics[*]

Carsten Lutz[1]**, Dirk Walther**[2]**, Frank Wolter**[2]
[1]Institut für Theoretische Informatik, TU Dresden, Germany
[2]Department of Computer Science, University of Liverpool, UK

Abstract

The notion of a conservative extension plays a central role in ontology design and integration: it can be used to formalize ontology refinements, safe mergings of two ontologies, and independent modules inside an ontology. Regarding reasoning support, the most basic task is to decide whether one ontology is a conservative extension of another. It has recently been proved that this problem is decidable and 2ExpTime-complete if ontologies are formulated in the basic description logic $\mathcal{ALC}$. We consider more expressive description logics and begin to map out the boundary between logics for which conservativity is decidable and those for which it is not. We prove that conservative extensions are 2ExpTime-complete in $\mathcal{ALCQI}$, but undecidable in $\mathcal{ALCQIO}$. We also show that if conservative extensions are defined model-theoretically rather than in terms of the consequence relation, they are undecidable already in $\mathcal{ALC}$.

1 Introduction

The design and integration of ontologies formulated in modern ontology languages such as OWL is a serious challenge. Experience shows that principled methodologies as well as automated reasoning support are required to ensure that the resulting ontologies are well-structured [6]. In the recent papers [4; 5; 3; 1], conservative extensions have been identified as a crucial notion for formalizing central tasks in ontology design and integration. Consequently, conservative extensions can play a key role in design and integration methodologies [3], and reasoning about conservative extensions can provide valuable support for the ontology designer.

Formally, an ontology $\mathcal{T}_1 \cup \mathcal{T}_2$ is a conservative extension of an ontology $\mathcal{T}_1$ w.r.t. a signature Γ iff every consequence of $\mathcal{T}_1 \cup \mathcal{T}_2$ formulated in Γ is already a consequence of $\mathcal{T}_1$. For example, assume that ontologies are formalized in a description logic (DL) such as OWL-DL and its fragments [2]. Then a signature is a set of concept and role names, an ontology

is a DL TBox, and a consequence of a TBox is a subsumption relationship between two concepts which follows from the TBox. Intuitively, $\mathcal{T}_1 \cup \mathcal{T}_2$ is a conservative extension of $\mathcal{T}_1$ w.r.t. Γ if adding $\mathcal{T}_2$ to $\mathcal{T}_1$ does not change the ontology $\mathcal{T}_1$ as far as concepts built only from concept and role names in Γ are concerned. We give three examples of ontology related tasks that can be understood in terms of conservative extensions.

– *Ontology refinement.* During ontology design, a frequent task is to add more details to a part of the ontology that has not yet been sufficiently described. Intuitively, such a refinement should have no impact on other, unrelated parts of the ontology. This requirement can be formalized by demanding that the refined ontology is a conservative extension of the original one w.r.t. the concept and role names that do not belong to the refined part [4; 1].

– *Ontology merging.* The most straightforward way to integrate two ontologies is to simply take their union. Such a merging should not compromise the original ontologies. One possible formalization of this requirement is to demand that the united ontology is a conservative extension of the component ontologies w.r.t. the set of all concept and role names used in the respective components. Weaker formalizations are obtained by excluding from the signature concept and role names for which an interaction between the component ontologies is expected (and intended) [4].

– *Defining Modules.* A module inside an ontology $\mathcal{T}$ that describes an independent part of the application domain can be defined as a subset $\mathcal{T}'$ of $\mathcal{T}$ such that $\mathcal{T}$ is a conservative extension of $\mathcal{T}'$ w.r.t. the concept names and role names that belong to $\mathcal{T}'$ [5].

The most basic reasoning task regarding conservative extensions is as follows: given ontologies $\mathcal{T}_1$ and $\mathcal{T}_2$ and a signature Γ, decide whether $\mathcal{T}_1 \cup \mathcal{T}_2$ is a conservative extension of $\mathcal{T}_1$ w.r.t. Γ. In the following, we refer to this task as *deciding conservative extensions*. In [4], this decision problem is investigated for the basic DL $\mathcal{ALC}$ and proved to be 2ExpTime-complete. The aim of the current paper is to *investigate conservative extensions in more expressive DLs such as the ones underlying the ontology language OWL-DL and to map out the boundary between decidable and undecidable.* Our main results are as follows: (i) in $\mathcal{ALCQI}$, the extension of $\mathcal{ALC}$ with inverse roles and qualifying number restrictions, decid-

---

Name	Syntax	Semantics
inverse role	r^-	$(r^{\mathcal{I}})^{-1}$
nominal	$\{a\}$	$\{a^{\mathcal{I}}\}$
negation	$\neg C$	$\Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}$
conjunction	$C \sqcap D$	$C^{\mathcal{I}} \cap D^{\mathcal{I}}$
at-most number restriction	$(\leqslant n\ r\ C)$	$\{d \mid \#\{e \mid (d,e) \in r^{\mathcal{I}} \wedge e \in C^{\mathcal{I}}\} \le n\}$

Figure 1: Syntax and semantics of $\mathcal{ALCQIO}$.

ing conservative extensions is 2-ExpTime complete and thus not more difficult than in $\mathcal{ALC}$; and (ii) if we further extend $\mathcal{ALCQI}$ with nominals, conservative extensions in the resulting DL $\mathcal{ALCQIO}$ are undecidable. This shows that conservative extensions in OWL-DL, of which $\mathcal{ALCQIO}$ is a fragment, is also undecidable. It also identifies $\mathcal{ALCQI}$ as a significant fragment of OWL-DL in which conservative extensions are still decidable.

In mathematical logic, there exist (at least) two versions of conservative extensions. One is based on the consequence relation as sketched above. An alternative, stronger version is defined in a model-theoretic way. We also consider deciding the latter kind of conservative extensions and show that, already in $\mathcal{ALC}$, this problem is highly undecidable. Details of all proofs can be found in the technical report [8].

2 Preliminaries

In DLs, *concepts* are inductively defined with the help of a set of *constructors*, starting with a set N_C of *concept names*, a set N_R of *role names*, and (possibly) a set N_I of *individual names*. In this paper, we consider the DL $\mathcal{ALCQIO}$ and its fragments. The constructors available in $\mathcal{ALCQIO}$ are shown in Figure 1. There, the inverse constructor is the only role constructor, whereas the remaining constructors are concept constructors. In Figure 1 and the remainder of this paper, we use $\#S$ to denote the cardinality of a set S, a and b to denote individual names, r and s to denote roles (i.e., role names and inverses thereof), A, B to denote concept names, and C, D to denote (possibly complex) concepts. For an inverse role $s = r^-$ we set $s^- := r$. As usual, we use $\top$ as abbreviation for an arbitrary (but fixed) propositional tautology, $\bot$ for $\neg\top$, $\sqcup$, $\rightarrow$, and $\leftrightarrow$ for the usual Boolean abbreviations, $(\geqslant n\ r\ C)$ (*at-least restriction*) for $\neg(\leqslant n-1\ r\ C)$ if $n > 0$ and for $\top$ if $n = 0$, $(= n\ r\ C)$ for $(\leqslant n\ r\ C) \sqcap (\geqslant n\ r\ C)$, $\exists r.C$ (*existential restriction*) for $(\geqslant 1\ r\ C)$, and $\forall r.C$ (*universal restriction*) for $(\leqslant 0\ r\ \neg C)$. We assume that the numbers inside number restrictions are coded in binary.

The DL that allows only for negation, conjunction, disjunction, and universal and existential restrictions is called $\mathcal{ALC}$. The availability of additional constructors is indicated by concatenation of a corresponding letter: $\mathcal{Q}$ stands for number restrictions; $\mathcal{I}$ stands for inverse roles, and $\mathcal{O}$ for nominals. This explains the name $\mathcal{ALCQIO}$, and also allows us to refer to its sublanguages in a simple way.

The formulation 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 $C \sqsubseteq D$.

The semantics of $\mathcal{ALCQIO}$-concepts is defined in terms of an *interpretation* $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$. The *domain* $\Delta^{\mathcal{I}}$ is a non-empty set of individuals and the *interpretation function* $\cdot^{\mathcal{I}}$ maps each concept name $A \in N_C$ to a subset $A^{\mathcal{I}}$ of $\Delta^{\mathcal{I}}$, each role name $r \in N_R$ to a binary relation $r^{\mathcal{I}}$ on $\Delta^{\mathcal{I}}$, and each individual name $a \in N_I$ to an individual $a^{\mathcal{I}} \in \Delta^{\mathcal{I}}$. The extension of $\cdot^{\mathcal{I}}$ to inverse roles and arbitrary concepts is defined inductively as shown in the third column of Figure 1.

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 $C \sqsubseteq D$.

Despite the fact that individual names are closer to constants than to predicates, we henceforth use the term *predicates* to refer to elements of $N_C \cup N_R \cup N_I$. A *signature* is a finite set of predicates. The signature $sig(\mathcal{T})$ *of a TBox* $\mathcal{T}$ is the set of all predicates that occur in $\mathcal{T}$. Given a description logic $\mathcal{L}$ and a signature Γ, we use $\mathcal{L}(\Gamma)$ to denote the set of $\mathcal{L}$-concepts that use only predicates from Γ.

Definition 1 (Conservative Extension) *Let $\mathcal{T}_1$ and $\mathcal{T}_2$ be TBoxes formulated in a DL $\mathcal{L}$, and let $\Gamma \subseteq sig(\mathcal{T}_1)$ be a signature. Then $\mathcal{T}_1 \cup \mathcal{T}_2$ is a Γ-conservative extension of $\mathcal{T}_1$ if for all $C_1, C_2 \in \mathcal{L}(\Gamma)$, we have $\mathcal{T}_1 \models C_1 \sqsubseteq C_2$ iff $\mathcal{T}_1 \cup \mathcal{T}_2 \models C_1 \sqsubseteq C_2$.*

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

If $\Gamma = sig(\mathcal{T}_1)$, we simply drop Γ and only talk about conservative extensions. It is not difficult to see that an alternative definition of conservative extensions is as follows: $\mathcal{T}_1 \cup \mathcal{T}_2$ is a Γ-conservative extension of $\mathcal{T}_1$ iff each concept $C \in \mathcal{L}(\Gamma)$ that is satisfiable relative to $\mathcal{T}_1$ is satisfiable relative to $\mathcal{T}_1 \cup \mathcal{T}_2$. Therefore, a concept $C \in \mathcal{L}(\Gamma)$ that is satisfiable relative to $\mathcal{T}_1$, but not relative to $\mathcal{T}_1 \cup \mathcal{T}_2$ witnesses that $\mathcal{T}_1 \cup \mathcal{T}_2$ is *not* a conservative extension of $\mathcal{T}_1$. We call such a concept a *witness concept*.

Let us give an example for conservative extensions in the description logic $\mathcal{ALCQI}$. Assume that $\mathcal{T}_1$ is a TBox formalizing knowledge about universities:

Lecture	$\sqsubseteq$	$\exists$has_subject.Subject $\sqcap$ $\exists$given_by.Lecturer
Intro_TCS	$\sqsubseteq$	Lecture
Lecturer	$\sqsubseteq$	Professor $\sqcup$ TeachingAssistant
Lecturer	$\sqsubseteq$	$\exists$employed_by.University
University	$\sqsubseteq$	$\forall$employed_by$^-$.(Academic $\sqcup$ Admin)

The upper part of $\mathcal{T}_1$ describes university lectures, saying, e.g., that every introductory lecture on theoretical computer science (TCS) is a lecture. The lower part of $\mathcal{T}_1$ describes universities and their employees. Suppose now that we want to refine the part of the ontology that is concerned with lectures. We extend the signature by adding the concept names AutomataTheory and ComplexityTheory and state in $\mathcal{T}_2$ that

these subjects are discussed in every introductory TCS lecture. We also say that automata theory and complexity theory are different things:

$$\begin{aligned} \mathsf{Intro\_TCS} &\sqsubseteq \exists\mathsf{has\_subject.AutomataTheory} \\ \mathsf{Intro\_TCS} &\sqsubseteq \exists\mathsf{has\_subject.ComplexityTheory} \\ \bot &\sqsupseteq \mathsf{AutomataTheory} \sqcap \mathsf{ComplexityTheory}. \end{aligned}$$

Intuitively, this addition should have an impact on the upper part of $\mathcal{T}_1$ since it adds information about lectures, but it should not affect the lower part which is not concerned with lectures. This intuition can be formally captured by conservative extensions: if we choose Γ to be the set of all predicates used in the lower part of $\mathcal{T}_1$, then $\mathcal{T}_1 \cup \mathcal{T}_2$ is a Γ-conservative extension of $\mathcal{T}_1$. Thus, the lower part of $\mathcal{T}_1$ is not affected by the addition of $\mathcal{T}_2$. If we choose Γ to be the predicates in the upper part of $\mathcal{T}_1$, then $\mathcal{T}_1 \cup \mathcal{T}_2$ is not a Γ-conservative extension, a witness concept being

$$\mathsf{IntroTCS} \sqcap (\leqslant 1\ \mathsf{has\_subject}\ \top).$$

By considering these two cases of conservative extensions, the ontology designer can thus verify that his modification changes the TBox (only) in the intended way.

This example also shows that conservative extensions depend on the descripion logic $\mathcal{L}$: the TBoxes $\mathcal{T}_1$ and $\mathcal{T}_2$ are actually formulated in $\mathcal{ALCI}$ and we have seen that if Γ is the set of predicates in the upper part of $\mathcal{T}_1$, then $\mathcal{T}_1 \cup \mathcal{T}_2$ is not a Γ-conservative extension of $\mathcal{T}_1$. However, this only holds since we assumed $\mathcal{ALCQI}$ to be the underlying DL and thus allowed number restrictions in the witness concept. If we switch the underlying DL to $\mathcal{ALCI}$, then $\mathcal{T}_1 \cup \mathcal{T}_2$ *is* a Γ-conservative extension of $\mathcal{T}_1$, for the same Γ. In the next section, we investigate a purely model-theoretic version of convervative extension, which does not depend on the language.

2.1 Model Conservative Extensions

In mathematical logic and software specification [9], there are two different kinds of conservative extensions: one that is based on the consequence relation "$\models$" as in Definition 1 and one that is based on models only . For simplicity, we formulate this second notion only for the case where $\Gamma = \mathsf{sig}(\mathcal{T}_1)$.

Definition 2 (Model Conservative Extension) *Let $\mathcal{T}_1$ and $\mathcal{T}_2$ be TBoxes. We say that $\mathcal{T}_1 \cup \mathcal{T}_2$ is a* model conservative *extension of $\mathcal{T}_1$ iff for every model $\mathcal{I}$ of $\mathcal{T}_1$, there exists a model of $\mathcal{T}_1 \cup \mathcal{T}_2$ which can be obtained from $\mathcal{I}$ by modifying the interpretation of the predicates in $\mathsf{sig}(\mathcal{T}_2) \setminus \mathsf{sig}(\mathcal{T}_1)$ while leaving the predicates in $\mathsf{sig}(\mathcal{T}_1)$ fixed.*

To distinguish the two versions of conservative extensions, in this section we call the one based on "$\models$" a *deductive conservative extension*.

The notion of a model conservative extension is more strict than the deductive one: if $\mathcal{T}_1 \cup \mathcal{T}_2$ is a model conservative extension of $\mathcal{T}_1$, then it is clearly also a deductive conservative extension of $\mathcal{T}_1$, but the converse does not hold. To see the latter, consider the TBoxes

$$\mathcal{T}_1 = \{\exists r.\top \sqcap \exists s.\top = \top\}, \quad \mathcal{T}_2 = \{\exists r.A \sqcap \exists s.\neg A = \top\}.$$

It is not hard to see that $\mathcal{T}_1 \cup \mathcal{T}_2$ is a deductive conservative extension of $\mathcal{T}_1$ if $\mathcal{ALC}$ (or even $\mathcal{ALCQI}$) is the language for witness concepts, but it is not a model conservative extension. The stronger notion of model conservative extensions is of interest for *query answering modulo ontologies*. In this case, one might want to ensure that under the addition of any ABox $\mathcal{A}$ (over the signature of $\mathcal{T}_1$) the answers to queries (over the signature of $\mathcal{T}_1$) to $\mathcal{T}_1 \cup \mathcal{T}_2 \cup \mathcal{A}$ coincide with those to $\mathcal{T}_1 \cup \mathcal{A}$. This immedialy follows if $\mathcal{T}_1 \cup \mathcal{T}_2$ is a model conservative extension of $\mathcal{T}_1$, but it does not follow if $\mathcal{T}_1 \cup \mathcal{T}_2$ is just a deductive conservative extension of $\mathcal{T}_1$.

However, from an algorithmic viewpoint model conservative extensions are a problematic choice: we show that they are highly undecidable even in the basic description logic $\mathcal{ALC}$ (and therefore also in all its extensions). The proof of the following result is by a reduction from the semantic consequence problem in modal logic and can be found in [8].

Theorem 3 *It is Π_1^1-hard to decide whether for two given $\mathcal{ALC}$ TBoxes $\mathcal{T}_1$ and $\mathcal{T}_2$, the TBox $\mathcal{T}_1 \cup \mathcal{T}_2$ is a model conservative extension of $\mathcal{T}_1$.*

3 Decidability in $\mathcal{ALCQI}$

We give a tight complexity bound for deciding conservative extensions in $\mathcal{ALCQI}$. We use $|C|$ to denote the length of a concept C, and $|\mathcal{T}|$ to denote the *size* $\sum_{C \sqsubseteq D \in \mathcal{T}}(|C| + |D|)$ of a TBox $\mathcal{T}$.

Theorem 4 *It is 2-ExpTime-complete to decide conservative extensions in $\mathcal{ALCQI}$. In the case that $\mathcal{T}_1 \cup \mathcal{T}_2$ is not a conservative extension of $\mathcal{T}_1$, 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 $|C|$.*

The lower bound can be proved exactly in the same way as the 2-ExpTime lower bound for conservative extensions in $\mathcal{ALC}$ [4]. However, the lower bounds from $\mathcal{ALC}$ do *not simply transfer* to $\mathcal{ALCQI}$ and it is necessary to walk through the proof in [4] and check that it also works for the case of $\mathcal{ALCQI}$. In the following, we concentrate on proving the upper bound. It is established by devising a 2-ExpTime algorithm that, for convenience, decides *non*-conservative extensions.

We start by reminding that $\mathcal{ALCQI}$ has the tree model property. More precisely, a *tree interpretation* is an interpretation $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}}, <^{\mathcal{I}})$ equipped with an additional relation $<^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$ such that (i) $(\Delta^{\mathcal{I}}, <^{\mathcal{I}})$ is a tree, (ii) $\bigcup_{r \in \mathsf{N_R}} r^{\mathcal{I}} \cup r^{-\mathcal{I}} = \ < \cup <^{-1}$, and (iii) $s^{\mathcal{I}}$ and $r^{\mathcal{I}}$ are disjoint for all distinct roles s and r. In $\mathcal{ALCQI}$, every concept C that is satisfiable relative to a TBox $\mathcal{T}$ is satisfiable in a *tree model* of $\mathcal{T}$, i.e., a model of $\mathcal{T}$ that is a tree interpretation [7]. In this section, when talking of an interpretation or model of a TBox we always mean a tree interpretation.

To develop the algorithm for deciding non-conservative extensions in $\mathcal{ALCQI}$, we introduce a new kind of witness for non-conservativity. The new witnesses are very similar to finite tree interpretations and easier to work with than witness concepts. For a signature Γ, let a *literal type S for Γ* be a subset of $\mathsf{lit}(\Gamma) := \{A, \neg A \mid A \in \Gamma \cap \mathsf{N_C}\}$ such that for each

$A \in \Gamma \cap \mathsf{N_C}$, $A \in S$ iff $\neg A \notin S$. A Γ-*role* is a role r such that r or r^- is in Γ.

Definition 5 (Γ-tree) *A Γ-tree $\mathfrak{T} = (W, <, L, O)$ is a finite intransitive tree $(W, <)$ such that each node $w \in W$ is labeled by a literal type $L(w)$ for Γ, each edge (w, w') is labeled by a Γ-role $L(w, w')$, and $O \subseteq W$ is a set of leafs of $(W, <)$.*

Essentially, a Γ-tree is a finite tree interpretation equipped with an additional unary predicate O denoting a subset of the leafs. The following definition provides a way to relate Γ-trees and actual interpretations.

Definition 6 (Γ-embedding) *Let $\mathfrak{T} = (W, <, L, O)$ be a Γ-tree with root $w \in W$, and $\mathcal{I}$ an interpretation with root $d \in \Delta^{\mathcal{I}}$. A Γ-embedding $f : \mathfrak{T} \to \mathcal{I}$ is an injection from W to $\Delta^{\mathcal{I}}$ such that*

- *$f(w) = d$,*
- *$L(v, v') = r$ iff $f(v) r^{\mathcal{I}} f(v')$, for all $v, v' \in W$ and Γ-roles r,*
- *$C \in L(v)$ iff $f(v) \in C^{\mathcal{I}}$, for all $v \in W$ and $C \in \mathsf{lit}(\Gamma)$,*
- *if $v \notin O$, then every $d' \in \Delta^{\mathcal{I}}$ with $f(v) r^{\mathcal{I}} d'$ for some Γ-role r is in the range of f.*

$\mathfrak{T}$ is called Γ-embeddable into $\mathcal{I}$ if there is a Γ-embedding $f : \mathfrak{T} \to \mathcal{I}$.

The definition illustrates that Γ-trees represent a (finite) initial part of (potentially infinite) tree interpretations. This explains the predicate O of Γ-trees: O marks those leafs in the Γ-tree that are not necessarily leafs in the tree interpretation $\mathcal{I}$ that we embed into. We can now establish Γ-trees as witnesses for non-conservativity.

Lemma 7 *$\mathcal{T}_1 \cup \mathcal{T}_2$ is not a conservative extension of $\mathcal{T}_1$ w.r.t. Γ iff there exists a Γ-tree $\mathfrak{T} = (W, <, L, O)$ which is Γ-embeddable into a model of $\mathcal{T}_1$ but not into any model of $\mathcal{T}_1 \cup \mathcal{T}_2$.*

The general idea behind the algorithm is as follows: by Lemma 7, to decide whether $\mathcal{T}_1 \cup \mathcal{T}_2$ is not a conservative extension of $\mathcal{T}_1$, it suffices to decide whether there exists a Γ-tree that is Γ-embeddable into a model of $\mathcal{T}_1$, but not into any model of $\mathcal{T}_1 \cup \mathcal{T}_2$. This is what our algorithm will do. Alas, we conjecture that there are cases in which the smallest such tree is 3-exponential in $|\mathcal{T}_1 \cup \mathcal{T}_2|$, and therefore a 2-exponential algorithm cannot simply try to construct such a tree. Instead, we check the existence of the Γ-tree by searching for certain witnesses for the existence of such a tree. Before we can introduce these witnesses (which should not be confused with Γ-trees as witnesses for non-conservativity), we need to introduce the notion of a type.

Definition 8 (Type) *Let $\mathcal{T}$ be a TBox. We use $\mathsf{cl}(\mathcal{T})$ to denote the smallest set that contains all concepts in $\mathcal{T}$ and is closed under single negations and under subconcepts. A $\mathcal{T}$-type t is a subset of $\mathsf{cl}(\mathcal{T})$ such that*

- *$\neg C \in t$ iff $C \notin t$, for all $\neg C \in \mathsf{cl}(\mathcal{T})$;*
- *$C_1 \sqcap C_2 \in t$ iff $C_1 \in t$ and $C_2 \in t$, for all $C_1 \sqcap C_2 \in \mathsf{cl}(\mathcal{T})$.*

Given an interpretation $\mathcal{I}$ and $u \in \Delta^{\mathcal{I}}$, the set

$$t_{\mathcal{I}}^{\mathcal{T}}(u) = \{C \in \mathsf{cl}(\mathcal{T}) \mid u \in C^{\mathcal{I}}\}$$

is a $\mathcal{T}$-type. In what follows, we will not always distinguish between the type t and the conjunction of all members of t. We now introduce a witness for the existence of a Γ-tree that is Γ-embeddable into a model of $\mathcal{T}_1$, but not into any model of $\mathcal{T}_1 \cup \mathcal{T}_2$. To avoid writing sub- and superscripts, from now on we assume the input $\mathcal{T}_1$, $\mathcal{T}_2$, and Γ to be fixed.

Definition 9 (Root pair, Internal pair) *A root pair (t, U) consists of a $\mathcal{T}_1$-type t and a set U of $\mathcal{T}_1 \cup \mathcal{T}_2$-types. An internal pair $(t' \to_r t, U)$ consists of a Γ-role r, $\mathcal{T}_1$-types t' and t, and a function U mapping each $\mathcal{T}_1 \cup \mathcal{T}_2$-type to a set of $\mathcal{T}_1 \cup \mathcal{T}_2$-types.*

Intuitively, each (root or internal) pair encodes relevant information about possible embeddings of a Γ-tree into models of $\mathcal{T}_1$ and $\mathcal{T}_1 \cup \mathcal{T}_2$. This is made precise by the notion of realizability.

Definition 10 (Realizable root pair) *Let $\mathfrak{T} = (W, <, L, O)$ be a Γ-tree. A root pair (t, U) is realized by $\mathfrak{T}$ iff*

1. *there exist a model $\mathcal{I}$ of $\mathcal{T}_1$ with root $d \in t^{\mathcal{I}}$ and a Γ-embedding $f : \mathfrak{T} \to \mathcal{I}$;*
2. *for every $\mathcal{T}_1 \cup \mathcal{T}_2$-type s, we have $s \in U$ iff there exist a model $\mathcal{I}$ of $\mathcal{T}_1 \cup \mathcal{T}_2$ with root $d \in s^{\mathcal{I}}$ and a Γ-embedding $f : \mathfrak{T} \to \mathcal{I}$.*

While root pairs encode information about possible embeddings of a Γ-tree into models of $\mathcal{T}_1$ and $\mathcal{T}_1 \cup \mathcal{T}_2$, internal pairs encode information about possible embeddings of a Γ-tree into *rooted submodels* of models of $\mathcal{T}_1$ and $\mathcal{T}_1 \cup \mathcal{T}_2$. In the following, if $\mathcal{I}$ is a (tree) interpretation and $d \in \Delta^{\mathcal{I}}$, we write $\mathcal{I}_d$ to denote the sub-tree interpretation of $\mathcal{I}$ rooted at d.

Definition 11 (Realizable internal pair) *Let $\mathfrak{T} = (W, <, L, O)$ be a Γ-tree. An internal pair $(t' \to_r t, U)$ is realized by $\mathfrak{T}$ iff*

- *there exist a model $\mathcal{I}$ of $\mathcal{T}_1$ and $d', d \in \Delta^{\mathcal{I}}$ such that $d' \in (t')^{\mathcal{I}}$, $d' r^{\mathcal{I}} d$, $d \in t^{\mathcal{I}}$, and there is a Γ-embedding $f : \mathfrak{T} \to \mathcal{I}_d$;*
- *for all $\mathcal{T}_1 \cup \mathcal{T}_2$-types s, s', we have $s' \in U(s)$ iff there exist a model $\mathcal{I}$ of $\mathcal{T}_1 \cup \mathcal{T}_2$ and $d', d \in \Delta^{\mathcal{I}}$ such that $d' \in (s')^{\mathcal{I}}$, $d' r^{\mathcal{I}} d$, $d \in s^{\mathcal{I}}$, and there is a Γ-embedding $f : \mathfrak{T} \to \mathcal{I}_d$.*

A (root or internal) pair is realizable if there exists a Γ-tree $\mathfrak{T}$ which realizes it.

Observe that internal pairs store information not only about the element $d \in \Delta^{\mathcal{I}}$ to which the root of $\mathfrak{T}$ is mapped, but also comprise the type t' of the predecessor d' of d in $\mathcal{I}$ and the (unique!) role r which connects d' and d. This is necessary due to the presence of inverse roles and number restrictions and bears some similarity to the *double blocking* technique in tableau algorithms; see [7]. Also note that the U-component of internal pairs is a function rather than a set because, intuitively, the possible types of d in models of $\mathcal{T}_1 \cup \mathcal{T}_2$ depend on the type of the predecessor d' in such models.

Let us now describe the algorithm. By Lemma 7 and definition of realizability, there exists a realizable root pair of the

form $(t, \emptyset)$ iff $\mathcal{T}_1 \cup \mathcal{T}_2$ is not a conservative extension of $\mathcal{T}_1$ w.r.t. Γ. The algorithm for deciding non-conservative extensions searches for such a root pair. The easiest case is that a root pair $(t, \emptyset)$ is realized by a *singleton* Γ-*tree*, i.e., a Γ-tree that consists of only a single node. This special case is tested first. If the test is not successful, we must check whether there is a root pair $(t, \emptyset)$ that is realized by a non-singleton tree $\mathfrak{T} = (W, <, L, O)$. Assume that this is the case and that the root of $\mathfrak{T}$ is w. Then each subtree of $\mathfrak{T}$ rooted at a succesor node w' of w realizes an internal pair $(\hat{t}' \to_{\hat{r}} \hat{t}, \hat{U})$ with $\hat{t}' = t$ and $\hat{r} = L(w, w')$. Intuitively, this means that we can check realization of the root pair $(t, \emptyset)$ in $\mathfrak{T}$ based on the realization of internal pairs in trees of strictly smaller height. Similarly, we can check the realizability of *internal* pairs in a Γ-tree based on the realizability of internal pairs in Γ-trees of strictly smaller height. Based on these observations, our algorithm repeatedly generates internal pairs that are realized by Γ-trees of larger and larger height until all such pairs are generated. It then checks whether there exists a root pair $(t, \emptyset)$ that is realizable based on the generated internal pairs. The following definition formalizes one step of the algorithm in which root pairs or new internal pairs are generated from an existing set of internal pairs.

In the following, if $\mathfrak{T}$ is a Γ-tree and $w \in W$, we write $\mathfrak{T}_w$ to denote the sub-tree of $\mathfrak{T}$ rooted at w.

Definition 12 (One step) *Let $\mathcal{R}$ be a set of internal pairs. A root pair (t, U) (resp. internal pair $(t' \to_r t, U)$) can be obtained in one step from $\mathcal{R}$ if there exists a Γ-tree $\mathfrak{T} = (W, <, L, O)$ with root w such that*

- *$\mathfrak{T}$ realizes (t, U) (resp. $(t' \to_r t, U)$);*

- *for all $w' \in W$ with $w < w'$, there exists an internal pair $p = (\hat{t}' \to_{\hat{r}} \hat{t}, \hat{U}) \in \mathcal{R}$ such that $\hat{t}' = t$, $\hat{r} = L(w, w')$, and p is realized by $\mathfrak{T}_{w'}$.*

The details of our algorithm are given in Figure 2, where

$$m_{\mathcal{T}_1, \mathcal{T}_2} := 2 \times |\mathcal{T}_1 \cup \mathcal{T}_2| \times 2^{3 \times |\mathcal{T}_1 \cup \mathcal{T}_2|}.$$

Intuitively, considering only a subset of $\mathcal{R}_i$ of cardinality $m_{\mathcal{T}_1, \mathcal{T}_2}$ means that we limit our attention to Γ-trees of outdegree $m_{\mathcal{T}_1, \mathcal{T}_2}$. This is justified by the following lemma.

Lemma 13 *If $\mathcal{T}_1 \cup \mathcal{T}_2$ is not a conservative extension of $\mathcal{T}_1$ w.r.t. Γ, then there exists a root pair $(t, \emptyset)$ realized by a Γ-tree $\mathfrak{T}$ of outdegree at most $m_{\mathcal{T}_1, \mathcal{T}_2}$.*

It remains to be shown that each step of the algorithm can be carried out effectively and that the algorihm yields the 2-EXPTIME upper bound stated in Theorem 4. We start with the former. The proof of the following lemma relies on the fact that satisfiability in $\mathcal{ALCQI}$ relative to TBoxes can be decided in EXPTIME [10].

Lemma 14 *It can be checked in 2-exponential time (in the size of $\mathcal{T}_1$, $\mathcal{T}_2$) whether a (root or internal) pair can be obtained in one step from a set $\mathcal{R}$ of realizable internal pairs with $|\mathcal{R}| \leq m_{\mathcal{T}_1, \mathcal{T}_2}$.*

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

1. Determine the set $\mathcal{R}_0^r$ of root pairs realized by singleton Γ-trees. If $\mathcal{R}_0^r = \emptyset$, then reject the input (because $\mathcal{T}_1$ is not satisfied in any model). If $\mathcal{R}_0^r$ contains a root pair (t, U) such that $U = \emptyset$, then accept. Else,

2. Determine the set $\mathcal{R}_0$ of internal pairs realizable by singleton Γ-trees. If $\mathcal{R}_0 = \emptyset$, then reject the input. Else,

3. Generate the sequence $\mathcal{R}_1, \mathcal{R}_2, \dots$ of sets of internal pairs such that

$$\mathcal{R}_{i+1} = \mathcal{R}_i \cup \mathcal{R}_i',$$

where $\mathcal{R}_i'$ is the set internal pairs which can be obtained from some non-empty subset of $\mathcal{R}_i$ of cardinality not exceeding $m_{\mathcal{T}_1, \mathcal{T}_2}$ in one step. This is done until $\mathcal{R}_i = \mathcal{R}_i \cup \mathcal{R}_i'$. Then accept the input if there exists a root pair (t, U) with $U = \emptyset$ which can be obtained in one step from some subset of $\mathcal{R}_i$ of cardinality not exceeding $m_{\mathcal{T}_1, \mathcal{T}_2}$. If no such root pair exists, reject the input.

Figure 2: Algorithm for non-conservativeness w.r.t. Γ.

The number of internal pairs is bounded double exponentially in the size of $|\mathcal{T}_1 \cup \mathcal{T}_2|$. Therefore, the third step of the algorithm stabilizes after at most double exponentially many rounds. Together with Lemma 14, it follows that our algorithm is a 2-ExpTime one.

Theorem 15 *The algorithm in Figure 2 accepts input $\mathcal{T}_1$, $\mathcal{T}_2$, Γ iff $\mathcal{T}_1 \cup \mathcal{T}_2$ is not a conservative extension of $\mathcal{T}_1$ w.r.t. Γ. It runs in 2-exponential time.*

To show the upper bound on the size of witness concepts stated in Theorem 4, we proceed as follows: first, we observe that if the algorithm finds a realizable root pair $(t, \emptyset)$, then this pair is realized by a Γ-tree of at most double exponential depth and single exponential outdegree. Second, we show how to convert such a Γ-tree into a witness concept of three-exponential size.

4 Undecidability in $\mathcal{ALCQIO}$

We show that conservative extensions are undecidable in $\mathcal{ALCQIO}$. The proof is by a reduction of the following undecidable tiling problem.

Definition 16 *A domino system $D = (T, H, V, R, L, T, B)$ consists of a finite set T of* tiles, *horizontal and vertical matching relations $H, V \subseteq T \times T$, and sets $R, L, T, B \subseteq T$ of* right *tiles,* left *tiles,* top *tiles, and* bottom *tiles. A solution to D is a triple (n, m, τ) where $n, m \in \mathbb{N}$ and $\tau : \{0, \dots, n-1\} \times \{0, \dots, m-1\} \to T$ such that the following hold:*

1. *$(\tau(i, j), \tau(i+1, j)) \in H$, for all $i < n$ and $j \leq m$;*

2. *$(\tau(i, j), \tau(i, j+1)) \in V$, for all $i \leq n$ and $j < m$;*

3. *$\tau(0, j) \in L$ and $\tau(n, j) \in R$, for all $j \leq m$;*

4. $\tau(i,0) \in B$ and $\tau(i,m) \in T$, for all $i \leq n$.

Using proof methods from [11], it is easy to show that it is undecidable whether a given domino system D has a solution. We show how to convert a domino system D into TBoxes $\mathcal{T}_1$ and $\mathcal{T}_2$ such that D has a solution iff $\mathcal{T}_1 \cup \mathcal{T}_2$ is not a conservative extension of $\mathcal{T}_1$. In particular, models of witness concepts will correspond to solutions of D.

Let $D = (T, H, V, R, L, T, B)$ be a domino system. The TBox $\mathcal{T}_1$ uses the following signature: an indidual name o, role names r_x and r_y, concept names top, bottom, left, and right and each element of T as a concept name. The TBox $\mathcal{T}_1$ contains the following:

- The roles r_x, r_y, and their inverses are functional:

$$\top \sqsubseteq (\leqslant 1 \ r \ \top), \text{ for } r \in \{r_x, r_y, r_x^-, r_y^-\}$$

- Every position in the $n \times m$ grid is labeled with exactly one tile and the matching conditions are satisfied:

$$\top \ \sqsubseteq \ \bigsqcup_{t \in T}(t \sqcap \bigsqcap_{t' \in T, \ t' \neq t} \neg t')$$

$$\top \ \sqsubseteq \ \bigsqcap_{t \in T}(t \rightarrow (\bigsqcup_{(t,t') \in H} \forall r_x.t' \sqcap \bigsqcup_{(t,t') \in V} \forall r_y.t'))$$

- The concepts left, right, top, bottom mark the boundaries of the grid in the expected way:

$$\begin{aligned} \text{right} \ &\sqsubseteq \ \neg \exists r_x.\top \sqcap \forall r_y.\text{right} \sqcap \forall r_y^{-1}.\text{right} \\ \neg\text{right} \ &\sqsubseteq \ \exists r_x.\top \end{aligned}$$

and similarly for left, top, and bottom.

- The individual name o marks the origin:

$$\{o\} \sqsubseteq \text{left} \sqcap \text{bottom}.$$

The TBox $\mathcal{T}_2$ introduces two new concept names Q and P. It contains the following two concept inclusions:

$$\{o\} \sqsubseteq Q \sqsubseteq \exists r_x.Q \sqcup \exists r_y.Q \sqcup (\exists r_x.\exists r_y.P \sqcap \exists r_y\exists r_x.\neg P)$$

The idea behind this definition of $\mathcal{T}_2$ is to enforce that models $\mathcal{I}$ of witness concepts are such that (i) there is no infinite outgoing r_x/r_y-path starting at $o^{\mathcal{I}}$ and (ii) r_x and r_y commute in the connected part of $\mathcal{I}$ rooted at $o^{\mathcal{I}}$. This is achieved as follows: if (i) is violated, then we can find an assignment of Q in $\mathcal{I}$ that satisfies $\mathcal{T}_2$. Similarly, if (ii) is violated, then we can find an assignment of (Q and) P in $\mathcal{I}$ that satisfies $\mathcal{T}_2$.

It can be checked that, as intended, D has a solution iff $\mathcal{T}_1 \cup \mathcal{T}_2$ is not a conservative extension of $\mathcal{T}_1$. Here, we only show how to construct a witness concept in the case that D has a solution. Such a witness concept C has to ensure that for all models $\mathcal{I}$ of C and $\mathcal{T}_1$, the connected part of $\mathcal{I}$ rooted at $o^{\mathcal{I}}$ is isomorphic to the $n \times m$-grid.

For every word $w \in \{r_x, r_y\}^*$, denote by $\overleftarrow{w}$ the word that is obtained by reversing w and then adding $\cdot^-$ to each symbol. Let $|w|_r$ denote the number of occurrences of the symbol r in w. Now, C is the conjunction of

$$\{o\} \sqcap \forall r_x^n.\text{right} \sqcap \forall r_y^m.\text{top}$$

and for every $w \in \{r_x, r_y\}^*$ such that $|w|_{r_x} < n$ and $|w|_{r_y} < m$, the concept

$$\exists(w \cdot r_x r_y r_x^- r_y^- \cdot \overleftarrow{w}).\{o\},$$

where $\exists w.D$ abbreviates $\exists r_1. \cdots \exists r_k.D$ if $w = r_1 \cdots r_k$. It is readily checked that C enforces an $n \times m$-grid as required.

Theorem 17 *In $\mathcal{ALCQIO}$, conservative extensions are undecidable.*

Note that the theorem applies even to the case where $\Gamma = \text{sig}(\mathcal{T}_1)$ and we allow $(\leqslant 1 \ r \ \top)$ as the only form of number restriction.

5 Conclusion

Although the high computational complexity suggests that efficient tools for deciding conservative extensions will be difficult to attain, our results and techniques lay theoretical foundations that are important for practical applications of conservative extensions. These could be based on approximations, semi-decision procedures, and on syntactic restrictions in a normative framework such as [3]. Finally, transitive roles are a main ingredient of DLs underlying OWL-DL. It remains an important open problem to investigate conservative extensions for DLs containing transitive roles and role hierarchies.

References

[1] G. Antoniou and K. Kehagias. A note on the refinement of ontologies. *International Journal of Intelligent Systems*, 15:623–632, 2000.

[2] F. Baader, I. Horrocks, and U. Sattler. Description logics as ontology languages for the semantic web. In *Mechanizing Mathematical Reasoning*, volume 2605 of *LNAI*, pages 228–248. Springer, 2005.

[3] B. Cuenca-Grau, I. Horrocks, O. Kutz, and U. Sattler. Will my ontologies fit together? In *Proc. of DL06*, number 189 in CEUR-WS (http://ceur-ws.org/), 2006.

[4] S. Ghilardi, C. Lutz, and F. Wolter. Did I damage my ontology? A case for conservative extensions in description logic. In *Proc. of KR2006*, pages 187–197, 2006.

[5] B. Grau, B. Parsia, E. Sirin, and A. Kalyanpur. Modularity and web ontologies. In *Proc. of KR2006*, pages 198–208, 2006.

[6] N. Guarino and C. Welty. An overview of OntoClean. In *Handbook of Ontologies*. Springer, 2004.

[7] I. Horrocks, U. Sattler, and S. Tobies. Practical reasoning for expressive description logics. In *Proc. of LPAR'99*, number 1705 in LNAI, pages 161–180. Springer, 1999.

[8] C. Lutz, D. Walther, and F. Wolter. Conservative extensions in expressive description logics. Technical report, 2006. Available online from http://www.csc.liv.ac.uk/∼frank.

[9] T. Maibaum. Conservative extensions, interpretations between theories and all that! In *Proc. of 7th CAAP/FASE Conf. on Theory and Practice of Software Development*, pages 40–66, 1997.

[10] S. Tobies. *Complexity Results and Practical Algorithms for Logics in Knowledge Representation*. PhD thesis, RWTH Aachen, 2001.

[11] P. van Emde Boas. The convenience of tilings. In A. Sorbi, editor, *Complexity, Logic, Recursion Theory*. 1997.