Unification, matching and disunification in the description logic $\mathcal{EL}$

an Stelle einer Habilitationsschrift

vorgelegt an der
Technischen Universität Dresden
Fakultät Informatik

eingereicht von
Dr. Barbara Morawska

Betreuender Hochschullehrer: Prof. Dr.-Ing. Franz Baader

Dresden, Januar 2017

Ich erkläre, dass die Habilitationsschrift vom mir selbst und ohne andere als die darin angegebenen Hilfsmittel angefertigt sowie die wörtlich oder inhaltlich übernommenen Stellen als solche gekennzeichnet wurden.

Hereby I declare that the habilitation thesis was written by myself and with no other means than those indicated in it. All literal or content citations are indicated as such.

Barbara Morawska

Contents

1 Introduction

The description logic $\mathcal{EL}$ belongs to a big family of Description Logics (DLs). DLs are logical formalisms designed specially to represent knowledge expressed in various concepts, definitions and statements describing different domains of interest. Such a formal, well-structured and high-level formalism should facilitate building intelligent applications automatizing many tasks concerned with using knowledge, maintaining it, extending it and reasoning based on this knowledge. Thus the research about DLs aims at a better understanding of the human way of thinking and reasoning (as an area of Artificial Intelligence), but on the other hand it is aimed at solving very practical problems of knowledge representation in various applications. One can mention for example the following areas, where DLs make significant contributions.

- Databases. DLs provide means for conceptual modeling of various entity-relationship models. They provide tools to analyze and optimize queries. They help at the conceptual level in integration of many data sources into one Domain Conceptual Schema [11, 54, 48].

- Semantic Web. DLs provide formal understanding of the expressions in the Web Ontology Language (OWL) and provide the established standards for the subsets of OWL.[1]

- Big biomedical ontologies like SNOMED CT[2] or GENE ONTOLOGY[3]. DLs provide tools to maintain and analyze these big knowledge bases.

The family of description logics comprises logics which differ one from the other with respect to their expressive power. The expressive power of a logic results from the language constructors that this logic offers and from the kind of axioms that can be expressed. There is a trade off between expressivity of a description logic and the complexity of reasonings that one can perform within such logic [46]. The very expressive description logics, which were proposed for the application in some real-word databases, have very bad *worst case* reasoning complexity, e.g., [38, 43].

[1] `https://www.w3.org/TR/owl2-profiles/`
[2] `http://www.ihtsdo.org/snomed-ct`
[3] `http://geneontology.org/`

On the other hand, some description logics with weak expressive power, show very good complexity of reasonings. Moreover they find applications in many areas where modeling of a domain does not require very rich and powerful logical constructs.

There are two of such small description logics which are of special interests for us. Namely, $\mathcal{EL}$ and $\mathcal{FL}_0$. As all other DLs, these logics provide a finite set of concept names and roles, from which they allow to construct more complex concept terms. The following table shows the constructors provided by $\mathcal{EL}$ or $\mathcal{FL}_0$ together with their interpretation $\mathcal{I}$ over a non-empty domain $\Delta^{\mathcal{I}}$.

name	notation	interpretation over a non-empty domain $\Delta^{\mathcal{I}}$
top	$\top$	$\Delta^{\mathcal{I}}$
concept name	A	$A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$
role name	r	$r^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$
intersection	$C \sqcap D$	$C^{\mathcal{I}} \cap D^{\mathcal{I}}$
value restriction	$\forall r.C$	$\{x \in \Delta^{\mathcal{I}} \mid$ for every y, $(x,y) \in r^{\mathcal{I}}$ implies $y \in C^{\mathcal{I}}\}$
existential restriction	$\exists r.C$	$\{x \in \Delta^{\mathcal{I}} \mid$ there is y, $(x,y) \in r^{\mathcal{I}}$ and $y \in C^{\mathcal{I}}\}$

Table 1.1: Constructors of $\mathcal{EL}$ and $\mathcal{FL}_0$

The description logic $\mathcal{EL}$ allows to construct concept terms from a set N_C of concept names and a set N_R of role names, using the constructors: $\top$ (top constructor), $\sqcap$ (concept intersection) and $\exists r.C$ (existential restriction). $\mathcal{FL}_0$ differs from $\mathcal{EL}$, in allowing $\forall r.C$ (value restriction) in place of the existential restriction.

For example, if `Pain` and `Head` are concept names and `location` is a role name, then $\texttt{Pain} \sqcap \exists \texttt{location}.\texttt{Head}$ is a concept term in $\mathcal{EL}$, which obviously describes a headache. On the other hand if `Doctor`, `Child` are concept names and `has_patient` is a role name, then $\texttt{Doctor} \sqcap \forall \texttt{has\_patient}.\texttt{Child}$ is an $\mathcal{FL}_0$-concept term, which describes pediatricians.

Since neither of these two small description logics has a negation, all concept terms constructed in these logics are consistent. The standard reasoning problem in these logics is deciding *subsumption* between concept terms.

Given two concept terms C, D, we say that D subsumes C (C is subsumed by D), $C \sqsubseteq D$, if for every interpretation $\mathcal{I}$, $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$.

For example the concept term $\exists \texttt{has\_patient}.(\texttt{Child} \sqcap \texttt{Female})$ is subsumed by a concept term $\exists \texttt{has\_patient}.\texttt{Child}$, because in all possible interpretations of concept names `Child`, `Female` and the role name `has_patient`, due to the properties of the logical constructors (defined in Table 1.1), the extension of $\exists \texttt{has\_patient}.(\texttt{Child} \sqcap \texttt{Female})$ will be a subset of the extension of $\exists \texttt{has\_patient}.\texttt{Child}$.

In both $\mathcal{FL}_0$ and $\mathcal{EL}$, the subsumption problem is decidable in polynomial time [46, 14].

Having defined subsumption between concepts, we define also the equivalence.

For two concept terms C, D, C is *equivalent* to D, $C \equiv D$ iff $C \sqsubseteq D$ and $D \sqsubseteq C$.[4]

Expressive power of the description logics $\mathcal{EL}$ and $\mathcal{FL}_0$ is determined by the above mentioned constructors, but also by a kind of axioms that can be expressed and assumed in order to capture more exactly the intended interpretation of a given domain of knowledge. The axioms in the form of statements of equivalences or of subsumptions between concept terms are a part of the so called *knowledge base*. The knowledge base contains usually a TBox (set of the just mentioned axioms), ABox (statements about individual elements of an interpretation) and sometimes also RBox (the statements about properties of relations between elements in an interpretation). In this thesis we will be concerned only with the first component of a knowledge base, i.e., a TBox, which is a set of equivalence or subsumption statements between concepts (in Section 5).

A TBox extends expressivity of a description logic and although similar, the logics $\mathcal{EL}$ and $\mathcal{FL}_0$ begin to differ much, when we extend their expressivity in this way. Standard reasoning tasks in $\mathcal{FL}_0$ become more complex, while $\mathcal{EL}$ retains a polynomial complexity. Therefore, in contrast to $\mathcal{FL}_0$, $\mathcal{EL}$ is used in practical applications. It is used to define biomedical ontologies. The biomedical ontologies like SNOMED CT or the GENE ONTOLOGY or part of GALEN[50] can be seen as TBoxes written in the language of $\mathcal{EL}$. An extension of $\mathcal{EL}$ became also a standard for a subset of OWL 2, OWL 2 EL[5],[10]. EL is also a subject of intensive research in the area of OBDA, [54].

Unification modulo an equational theory $\mathcal{E}$ is concerned with the question of how to make a pair of terms (or a set of such pairs) equivalent modulo $\mathcal{E}$, using substitution of terms for variables.

For example, the concept term $\texttt{Woman} \sqcap \texttt{Doctor} \sqcap \exists \texttt{has\_patient}.\texttt{Child}$ is not equivalent to the concept term $\texttt{Female} \sqcap \texttt{Human} \sqcap \texttt{Pediatritian}$, although they intuitively denote the same concept. We discover that they are in fact equivalent, if we see that `Woman` is equivalent to $\texttt{Female} \sqcap \texttt{Human}$ and `Pediatritian` is equivalent to $\texttt{Doctor} \sqcap \exists.\texttt{has\_patient}.\texttt{Child}$. Thus we can say that a mapping which substitutes `Woman` by the concept term $\texttt{Human} \sqcap \texttt{Female}$ and `Pediatritian` by the concept term $\texttt{Doctor} \sqcap \exists.\texttt{has\_patient}.\texttt{Child}$ shows the intuitive equivalence of the above concepts. From the point of view of unification the concept names `Woman` and `Pediatritian` are treated in this example as variables.

Such pairs of *intentionally equivalent* concept terms can occur, when differ-

[4] In this thesis we will use $\equiv$ (equivalence) in the meaning just defined. The equality sign $=$ denotes a syntactic identity.

[5] `http://www.w3.org/TR/2012/REC-owl2-profiles-20121211/#OWL_2_EL`

ent ontology engineers introduce extensions of an ontology independent of each other. Unification could be used here to detect and prevent such redundancies, by providing unifiers in the form of sets of definitions for the previously undefined concepts. Such a set of definitions can be used to *unfold* redundant concepts and replace them in an ontology with the result, or can be treated as an extension of a knowledge base.

Unification was first considered in the context of description logics as a new non-standard reasoning task with the possible application in detecting redundant concepts in big ontologies. The theory $\mathcal{E}$ which we consider here is the equational theory defined by equivalence of concepts modulo properties of a description logic. In the case of $\mathcal{EL}$ it is the theory of an idempotent Abelian monoid where unity is top, with monotonic operators [53], while in the case of $\mathcal{FL}_0$, it is the theory of an idempotent Abelian monoid with homomorphisms, [20].

At first the focus of this kind of research was put on the logic $\mathcal{FL}_0$. In [20] it was shown that the unification problem (modulo the equational theory of $\mathcal{FL}_0$) is decidable, but ExpTime-complete. In this thesis, we focus on unification in the description logic $\mathcal{EL}$ (i.e., modulo the equational theory of $\mathcal{EL}$), which turns out to be more promising for applications due to its better complexity.

Not much is known about the complexity of unification problem in more expressive DLs. If we add negation to $\mathcal{EL}$, then we obtain the description logic $\mathcal{ALC}$, which corresponds to the basic (multi)-modal logic K [52]. Decidability of unification in K is a long-standing open problem. Undecidability of unification in some extensions of K was shown in [55]. These undecidability results imply undecidability of unification in more expressive DLs (e.g., SHIQ, [39])

1.1 Overview of the thesis

Before delving into the main subject of the thesis, in Section 2 we briefly recall a syntactic characterization of subsumption. Each of our results for the unification in $\mathcal{EL}$ is based on a convenient characterization of subsumption in $\mathcal{EL}$ or in extensions thereof. Such characterizations contain basic intuitions needed to understand subsequent constructions.

This thesis presents a selection of 9 publications, containing our results on unification in $\mathcal{EL}$.

1. In Section 3 we present the *type zero* result and then prove decidability of unification in $\mathcal{EL}$. For the decidability result we present first a brute-force algorithm deciding unifiability of an $\mathcal{EL}$-unification problem. This algorithm does not provide a basis for a practical implementation, therefore we present also two other algorithms, which are more practical: one based on transformation rules and the other based on SAT-reduction. Both these

alogrithms show complementary behavior: the first is faster in obtaining a positive answer, and the second is faster obtaining the negative one. Papers: [18] (type zero, brute-force algorithm), [17] (SAT-based algorithm), [19] (rule-based algorithm).

2. In Section 4 we present unification in $\mathcal{EL}^{-\top}$, i.e., in $\mathcal{EL}$ without the top constructor. For the reasons explained later, removing top from constructors of $\mathcal{EL}$, brings us nearer to the practical applications of unification algorithms. Surprisingly, this small change in the logic, causes a significant increase in the complexity of a decision procedure. Paper: [4].

3. In Section 5, we present various attempts to extend the unification procedures to $\mathcal{EL}$ with a TBox, which might not be acyclic. Papers: [19] (unification in $\mathcal{EL}$ with acyclic TBoxes), [16] (matching in $\mathcal{EL}$ with general TBoxes), [8](unification in $\mathcal{EL}$ modulo cycle-restricted TBoxes), [6] (rule-based algorithm for this case), [9] (SAT-based algorithm for this case).

4. Finally, in Section 6, we present partial results on a generalization of unification to include negative constraints, i.e., disunification in $\mathcal{EL}$ (without considering a general or restricted TBox). Paper: [7] (dismatching and local disunification).

2 Subsumption in $\mathcal{EL}$

As was already mentioned in the Introduction, deciding subsumption between concept terms is the standard problem in the description logic $\mathcal{EL}$. The problem is decidable in polynomial time [1, 10]. Since all the results presented in this thesis depend on the properties of subsumption, we explain here those of them that are necessary to understand the following sections. All these properties follow from the semantic interpretation of the logical constructors as presented in Table 1.1 and the following semantic definition of subsumption.

Definition 1. Given two concept terms C, D, $C \sqsubseteq D$ iff for every interpretation $\mathcal{I}$ over a non-empty domain $\Delta^{\mathcal{I}}$, $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ holds.[1]

Concept intersection and existential restrictions are monotonic w.r.t. subsumption: for each concept terms C, D and a role name r, $C \sqcap D \sqsubseteq D$ and $\exists r.(C \sqcap D) \sqcap \exists r.D \sqsubseteq \exists r.(C \sqcap D)$.

Because of these monotonicity properties, the following reduction rules preserve the equivalence (subsumption in both directions) of concept terms.

For all $\mathcal{EL}$-concept terms C, D, each concept name A and each role name r:

- $C \sqcap \top \equiv \top \sqcap C \longrightarrow C$,
- $A \sqcap A \longrightarrow A$,
- if $C \sqsubseteq D$, then $\exists r.C \sqcap \exists r.D \longrightarrow \exists r.C$.

Because these rules preserve the equivalence between concept terms and any reduction chain of concept terms using these rules terminates, we can assume that the concept terms we are working with are reduced.

Associativity and commutativity of conjunction constructor allows to present each concept term C in the form of a conjunction:

$$A_1 \sqcap \cdots \sqcap A_n \sqcap \exists r_1.D_1 \sqcap \cdots \sqcap \exists r_m.D_m$$

where $A_1, \ldots, A_n$ are concept names and $D_1, \ldots, D_m$ are concept terms and $n, m \geq 0$.

Using this form of concept terms, we can formulate a syntactic characterization of the subsumption in $\mathcal{EL}$, in the following theorem.

[1] From Table 1.1, $C^{\mathcal{I}}, D^{\mathcal{I}}$ are sets of elements in $\Delta^{\mathcal{I}}$.

Theorem 1. *(characterization of subsumption)*
Let $C := A_1 \sqcap \cdots \sqcap A_n \sqcap \exists r_1.D_1 \sqcap \cdots \sqcap \exists r_m.D_m$ *and*
$D := B_1 \sqcap \cdots \sqcap B_k \sqcap \exists r_1.E_1 \sqcap \cdots \sqcap \exists r_l.E_l$.
Then $C \sqsubseteq D$ *iff*

- $\{B_1, \ldots, B_k\} \subseteq \{A_1, \ldots, A_n\}$ *and*
- *for each* $i \in \{1, \ldots, l\}$, *there is* $j \in \{1, \ldots, m\}$ *such that* $r_i = r_j$ *and* $D_j \sqsubseteq E_i$.

This theorem follows from a similar characterization of equivalence in $\mathcal{EL}$ in [45].

An important property of subsumption is that the order defined by its inverse is well-founded, [18].

Definition 2. Let C, D be $\mathcal{EL}$-concept terms. We say that $C \succ_{is} D$ (C is bigger than D w.r.t. subsumption inverse order) iff $C \sqsubset D$.

The well-foundedness of this order leads to a notion of *minimal unifiers*, (Definition 9).

3 Unification in $\mathcal{EL}$

In order to formally define unification in the description logic $\mathcal{EL}$, we divide the set of concept names N_C into disjoint subsets: set of concept constants N_c and a set of concept variables *Var*. Intuitively, the concepts in *Var* are those that have definitions or which we allow to be defined in an extension of the knowledge base.

A substitution σ is a mapping from *Var* into the set of $\mathcal{EL}$-concept terms. This mapping is extended to all $\mathcal{EL}$-concept terms in a standard way.

In the definition of an $\mathcal{EL}$-unification problem, instead of referring to the equivalence of concept terms, we refer to subsumption between such terms. Since the equivalence between concept terms is defined by two subsumptions between them holding in both directions, and since any subsumption can be defined as equivalence between some concept terms, we propose here an approach to unification in $\mathcal{EL}$ that is equivalent to the standard one, but turns out to be more convenient. We have adopted this approach for the first time in [8].

Definition 3. An *$\mathcal{EL}$-unification problem* is a set

$$\Gamma := \{C_1 \sqsubseteq^? D_1, \ldots, C_n \sqsubseteq^? D_n\}$$

where $C_1, \ldots, D_n$ are $\mathcal{EL}$-concept terms. The substitution σ is a unifier of Γ, if $\sigma(C_1) \sqsubseteq \sigma(D_1), \ldots, \sigma(C_n) \sqsubseteq \sigma(D_n)$. We say that in this case Γ is *solvable* or *unifiable*.

The subsumption symbols decorated with $\cdot^?$, denote *subsumption constraints* as opposed to subsumption statements. They are also called *goal subsumptions*, as the goal of a unification algorithm is to find a substitution that makes them true.

The set of concept variables occurring in Γ will be denoted by *Var*(Γ). Some general remarks about substitutions are useful here. We say that a substitution σ is *ground* iff for each variable X, where X is in the domain of σ (i.e., $X \neq \sigma(X)$), $\sigma(X)$ does not contain variables.

Since for any substitution σ and any variable X, $\sigma(X)$ is a finite term, there is a partial strict (acyclic) order on variables determined by σ.

Definition 4. Let X, Y be two variables. $X \succ_\sigma Y$ iff $\sigma(Y)$ is a proper subterm of $\sigma(X)$. We say that $\succ_\sigma$ is the partial order on variables induced by σ.

It is very convenient to restrict a unification problem to a *flat* form. This is possible without any loss of generality, but greatly simplifies concepts and proofs, which otherwise would have to use involved arguments based on induction on the structure of terms. Hence here we introduce the notion of a *flat* concept term, and a *flat* unification problem.

Definition 5. A concept term C is *flat* iff one of the following conditions holds:

- C is a concept name (i.e., a constant or a variable);
- C is of the form $\exists r.D$ and D is either a variable or $\top$;
- C is a conjunction of flat concept terms.

An $\mathcal{EL}$-unification problem $\Gamma := \{C_1 \sqsubseteq^? D_1, \ldots, C_n \sqsubseteq^? D_n\}$ is *flat* iff $C_1, \ldots C_n, D_1, \ldots, D_n$ are flat concept terms.

By introducing new variables and subsumptions, we can transform in polynomial time any $\mathcal{EL}$-unification problem Γ into a flat $\mathcal{EL}$-unification problem Γ' such that Γ is solvable iff Γ' is solvable [18]. We will therefore assume from now on that all $\mathcal{EL}$-unification problems are flat, if not stated otherwise.

3.1 Type zero

As in the general theory of equational unification, $\mathcal{EL}$-subsitutions can be partially ordered with respect to a given set of variables $\mathcal{X}$ and the instantiation preorder [21]. We define:
$\sigma \leq_{\mathcal{X}} \gamma$ iff there is a substitution λ such that, for each $X \in \mathcal{X}$, $\gamma(X) \equiv \lambda(\sigma(X))$.

A given $\mathcal{EL}$-unification problem determines a complete set of its unifiers. The set is empty, if the problem is not unifiable and otherwise it can be finite or infinite. Formally we define this set as follows.

Definition 6. Let Γ be an $\mathcal{EL}$-unification problem and let $\mathcal{X}$ be the set of variables in Γ. The set of substitutions M is called a *complete set of unifiers* of Γ iff it satisfies the following properties:

- every element of M is a unifier of Γ;
- if γ is a unifier of Γ, then there is $\sigma \in M$ such that $\sigma \leq_{\mathcal{X}} \gamma$;

If moreover for every $\sigma, \gamma \in M$, $\sigma \leq_{\mathcal{X}} \gamma$ implies $\sigma = \gamma$, we say that M is a *minimal complete set of unifiers* of Γ. In this case, we call the unifiers in M *most general* unifiers of Γ.

Equational theories can be divided into 4 main types, according to the cardinality of minimal complete sets of unifiers of unification problems in those theories, [21].

We say that a given equational theory $\mathcal{E}$ is of type:

- *unitary* iff for each $\mathcal{E}$-unification problem a minimal complete set of $\mathcal{E}$-unifiers exists and it has cardinality 1,
- *finitary* iff for each $\mathcal{E}$-unification problem a minimal compete set of $\mathcal{E}$-unifiers exists and it has finite cardinality,
- *infinitary* iff for each $\mathcal{E}$-unification problem a minimal complete set of $\mathcal{E}$-unifiers exists and there is an $\mathcal{E}$-unification problem for which this set is infinite,
- *zero* iff there is at least one $\mathcal{E}$-unification problem which does not have a minimal complete set of $\mathcal{E}$-unifiers.

A natural question is to ask what type has the equational theory of $\mathcal{EL}$. Interestingly, for the related description logic $\mathcal{FL}_0$, it was discovered, in [2], that the equational theory of $\mathcal{FL}_0$ is of type zero. In fact it was shown in this paper that the theory of idempotent Abelian monoids with only one homomorphism is of type zero. In our first paper on unification in $\mathcal{EL}$, [18], we show that the equational theory of $\mathcal{EL}$ is also of type zero.

Theorem 2. *$\mathcal{EL}$-unification is of type zero.*

In the proof of this theorem we show a simple $\mathcal{EL}$-unification problem that has type zero. The problem is the following.

$$\Gamma := \{\exists r.Y \sqsubseteq^? X\}$$

Γ is unifiable, and the simplest[1] unifier maps X, Y to $\top$. But for every unifier σ of Γ we can find another one which is strictly more general than σ. Hence there cannot exists a non-empty set containing only most general unifiers.

3.2 NP-completeness

The above type zero result means that one usual approach to unification modulo equational theories does not work. Namely, we cannot compute the minimal complete set of unifiers for any given problem in a theory. Type zero says that this would not be complete in the case of $\mathcal{EL}$-unification problems. There are

[1] This is the only minimal unifier for this problem, Definition 9.

$\mathcal{EL}$-unification problems which do not have a minimal complete set of unifiers, but are unifiable anyway.

Despite this negative result, as in the case of $\mathcal{FL}_0$, we can decide if an $\mathcal{EL}$-unification problem is solvable. We cannot compute a set of most general unifiers, but we compute a set of *local* unifiers instead.

In order to explain this notion, we use the notion of an atom.

Definition 7. An $\mathcal{EL}$-concept term is called an *atom* iff it is a concept name (constant or variable) or an existential restriction of the form $\exists r.D$.
The set of atoms of a concept term C, $\text{AT}(C)$ is defined recursively:

- if $C = \top$, $\text{AT}(C) := \emptyset$,
- if C is a concept name, then $\text{AT}(C) := \{C\}$,
- if $C = \exists r.D$, then $\text{AT}(C) := \{C\} \cup \text{AT}(D)$,
- if $C = C_1 \sqcap C_2$, then $\text{AT}(C) := \text{AT}(C_1) \cup \text{AT}(C_2)$.

The set of atoms of an $\mathcal{EL}$-unification problem $\Gamma := \{C_1 \sqsubseteq^? D_1, \ldots, C_n \sqsubseteq^? D_n\}$ is $\text{AT}(\Gamma) := \text{AT}(C_1) \cup \cdots \cup \text{AT}(C_n) \cup \text{AT}(D_1) \cup \cdots \cup \text{AT}(D_n)$.

With the notion of an *atom*, we can reformulate Theorem 1 in the following way.

Theorem 3. *Let C, D be $\mathcal{EL}$-concept terms and $C = C_1 \sqcap \cdots \sqcap C_m$, $D = D_1 \sqcap \cdots \sqcap D_n$, where $C_1, \ldots, D_n$ are atoms. Then $C \sqsubseteq D$ iff for each index j, $1 \leq j \leq n$, there is an index i, $1 \leq i \leq m$, such that $C_i \sqsubseteq D_j$.*

Notice that since we have restricted (wlg) $\mathcal{EL}$-unification problems to flat $\mathcal{EL}$-unification problems (Definition 5), the set $\text{AT}(\Gamma)$ contains only *flat atoms*.

Now we define *local* substitutions.

Definition 8. Let AT be a set of $\mathcal{EL}$-atoms and *Var* a set of concept variables, and let σ be a substitution. Let S_σ be an assignment of atoms from AT to variables defined in the following way.

$$S_\sigma(X) := \{A \in \text{AT} \mid A \notin \mathit{Var} \text{ and } \sigma(X) \sqsubseteq \sigma(A)\}$$

Then we say that σ is *local* for variables in *Var* w.r.t. *local* atoms in AT iff for each variable X in *Var*, $\sigma(X) \equiv \bigsqcap_{A \in S(X)} \sigma(A)$.[2]

We say that σ is a *local solution* of a unification problem Γ, iff σ is a unifier of Γ and σ is local for variables in $\mathit{Var}(\Gamma)$ w.r.t. $\text{AT}(\Gamma)$.

[2]Notice that we have defined a *local* assignment S with atoms that are not variables. In our papers attached to this thesis, we use the notion of *non-variable atoms* in such context.

Given σ as in the above definition, we say that S_σ is *induced* by σ. Later we will see that our unification algorithms first construct S which satisfies an acyclic order on variables (Definition 4), which then allows us to define a substitution σ_S. We say then, that σ_S is *induced* by S.

In our papers on which this thesis is based, we defined *locality* of a substitution in a slightly different, but equivalent way to Definition 8 (e.g., [6]). Basically, we define there a *local* substitution as one, that is induced by a non-cyclic assignment S of non-variable atoms to variables.

Obviously, for a finite set of atoms At and a finite set of variables, there are finitely many (exponentially many) local substitutions. If we consider only variables in an $\mathcal{EL}$-unification problem Γ and $\text{At} := \text{At}(\Gamma)$, then obviously, there are only finitely many *local* substitutions defined in this way. The main idea behind the $\mathcal{EL}$-unification decision procedure in [18] is to show that if a problem Γ has solution, then it has a *local solution*. This leads to the following result.

Theorem 4 ([18])**.** *The $\mathcal{EL}$-unification problem is NP-complete.*

The hardness part of this claim follows from the fact that already $\mathcal{EL}$-matching, a simpler problem that can be solved by a unification procedure, is NP-hard.[3]

For the *in NP* part of the claim, we describe a non-deterministic polynomial procedure that decides if an $\mathcal{EL}$-unification problem is unifiable. The procedure is a brute force, *guess and then check* algorithm, (Algorithm 1).

In the step 3(a), the algorithm checks if the order on variables induced by S (Definition 4) is really acyclic. Hence this step contains an Occur Check, which can be preformed as a reachability test on a graph in polynomial time.

The algorithm is obviously sound. In [18] we have proved that it is also complete, because for every unifier of Γ, there is a local unifier.

The algorithm is obviously nondeterministic, but in order for it to run in polynomial time, one has to be careful about how the checking in 3(b) is done. If we allow to define a fully expanded substitution σ based on the sets S_X, then the size of such substitution may already be exponential. If we check if the subsumptions in $\sigma(\Gamma)$ hold after having performed an unfolding of σ on Γ, then the algorithm is worst-case exponential (the nondeterministic polynomial time for choice in steps 1, 2 and 3(a) are dominated by the exponential deterministic time needed for step 3(b)). This problem of exploding size of a unification problem under a substitution, appears already in the *syntactic* unification. The exponential explosion could be avoided in this case by *structure sharing*, or in other words, by preventing a substitution from expanding, and instead presenting it in the so called *triangular form* [25].

[3] We come back to the matching problem in Section 5.2.

Algorithm 1 *(Brute Force Algorithm)*

Input: $\mathcal{EL}$-unification problem Γ

Output: *YES* if Γ is unifiable, *NO* otherwise.

1: *(Guessing part)* For each variable $X \in \mathit{Var}(\Gamma)$ guess an assignment of sets of atoms S_X, $S_X \subseteq \text{AT}(\Gamma)$.

2: Define an order on variables: $X \succ Y$ iff

- Y occurs in S_X or
- there is a variable Z, Z occurs in S_X and $Z \succ Y$

3: *(Checking part)*

(a) If $\succ$ is acyclic, define a subsitution σ for each $X \in \mathit{Var}(\Gamma)$:

- if X is minimal w.r.t. $\succ$ and $S_X = \{C_1, \ldots, C_n\}$, $\sigma(X) := C_1 \sqcap \cdots \sqcap C_n$ and for $S_X = \emptyset$, $\sigma(X) := \top$,
- if $S_X = \{C_1, \ldots, C_n\} \neq \emptyset$ and for every Y, $X \succ Y$, $\sigma(Y)$ is defined, $\sigma(X) := \sigma(C_1) \sqcap \cdots \sqcap \sigma(C_n)$.

(b) if σ is a unifier of Γ, **return** *YES*;

otherwise **return** *NO*.

Here we have used the same idea, by preventing the algorithm from expanding the concept terms with the substitution σ. Instead, we treat σ as a set of definitions of concept names occurring in Γ. The checking in step 3(b) can be then done by the subsumption decision procedure which decides the subsumptions modulo definitions in σ [31].

We have provided two kinds of proofs for the completeness of Algorithm 1. The first one presented in [18] is quite involved, based on a replacement of each non-local atom in the range of a solution, by a conjunction of local atoms. We show that due to the monotonic properties of constructors, such a replacement preserves subsumptions, and thus preserves unifying properties of a solution. This replacing process has to terminate, because the terms used in the range of a substitution are finite.

We can also notice that a replacement of non-local atoms by a conjunction of local ones yields a solution with is smaller w.r.t. the inverse of subsumption order. This order is well-founded. The minimal solutions w.r.t. this order are the so called *minimal unifiers* which have to be local ([18], Proposition 2).

We recall here the definition of a *minimal* unifier, as its role is similar to the one of the most general unifier (Definition 6), not with respect to all unifiers, but with respect to the local ones.

Definition 9. (minimal unifiers)

- Let σ, θ be two $\mathcal{EL}$-substitutions and let $\mathcal{X}$ be a set of variables. We say that $\sigma \prec_{\mathcal{X}} \theta$ iff for each variable $X \in \mathcal{X}$, $\sigma(X) \sqsubseteq \theta(X)$ and there is at least one variable $Y \in \mathcal{X}$ such that $\sigma(Y) \sqsubset \theta(Y)$.

- Let σ an $\mathcal{EL}$-unifier of a unification problem Γ. We say that σ is a *minimal unifier* of Γ if there is no $\mathcal{EL}$-unifier of Γ, θ, such that $\theta \prec_{Var(\Gamma)} \sigma$.

The second type of completeness proof of Algorithm 1 appeared e.g., in [17]. It shows that having any solution, one can construct another solution using only atoms from $\text{A}\text{T}(\Gamma)$, which is thus local and *smaller* than the original one, but not necessary minimal.

Rule-based algorithm

Algorithm 1 depends on blind guessing and checking and therefore is not fit for a practical implementation. In [19] we introduce a goal-oriented algorithm, which makes choices only if necessary for finding a solution. The algorithm presented there is formulated as working on goal equivalences instead of goal subsumptions as presented here, (see Definition 3 and the explanations there).

The goal-oriented algorithm (Algorithm 2) is defined based on a set of transformation rules, which are to be applied exhaustively to a given $\mathcal{EL}$-unification

problem and transform it into a solved form much in the style of the Martelli and Montanari algorithm for the syntactic unification, [47].

Some of the rules are to be applied *eagerly* and some are chosen in a *don't know*-non-deterministic way. If a non-eager rule is chosen to solve a goal subsumption, it can also involve a *don't know* kind of non-deterministic choice of the way it is applied. On the other hand, like in the Martelli and Montanari algorithm [47], a choice of unsolved subsumption for a non-eager rule application is *don't care*-non-deterministic.

At first all subsumptions, with the exception of the subsumptions with a variable on the right-hand, side are *unsolved*. This means that the subsumptions of the form $C_1 \sqcap \cdots \sqcap C_n \sqsubseteq^? X$ are solved by definition and this is maintained throughout the run. Notice that if there are no other constraints on the variable X in the unification problem, then the variable is substituted by $\top$ in any computed unifiers.

Each rule application, either fails or solves one goal subsumption. The algorithm applies rules to *unsolved* goal subsumptions only.

The algorithm maintains an assignment S of non-variable atoms in $\text{AT}(\Gamma)$ to each variable. Initially, all variables are assigned the empty set of atoms. This assignment is basically the same assignment as in Algorithm 1, guessed in step 1. Here this guess is restricted by the process of solving the goal subsumptions. Extending S produces a dependency order on variables. Namely, $X \succ_S Y$ iff

- $\exists r.Y \in S(X)$ for a role name $r \in N_r$ or
- there is a variable Z, such that $\exists r.Z \in S(X)$ for a role name $r \in N_r$, and $Z \succ_S Y$

The algorithm while defining the assignment for variables tries to construct an order on variables which could be induced by a substitution as in Definition 4.

If a rule application requires an extension of the assignment $S(X)$ for a variable X, the algorithm performs a kind of the Occurs Check, by checking if the dependency order $\succ_X$ on variables defined by S becomes cyclic by such an extension. If yes, it fails on the given non-deterministic choices made already in the run.

If by a rule application a local non-variable atom D is added to a set $S(X)$, then for each solved goal subsumption of the form $C_1 \sqcap \cdots \sqcap C_n \sqsubseteq^? X$, a new goal subsumption $C_1 \sqcap \cdots \sqcap C_n \sqsubseteq^? D$ is added to the current set of unsolved goal subsumptions.

The completeness argument shows that the unifier computed by a successful run of the algorithm is smaller w.r.t. the order on substitutions defined in Definition 9 than a unifier which could guide the non-deterministic choices in this run.

Algorithm 2 *(Rule-based Algorithm)*

Input: flat $\mathcal{EL}$-unification problem Γ_0

Output: "YES" if Γ is unifiable, "NO" otherwise.

1: $\Gamma := \Gamma_0$

2: for each $X \in Var(\Gamma)$, $S(X) := \emptyset$

3: While Γ contains an unsolved subsumption, apply the steps (1) and (2).

1. Apply eager rules to the unsolved subsumptions;

2. If no eager rule is applicable, choose an unsolved subsumption s. Choose a rule applicable to s and apply it. If no rule is applicable to s or the rule applied fails, then return *NO*.

4: Return *YES*

The algorithm, as presented here is a decision procedure, but it can be easily modified to compute local unifiers. Nevertheless it will not compute *all* local unifiers. The set of computed unifiers is a subset of all local unifiers and contains the set of all *minimal unifiers*. For each local unifier that is not discovered by the algorithm, there is a minimal one that is *smaller* w.r.t. the order defined in Definition 9, than the first one.

Sat reduction

An algorithm which is somewhat more similar to the "guess and check" strategy of Algorithm 1, is the one we proposed in [17]. It transforms a given flat $\mathcal{EL}$-unification problem Γ into a set of propositional clauses $C(\Gamma)$ such that the size of $C(\Gamma)$ is polynomial in the size of Γ and Γ is unifiable iff $C(\Gamma)$ is satisfiable. Hence we reduce $\mathcal{EL}$-unification problem to satisfiability in propositional logic (SAT). From the point of view of applications, this enables us to use highly-optimized SAT solvers, [37].

This translation into SAT is inspired by the Kapur and Narendran's translation of ACIU-unification problems into satisfiability in propositional Horn logic (HornSAT) in [42].[4] In fact $\mathcal{EL}$-equational theory defined by equivalence between concept terms, is an ACIU theory, where the only associative, commutative, idempotent operator is the concept intersection ($\sqcap$) with top ($\top$) as its unit element, extended with existential restrictions, which are in fact very much like free functions, except that they have the property of being monotonic with respect to concept subsumption.

The translation to SAT requires coding each subsumption between atoms in $\text{At}(\Gamma)$ as a propositional variable. In [17], we decided to introduce propositional

[4]This reduction was extended to ACI-disunification in [24].

variables to express dis-subsumptions rather than subsumptions. Hence the propositional variables are of the form $[C \not\sqsubseteq D]$. The meaning of these variables is determined by the following requirement on a valuation τ satisfying the set of clauses $C(\Gamma)$, [17, Lemma 5]:

$$\text{if } \tau([C \not\sqsubseteq D]) = 0, \text{ then } \sigma(C) \sqsubseteq \sigma(D).$$

For a valuation satisfying $C(\Gamma)$, it is not necessary that the converse of this implication holds. τ will correctly determine a unifier σ, for which maybe the converse implication does not hold. We allowed this flexibility of encoding in order to have smaller set of clauses $C(\Gamma)$ and thus to get the solution faster.

The choice of encoding dis-subsumptions rather than subsumptions, followed from an idea to avoid as much as possible non-Horn clauses in $C(\Gamma)$. Non-Horn clauses are a source of non-determinism in solving SAT problems, and since $\mathcal{EL}$-unification is NP-complete, it is not possible to eliminate completely Horn clauses from the encoding.

For example a goal subsumption $C_1 \sqcap \cdots \sqcap C_n \sqsubseteq^? D$, where D is a non-variable atom, holds according to the characterization of subsumption in Theorem 3, if and only if $C_1 \sqsubseteq D, \ldots C_n \sqsubseteq D$. Moreover, if $D \sqsubseteq E$ for a non-variable atom E, then $C_1 \sqsubseteq E, \ldots, C_n \sqsubseteq E$.

These properties (constraints) are encoded with dis-subsumption variables into the clauses:

1. $[C_1 \not\sqsubseteq D] \wedge \cdots \wedge [C_n \not\sqsubseteq D] \rightarrow$
2. for each non-variable atom E of Γ: $[C_1 \not\sqsubseteq E] \wedge \cdots \wedge [C_n \not\sqsubseteq E] \rightarrow [D \not\sqsubseteq E]$

Obviously, these are Horn clauses, hence they are *easy* for a SAT procedure. If we choose rather to encode the subsumption using propositional variables of the type: $[C \sqsubseteq D]$, then the above clauses would have to be reversed and thus become non-Horn.

Avoiding explosion of non-deterministic choices in this place, does not mean we can do it everywhere. In fact, our encoding has to deal also with other properties of subsumption or rather dis-subsumption in our case. And the unavoidable non-Horn clauses appear while encoding the transitivity of subsumption using propositional variables expressing dis-subsumption.

For each 3 atoms of Γ, C_1, C_2, C_3,

$$[C_1 \not\sqsubseteq C_3] \rightarrow [C_1 \not\sqsubseteq C_2] \vee [C_2 \not\sqsubseteq C_3]$$

This last part of the encoding shows that the set $C(\Gamma)$ contains more than n^3 clauses, if n is the number of all atoms of Γ.

In our implementations of the SAT-based algorithm [5], we have noticed that working with dis-subsumptions or subsumptions does not make a difference.

Algorithm 3 *(SAT-based Algorithm)*

Input: flat $\mathcal{EL}$-unification problem Γ
Output: "YES" if Γ is unifiable, "NO" otherwise.

1: Create a set of propositional clauses $C(\Gamma)$ encoding:

- goal subsumptions in Γ,
- properties of subsumption in $\mathcal{EL}$,
- partial strict order on variables.

2: Run a highly-optimized SAT-solver on $C(\Gamma)$ and if the solver answers "satisfiable", **return** *YES*, otherwise **return** *NO*.

This is maybe due to the power of the modern SAT-solvers. In [9], while extending this procedure, we returned to a more intuitive encoding of the goal subsumptions as propositional variables representing subsumptions rather than dis-subsumptions. The latest version of UEL[5], uses this last encoding in the SAT-reduction.

Apart from propositional variables encoding dis-subsumption, we have propositional variables of the form $[X > Y]$ for each pair of variables $X, Y \in \mathit{Var}(\Gamma)$. The clauses in $C(\Gamma)$ containing these propositional variables have to encode the partial strict order of Definition 4. The clauses encoding irreflexivity and transitivity are Horn. The non-Horn clauses are needed to express connection between dis-subsumption and the order. For each variable X and a non-variable atom $\exists r.Y$ which are in the set of atoms of Γ:

$$\rightarrow [X > Y] \vee [X \not\sqsubseteq \exists r.Y]$$

The schema of the algorithm is presented as Algorithm 3.

In the SAT-based approach we can take advantage of the highly-optimized, state of the art SAT solvers. For example, in our implementation of Algorithm 3 in UEL, we used MiniSat[6] and the recent version uses SAT4J[7]. It turns out that in our implementation of both *rule-based* and *SAT-based* algorithms, the SAT-based algorithm returns a negative answer faster than the rule-based algorithm. But the SAT-based algorithm is often worse than the rule-based, when the goal is unifiable. The rule-based algorithm usually finds a positive answer faster.

[5] `http://uel.sourceforge.net`
[6] `http://minisat.se`
[7] `http://www.sat4j.org/`

4 Without Top: unification in $\mathcal{EL}^{-\top}$

Unification algorithms for the description logic $\mathcal{EL}$ were meant to find applications in maintenance of large medical ontologies like SNOMED CT. A unification algorithm should help to find concepts equivalent w.r.t. a plausible interpretation. It would moreover propose a set of definitions (a solution), which makes such equivalence explicit.

Alas, a crucial role in this algorithms plays the logical constructor $\top$, which is always interpreted as the set containing *everything*, i.e., the whole domain. A variable, with an empty set of non-variable atoms assigned to it by an algorithm would be *defined* by $\top$.

SNOMED CT on the other hand, does not allow $\top$ into definitions of its concepts. There is only one *top* concept in SNOMED, namely SNOMED CT CONCEPT. It is used as the root of a concept hierarchy. To define another concept as equivalent to $\top$ would make this concept redundant in the hierarchy. But on the other hand a usual $\mathcal{EL}$-unifier may suggest to a user to define a concept name as $\top$. This would not make any sense to the user.

To address this problem in [4] and [3], we define there the description logic $\mathcal{EL}^{-\top}$ as providing all the constructors of $\mathcal{EL}$ with the exception of $\top$.

It is also interesting from a theoretical point of view to see how much influence removing the top constructor from the logic has on the unification problem. For related equational theories such a change does not affect the complexity of unification. For example, consider the simple equational theory ACIU i.e., $\mathcal{EL}$ without existential restrictions. As mentioned above, unification in ACIU is in P.[1] If the unit (U), and thus $\top$ is removed from ACIU, we obtain the theory ACI.[2] The complexity of unification in ACI is the same as that for ACIU [42].

In a more general context, where multiple functions are allowed, some of which are *uninterpreted* or satisfy ACI, or else ACIU properties, unification becomes NP-complete. The complexity of unification in this context is the same if unit is not allowed.

[1] This complexity is obtained by reducing this kind of unification problems to propositional Horn-SAT problems.

[2] The solution is obtained by the same procedure as in the case of ACIU, but without the prior *guessing* which variables are top that causes their elimination from the goal.

As another example, let us consider the small description logic $\mathcal{FL}_0$. Unification in $\mathcal{FL}_0$ is EXPTIME-complete, and it is the same if $\top$ is removed form $\mathcal{FL}_0$.

One can obtain the effect of removal $\top$ from a unifier, by adding negative constraints $\top \not\sqsubseteq^? X$ to an $\mathcal{FL}_0$-unification problem and then use disunification techniques from [22]. This way of solving unification in $\mathcal{EL}^{-\top}$, i.e., using dismatching for $\mathcal{EL}$, is not possible. In $\mathcal{FL}_0$ we need to prevent only substitution of $\top$ for variables given in the unification problem and all other occurrences of $\top$ in the range of solution are reducible due to the properties of universal restriction ($\forall$) provided by $\mathcal{FL}_0$. In contrast, in $\mathcal{EL}$ $\top$ cannot be reduced in this way. Hence we have to deal with possible occurrences of $\top$ at any depth of existential restrictions in the range of a unifier. Such occurrences of $\top$ cannot be prevented by adding negative constraints for goal variables as in the case of $\mathcal{FL}_0$.

Surprisingly, it turns out that in the case of unification in $\mathcal{EL}$ the complexity of the problem increases from NP to PSPACE if $\top$ is removed from the logic.

The methods used for solving the problem are quite different from those used for $\mathcal{EL}$ when top is available. We have to proceed similar as in case of $\mathcal{FL}_0$, where a unification problem has been reduced to a problem of solving formal languages equations, [20]. The equations in the case of $\mathcal{EL}^{-\top}$ are of a type that can be solved in PSPACE, while in the case of $\mathcal{FL}_0$, solving the equations occurring there is EXPTIME complete. We show the *in* PSPACE result by a reduction to the emptiness problem for alternating automata. PSPACE *hardness* is obtained by a reduction of the emptiness problem of the intersection of deterministic finite automata (DFA) [44, 35], to the unification problem in $\mathcal{EL}^{-\top}$.

Theorem 5. *Unification in $\mathcal{EL}^{-\top}$ is* PSPACE *complete.*

4.1 In PSpace

Removing $\top$ from the set of constructors of $\mathcal{EL}^{-\top}$ restricts its expressive power, but does not restrict the allowed interpretations in this logic. Hence Theorem 1 characterizing subsumption remains valid for $\mathcal{EL}^{-\top}$-concept terms. Since an $\mathcal{EL}^{-\top}$-unification problem is also an $\mathcal{EL}$-unification problem, the unification algorithms may compute $\mathcal{EL}$-unifiers for $\mathcal{EL}^{-\top}$-unification problems. If such unifiers contain $\top$, they are not allowed in $\mathcal{EL}^{-\top}$ though.

Hence, by the completeness result in [18], we know that for each solution γ of a unification problem Γ in $\mathcal{EL}^{-\top}$, there is a local unifier σ of Γ in $\mathcal{EL}$, such that $\gamma \succeq_{Var(\Gamma)} \sigma$ (Definition 9). Our algorithm tries to correct σ by adding concept terms to the conjunctions of concept terms substituted by it for variables. The concept terms that are used in this process are *minimal* w.r.t. the inverse of the subsumption order. If top is present, it is the smallest element in this order.

But since we do not have top, the minimal elements are concept names and (ground) existential restrictions of the form $\exists r_1.(\dots \exists r_n.A)\dots)$, where A is a concept name. We call such concept terms *particles*.

By introducing particles, we depart from the realm of *locality*, but only in a controlled way. Especially, we allow only ground particles. If we want to add such a particle as a conjunct to the conjunction $\sigma(X)$, we have to consider all goal subsumptions of the form: $C_1 \sqcap \dots \sqcap C_n \sqsubseteq^? X$. They are already satisfied by σ, especially if X is send by σ to $\top$. These subsumptions can be used to define an alternating automaton, which accepts all particles with which we can extend σ. Such an automaton has universal states denoting concept variables and existential states for subsumptions. We need ϵ-transitions from states representing variables to states representing subsumptions and also from states representing subsumptions to states representing concept names which are top-level conjuncts of the subsumptions. Hence in order to decide if a particle can be used to extend a local unifier σ, we use the emptiness test for an alternating automaton with ϵ-transitions. Such test can be done in PSPACE [41].

4.2 PSpace hardness

In [4] and [3] we show how the PSPACE complete problem of the intersection emptiness for deterministic finite automata (DFA) can be reduced to a unification problem in $\mathcal{EL}^{-\top}$. For such a reduction we consider only automata which accept a non-empty language and in which all states are reachable from a start state.

Let $\mathcal{A}$ be a DFA. We construct an $\mathcal{EL}^{-\top}$-unification problem Γ in the following way. Every state in $\mathcal{A}$ is represented by a concept variable, and we use only one concept name A, which represents every accept state. We define the set of role names N_R as the set of alphabet symbols of the automaton, and define the goal subsumptions: $C_1 \sqcap \dots \sqcap C_n \sqsubseteq^? X$, where $C_i = \exists\alpha.Y$ if α is a symbol in the alphabet of the given automaton, and a state represented by Y is reached by reading α from the state represented by X.

Let X be a variable in Γ corresponding to the initial state of $\mathcal{A}$. We have the following connection between $\mathcal{A}$ and Γ:

- If γ is an $\mathcal{EL}^{-\top}$-unifier of Γ, then for every particle $\exists\omega.A$ of $\gamma(X)$, $\omega \in L(\mathcal{A})$, where $L(\mathcal{A})$ is the language accepted by $\mathcal{A}$.
- If $\omega \in L(\mathcal{A})$, then there is a ground substitution, which is a unifier of Γ in $\mathcal{EL}^{-\top}$.

The intersection emptiness problem is given by finitely many DFA's $\mathcal{A}_1, \dots, \mathcal{A}_k$. We ask if the intersection of the languages accepted by these DFA's

is empty. We can assume that each of the languages is non-empty and that the automata $\mathcal{A}_1, \ldots, \mathcal{A}_k$ have disjoint sets of states. The flat $\mathcal{EL}^{-\top}$ unification problem is now defined by:

$$\Gamma := \bigcup_{i \in \{1,\ldots,k\}} (\Gamma_{\mathcal{A}_i} \cup \{X_{\mathcal{A}_i} \sqsubseteq^? Y\})$$

where Y is a new variable, $\Gamma_{\mathcal{A}_i}$ is a $\mathcal{EL}^{-\top}$-unification problem for the automaton $\mathcal{A}_i$, and $X_{\mathcal{A}_i}$ is a variable of $\Gamma_{\mathcal{A}_i}$ representing the start state of $\mathcal{A}_i$. For thus defined Γ, we can show that Γ is unifiable in $\mathcal{EL}^{-\top}$ iff the intersection of the languages of $\mathcal{A}_1, \ldots, \mathcal{A}_k$ is non-empty.

5 Unification and matching in $\mathcal{EL}$ with a TBox

In the introduction, we have mentioned that the expressive power of a description logic depends on the one hand, on the language tools, i.e., logical constructors, and on the other hand, on the sort of axioms that can be assumed for given modeling applications.

A set of such axioms is called a *TBox* or an *ontology*. The axioms are *terminological*, if they can be treated as a set of definitions. A concept definition is an equation statement $A \equiv C$, which has on its left-hand side a concept name A, while the right-hand side is not restricted i.e., an arbitrary concept term. We say that a TBox is a *terminology* or an *acyclic* TBox, if

- it is a set of definitions,
- every defined concept has a unique definition, and
- the set does not contain cyclic dependencies between the defined concepts.

In the case of acyclic TBoxes, we can clearly separate *defined* concept names from the undefined, *primitive* ones.

If cyclic dependencies between defined concepts do appear, we say that the terminology is *cyclic*, or that the TBox is a *cyclic TBox*. If the assumption about unique name definition is satisfied, we can still separate *defined* from *primitive* concepts.

More general than all these kinds of TBoxes mentioned above are the so called *general TBoxes*, which can contain any subsumption statements between concept terms. Such statements are called *concept inclusions* or *general concept inclusions*, abbreviated with the name *GCIs*.

Assuming a TBox $\mathcal{T}$ restricts the set of interpretations admissible in a logic to models of $\mathcal{T}$. We say that an interpretation $\mathcal{I}$ is a model of $\mathcal{T}$ if it satisfies each definition (concept inclusion) contained in $\mathcal{T}$.

In this section we present our results on unification in $\mathcal{EL}$ extended with different kinds of TBoxes ([19, 8, 9, 6]). As generalizations of $\mathcal{EL}$, unification in all these extensions of $\mathcal{EL}$ (if decidable) inherits the lower complexity bound, i.e., NP-hardness, from unification in $\mathcal{EL}$.

In the context of unification, we generally assume that every TBox is ground, i.e., we do not allow the concept names occurring in the TBox to be variables. The intuition behind this restriction is that the unifier can define an extension of a TBox, but cannot change the definitions present in it. In the case of acyclic TBoxes, we can assume some primitive concepts from the ontology to be variables. This corresponds to the intuition, that a unifier may suggest some definitions for undefined concepts in an ontology.

5.1 Unification modulo acyclic TBoxes

The unification problem in $\mathcal{EL}$ modulo an acyclic TBox was addressed in [19]. It is obvious that subsumption w.r.t. such a TBox can be reduced to subsumption without TBox, by *expanding* the occurrences of the defined concepts with their definitions and then checking subsumption. This however may result in an exponential blow-up and thus the procedure for checking subsumption would not be polynomial. There is however another, polynomial procedure to decide subsumption in $\mathcal{EL}$ w.r.t. an acyclic TBox, [1].

Using the same expansion, we can also reduce unification in $\mathcal{EL}$ modulo an acyclic TBox to unification without such a TBox. But then, because of a possible exponential blow-up, the procedure would be nondeterministic exponential.

Nevertheless, in [19], we show that unification in $\mathcal{EL}$ modulo an acyclic TBox is of the same complexity as the unification without a TBox, i.e., it is NP-complete, when the size of TBox is either counted as constant or is part of the input. This is obtained by reducing $\mathcal{EL}$-unification modulo acyclic TBoxes to $\mathcal{EL}$-unification without a TBox.

Theorem 6. *$\mathcal{EL}$-unification modulo acyclic TBoxes can be reduced in polynomial time to $\mathcal{EL}$-unification.*

We do not extend the unification problem with the TBox, but treat the TBox as a part of the problem, and thus avoid a possible exponential blow-up in the size of the input.

Basically, we define $\Gamma(\mathcal{T})$, a unification problem in *dag-solved form*, that corresponds to the TBox $\mathcal{T}$ and is solved together with the given $\mathcal{EL}$-unification problem Γ as a part of the problem: $\Gamma \cup \Gamma(\mathcal{T})$. Variables in $\Gamma(\mathcal{T})$ correspond to the defined concept names and are called *system variables*. Adding $\Gamma(\mathcal{T})$ to the unification problem is equivalent to a succinct encoding (by structure sharing) of the unfolding of a given unification problem with the TBox $\mathcal{T}$. Since $\mathcal{T}$ is part of the input (or of constant size), and $\Gamma(\mathcal{T})$ is of the same size as $\mathcal{T}$, adding $\Gamma(\mathcal{T})$ to the unification problem can be done in polynomial time.

We show in [19] that this reduction provides a sound and complete decision procedure for $\mathcal{EL}$-unification modulo an acyclic TBox.

It is worthwhile to mention that the medical ontology SNOMED CT, which is the main target of the intended applications of $\mathcal{EL}$-unification algorithms, is an acyclic $\mathcal{EL}$-TBox, or rather an acyclic $\mathcal{EL}^{-\top}$- TBox. In the implementation UEL [15], we adopt the above described way of dealing with acyclic TBoxes. We pull into a given unification problem an ontology and integrate it into the problem. Since the ontology is very big, it would be impossible to do it in a straightforward way. Instead, we extract a *module* which is a part of the entire ontology relevant for our problem. To be more precise, the module contains definitions of the concepts used in the problem and is closed under the action of adding the definitions of the concepts used in the module.

5.2 Matching in $\mathcal{EL}$ with a TBox

Before considering unification with general TBoxes, we turn here to an *easier* and less general problem, namely, the problem of matching. Usually a matching problem is defined as a set of pairs of terms containing variables. A solution for such a problem is a substitution, which is to be applied to the first elements of the pairs of terms. Such an application of the substitution should make the terms in each pair equivalent to each other [49].

Since the variables in the second elements of the pairs cannot be substituted, we can rename them and treat them as constants. In this way, one can reduce each matching problem to a unification problem and apply a unification algorithm to solve it. This may be not optimal from the point of view of complexity, since maybe there are ways to solve matching faster than using tools for unification. In the case of $\mathcal{EL}$ with a general TBox such a reduction of a matching problem is not possible, since we do not have a unification algorithm for unification in $\mathcal{EL}$ modulo a general TBox, nor do we know if this problem is decidable. Nevertheless, in [16], we present an NP decision procedure for matching in $\mathcal{EL}$ with a TBox.

In the context of the description logic $\mathcal{EL}$, matching is defined on pairs of concept terms, where one of these terms is *ground*. As usual in the case of $\mathcal{EL}$, we define matching using *subsumption* rather than *equivalence*. The following is the formal definition.

Definition 10. An $\mathcal{EL}$-matching problem is a finite set

$$\Gamma = \{C_1 \sqsubseteq^? D_1, \ldots, C_n \sqsubseteq^? D_n\}$$

of subsumptions between $\mathcal{EL}$-concept terms which may contain variables, where for each i, $1 \leq i \leq n$, either C_i or D_i is ground.
A substitution σ is a *matcher* of Γ iff it solves all subsumptions in Γ, i.e., $\sigma(C_1) \sqsubseteq \sigma(D_1), \ldots, \sigma(C_n) \sqsubseteq \sigma(D_n)$.

One major difference between our approach to unification problems in $\mathcal{EL}$ and the approach to matching is that we cannot conveniently *flatten* matching problems without losing the property of *one side ground* goal subsumptions.

It should be noticed that matching has been since a long time a subject of interest in the context of DLs. It first appeared in applications of the Classic system [28]. In [29], Borgida and McGuinnness proposed to use matching as a special query, which returns a concept. Such queries could be used as a means to filter out the unimportant aspects of large concept descriptions appearing in knowledge bases of Classic. Another major application of matching in the context of DLs is the integration of knowledge bases by prompting interschema assertions to the integrator [27].

Originally matching in DLs was considered without TBoxes, because there were only acyclic TBoxes considered at that time, and as in the case of unification, an acyclic TBox can be reduced away by unfolding.

There are three special cases of matching problems defined in $\mathcal{EL}$.

1. An $\mathcal{EL}$-matching problem Γ is a *matching problem modulo equivalence* if $C \sqsubseteq^? D \in \Gamma$ implies $D \sqsubseteq^? C \in \Gamma$. This kind of $\mathcal{EL}$-matching problems (without TBoxes) were considered in [12, 13].

2. An $\mathcal{EL}$-matching problem Γ is a *left-ground matching problem modulo subsumption* if $C \sqsubseteq^? D \in \Gamma$ implies that C is ground. This kind of $\mathcal{EL}$-matching problems coincides with *matching modulo subsumption* considered (without TBoxes) also in [12, 13].

3. An $\mathcal{EL}$-matching problem Γ is a *right-ground matching problem modulo subsumption* if $C \sqsubseteq^? D \in \Gamma$ implies that D is ground. This kind of $\mathcal{EL}$-matching has not been considered before.

We show the following results in [16].

Theorem 7. *Let Γ be an $\mathcal{EL}$-matching problem and $\mathcal{T}$ a general $\mathcal{EL}$-TBox.*

1. *If Γ is a* left-ground $\mathcal{EL}$-matching problem*, then we can decide in polynomial time whether Γ has a matcher w.r.t. $\mathcal{T}$ or not.*

2. *If Γ is a* right-ground $\mathcal{EL}$-matching problem*, then we can decide in polynomial time whether Γ has a matcher w.r.t. $\mathcal{T}$ or not.*

3. *In general, the problem of deciding if Γ has a matcher w.r.t. $\mathcal{T}$ or not is* NP*-complete.*

The first two statements are shown by proving that Γ is matchable iff a special kind of substitution is a matcher. Checking if this substitution is a matcher takes polynomial time [31]. To prove the third statement, we notice that matching in $\mathcal{EL}$ is NP-hard even without general TBoxes [45], and then we introduce a rule based procedure that solves the problem in NP.

5.3 Unification modulo cycle-restricted TBoxes

The problem of unification in $\mathcal{EL}$ extended with general TBoxes is a long standing open problem. Research in this area allowed us to find cases where the *local* approach (as in the case of $\mathcal{EL}$ alone) still works. The case we describe here is the unification problem in $\mathcal{EL}$ with a cycle-restricted TBox. Such a TBox is a kind of general TBox, i.e., a set of GCIs, which satisfy a restriction on cycles in the subsumption chains between concepts which the TBox implies. Namely, we forbid that subsumptions of the form $C \sqsubseteq \exists\omega.C$, where C is any concept term and ω is a sequence of role names, are consequences of the TBox. For example a GCI $\exists\texttt{child.Human} \sqsubseteq \texttt{Human}$ is allowed, but a GCI $\texttt{Human} \sqsubseteq \exists\texttt{parent.Human}$ does not satisfy this restriction. This restriction is sufficient to allow us to prove the *locality* of unification: any unifiable $\mathcal{EL}$-unification problem w.r.t. a cycle-restricted TBox has a local unifier. On the other hand, if the restriction is not satisfied, unification is not *local* any more. We illustrate it with the following example.

Let $\mathcal{T} := \{B \sqsubseteq \exists s.D, D \sqsubseteq B\}$. Let the unification problem be: $\Gamma := \{A_1 \sqcap B \equiv^? Y_1, A_2 \sqcap B \equiv^? Y_2, \exists s.Y_1 \sqsubseteq^? X, \exists s.Y_2 \sqsubseteq^? X, X \sqsubseteq^? \exists s.X\}$

It is obvious that Γ has a unifier: $\gamma := \{Y_1 \mapsto A_1 \sqcap B, Y_2 \mapsto A_2 \sqcap B, X \mapsto \exists s.B\}$. This solution is not *local*, since $\exists s.B$ is not a substitution of any non-variable atom in Γ or in $\mathcal{T}$. In [9] we show that Γ does not have any local solutions.

- In [8], we presented a brute force algorithm in the style of Algorithm 1 for the case of $\mathcal{EL}$ with a cycle-restricted TBox.
- In [9] we presented an algorithm based on a translation into SAT, in the spirit of Algorithm 3.
- In [6] we presented an algorithm based on a set of transformation rules similar to Algorithm 2.

In the last two papers presented in this thesis ([9, 6]), we show that the solution works also if a cycle-restricted TBox contains *role inclusion axioms* of the form $r \sqsubseteq s$ for role names $r, s \in N_R$ and *transitivity axioms* of the form $r \circ r \sqsubseteq r$, for a role name $r \in N_R$. When $\mathcal{EL}$ is extended with such a TBox, it is called $\mathcal{ELH}_{\mathcal{R}^+}$ [10], where $\mathcal{H}$ stands for *role hierarchy* induced by the role inclusion axioms and $\mathcal{R}^+$ indicates allowing role transitivity axioms. In [9] we show that checking if a TBox containing axioms of $\mathcal{ELH}_{\mathcal{R}^+}$ is cycle-restricted can be done in polynomial time, by looking for the forbidden cycles only within the set of concept names together with the top constructor.

The above mentioned rule-based unification algorithm relies on a polynomial subsumption procedure for this logic. Such a procedure for subsumption in $\mathcal{ELH}_{\mathcal{R}^+}$ was provided in [31, 10].

The new proof techniques required to solve unification in $\mathcal{ELH}_{\mathcal{R}^+}$ with cycle-restricted TBoxes, lead us to develop new and interesting characterizations of subsumption between concept terms in $\mathcal{ELH}_{\mathcal{R}^+}$.

For the SAT-based algorihtm in [9] we have designed a term rewriting system that characterizes subsumption. For the rule-based algorithm in [6] we have designed a characterization of subsumption in $\mathcal{ELH}_{\mathcal{R}^+}$ by a Gentzen-style calculus on expressions of the form $C \vdash_{\mathcal{T}} D$ which are sequents in our calculus. We show that $C \vdash_{\mathcal{T}} D$ iff $C \sqsubseteq_{\mathcal{T}} D$. A similar Gentzen-style characterization of subsumption in $\mathcal{EL}$ w.r.t. TBoxes was developed in [36].

6 Disunification in $\mathcal{EL}$

In this section we come back to consider the description logic $\mathcal{EL}$ without TBoxes. We will focuse on another generalization of the unification problem in $\mathcal{EL}$, namely *disunification* in $\mathcal{EL}$.

In [7], we have proposed disunification as a means to restrict the set of local unifiers to those that satisfy some reasonable constraints, from a point of view of a user, e.g., an ontology engineer. A disunification problem is defined by positive and negative constraints. The positive part is a usual unification problem. By adding negative constraints, one can prevent many undesirable solutions of this problem and present a user with a smaller and thus more manageable set of unifiers.

We do not know yet if the problem of disunification in $\mathcal{EL}$ is in general decidable, but we have some decidability results about interesting, restricted cases: *local* disunification and *dismatching* in $\mathcal{EL}$.

As was mentioned above, disunification and unification in a logic are special cases of unification and disunification modulo equational theories (defined w.r.t. logical equivalence, [19]). Disunification modulo equational theories has been a topic of extended research, e.g., [33, 32, 23, 34]. Informally speaking, a disunification problem consists of a positive part, which is a set of goal equations, and a negative part which is a set of dis-equations. The first part is a unification problem modulo an equational theory. A substitution for variables is a solution of a disunification problem, if it solves the unification part while the negative part is also satisfied. A dis-equation in the negative part is satisfied, if it is not valid (not true in all models). Disunification in the equational theories of finitary unification type is reducible to unification and if one can effectively compute the finite complete set of unifiers of a unification problem, it is also decidable. For such theories the following procedure solves disunification. In the first step we compute a finite complete set of unifiers of the positive part of a disunification problem and in the second step, we check if any one of them is a solution for the negative part. Unfortunately, the unification in the equational theory corresponding to $\mathcal{EL}$ is of type zero. This means that a unification problem in $\mathcal{EL}$ may not have a finite complete set of most general solutions. Hence it is not possible to use the above described reduction to solve disunification in $\mathcal{EL}$. Another small description logic $\mathcal{FL}_0$ also has unification type zero. Nevertheless it is possible to tweak the decision procedure for unification in $\mathcal{FL}_0$ in such a way that it solves also disunification, [22]. This proves that the disunification

problem in $\mathcal{FL}_0$ has the same complexity as unification in $\mathcal{FL}_0$, i.e., complete for the class ExpTime. The DL $\mathcal{EL}$ seems to be more difficult in this respect. Our results presented here are obtained also by extending the methods developed for solving unification in $\mathcal{EL}$, but they do not work for the general case. The reason is that the decidability proof of unification in $\mathcal{EL}$ relies heavily on *locality* of the problem. Disunification problems in $\mathcal{EL}$ are not necessarily *local*, in the sense that there are disunification problems which have solutions but do not have *local* solutions (Definition 8).

As usually in the case of $\mathcal{EL}$, we define the disunification problem in terms of subsumptions and equivalence is treated as an abbreviation for two way subsumptions. We define here the disunification problem in an as general way as possible, taking advantage of the negation and combining the constraints with all boolean constructors.

Definition 11. An $\mathcal{EL}$-disunification problem Γ is a formula built from goal subsumptions ($C \sqsubseteq^? D$, where C, D are $\mathcal{EL}$-concept terms possibly containing variables) using the logical connectives $\wedge, \vee$ and $\neg$.
$C \equiv^? D$ is an abbreviation of $(C \sqsubseteq^? D) \wedge (D \sqsubseteq^? C)$.
$C \not\equiv^? D$ is an abbreviation of $(C \not\sqsubseteq^? D) \vee (D \not\sqsubseteq^? C)$.
$C \not\sqsubseteq^? D$ is an abbreviation of $\neg(C \sqsubseteq^? D)$ and is called a *goal dissubsumption*.
A *basic disunification problem* is a conjunction of goal subsumptions and dissubsumptions.

A substitution σ is a solution for Γ if:

- $\Gamma = \Gamma_1 \vee \Gamma_2$ and σ solves Γ_1 or σ solves Γ_2;
- $\Gamma = \Gamma_1 \wedge \Gamma_2$ and σ solves Γ_1 and σ solves Γ_2;
- $\Gamma = \neg\Gamma_1$ and σ is not a solution for Γ_1;
- $\Gamma = C \sqsubseteq^? D$ and $\sigma(C) \sqsubseteq \sigma(D)$.

There is a straightforward reduction of disunification problems to basic disunification problems. If we view all subsumptions and dissubsumptions in a disunification problem Γ as propositional variables, we obtain a propositional formula, which is equivalent to a disjunction of satisfying valuations for the variables. We non-deterministically choose one such valuation which corresponds to a basic disunification problem. A solution to this problem is obviously a solution to the original Γ. On the other hand, a solution of Γ defines a satisfying valuation for the propositional formula mentioned above. This reduction is of the non-deterministic polynomial complexity, which matches the lower bound (NP *hardness*) for any procedure for disunification in $\mathcal{EL}$, since disunification generalizes unification and thus inherits the lower complexity bounds.

In the following we restrict the disunification problems to basic disunification problems and consider them as sets of goal subsumptions and dissubsumptions.

The characterization of dissubsumption, which we use here, is based on the contrapositive of Theorem 3.

Theorem 8. *For two concepts terms C, D, $C \not\sqsubseteq D$ iff there is a top level atom D' of D such that for all top-level atoms C' of C, $C' \not\sqsubseteq D'$.*
For two atoms C, D, $C \not\sqsubseteq D$ iff either

1. *C or D is a concept name and $C \neq D$,*
2. *$C = \exists s.C', D = \exists r.D'$ and $s \neq r$,*
3. *$C = \exists r.C', D = \exists r.D'$ and $C' \not\sqsubseteq D'$*

In contrast to unification, disunification is not equivalent to *ground* disunification, which asks for ground solutions only. Disunification is also more dependent on a given signature than unification. The intuition that justifies this statement stems from the first statement of the above theorem that requires a top-level atom to witness a dissubsumption. A disunification algorithm should either guess or compute such atom for a given goal dissubsumption, and this construction depends very much on the symbols provided by a signature.

6.1 Connection to admissibility problem

Since Description Logics and Modal Logics are so closely related, results from both areas carry over to one another [52]. There is an interesting connection between the disunification problem in $\mathcal{EL}$ and the admissibility problem in the modal logic corresponding to $\mathcal{EL}$ [26, 40, 51]. Informally speaking, an inference rule is admissible in a logic, if it does not increase the set of theorems of this logic. An inference rule is of the form:

$$\frac{A_1, \ldots, A_m}{B_1, \ldots, B_n}$$

where $A_1, \ldots, B_n$ are formulas that may contain variables. An inference rule is thus a schema for its infinitely many instances. The meaning of such a rule corresponds to the meaning of a clause: $A_1 \wedge \cdots \wedge A_m \rightarrow B_1 \vee \cdots \vee B_n$. In order for the rule to be admissible, even if the implication is not true in all models, the following implication must be true: if there is a substitution σ, such that $\sigma(A_1) \wedge \cdots \wedge \sigma(A_m) \equiv \top$, then $\sigma(B_1) \equiv \top$ or $\ldots$ or $\sigma(B_n) \equiv \top$.

Obviously, this is the case iff the disunification problem

$$\{A_1 \equiv^? \top, \ldots, A_m \equiv^? \top, B_1 \not\equiv^? \top, \ldots, B_n \not\equiv^? \top\} \tag{6.1}$$

does not have a solution.

6.2 Dismatching

Dismatching problems in $\mathcal{EL}$ are the disunification problems in which the negative part may contain a dissubsumption $C \not\sqsubseteq^? D$ only if C or D is a ground concept.

Note that the disunification problem (6.1) which corresponds to a non-addmissible inference rule in a modal logic, is in fact a dismatching problem. Hence since we show in [7] that dismatching in $\mathcal{EL}$ is decidable, and in fact, NP-complete, we can transfer this result immediately to the corresponding modal logic.

Although dismatching in $\mathcal{EL}$ is a restricted form of disunification, we have found out that it is sufficient for many practical applications. It is enough to add to a unification problem some dismatching constraints to improve results.

Theorem 9. *Dismatching in $\mathcal{EL}$ is* NP*-complete.*

A dismatching problem, just like it is in the case of matching, cannot be conveniently *flattened* as it is the case for unification problems, without ceasing to have the property of one side ground dissubsumptions. Nevertheless, the goal subsumptions (the positive part) that it contains can be flattened. Moreover we can flatten the non-ground side of a dissubsumption: if $C \not\sqsubseteq^? D$ is a part of a problem Γ and e.g., D is ground while C is not, then we can flatten C. This yields a new dismatching problem Γ', because flattening adds only subsumptions to the positive part of a dismatching problem. It is easy to see that a solution for Γ can be extended so as to be a solution for Γ'. And a solution for Γ' is a solution for Γ.

In order to prove the theorem we need only to show the *in* NP part of the statement. In order to do it, we present a non-deterministic polynomial procedure of *flattening* the dissubsumptions, reducing the problem to a *local disunification* problem. Contrary to the usual flattening of a unification problem, the flattening in this reduction involves a *don't know* kind of non-determinism.

The problem of *local disunification* is NP-complete, which is explained in the next section.

6.3 Local disunification

A local disunification problem in $\mathcal{EL}$ is given by a set of goal subsumptions and dissubsumptions exactly like in the case of disunifcation in $\mathcal{EL}$, Definition 11. The only difference is that here we check for the existence of *local* solutions for a problem (see Definition 8).

Theorem 10. *Local disunification in $\mathcal{EL}$ is* NP*-complete.*

The statement of the theorem should now be obvious. The NP-hardness follows from the NP-hardness of unification in $\mathcal{EL}$, and for the *in* NP part, we can just guess a local assignment S for variables in a problem and then check if the substitution induced by S is a solution.

7 Conclusions

This thesis presents a selection of the main results in connection to unification in the description logic $\mathcal{EL}$. All of them:

- the unification problem in $\mathcal{EL}$,
- its extension to include acyclic or cycle-restricted TBoxes,
- its restriction to matching with general TBoxes,
- dismatching and local disunifictation

are contained in the NP complexity class. Only removing the top constructor ($\top$) took us beyond NP into PSPACE complexity.

The main principle used to achieve these results was to show that it is sufficient to restrict the search space for solutions to the *local* solutions. And since each one of them is of polynomial size, constructed from a polynomial size set of local atoms, we can guess one in polynomial non-deterministic time.

The interesting unification problems that are still unsolved, like unification in $\mathcal{EL}$ with general TBoxes, disunification in $\mathcal{EL}$ or the unification problem in a combination of $\mathcal{EL}$ and $\mathcal{FL}_0$, require perhaps different techniques. A promising approach is perhaps a new computational model in [30], which provided us with a practical way to solve unification in $\mathcal{FL}_0$.

The algorithms for unification in $\mathcal{EL}$, together with a local disunification algorithm, are implemented in our system UEL [15]. UEL takes as input a unification problem and an acyclic ontology and outputs a unifier, or computes all unifiers, if the problem is unifiable. UEL gives its user the choice of using the SAT-based algorithm or the rule-based algorithm. A new ASP-based algorithm, which should improve the one based on SAT is being developed. As mentioned above, the behavior of the SAT-based and the rule-based algorithms is complementary in our experiments. SAT is better in detecting negative answers, while the rule-based approach is better in computing the first unifier if there is one.

UEL can work as a Protégé[1] plug-in and its recent version has a special option for facilitating working with the SNOMED CT ontology. It is constantly being improved with the intent to make it more suitable for practical purposes. These practical goals are situated in the realm of ontology maintenance: e.g., checking

[1] `http://protege.stanford.edu/`

for redundancies in extensions of SNOMED CT, or designing new concepts that are to be located in a given place in the hierarchy of concepts in the ontology. UEL is open source and can be downloaded at `https://sourceforge.net/projects/uel/`.

Bibliography

[1] Franz Baader. "Terminological Cycles in a Description Logic with Existential Restrictions". In: *IJCAI-03, Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence, Acapulco, Mexico, August 9-15, 2003*. Ed. by Georg Gottlob and Toby Walsh. Morgan Kaufmann, 2003, pp. 325–330. URL: http://ijcai.org/Proceedings/03/Papers/048.pdf#A#.

[2] Franz Baader. "Unification in Commutative Theories". In: *J. Symb. Comput.* 8.5 (1989), pp. 479–497. DOI: 10.1016/S0747-7171(89)80055-0. URL: http://dx.doi.org/10.1016/S0747-7171(89)80055-0.

[3] Franz Baader, Nguyen Thanh Binh, Stefan Borgwardt, and Barbara Morawska. "Deciding Unifiability and Computing Local Unifiers in the Description Logic $\mathcal{EL}$ without Top Constructor". In: *Notre Dame Journal of Formal Logic* (2016). In press.

[4] Franz Baader, Nguyen Thanh Binh, Stefan Borgwardt, and Barbara Morawska. "Unification in the Description Logic $\mathcal{EL}$ without the Top Concept". In: *Proceedings of the 23rd International Conference on Automated Deduction (CADE 2011)*. Ed. by Nikolaj Bjørner and Viorica Sofronie-Stokkermans. Vol. 6803. Lecture Notes in Computer Science. Wroclaw, Poland: Springer-Verlag, 2011, pp. 70–84.

[5] Franz Baader, Stefan Borgwardt, Julian Mendez, and Barbara Morawska. "UEL: Unification Solver for EL". In: *Proceedings of the 2012 International Workshop on Description Logics, DL-2012, Rome, Italy, June 7-10, 2012*. Ed. by Yevgeny Kazakov, Domenico Lembo, and Frank Wolter. Vol. 846. CEUR Workshop Proceedings. CEUR-WS.org, 2012. URL: http://ceur-ws.org/Vol-846/paper_8.pdf.

[6] Franz Baader, Stefan Borgwardt, and Barbara Morawska. "A Goal-Oriented Algorithm for Unification in $\mathcal{ELH}_{R+}$ w.r.t. Cycle-Restricted Ontologies". In: *AI 2012: Advances in Artificial Intelligence - 25th Australasian Joint Conference, Sydney, Australia, December 4-7, 2012. Proceedings*. Ed. by Michael Thielscher and Dongmo Zhang. Vol. 7691. Lecture Notes in Computer Science. Springer, 2012, pp. 493–504. ISBN: 978-3-642-35100-6. DOI: 10.1007/978-3-642-35101-3_42.

[7] Franz Baader, Stefan Borgwardt, and Barbara Morawska. "Dismatching and Local Disunification in $\mathcal{EL}$". In: *Proceedings of the 26th International Conference on Rewriting Techniques and Applications (RTA'15)*. Ed. by Maribel Fernández. Vol. 36. Leibniz International Proceedings in Informatics. Warsaw, Poland: Dagstuhl Publishing, 2015, pp. 40–56.

[8] Franz Baader, Stefan Borgwardt, and Barbara Morawska. "Extending Unification in EL Towards General TBoxes". In: *Principles of Knowledge Representation and Reasoning: Proceedings of the Thirteenth International Conference, KR 2012, Rome, Italy, June 10-14, 2012*. Ed. by Gerhard Brewka, Thomas Eiter, and Sheila A. McIlraith. AAAI Press, 2012. ISBN: 978-1-57735-560-1. URL: http://www.aaai.org/ocs/index.php/KR/KR12/paper/view/4491.

[9] Franz Baader, Stefan Borgwardt, and Barbara Morawska. "SAT-Encoding of Unification in $\mathcal{ELH}_R^+$ w.r.t. Cycle-Restricted Ontologies". In: *Proceedings of the 6th International Joint Conference on Automated Reasoning (IJCAR'12)*. Vol. 7364. Lecture Notes in Artificial Intelligence. Manchester, UK: Springer-Verlag, 2012, pp. 30–44.

[10] Franz Baader, Sebastian Brandt, and Carsten Lutz. "Pushing the EL Envelope". In: *IJCAI-05, Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence, Edinburgh, Scotland, UK, July 30 - August 5, 2005*. Ed. by Leslie Pack Kaelbling and Alessandro Saffiotti. Professional Book Center, 2005, pp. 364–369. ISBN: 978-0-938075-93-6. URL: http://ijcai.org/Proceedings/05/Papers/0372.pdf.

[11] Franz Baader, Diego Calvanese, Deborah L. McGuinness, Daniele Nardi, and Peter F. Patel-Schneider, eds. *The Description Logic Handbook: Theory, Implementation, and Applications*. New York, NY, USA: Cambridge University Press, 2003. ISBN: 0-521-78176-0.

[12] Franz Baader and Ralf Küsters. "Matching Concept Descriptions with Existential Restrictions". In: *KR 2000, Principles of Knowledge Representation and Reasoning Proceedings of the Seventh International Conference, Breckenridge, Colorado, USA, April 11-15, 2000*. Ed. by Anthony G. Cohn, Fausto Giunchiglia, and Bart Selman. Morgan Kaufmann, 2000, pp. 261–272.

[13] Franz Baader, Ralf Küsters, Alexander Borgida, and Deborah L. McGuinness. "Matching in Description Logics". In: *J. Log. Comput.* 9.3 (1999), pp. 411–447. DOI: 10.1093/logcom/9.3.411.

[14] Franz Baader, Ralf Küsters, and Ralf Molitor. "Computing Least Common Subsumers in Description Logics with Existential Restrictions". In: *Proceedings of the Sixteenth International Joint Conference on Artificial In-*

telligence, IJCAI 99, Stockholm, Sweden, July 31 - August 6, 1999. 2 Volumes, 1450 pages. Ed. by Thomas Dean. Morgan Kaufmann, 1999, pp. 96–103. ISBN: 978-1-55860-613-5. URL: http://ijcai.org/Proceedings/99-1/Papers/015.pdf.

[15] Franz Baader, Julian Mendez, and Barbara Morawska. "UEL: Unification Solver for the Description Logic $\mathcal{EL}$ – System Description". In: *Proceedings of the 6th International Joint Conference on Automated Reasoning (IJCAR'12)*. Vol. 7364. Lecture Notes in Artificial Intelligence. Manchester, UK: Springer-Verlag, 2012, pp. 45–51.

[16] Franz Baader and Barbara Morawska. "Matching with Respect to General Concept Inclusions in the Description Logic $\mathcal{EL}$". In: *KI 2014: Advances in Artificial Intelligence - 37th Annual German Conference on AI, Stuttgart, Germany, September 22-26, 2014. Proceedings*. Ed. by Carsten Lutz and Michael Thielscher. Vol. 8736. Lecture Notes in Computer Science. Springer, 2014, pp. 135–146. ISBN: 978-3-319-11205-3. DOI: 10.1007/978-3-319-11206-0_14.

[17] Franz Baader and Barbara Morawska. "SAT Encoding of Unification in $\mathcal{EL}$". In: *Proceedings of the 17th International Conference on Logic for Programming, Artifical Intelligence, and Reasoning (LPAR-17)*. Ed. by Christian G. Fermüller and Andrei Voronkov. Vol. 6397. Lecture Notes in Computer Science (subline Advanced Research in Computing and Software Science). Yogyakarta, Indonesia: Springer-Verlag, Oct. 2010, pp. 97–111.

[18] Franz Baader and Barbara Morawska. "Unification in the Description Logic $\mathcal{EL}$". In: *Proceedings of the 20th International Conference on Rewriting Techniques and Applications (RTA 2009)*. Ed. by Ralf Treinen. Vol. 5595. Lecture Notes in Computer Science. Springer-Verlag, 2009, pp. 350–364.

[19] Franz Baader and Barbara Morawska. "Unification in the Description Logic $\mathcal{EL}$". In: *Logical Methods in Computer Science* 6.3 (2010). Special Issue of the 20th International Conference on Rewriting Techniques and Applications; also available at http://arxiv.org/abs/1006.2289.

[20] Franz Baader and Paliath Narendran. "Unification of Concept Terms in Description Logics". In: *J. Symbolic Computation* 31.3 (2001), pp. 277–305.

[21] Franz Baader and Tobias Nipkow. *Term rewriting and all that*. Cambridge University Press, 1998. ISBN: 978-0-521-45520-6.

[22] Franz Baader and Alexander Okhotin. "Solving Language Equations and Disequations with Applications to Disunification in Description Logics and Monadic Set Constraints". In: *Logic for Programming, Artificial Intelligence, and Reasoning - 18th International Conference, LPAR-18, Mérida, Venezuela, March 11-15, 2012. Proceedings*. Ed. by Nikolaj Bjørner and Andrei Voronkov. Vol. 7180. Lecture Notes in Computer Science. Springer, 2012, pp. 107–121. ISBN: 978-3-642-28716-9. DOI: 10.1007/978-3-642-28717-6_11.

[23] Franz Baader and Klaus U. Schulz. "Combination Techniques and Decision Problems for Disunification". In: *Rewriting Techniques and Applications, 5th International Conference, RTA-93, Montreal, Canada, June 16-18, 1993, Proceedings*. Ed. by Claude Kirchner. Vol. 690. Lecture Notes in Computer Science. Springer, 1993, pp. 301–315. ISBN: 978-3-540-56868-1. DOI: 10.1007/3-540-56868-9_23.

[24] Franz Baader and Klaus U. Schulz. "Combination techniques and decision problems for disunification". In: *Theoretical Computer Science* 142.2 (1995), pp. 229–255. ISSN: 0304-3975. DOI: http://dx.doi.org/10.1016/0304-3975(94)00277-0. URL: http://www.sciencedirect.com/science/article/pii/0304397594002770.

[25] Franz Baader and Wayne Snyder. "Unification Theory". In: *Handbook of Automated Reasoning (in 2 volumes)*. Ed. by John Alan Robinson and Andrei Voronkov. Elsevier and MIT Press, 2001, pp. 445–532.

[26] Sergey Babenyshev, Vladimir V. Rybakov, Renate A. Schmidt, and Dmitry Tishkovsky. "A Tableau Method for Checking Rule Admissibility in S4". In: *Electr. Notes Theor. Comput. Sci.* 262 (2010), pp. 17–32. DOI: 10.1016/j.entcs.2010.04.003.

[27] A. Borgida and R. Küsters. *What's not in a name? Initial Explorations of a Structural Approach to Integrating Large Concept Knowledge-Bases*. Tech. rep. DCS-TR-391. Rutgers University, USA, 1999.

[28] Alexander Borgida, Ronald J. Brachman, Deborah L. McGuinness, and Lori Alperin Resnick. "CLASSIC: A Structural Data Model for Objects". In: *Proceedings of the 1989 ACM SIGMOD International Conference on Management of Data, Portland, Oregon, May 31 - June 2, 1989*. Ed. by James Clifford, Bruce G. Lindsay, and David Maier. ACM Press, 1989, pp. 58–67. DOI: 10.1145/67544.66932. URL: http://doi.acm.org/10.1145/67544.66932.

[29] Alexander Borgida and Deborah L. McGuinness. "Asking Queries about Frames". In: *Proceedings of the 1996 International Workshop on Description Logics, November 2-4, 1996, Cambridge, MA, USA*. Ed. by Lin

Padgham, Enrico Franconi, Manfred Gehrke, Deborah L. McGuinness, and Peter F. Patel-Schneider. Vol. WS-96-05. AAAI Technical Report. AAAI Press, 1996, pp. 15–24. ISBN: 978-1-57735-014-9.

[30] Stefan Borgwardt and Barbara Morawska. "Finding Finite Herbrand Models". In: *Logic for Programming, Artificial Intelligence, and Reasoning - 18th International Conference, LPAR-18, Mérida, Venezuela, March 11-15, 2012. Proceedings*. Ed. by Nikolaj Bjørner and Andrei Voronkov. Vol. 7180. Lecture Notes in Computer Science. Springer, 2012, pp. 138–152. ISBN: 978-3-642-28716-9. DOI: 10.1007/978-3-642-28717-6_13.

[31] Sebastian Brandt. "On Subsumption and Instance Problem in ELH w.r.t. General TBoxes". In: *Proceedings of the 2004 International Workshop on Description Logics (DL2004),* Whistler, British Columbia, Canada, June 2004.

[32] Wray L. Buntine and Hans-Jürgen Bürckert. "On Solving Equations and Disequations". In: *J. ACM* 41.4 (1994), pp. 591–629. DOI: 10.1145/179812.179813. URL: http://doi.acm.org/10.1145/179812.179813.

[33] Hubert Comon. "Disunification: A Survey". In: *Computational Logic - Essays in Honor of Alan Robinson*. 1991, pp. 322–359.

[34] Agostino Dovier, Carla Piazza, and Enrico Pontelli. "Disunification in ACI1 Theories". In: *Constraints* 9.1 (Jan. 2004), pp. 35–91. ISSN: 1383-7133. DOI: 10.1023/B:CONS.0000006182.84033.6e. URL: http://dx.doi.org/10.1023/B:CONS.0000006182.84033.6e.

[35] Michael R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, 1979. ISBN: 978-0-7167-1044-8.

[36] Martin Hofmann. "Proof-Theoretic Approach to Description-Logic". In: *20th IEEE Symposium on Logic in Computer Science (LICS 2005), 26-29 June 2005, Chicago, IL, USA, Proceedings*. 2005, pp. 229–237. DOI: 10.1109/LICS.2005.38. URL: http://doi.ieeecomputersociety.org/10.1109/LICS.2005.38.

[37] S. Hölldobler, N. Manthey, V.H. Nguyen, J. Stecklina, and P. Steinke. "A Short Overview on Modern Parallel SAT-Solvers". In: *Proceedings of the International Conference on Advanced Computer Science and Information Systems*. Ed. by I. Wasito et.al. 2011, pp. 201–206.

[38] Ian Horrocks and Ulrike Sattler. "A Tableaux Decision Procedure for SHOIQ". In: *IJCAI-05, Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence, Edinburgh, Scotland, UK, July 30 - August 5, 2005*. 2005, pp. 448–453. URL: http://ijcai.org/Proceedings/05/Papers/0759.pdf.

[39] Ian Horrocks, Ulrike Sattler, and Stephan Tobies. "Practical Reasoning for Very Expressive Description Logics". In: *CoRR* cs.LO/0005013 (2000). URL: http://arxiv.org/abs/cs.LO/0005013.

[40] Rosalie Iemhoff and George Metcalfe. "Proof theory for admissible rules". In: *Ann. Pure Appl. Logic* 159.1 (2009), pp. 171–186. DOI: 10.1016/j.apal.2008.10.011.

[41] Tao Jiang and B. Ravikumar. "A Note on the Space Complexity of Some Decision Problems for Finite Automata". In: *Inf. Process. Lett.* 40.1 (Oct. 1991), pp. 25–31. ISSN: 0020-0190. DOI: 10.1016/S0020-0190(05)80006-7. URL: http://dx.doi.org/10.1016/S0020-0190(05)80006-7.

[42] Deepak Kapur and Paliath Narendran. "Complexity of Unification Problems with Associative-Commutative Operators". In: *J. Autom. Reasoning* 9.2 (1992), pp. 261–288. DOI: 10.1007/BF00245463.

[43] Yevgeny Kazakov. "SRIQ and SROIQ are Harder than SHOIQ". In: *Proceedings of the 21st International Workshop on Description Logics (DL2008), Dresden, Germany, May 13-16, 2008*. 2008. URL: http://ceur-ws.org/Vol-353/Kazakov.pdf.

[44] Dexter Kozen. "Lower Bounds for Natural Proof Systems". In: *18th Annual Symposium on Foundations of Computer Science, Providence, Rhode Island, USA, 31 October - 1 November 1977*. IEEE Computer Society, 1977, pp. 254–266. DOI: 10.1109/SFCS.1977.16.

[45] Ralf Küsters. *Non-Standard Inferences in Description Logics*. Vol. 2100. Lecture Notes in Computer Science. Springer, 2001. ISBN: 3-540-42397-4. DOI: 10.1007/3-540-44613-3. URL: http://dx.doi.org/10.1007/3-540-44613-3.

[46] Hector J. Levesque and Ronald J. Brachman. "Expressiveness and tractability in knowledge representation and reasoning." In: *Computational Intelligence* 3.1 (1987), pp. 78–93. ISSN: 1467-8640. DOI: 10.1111/j.1467-8640.1987.tb00176.x. URL: http://dx.doi.org/10.1111/j.1467-8640.1987.tb00176.x.

[47] Alberto Martelli and Ugo Montanari. "An Efficient Unification Algorithm". In: *ACM Trans. Program. Lang. Syst.* 4.2 (Apr. 1982), pp. 258–282. ISSN: 0164-0925. DOI: 10.1145/357162.357169. URL: http://doi.acm.org/10.1145/357162.357169.

[48] Antonella Poggi, Domenico Lembo, Diego Calvanese, Giuseppe De Giacomo, Maurizio Lenzerini, and Riccardo Rosati. "Linking Data to Ontologies". In: *J. Data Semantics* 10 (2008), pp. 133–173. DOI: 10.1007/978-3-540-77688-8_5. URL: http://dx.doi.org/10.1007/978-3-540-77688-8_5.

[49] P. Raulefs, J. Siekmann, P. Szabó, and E. Unvericht. "A Short Survey on the State of the Art in Matching and Unification Problems". In: *SIGSAM Bull.* 13.2 (May 1979), pp. 14–20. ISSN: 0163-5824. DOI: 10.1145/1089208.1089210. URL: http://doi.acm.org/10.1145/1089208.1089210.

[50] Alan L. Rector and Ian R. Horrocks. "Experience building a large, reusable medical ontology using a description logic with transitivity and concept inclusions". In: *In Proc. of the Workshop on Ontological Engineering*. 1997, pp. 414–418.

[51] V. V. Rybakov. *Admissibility of Inference Rules*. Vol. 136. Studies in Logic and the Foundations of Mathematics. Amsterdam: North-Holland Publishing Co., 1997.

[52] Klaus Schild. "A Correspondence Theory for Terminological Logics: Preliminary Report". In: *Proceedings of the 12th International Joint Conference on Artificial Intelligence. Sydney, Australia, August 24-30, 1991*. 1991, pp. 466–471. URL: http://ijcai.org/Proceedings/91-1/Papers/072.pdf.

[53] Viorica Sofronie-Stokkermans. "Locality and subsumption testing in EL and some of its extensions". In: *Advances in Modal Logic 7, papers from the seventh conference on "Advances in Modal Logic," held in Nancy, France, 9-12 September 2008*. 2008, pp. 315–339. URL: http://www.aiml.net/volumes/volume7/Sofronie-Stokkermans.pdf.

[54] Giorgio Stefanoni and Boris Motik. "Answering Conjunctive Queries over EL Knowledge Bases with Transitive and Reflexive Roles". In: *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, January 25-30, 2015, Austin, Texas, USA*. 2015, pp. 1611–1617. URL: http://www.aaai.org/ocs/index.php/AAAI/AAAI15/paper/view/9310.

[55] Frank Wolter and Michael Zakharyaschev. "Undecidability of the unification and admissibility problems for modal and description logics". In: *ACM Trans. Comput. Log.* 9.4 (2008). DOI: 10.1145/1380572.1380574. URL: http://doi.acm.org/10.1145/1380572.1380574.

Appendix: submitted publications

Unification in the Description Logic $\mathcal{EL}$

Franz Baader and Barbara Morawska

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

Abstract. The Description Logic $\mathcal{EL}$ has recently drawn considerable attention since, on the one hand, important inference problems such as the subsumption problem are polynomial. On the other hand, $\mathcal{EL}$ is used to define large biomedical ontologies. Unification in Description Logics has been proposed as a novel inference service that can, for example, be used to detect redundancies in ontologies. The main result of this paper is that unification in $\mathcal{EL}$ is decidable. More precisely, $\mathcal{EL}$-unification is NP-complete, and thus has the same complexity as $\mathcal{EL}$-matching. We also show that, w.r.t. the unification type, $\mathcal{EL}$ is less well-behaved: it is of type zero, which in particular implies that there are unification problems that have no finite complete set of unifiers.

1 Introduction

Description logics (DLs) [5] are a family of logic-based knowledge representation formalisms, which can be used to represent the conceptual knowledge of an application domain in a structured and formally well-understood way. They are employed in various application domains, such as natural language processing, configuration, databases, and biomedical ontologies, but their most notable success so far is the adoption of the DL-based language OWL [15] as standard ontology language for the semantic web.

In DLs, concepts are formally described by *concept terms*, i.e., expressions that are built from concept names (unary predicates) and role names (binary predicates) using concept constructors. The expressivity of a particular DL is determined by which concept constructors are available in it. From a semantic point of view, concept names and concept terms represent sets of individuals, whereas roles represent binary relations between individuals. For example, using the concept name Woman, and the role name child, the concept of all *women having a daughter* can be represented by the concept term

$$\mathsf{Woman} \sqcap \exists \mathsf{child}.\mathsf{Woman},$$

and the concept of all *women having only daughters* by

$$\mathsf{Woman} \sqcap \forall \mathsf{child}.\mathsf{Woman}.$$

Knowledge representation systems based on DLs provide their users with various inference services that allow them to deduce implicit knowledge from the explicitly represented knowledge. For instance, the subsumption algorithm allows one to

R. Treinen (Ed.): RTA 2009, LNCS 5595, pp. 350–364, 2009.

determine subconcept-superconcept relationships. For example, the concept term Woman subsumes the concept term Woman $\sqcap \exists$child.Woman since all instances of the second term are also instances of the first term, i.e., the second term is always interpreted as a subset of the first term. With the help of the subsumption algorithm, a newly introduced concept term can automatically be placed at the correct position in the hierarchy of the already existing concept terms.

Two concept terms C, D are *equivalent* ($C \equiv D$) if they subsume each other, i.e., if they always represent the same set of individuals. For example, the terms $\forall$child.Rich $\sqcap \forall$child.Woman and $\forall$child.(Rich $\sqcap$ Woman) are equivalent since the value restriction operator ($\forall r.C$) distributes over the conjunction operator ($\sqcap$). If we replaced the value restriction operator by the existential restriction operator ($\exists r.C$), then this equivalence would no longer hold. However, for this operator, we still have the equivalence

$$\exists \text{child.Rich} \sqcap \exists \text{child.}(\text{Woman} \sqcap \text{Rich}) \equiv \exists \text{child.}(\text{Woman} \sqcap \text{Rich}).$$

The equivalence test can, for example, be used to find out whether a concept term representing a particular notion has already been introduced, thus avoiding multiple introduction of the same concept into the concept hierarchy. This inference capability is very important if the knowledge base containing the concept terms is very large, evolves during a long time period, and is extended and maintained by several knowledge engineers. However, testing for equivalence of concepts is not always sufficient to find out whether, for a given concept term, there already exists another concept term in the knowledge base describing the same notion. For example, assume that one knowledge engineer has defined the concept of all *women having a daughter* by the concept term

$$\text{Woman} \sqcap \exists \text{child.Woman}.$$

A second knowledge engineer might represent this notion in a somewhat more fine-grained way, e.g., by using the term Female $\sqcap$ Human in place of Woman. The concept terms Woman $\sqcap \exists$child.Woman and

$$\text{Female} \sqcap \text{Human} \sqcap \exists \text{child.}(\text{Female} \sqcap \text{Human})$$

are not equivalent, but they are meant to represent the same concept. The two terms can obviously be made equivalent by substituting the concept name Woman in the first term by the concept term Female $\sqcap$ Human. This leads us to *unification of concept terms*, i.e., the question whether two concept terms can be made equivalent by applying an appropriate substitution, where a substitution replaces (some of the) concept names by concept terms. Of course, it is not necessarily the case that unifiable concept terms are meant to represent the same notion. A unifiability test can, however, suggest to the knowledge engineer possible candidate terms.

Unification in DLs was first considered in [9] for a DL called $\mathcal{FL}_0$, which has the concept constructors *conjunction* ($\sqcap$), *value restriction* ($\forall r.C$), and the *top concept* ($\top$). It was shown that unification in $\mathcal{FL}_0$ is decidable and

ExpTime-complete, i.e., given an $\mathcal{FL}_0$-unification problem, we can effectively decide whether it has a solution or not, but in the worst-case, any such decision procedure needs exponential time. This result was extended in [7] to a more expressive DL, which additional has the role constructor *transitive closure*. Interestingly, the *unification type* of $\mathcal{FL}_0$ had been determined almost a decade earlier in [1]. In fact, as shown in [9], unification in $\mathcal{FL}_0$ corresponds to unification modulo the equational theory of idempotent Abelian monoids with several homomorphisms. In [1] it was shown that, already for a single homomorphism, unification modulo this theory has unification type zero, i.e., there are unification problems for this theory that do not have a minimal complete set of unifiers. In particular, such unification problems cannot have a finite complete set of unifiers.

In this paper, we consider unification in the DL $\mathcal{EL}$. The $\mathcal{EL}$-family consists of inexpressive DLs whose main distinguishing feature is that they provide their users with *existential restrictions* ($\exists r.C$) rather than value restrictions ($\forall r.C$) as the main concept constructor involving roles. The core language of this family is $\mathcal{EL}$, which has the top concept, conjunction, and existential restrictions as concept constructors. This family has recently drawn considerable attention since, on the one hand, the subsumption problem stays tractable (i.e., decidable in polynomial time) in situations where $\mathcal{FL}_0$, the corresponding DL with value restrictions, becomes intractable: subsumption between concept terms is tractable for both $\mathcal{FL}_0$ and $\mathcal{EL}$, but allowing the use of concept definitions or even more expressive terminological formalisms makes $\mathcal{FL}_0$ intractable [2,16,4], whereas it leaves $\mathcal{EL}$ tractable [3,13,4]. On the other hand, although of limited expressive power, $\mathcal{EL}$ is nevertheless used in applications, e.g., to define biomedical ontologies. For example, both the large medical ontology SNOMED CT[1] and the Gene Ontology[2] can be expressed in $\mathcal{EL}$, and the same is true for large parts of the medical ontology GALEN [18]. The importance of $\mathcal{EL}$ can also be seen from the fact that the new OWL 2 standard[3] contains a sub-profile OWL 2 EL, which is based on (an extension of) $\mathcal{EL}$.

Unification in $\mathcal{EL}$ has, to the best of our knowledge, not been investigated before, but matching (where one side of the equation(s) to be solved does not contain variables) has been considered in [6,17]. In particular, it was shown in [17] that the decision problem, i.e., the problem of deciding whether a given $\mathcal{EL}$-matching problem has a matcher or not, is NP-complete. Interestingly, $\mathcal{FL}_0$ behaves better w.r.t. matching than $\mathcal{EL}$: for $\mathcal{FL}_0$, the decision problem is tractable [8]. In this paper, we show that, w.r.t. the unification type, $\mathcal{FL}_0$ and $\mathcal{EL}$ behave the same: just as $\mathcal{FL}_0$, the DL $\mathcal{EL}$ has unification type zero. However, w.r.t. the decision problem, $\mathcal{EL}$ behaves much better than $\mathcal{FL}_0$: $\mathcal{EL}$-unification is NP-complete, and thus has the same complexity as $\mathcal{EL}$-matching.

In the next section, we define the DL $\mathcal{EL}$ and unification in $\mathcal{EL}$ more formally. In Section 3, we recall the characterisation of subsumption and equivalence in

[1] http://www.ihtsdo.org/snomed-ct/
[2] http://www.geneontology.org/
[3] See http://www.w3.org/TR/owl2-profiles/

$\mathcal{EL}$ from [17], and in Section 4 we use this to show that unification in $\mathcal{EL}$ has type zero. In Section 5, we show that unification in $\mathcal{EL}$ is NP-complete, and in Section 6 we point out that our results for $\mathcal{EL}$-unification imply that unification modulo the equational theory of semilattices with monotone operators [19] is NP-complete and of unification type zero.

More information about Description Logics can be found in [5], and about unification theory in [12].

2 Unification in $\mathcal{EL}$

First, we define the syntax and semantics of $\mathcal{EL}$-concept terms as well as the subsumption and the equivalence relation on these terms.

Starting with a set N_{con} of concept names and a set N_{role} of role names, *$\mathcal{EL}$-concept terms* are built using the concept constructors top concept ($\top$), conjunction ($\sqcap$), and existential restriction ($\exists r.C$). The semantics of $\mathcal{EL}$ is defined in the usual way, using the notion of an interpretation $\mathcal{I} = (\mathcal{D}_\mathcal{I}, \cdot^\mathcal{I})$, which consists of a nonempty domain $\mathcal{D}_\mathcal{I}$ and an interpretation function $\cdot^\mathcal{I}$ that assigns binary relations on $\mathcal{D}_\mathcal{I}$ to role names and subsets of $\mathcal{D}_\mathcal{I}$ to concept terms, as shown in the semantics column of Table 1.

Table 1. Syntax and semantics of $\mathcal{EL}$

Name	Syntax	Semantics
concept name	A	$A^\mathcal{I} \subseteq \mathcal{D}_\mathcal{I}$
role name	r	$r^\mathcal{I} \subseteq \mathcal{D}_\mathcal{I} \times \mathcal{D}_\mathcal{I}$
top-concept	$\top$	$\top^\mathcal{I} = \mathcal{D}_\mathcal{I}$
conjunction	$C \sqcap D$	$(C \sqcap D)^\mathcal{I} = C^\mathcal{I} \cap D^\mathcal{I}$
existential restriction	$\exists r.C$	$(\exists r.C)^\mathcal{I} = \{x \mid \exists y : (x,y) \in r^\mathcal{I} \wedge y \in C^\mathcal{I}\}$
subsumption	$C \sqsubseteq D$	$C^\mathcal{I} \subseteq D^\mathcal{I}$
equivalence	$C \equiv D$	$C^\mathcal{I} = D^\mathcal{I}$

The concept term C *is subsumed by* the concept term D (written $C \sqsubseteq D$) iff $C^\mathcal{I} \subseteq D^\mathcal{I}$ holds for all interpretations $\mathcal{I}$. We say that C *is equivalent to* D (written $C \equiv D$) iff $C \sqsubseteq D$ and $D \sqsubseteq C$, i.e., iff $C^\mathcal{I} = D^\mathcal{I}$ holds for all interpretations $\mathcal{I}$. The concept term C *is strictly subsumed by* the concept term D (written $C \sqsubset D$) iff $C \sqsubseteq D$ and $C \not\equiv D$.

A *concept definition* is of the form $A \doteq C$ where A is a concept name and C is a concept term. A *TBox* $\mathcal{T}$ is a finite set of concept definitions such that no concept name occurs more than once on the left-hand side of a concept definition in $\mathcal{T}$. The TBox $\mathcal{T}$ is called *acyclic* if there are no cyclic dependencies between its concept definitions. The interpretation $\mathcal{I}$ is a model of the TBox $\mathcal{T}$ iff $A^\mathcal{I} = C^\mathcal{I}$ holds for all concept definitions $A \doteq C$ in $\mathcal{T}$. Subsumption and equivalence w.r.t. a TBox are defined as follows: $C \sqsubseteq_\mathcal{T} D$ ($C \equiv_\mathcal{T} D$) iff $C^\mathcal{I} \subseteq D^\mathcal{I}$ ($C^\mathcal{I} = D^\mathcal{I}$) holds for all models $\mathcal{I}$ of $\mathcal{T}$. Subsumption and equivalence

w.r.t. an acyclic TBox can be reduced to subsumption and equivalence of concept terms (without TBox) by *expanding* the concept terms w.r.t. the TBox, i.e., by replacing defined concepts (i.e., concept names occurring on the left-hand side of a definition) by their definitions (i.e., the corresponding right-hand sides) until all defined concepts have been replaced. This expansion process may, however, result in an exponential blow-up [10].

In order to define unification of concept terms, we first introduce the notion of a substitution operating on concept terms. To this purpose, we partition the set of concepts names into a set N_v of concept variables (which may be replaced by substitutions) and a set N_c of concept constants (which must not be replaced by substitutions). Intuitively, N_v are the concept names that have possibly been given another name or been specified in more detail in another concept term describing the same notion. The elements of N_c are the ones of which it is assumed that the same name is used by all knowledge engineers (e.g., standardised names in a certain domain).

A *substitution* σ is a mapping from N_v into the set of all $\mathcal{EL}$-concept terms. This mapping is extended to concept terms in the obvious way, i.e.,

- $\sigma(A) := A$ for all $A \in N_c$,
- $\sigma(\top) := \top$,
- $\sigma(C \sqcap D) := \sigma(C) \sqcap \sigma(D)$, and
- $\sigma(\exists r.C) := \exists r.\sigma(C)$.

Definition 1. *An* $\mathcal{EL}$-unification problem *is of the form* $\Gamma = \{C_1 \equiv^? D_1, \ldots, C_n \equiv^? D_n\}$, *where* $C_1, D_1, \ldots C_n, D_n$ *are* $\mathcal{EL}$*-concept terms. The substitution* σ *is a* unifier *(or* solution*) of* Γ *iff* $\sigma(C_i) \equiv \sigma(D_i)$ *for* $i = 1, \ldots, n$. *In this case,* Γ *is called* solvable *or* unifiable.

When we say that $\mathcal{EL}$-unification is *decidable (NP-complete)*, then we mean that the following decision problem is decidable (NP-complete): given an $\mathcal{EL}$-unification problem Γ, decide whether Γ is solvable or not.

As usual, unifiers can be compared using the instantiation preorder $\preceq^\bullet$. Let Γ be an $\mathcal{EL}$-unification problem, V the set of variables occurring in Γ, and σ, θ two unifiers of this problem. We define

$\sigma \preceq^\bullet \theta$ iff there is a substitution λ such that $\theta(X) \equiv \lambda(\sigma(X))$ for all $X \in V$.

If $\sigma \preceq^\bullet \theta$, then we say that θ is an *instance* of σ.

Definition 2. *Let* Γ *be an* $\mathcal{EL}$*-unification problem. The set of substitutions* M *is called a* complete set of unifiers *for* Γ *iff it satisfies the following two properties:*

1. *every element of* M *is a unifier of* Γ*;*
2. *if* θ *is a unifier of* Γ*, then there exists a unifier* $\sigma \in M$ *such that* $\sigma \preceq^\bullet \theta$.

The set M *is called a* minimal complete set of unifiers *for* Γ *iff it additionally satisfies*

3. *if* $\sigma, \theta \in M$*, then* $\sigma \preceq^\bullet \theta$ *implies* $\sigma = \theta$.

The unification type of a given unification problem is determined by the existence and cardinality of such a minimal complete set.

Definition 3. *Let Γ be an $\mathcal{EL}$-unification problem. This problem has type* unitary *(*finitary*,* infinitary*) iff it has a minimal complete set of unifiers of cardinality* 1 *(finite cardinality, infinite cardinality). If Γ does not have a minimal complete set of unifiers, then it is of type* zero*.*

Note that the set of all unifiers of a given $\mathcal{EL}$-unification problem is always a complete set of unifiers. However, this set is usually infinite and redundant (in the sense that some unifiers are instances of others). For a unitary or finitary $\mathcal{EL}$-unification problem, all unifiers can be represented by a finite complete set of unifiers, whereas for problems of type infinitary or zero this is no longer possible. In fact, if a problem has a finite complete set of unifiers M, then it also has a finite *minimal* complete set of unifiers, which can be obtained by iteratively removing redundant elements from M. For an infinite complete set of unifiers, this approach of removing redundant unifiers may be infinite, and the set reached in the limit need no longer be complete. This is what happens for problems of type zero. The difference between infinitary and type zero is that a unification problem of type zero cannot even have a non-redundant complete set of unifiers, i.e., every complete set of unifiers must contain different unifiers σ, θ such that $\sigma \preceq \theta$.

When we say that $\mathcal{EL}$ *has unification type zero*, we mean that there exists an $\mathcal{EL}$-unification problem that has type zero. Before we can prove that this is indeed the case, we must first have a closer look at equivalence in $\mathcal{EL}$.

3 Equivalence and Subsumption in $\mathcal{EL}$

In order to characterise equivalence of $\mathcal{EL}$-concept terms, the notion of a reduced $\mathcal{EL}$-concept term is introduced in [17]. A given $\mathcal{EL}$-concept term can be transformed into an equivalent reduced term by applying the following rules modulo associativity and commutativity of conjunction:

$C \sqcap \top \to C$	for all $\mathcal{EL}$-concept terms C
$A \sqcap A \to A$	for all concept names $A \in N_{con}$
$\exists r.C \sqcap \exists r.D \to \exists r.C$	for all $\mathcal{EL}$-concept terms C, D with $C \sqsubseteq D$

Obviously, these rules are equivalence preserving. We say that the $\mathcal{EL}$-concept term C is *reduced* if none of the above rules is applicable to it (modulo associativity and commutativity of $\sqcap$). The $\mathcal{EL}$-concept term D is a *reduced form* of C if D is reduced and can be obtained from C by applying the above rules (modulo associativity and commutativity of $\sqcap$). The following theorem is an easy consequence of Theorem 6.3.1 on page 181 of [17].

Theorem 1. *Let C, D be $\mathcal{EL}$-concept terms, and $\widehat{C}, \widehat{D}$ reduced forms of C, D, respectively. Then $C \equiv D$ iff $\widehat{C}$ is identical to $\widehat{D}$ up to associativity and commutativity of $\sqcap$.*

This theorem can also be used to derive a recursive characterisation of subsumption in $\mathcal{EL}$. In fact, if $C \sqsubseteq D$, then $C \sqcap D \equiv C$, and thus C and $C \sqcap D$ have the same reduced form. Thus, during reduction, all concept names and existential restrictions of D must be "eaten up" by corresponding concept names and existential restrictions of C.

Corollary 1. *Let $C = A_1 \sqcap \ldots \sqcap A_k \sqcap \exists r_1.C_1 \sqcap \ldots \sqcap \exists r_m.C_m$ and $D = B_1 \sqcap \ldots \sqcap B_\ell \sqcap \exists s_1.D_1 \sqcap \ldots \sqcap \exists s_n.D_n$, where $A_1, \ldots, A_k, B_1, \ldots, B_\ell$ are concept names. Then $C \sqsubseteq D$ iff $\{B_1, \ldots, B_\ell\} \subseteq \{A_1, \ldots, A_k\}$ and for every $j, 1 \leq j \leq n$, there exists an $i, 1 \leq i \leq m$, such that $r_i = s_j$ and $C_i \sqsubseteq D_j$.*

Note that this corollary also covers the cases where some of the numbers k, ℓ, m, n are zero. The empty conjunction should then be read as $\top$. The following lemma, which is an immediate consequence of this corollary, will be used in our proof that $\mathcal{EL}$ has unification type zero.

Lemma 1. *If C, D are reduced $\mathcal{EL}$-concept terms such that $\exists r.D \sqsubseteq C$, then C is either $\top$, or of the form $C = \exists r.C_1 \sqcap \ldots \sqcap \exists r.C_n$ where $n \geq 1$; $C_1, \ldots, C_n$ are reduced and pairwise incomparable w.r.t. subsumption; and $D \sqsubseteq C_1, \ldots, D \sqsubseteq C_n$. Conversely, if C, D are $\mathcal{EL}$-concept terms such that $C = \exists r.C_1 \sqcap \ldots \sqcap \exists r.C_n$ and $D \sqsubseteq C_1, \ldots, D \sqsubseteq C_n$, then $\exists r.D \sqsubseteq C$.*

In the proof of decidability of $\mathcal{EL}$-unification, we will make use of the fact that the inverse strict subsumption order is well-founded.

Proposition 1. *There is no infinite sequence $C_0, C_1, C_2, C_3, \ldots$ of $\mathcal{EL}$-concept terms such that $C_0 \sqsubset C_1 \sqsubset C_2 \sqsubset C_3 \sqsubset \cdots$.*

Proof. We define the *role depth* of an $\mathcal{EL}$-concept term C as the maximal nesting of existential restrictions in C. Let n_0 be the role depth of C_0. Since $C_0 \sqsubseteq C_i$ for $i \geq 1$, it is an easy consequence of Corollary 1 that the role depth of C_i is bounded by n_0, and that C_i contains only concept and role names occurring in C_0. In addition, it is known that, for a given natural number n_0 and finite sets of concept names $\mathcal{C}$ and role names $\mathcal{R}$, there are, up to equivalence, only finitely many $\mathcal{EL}$-concept term built using concept names from $\mathcal{C}$ and role names from $\mathcal{R}$ and of a role depth bounded by n_0 [11]. Consequently, there are indices $i < j$ such that $C_i \equiv C_j$. This contradicts our assumption that $C_i \sqsubset C_j$. □

4 An $\mathcal{EL}$-Unification Problem of Type Zero

To show that $\mathcal{EL}$ has unification type zero, we exhibit an $\mathcal{EL}$-unification problem that has this type.

Theorem 2. *Let X, Y be variables. The $\mathcal{EL}$-unification problem $\Gamma := \{X \sqcap \exists r.Y \equiv^? \exists r.Y\}$ has unification type zero.*

Proof. It is enough to show that any complete set of unifiers for this problem is redundant, i.e., contains two different unifiers that are comparable w.r.t. the instantiation preorder. Thus, let M be a complete set of unifiers for Γ.

First, note that M must contain a unifier that maps X to an $\mathcal{EL}$-concept term not equivalent to $\top$ or $\exists r.\top$. In fact, consider a substitution τ such that $\tau(X) = \exists r.A$ and $\tau(Y) = A$. Obviously, τ is a unifier of Γ. Thus, M must contain a unifier σ such that $\sigma \preceq \tau$. In particular, this means that there is a substitution λ such that $\exists r.A = \tau(X) \equiv \lambda(\sigma(X))$. Obviously, $\sigma(X) \equiv \top$ ($\sigma(X) \equiv \exists r.\top$) would imply $\lambda(\sigma(X)) \equiv \top$ ($\lambda(\sigma(X)) \equiv \exists r.\top$), and thus $\exists r.A \equiv \top$ ($\exists r.A \equiv \exists r.\top$), which is, however, not the case.

Thus, let $\sigma \in M$ be such that $\sigma(X) \not\equiv \top$ and $\sigma(X) \not\equiv \exists r.\top$. Without loss of generality, we assume that $C := \sigma(X)$ and $D := \sigma(Y)$ are reduced. Since σ is a unifier of Γ, we have $\exists r.D \sqsubseteq C$. Consequently, Lemma 1 yields that C is of the form $C = \exists r.C_1 \sqcap \ldots \sqcap \exists r.C_n$ where $n \geq 1$, $C_1, \ldots, C_n$ are reduced and pairwise incomparable w.r.t. subsumption, and $D \sqsubseteq C_1, \ldots, D \sqsubseteq C_n$.

We use σ to construct a new unifier $\widehat{\sigma}$ as follows:

$$\begin{aligned} \widehat{\sigma}(X) &:= \exists r.C_1 \sqcap \ldots \sqcap \exists r.C_n \sqcap \exists r.Z \\ \widehat{\sigma}(Y) &:= D \sqcap Z \end{aligned}$$

where Z is a new variable (i.e., one not occurring in C, D). The second part of Lemma 1 implies that $\widehat{\sigma}$ is indeed a unifier of Γ.

Next, we show that $\widehat{\sigma} \preceq \sigma$. To this purpose, we consider the substitution λ that maps Z to C_1, and does not change any of the other variables. Then we have $\lambda(\widehat{\sigma}(X)) = \exists r.C_1 \sqcap \ldots \sqcap \exists r.C_n \sqcap \exists r.C_1 \equiv \exists r.C_1 \sqcap \ldots \sqcap \exists r.C_n = \sigma(X)$ and $\lambda(\widehat{\sigma}(Y)) = D \sqcap C_1 \equiv D = \sigma(Y)$. Note that the second equivalence holds since we have $D \sqsubseteq C_1$.

Since M is complete, there exists a unifier $\theta \in M$ such that $\theta \preceq \widehat{\sigma}$. Transitivity of the relation $\preceq$ thus yields $\theta \preceq \sigma$. Since σ and θ both belong to M, we have completed the proof of the theorem once we have shown that $\sigma \neq \theta$. Assume to the contrary that $\sigma = \theta$. Then we have $\sigma \preceq \widehat{\sigma}$, and thus there exists a substitution μ such that $\mu(\sigma(X)) \equiv \widehat{\sigma}(X)$, i.e.,

$$\exists r.\mu(C_1) \sqcap \ldots \sqcap \exists r.\mu(C_n) \equiv \exists r.C_1 \sqcap \ldots \sqcap \exists r.C_n \sqcap \exists r.Z. \tag{1}$$

Recall that the concept terms $C_1, \ldots, C_n$ are reduced and pairwise incomparable w.r.t. subsumption. In addition, since $\sigma(X) = \exists r.C_1 \sqcap \ldots \sqcap \exists r.C_n$ is reduced and not equivalent to $\exists r.\top$, none of the concept terms $C_1, \ldots, C_n$ can be equivalent to $\top$. Finally, Z is a concept name that does not occur in $C_1, \ldots, C_n$. All this implies that $\exists r.C_1 \sqcap \ldots \sqcap \exists r.C_n \sqcap \exists r.Z$ is reduced. Obviously, any reduced form for $\exists r.\mu(C_1) \sqcap \ldots \sqcap \exists r.\mu(C_n)$ is a conjunction of at most n existential restrictions. Thus, Theorem 1 shows that the above equivalence (1) actually cannot hold.

To sum up, we have shown that M contains two distinct unifiers σ, θ such that $\theta \preceq \sigma$. Since M was an arbitrary complete set of unifiers for Γ, this shows that this unification problem cannot have a minimal complete set of unifiers. □

5 The Decision Problem

Before we can describe our decision procedure for $\mathcal{EL}$-unification, we must introduce some notation. An $\mathcal{EL}$-concept term is called an *atom* iff it is a concept name (i.e., concept constant or concept variable) or an existential restriction $\exists r.D$. Obviously, any $\mathcal{EL}$-concept term is (equivalent to) a conjunction of atoms, where the empty conjunction is $\top$. The set $At(C)$ of *atoms of an $\mathcal{EL}$-concept term C* is defined inductively: if $C = \top$, then $At(C) := \emptyset$; if C is a concept name, then $At(C) := \{C\}$; if $C = \exists r.D$ then $At(C) := \{C\} \cup At(D)$; if $C = C_1 \sqcap C_2$, then $At(C) := At(C_1) \cup At(C_2)$.

Concept names and existential restrictions $\exists r.D$ where D is a concept name or $\top$ are called *flat atoms*. The $\mathcal{EL}$-unification problem Γ is *flat* iff it only contains equations of the following form:

- $X \equiv^? C$ where X is a variable and C is a non-variable flat atom;
- $X_1 \sqcap \ldots \sqcap X_m \equiv^? Y_1 \sqcap \ldots \sqcap Y_n$ where $X_1, \ldots, X_m, Y_1, \ldots, Y_n$ are variables.

By introducing new concept variables and eliminating $\top$, any $\mathcal{EL}$-unification problem Γ can be transformed in polynomial time into a flat $\mathcal{EL}$-unification problem Γ' such that Γ is solvable iff Γ' is solvable. Thus, we may assume without loss of generality that our input $\mathcal{EL}$-unification problems are flat. Given a flat $\mathcal{EL}$-unification problem $\Gamma = \{C_1 \equiv^? D_1, \ldots, C_n \equiv^? D_n\}$, we call the atoms of $C_1, D_1, \ldots, C_n, D_n$ the *atoms of Γ*.

The unifier σ of Γ is called *reduced* (*ground*) iff, for all concept variables X occurring in Γ, the $\mathcal{EL}$-concept term $\sigma(X)$ is reduced (does not contain variables). Obviously, Γ is solvable iff it has a reduced ground unifier. Given a ground unifier σ of Γ, we consider the set $At(\sigma)$ of all atoms of $\sigma(X)$, where X ranges over all variables occurring in Γ. We call the elements of $At(\sigma)$ the *atoms of σ*.

Given $\mathcal{EL}$-concept terms C, D, we define $C >_{is} D$ iff $C \sqsubset D$. Proposition 1 says that the strict order $>_{is}$ defined this way is well-founded. This order is monotone in the following sense.

Lemma 2. *Let C, D, D' be $\mathcal{EL}$-concept terms such that $D >_{is} D'$ and C is reduced and contains at least one occurrence of D. If C' is obtained from C by replacing all occurrences of D by D', then $C >_{is} C'$.*

Proof. We prove the lemma by induction on the size of C. If $C = D$, then $C' = D'$, and thus $C = D >_{is} D' = C'$. Thus, assume that $C \neq D$. In this case, C obviously cannot be a concept name. If $C = \exists r.C_1$, then D occurs in C_1. By induction, we can assume that $C_1 >_{is} C_1'$, where C_1' is obtained from C_1 by replacing all occurrences of D by D'. Thus, we have $C = \exists r.C_1 >_{is} \exists r.C_1' = C'$ by Corollary 1. Finally, assume that $C = C_1 \sqcap \ldots \sqcap C_n$ for $n > 1$ atoms $C_1, \ldots, C_n$. Since C is reduced, these atoms are incomparable w.r.t. subsumption, and since D occurs in C we can assume without loss of generality that D occurs in C_1. Let $C_1', \ldots, C_n'$ be respectively obtained from $C_1, \ldots, C_n$ by replacing every occurrence of D by D', and then reducing the concept term obtained this way. By induction, we have $C_1 >_{is} C_1'$. Assume that $C \not>_{is} C'$.

Since the concept constructors of $\mathcal{EL}$ are monotone w.r.t. subsumption $\sqsubseteq$, we have $C \sqsubseteq C'$, and thus $C \not>_{is} C'$ means that $C \equiv C'$. Consequently, $C = C_1 \sqcap \ldots \sqcap C_n$ and the reduced form of $C'_1 \sqcap \ldots \sqcap C'_n$ must be equal up to associativity and commutativity of $\sqcap$. If $C'_1 \sqcap \ldots \sqcap C'_n$ is not reduced, then its reduced form is actually a conjunction of $m < n$ atoms, which contradicts $C \equiv C'$. If $C'_1 \sqcap \ldots \sqcap C'_n$ is reduced, then $C_1 >_{is} C'_1$ implies that there is an $i \neq 1$ such that $C_i \equiv C'_1$. However, then $C_i \equiv C'_1 \sqsupset C_1$ contradicts the fact that the atoms $C_1, \ldots, C_n$ are incomparable w.r.t. subsumption. □

We use the order $>_{is}$ on $\mathcal{EL}$-concept terms to define a well-founded order on ground unifiers. Since $>_{is}$ is well-founded, its multiset extension $>_m$ is also well-founded. Given a ground unifier σ of Γ, we consider the multiset $S(\sigma)$ of all $\mathcal{EL}$-concept terms $\sigma(X)$, where X ranges over all concept variables occurring in Γ. For two ground unifiers σ, θ of Γ, we define $\sigma \succ \theta$ iff $S(\sigma) >_m S(\theta)$. The ground unifier σ of Γ is *minimal* iff there is no ground unifier θ of Γ such that $\sigma \succ \theta$. The following proposition is an easy consequence of the fact that $\succ$ is well-founded.

Proposition 2. *Let Γ be an $\mathcal{EL}$-unification problem. Then Γ is solvable iff it has a minimal reduced ground unifier.*

In the following, we show that minimal reduced ground unifiers of flat $\mathcal{EL}$-unification problems satisfy properties that make it easy to check (with an NP-algorithm) whether such a unifier exists or not.

Lemma 3. *Let Γ be a flat $\mathcal{EL}$-unification problem and γ a minimal reduced ground unifier of Γ. If C is an atom of γ, then there is a non-variable atom D of Γ such that $C \equiv \gamma(D)$.*

Proof. Since γ is ground, C is either a concept constant or an existential restriction. First, assume that $C = A$ for a concept constant A, but there is no non-variable atom D of Γ such that $A \equiv \gamma(D)$. This simply means that A does not occur in Γ. Let γ' be the substitution obtained from γ by replacing every occurrence of A by $\top$. Since equivalence in $\mathcal{EL}$ is preserved under replacing concept names by $\top$, and since A does not occur in Γ, it is easy to see that γ' is also a unifier of Γ. However, since $\gamma \succ \gamma'$, this contradicts our assumption that γ is minimal.

Second, assume that $C = \exists r.C_1$, but there is no non-variable atom D of Γ such that $C \equiv \gamma(D)$. We assume that C is maximal (w.r.t. subsumption) with this property, i.e., for every atom C' of γ with $C \sqsubset C'$, there is a non-variable atom D' of Γ such that $C' \equiv \gamma(D')$. Let $D_1, \ldots, D_n$ be all the atoms of Γ with $C \sqsubseteq \gamma(D_i)$ $(i = 1, \ldots, n)$. By our assumptions on C, we actually have $C \sqsubset \gamma(D_i)$ and, by Lemma 1, the atom D_i is also an existential restriction $D_i = \exists r.D'_i$ $(i = 1, \ldots, n)$. The conjunction $\widehat{D} := \gamma(D_1) \sqcap \ldots \sqcap \gamma(D_n)$ obviously subsumes C. We claim that this subsumption relationship is actually strict. In fact, if $n = 0$, then $\widehat{D} = \top$, and since C is an atom, it is not equivalent to $\top$. If $n \geq 1$, then $C = \exists r.C_1 \sqsupseteq \exists r.\gamma(D'_1) \sqcap \ldots \sqcap \exists r.\gamma(D_n)$ would imply (by

Corollary 1) that there is an $i, 1 \leq i \leq n$, with $C_1 \sqsupseteq \gamma(D'_i)$. However, this would yield $C = \exists r.C_1 \sqsupseteq \exists r.\gamma(D'_i) = \gamma(D_i)$, which contradicts the fact that $C \sqsubset \gamma(D_i)$. Thus, we have shown that $C \sqsubset \widehat{D}$. The substitution γ' is obtained from γ by replacing every occurrence of C by $\widehat{D}$. Lemma 2 implies that $\gamma \succ \gamma'$. Thus, to obtain the desired contradiction, it is sufficient to show that γ' is a unifier of Γ.

First, consider an equation of the form $X \equiv^? E$ in Γ, where X is a variable and E is a non-variable flat atom. If E is a concept constant, then $\gamma(X) = E$, and thus $\gamma'(X) = \gamma(X)$, which shows that γ' solves this equation. Thus, assume that $E = \exists r.E'$. Since γ is reduced, we actually have $\gamma(X) = \exists r.\gamma(E')$. If C occurs in $\gamma(E')$, then each replacement of C by $\widehat{D}$ in $\gamma(E')$ is matched by the corresponding replacement in $\gamma(X)$. Thus, in this case γ' again solves the equation. Finally, assume that $C = \gamma(X)$. But then $C \equiv \gamma(E)$ for a non-variable atom E of Γ, which contradicts our assumption on C.

Second, consider an equation of the form $X_1 \sqcap \ldots \sqcap X_m \equiv^? Y_1 \sqcap \ldots \sqcap Y_n$ where $X_1, \ldots, X_m, Y_1, \ldots, Y_n$ are variables. Then $L := \gamma(X_1 \sqcap \ldots \sqcap X_m)$ and $R := \gamma(Y_1 \sqcap \ldots \sqcap Y_n)$ reduce to the same reduced $\mathcal{EL}$-concept term J. Let L', R', J' be the $\mathcal{EL}$-concept terms respectively obtained from L, R, J by replacing every occurrence of C by $\widehat{D}$. We prove that $L' = \gamma'(X_1 \sqcap \ldots \sqcap X_m)$ and $R' = \gamma'(Y_1 \sqcap \ldots \sqcap Y_n)$ both reduce to J', which shows that γ' solves this equation. It is enough to show that the reductions are invariant under the replacement of C by $\widehat{D}$. Obviously, all the interesting reductions are of the form $E_1 \sqcap E_2 \to E_1$ where E_1, E_2 are existential restrictions such that $E_1 \sqsubseteq E_2$. Since γ is reduced, we can assume that E_1, E_2 are reduced. Let E'_1, E'_2 be respectively obtained from E_1, E_2 by replacing every occurrence of C by $\widehat{D}$. We must show that $E'_1 \sqcap E'_2$ reduces to E'_1. For this, it is enough to show that $E'_1 \sqsubseteq E'_2$. Assume that an occurrence of C in E_1 is actually needed to have the subsumption $E_1 \sqsubseteq E_2$. Then there is an existential restriction C' in E_2 such that $C \sqsubseteq C'$. If $C = C'$, then both are replaced by $\widehat{D}$, and thus this replacement is harmless. Otherwise, $C \sqsubset C'$. Since C' is an atom of γ, maximality of C yields that there is a non-variable atom D' of Γ such that $C' \equiv \gamma(D')$. Now $C \sqsubset C' \equiv \gamma(D')$ implies that there is an $i, 1 \leq i \leq n$, such that $D' = D_i$. Thus, C' is actually one of the conjuncts of $\widehat{D}$, which again shows that replacing C by $\widehat{D}$ is harmless. Thus, we have shown that $E'_1 \sqsubseteq E'_2$, which completes the proof of the lemma. □

The next proposition is an easy consequence of this lemma.

Proposition 3. *Let Γ be a flat $\mathcal{EL}$-unification problem and γ a minimal reduced ground unifier of Γ. If X is a concept variable occurring in Γ, then $\gamma(X) \equiv \top$ or there are non-variable atoms $D_1, \ldots, D_n$ ($n \geq 1$) of Γ such that $\gamma(X) \equiv \gamma(D_1) \sqcap \ldots \sqcap \gamma(D_n)$.*

Proof. If $\gamma(X) \not\equiv \top$, then it is a non-empty conjunction of atoms, i.e., there are atoms $C_1, \ldots, C_n$ ($n \geq 1$) such that $\gamma(X) = C_1 \sqcap \ldots \sqcap C_n$. Then $C_1, \ldots, C_n$ are atoms of γ, and thus Lemma 3 yields non-variable atoms $D_1, \ldots, D_n$ of Γ such that $C_i \equiv \gamma(D_i)$ for $i = 1, \ldots n$. Consequently, $\gamma(X) \equiv \gamma(D_1) \sqcap \ldots \sqcap \gamma(D_n)$. □

This proposition suggests the following *non-deterministic algorithm for deciding solvability of a given flat $\mathcal{EL}$-unification problem Γ:*

1. For every variable X occurring in Γ, guess a finite, possibly empty, set S_X of non-variable atoms of Γ.
2. We say that the variable X *directly depends on* the variable Y if Y occurs in an atom of S_X. Let *depends on* be the transitive closure of *directly depends on*. If there is a variable that depends on itself, then the algorithm returns "fail." Otherwise, there exists a strict linear order $>$ on the variables occurring in Γ such that $X > Y$ if X depends on Y.
3. We define the substitution σ along the linear order $>$:
   - If X is the least variable w.r.t. $>$, then S_X does not contain any variables. We define $\sigma(X)$ to be the conjunction of the elements of S_X, where the empty conjunction is $\top$.
   - Assume that $\sigma(Y)$ is defined for all variables $Y < X$. Then S_X only contains variables Y for which $\sigma(Y)$ is already defined. If S_X is empty, then we define $\sigma(X) := \top$. Otherwise, let $S_X = \{D_1, \ldots, D_n\}$. We define $\sigma(X) := \sigma(D_1) \sqcap \ldots \sqcap \sigma(D_n)$.
4. Test whether the substitution σ computed in the previous step is a unifier of Γ. If this is the case, then return σ; otherwise, return "fail."

This algorithm is trivially *sound* since it only returns substitutions that are unifiers of Γ. In addition, it obviously always terminates. Thus, to show correctness of our algorithm, it is sufficient to show that it is complete.

Lemma 4 (completeness). *If Γ is solvable, then there is a way of guessing in Step 1 subsets S_X of the non-variable atoms of Γ such that the* depends on *relation determined in Step 2 is acyclic and the substitution σ computed in Step 3 is a unifier of Γ.*

Proof. If Γ is solvable, then it has a minimal reduced ground unifier γ. By Proposition 3, for every variable X occurring in Γ we have $\gamma(X) \equiv \top$ or there are non-variable atoms $D_1, \ldots, D_n$ $(n \geq 1)$ of Γ such that $\gamma(X) \equiv \gamma(D_1) \sqcap \ldots \sqcap \gamma(D_n)$. If $\gamma(X) \equiv \top$, then we define $S_X := \emptyset$. Otherwise, we define $S_X := \{D_1, \ldots, D_n\}$.

We show that the relation *depends on* induced by these sets S_X is acyclic, i.e., there is no variable X such that X depends on itself. If X directly depends on Y, then Y occurs in an element of S_X. Since S_X consists of non-variable atoms of the flat unification problem Γ, this means that there is a role name r such that $\exists r.Y \in S_X$. Consequently, we have $\gamma(X) \sqsubseteq \exists r.\gamma(Y)$. Thus, if X depends on X, then there are $k \geq 1$ role names $r_1, \ldots, r_k$ such that $\gamma(X) \sqsubseteq \exists r_1. \cdots \exists r_k.\gamma(X)$. This is clearly not possible since $\gamma(X)$ cannot be subsumed by an $\mathcal{EL}$-concept term whose role depth is larger than the role depth of $\gamma(X)$.

To show that the substitution σ induced by the sets S_X is a unifier of Γ, we prove that σ is equivalent to γ, i.e., $\sigma(X) \equiv \gamma(X)$ holds for all variables X occurring in Γ. The substitution σ is defined along the linear order $>$. If X is the least variable w.r.t. $>$, then S_X does not contain any variables. If S_X is empty, then $\sigma(X) = \top \equiv \gamma(X)$. Otherwise, let $S_X = \{D_1, \ldots, D_n\}$. Since the atoms D_i do not contain variables, we have $D_i = \gamma(D_i)$. Thus, the definitions of S_X and of σ yield $\sigma(X) = D_1 \sqcap \ldots \sqcap D_n = \gamma(D_1) \sqcap \ldots \sqcap \gamma(D_n) \equiv \gamma(X)$.

Assume that $\sigma(Y) \equiv \gamma(Y)$ holds for all variables $Y < X$. If $S_X = \emptyset$, then we have again $\sigma(X) = \top \equiv \gamma(X)$. Otherwise, let $S_X = \{D_1, \ldots, D_n\}$. Since the atoms D_i contain only variables that are smaller than X, we have $\sigma(D_i) \equiv \gamma(D_i)$ by induction. Thus, the definitions of S_X and of σ yield $\sigma(X) = \sigma(D_1) \sqcap \ldots \sqcap \sigma(D_n) \equiv \gamma(D_1) \sqcap \ldots \sqcap \gamma(D_n) \equiv \gamma(X)$. □

Note that our proof of completeness actually shows that, up to equivalence, the algorithm returns all minimal reduced ground unifiers of Γ.

Theorem 3. *$\mathcal{EL}$-unification is NP-complete.*

Proof. NP-hardness follows from the fact that $\mathcal{EL}$-matching is NP-complete [17]. To show that the problem can be decided by a non-deterministic polynomial-time algorithm, we analyse the complexity of our algorithm. Obviously, guessing the sets S_X (Step 1) can be done within NP. Computing the *depends on* relation and checking it for acyclicity (Step 2) is clearly polynomial.

Steps 3 and 4 are more problematic. In fact, since a variable may occur in different atoms of Γ, the substitution σ computed in Step 3 may be of exponential size. This is actually the same reason that makes a naive algorithm for syntactic unification compute an exponentially large most general unifier [12]. As in the case of syntactic unification, the solution to this problem is basically structure sharing. Instead of computing the substitution σ explicitly, we view its definition as an acyclic TBox. To be more precise, for every concept variable X occurring in Γ, the TBox $\mathcal{T}_\sigma$ contains the concept definition $X \doteq \top$ if $S_X = \emptyset$ and $X \doteq D_1 \sqcap \ldots \sqcap D_n$ if $S_X = \{D_1, \ldots, D_n\}$ $(n \geq 1)$. Instead of computing σ in Step 3, we compute $\mathcal{T}_\sigma$. Because of the acyclicity test in Step 2, we know that $\mathcal{T}_\sigma$ is an acyclic TBox. The size of $\mathcal{T}_\sigma$ is obviously polynomial in the size of Γ, and thus this modified Step 3 is polynomial. It is easy to see that applying the substitution σ is the same as expanding the concept terms C, D w.r.t. the TBox $\mathcal{T}_\sigma$. This implies that, for every equation $C \equiv^? D$ in Γ, we have $C \equiv_{\mathcal{T}_\sigma} D$ iff $\sigma(C) \equiv \sigma(D)$. Thus, testing whether σ is a unifier of Γ can be reduced to testing whether $C \equiv_{\mathcal{T}_\sigma} D$ holds for every equation $C \equiv^? D$ in Γ. Since subsumption (and thus equivalence) in $\mathcal{EL}$ w.r.t. acyclic TBoxes can be decided in polynomial time [3],[4] this completes the proof of the theorem. □

6 Unification in Semilattices with Monotone Operators

Unification problems and their types were originally not introduced for Description Logics, but for equational theories [12]. In this section, we show that the above results for unification in $\mathcal{EL}$ can actually be viewed as results for an equational theory. As shown in [19], the equivalence problem for $\mathcal{EL}$-concept terms corresponds to the word problem for the equational theory of semilattices with monotone operators. In order to define this theory, we consider a signature Σ_{SLmO} consisting of a binary function symbol $\wedge$, a constant symbol 1, and finitely many unary function symbols $f_1, \ldots, f_n$. Terms can then be built using these symbols and additional variable symbols and free constant symbols.

[4] Of course, the polynomial-time subsumption algorithm does not expand the TBox.

Definition 4. *The equational theory of* semilattices with monotone operators *is defined by the following identities:*

$$SLmO := \{x \wedge (y \wedge z) = (x \wedge y) \wedge z,\ x \wedge y = y \wedge x,\ x \wedge x = x,\ x \wedge 1 = x\} \cup$$
$$\{f_i(x \wedge y) \wedge f_i(y) = f_i(x \wedge y) \mid 1 \leq i \leq n\}$$

A given $\mathcal{EL}$-concept term C using only roles $r_1, \ldots, r_n$ can be translated into a term t_C over the signature Σ_{SLmO} by replacing each concept constant A by a corresponding free constants a, each concept variable X by a corresponding variable x, $\top$ by 1, $\sqcap$ by $\wedge$, and $\exists r_i$ by f_i. For example, the $\mathcal{EL}$-concept term $C = A \sqcap \exists r_1.\top \sqcap \exists r_3.(X \sqcap B)$ is translated into $t_C = a \wedge f_1(1) \wedge f_3(x \wedge b)$. Conversely, any term over the signature Σ_{SLmO} can be translated back into an $\mathcal{EL}$-concept term.

Lemma 5. *Let C, D be $\mathcal{EL}$-concept term using only roles $r_1, \ldots, r_n$. Then $C \equiv D$ iff $t_C =_{SLmO} t_D$.*

As an immediate consequence of this lemma, we have that unification in the DL $\mathcal{EL}$ corresponds to unification modulo the equational theory *SLmO*. Thus, Theorem 2 implies that *SLmO* has unification type zero, and Theorem 3 implies that *SLmO*-unification is NP-complete.

Corollary 2. *The equational theory SLmO of semilattices with monotone operators has unification type zero, and deciding solvability of an SLmO-unification problem is an NP-complete problem.*

7 Conclusion

In this paper, we have shown that unification in the DL $\mathcal{EL}$ is of type zero and NP-complete. There are interesting differences between the behaviour of $\mathcal{EL}$ and the closely related DL $\mathcal{FL}_0$ w.r.t. unification and matching. Though the unification types coincide for these two DLs, the complexities of the decision problems differ: $\mathcal{FL}_0$-unification is ExpTime-complete, and thus considerably harder than $\mathcal{EL}$-unification. In contrast, $\mathcal{FL}_0$-matching is polynomial, and thus considerably easier than $\mathcal{EL}$-matching, which is NP-complete.

It is well-known that there is a close connection between modal logics and DLs [5]. For example, the DL $\mathcal{ALC}$, which can be obtained by adding negation to $\mathcal{EL}$ or $\mathcal{FL}_0$, corresponds to the basic (multi-)modal logic K. Decidability of unification in K is a long-standing open problem. Recently, undecidability of unification in some extensions of K (for example, by the universal modality) was shown in [20]. The undecidability results in [20] also imply undecidability of unification in some expressive DLs (e.g., $\mathcal{SHIQ}$). The unification types of some modal (and related) logics have been determined by Ghilardi; for example in [14] he shows that $\mathsf{K4}$ and $\mathsf{S4}$ have unification type finitary. Unification in sub-Boolean modal logics (i.e., modal logics that are not closed under all Boolean operations, such as the modal logic equivalent of $\mathcal{EL}$) has, to the best of our knowledge, not been considered in the modal logic literature.

References

1. Baader, F.: Unification in commutative theories. J. of Symbolic Computation 8(5) (1989)
2. Baader, F.: Terminological cycles in KL-ONE-based knowledge representation languages. In: Proc. AAAI 1990 (1990)
3. Baader, F.: Terminological cycles in a description logic with existential restrictions. In: Proc. IJCAI 2003 (2003)
4. Baader, F., Brandt, S., Lutz, C.: Pushing the $\mathcal{EL}$ envelope. In: Proc. IJCAI 2005 (2005)
5. Baader, F., Calvanese, D., McGuinness, D., Nardi, D., Patel-Schneider, P.F. (eds.): The Description Logic Handbook: Theory, Implementation, and Applications. Cambridge University Press, Cambridge (2003)
6. Baader, F., Küsters, R.: Matching in description logics with existential restrictions. In: Proc. KR 2000 (2000)
7. Baader, F., Küsters, R.: Unification in a description logic with transitive closure of roles. In: Nieuwenhuis, R., Voronkov, A. (eds.) LPAR 2001. LNCS (LNAI), vol. 2250, p. 217. Springer, Heidelberg (2001)
8. Baader, F., Küsters, R., Borgida, A., McGuinness, D.L.: Matching in description logics. J. of Logic and Computation 9(3) (1999)
9. Baader, F., Narendran, P.: Unification of concepts terms in description logics. J. of Symbolic Computation 31(3) (2001)
10. Baader, F., Nutt, W.: Basic description logics. In: [5] (2003)
11. Baader, F., Sertkaya, B., Turhan, A.-Y.: Computing the least common subsumer w.r.t. a background terminology. J. of Applied Logic 5(3) (2007)
12. Baader, F., Snyder, W.: Unification theory. In: Handbook of Automated Reasoning, vol. I. Elsevier Science Publishers, Amsterdam (2001)
13. Brandt, S.: Polynomial time reasoning in a description logic with existential restrictions, GCI axioms, and—what else. In: Proc. ECAI 2004 (2004)
14. Ghilardi, S.: Best solving modal equations. Ann. Pure Appl. Logic 102(3) (2000)
15. Horrocks, I., Patel-Schneider, P.F., van Harmelen, F.: From SHIQ and RDF to OWL: The making of a web ontology language. Journal of Web Semantics 1(1) (2003)
16. Kazakov, Y., de Nivelle, H.: Subsumption of concepts in $\mathcal{FL}_0$ for (cyclic) terminologies with respect to descriptive semantics is PSPACE-complete. In: Proc. DL 2003. CEUR Electronic Workshop Proceedings (2003), `http://CEUR-WS.org/Vol-81/`
17. Küsters, R.: Non-Standard Inferences in Description Logics. LNCS (LNAI), vol. 2100. Springer, Heidelberg (2001)
18. Rector, A., Horrocks, I.: Experience building a large, re-usable medical ontology using a description logic with transitivity and concept inclusions. In: Proc. AAAI 1997 (1997)
19. Sofronie-Stokkermans, V.: Locality and subsumption testing in $\mathcal{EL}$ and some of its extensions. In: Proc. AiML 2008 (2008)
20. Wolter, F., Zakharyaschev, M.: Undecidability of the unification and admissibility problems for modal and description logics. ACM Trans. Comput. Log. 9(4) (2008)

SAT Encoding of Unification in $\mathcal{EL}$

Franz Baader and Barbara Morawska*

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

Abstract. Unification in Description Logics has been proposed as a novel inference service that can, for example, be used to detect redundancies in ontologies. In a recent paper, we have shown that unification in $\mathcal{EL}$ is NP-complete, and thus of a complexity that is considerably lower than in other Description Logics of comparably restricted expressive power. In this paper, we introduce a new NP-algorithm for solving unification problems in $\mathcal{EL}$, which is based on a reduction to satisfiability in propositional logic (SAT). The advantage of this new algorithm is, on the one hand, that it allows us to employ highly optimized state-of-the-art SAT solvers when implementing an $\mathcal{EL}$-unification algorithm. On the other hand, this reduction provides us with a proof of the fact that $\mathcal{EL}$-unification is in NP that is much simpler than the one given in our previous paper on $\mathcal{EL}$-unification.

1 Introduction

Description logics (DLs) [3] are a well-investigated family of logic-based knowledge representation formalisms. They can be used to represent the relevant concepts of an application domain using concept terms, which are built from concept names and role names using certain concept constructors. The DL $\mathcal{EL}$ offers the constructors conjunction ($\sqcap$), existential restriction ($\exists r.C$), and the top concept ($\top$). This description logic has recently drawn considerable attention since, on the one hand, important inference problems such as the subsumption problem are polynomial in $\mathcal{EL}$ [1,2]. On the other hand, though quite inexpressive, $\mathcal{EL}$ can be used to define biomedical ontologies. For example, both the large medical ontology SNOMED CT and the Gene Ontology[1] can be expressed in $\mathcal{EL}$.

Unification in description logics has been proposed in [6] as a novel inference service that can, for example, be used to detect redundancies in ontologies. There, it was shown that, for the DL $\mathcal{FL}_0$, which differs from $\mathcal{EL}$ by offering value restrictions ($\forall r.C$) in place of existential restrictions, deciding unifiability is an ExpTime-complete problem. In [4], we were able to show that unification in $\mathcal{EL}$ is of considerably lower complexity: the decision problem is "only" NP-complete. However, the unification algorithm introduced in [4] to establish the NP upper bound is a brutal "guess and then test" NP-algorithm, and thus it is unlikely that a direct implementation of it will perform well in practice.

* Supported by DFG under grant BA 1122/14-1

[1] See http://www.ihtsdo.org/snomed-ct/ and http://www.geneontology.org/

C. Fermüller and A. Voronkov (Eds.): LPAR-17, LNCS 6397, pp. 97–111, 2010.

In this report, we present a new decision procedure for $\mathcal{EL}$-unification that takes a given $\mathcal{EL}$-unification problem Γ and translates it into a set of propositional clauses $C(\Gamma)$ such that (i) the size of $C(\Gamma)$ is polynomial in the size of Γ, and (ii) Γ is unifiable iff $C(\Gamma)$ is satisfiable. This allows us to use a highly-optimized SAT-solver such as MiniSat[2] to decide solvability of $\mathcal{EL}$-unification problems. Our SAT-translation is inspired by Kapur and Narendran's translation of ACIU-unification problems into satisfiability in propositional Horn logic (HornSAT) [9]. The connection between $\mathcal{EL}$-unification and ACIU-unification is due to the fact that (modulo equivalence) the conjunction constructor in $\mathcal{EL}$ is associative, commutative, and idempotent, and has the top concept $\top$ as a unit. Existential restrictions are similar to free unary functions symbols in ACIU, with the difference that existential restrictions are monotonic w.r.t. subsumption.

It should be noted that the proof of correctness of our translation into SAT does *not* depend on the results in [4]. Consequently, this translation provides us with a new proof of the fact that $\mathcal{EL}$-unification is in NP. This proof is much simpler than the original proof of this fact in [4].

2 Unification in $\mathcal{EL}$

Starting with a set N_{con} of concept names and a set N_{role} of role names, $\mathcal{EL}$-*concept terms* are built using the following concept constructors: the nullary constructor *top-concept* ($\top$), the binary constructor *conjunction* ($C \sqcap D$), and for every role name $r \in N_{role}$, the unary constructor *existential restriction* ($\exists r.C$). The semantics of $\mathcal{EL}$ is defined in the usual way, using the notion of an interpretation $\mathcal{I} = (\mathcal{D}_\mathcal{I}, \cdot^\mathcal{I})$, which consists of a nonempty domain $\mathcal{D}_\mathcal{I}$ and an interpretation function $\cdot^\mathcal{I}$ that assigns binary relations on $\mathcal{D}_\mathcal{I}$ to role names and subsets of $\mathcal{D}_\mathcal{I}$ to concept terms, as shown in the semantics column of Table 1.

Table 1. Syntax and semantics of $\mathcal{EL}$

Name	Syntax	Semantics
concept name	A	$A^\mathcal{I} \subseteq \mathcal{D}_\mathcal{I}$
role name	r	$r^\mathcal{I} \subseteq \mathcal{D}_\mathcal{I} \times \mathcal{D}_\mathcal{I}$
top-concept	$\top$	$\top^\mathcal{I} = \mathcal{D}_\mathcal{I}$
conjunction	$C \sqcap D$	$(C \sqcap D)^\mathcal{I} = C^\mathcal{I} \cap D^\mathcal{I}$
existential restriction	$\exists r.C$	$(\exists r.C)^\mathcal{I} = \{x \mid \exists y : (x,y) \in r^\mathcal{I} \wedge y \in C^\mathcal{I}\}$
subsumption	$C \sqsubseteq D$	$C^\mathcal{I} \subseteq D^\mathcal{I}$
equivalence	$C \equiv D$	$C^\mathcal{I} = D^\mathcal{I}$

The concept term C *is subsumed by* the concept term D (written $C \sqsubseteq D$) iff $C^\mathcal{I} \subseteq D^\mathcal{I}$ holds for all interpretations $\mathcal{I}$. We say that C *is equivalent to* D (written $C \equiv D$) iff $C \sqsubseteq D$ and $D \sqsubseteq C$, i.e., iff $C^\mathcal{I} = D^\mathcal{I}$ holds for all interpretations $\mathcal{I}$.

[2] http://minisat.se/

The following lemma provides us with a useful *characterization of subsumption in* $\mathcal{EL}$ [4].

Lemma 1. *Let* C, D *be* $\mathcal{EL}$*-concept terms such that*

$$C = A_1 \sqcap \ldots \sqcap A_k \sqcap \exists r_1.C_1 \sqcap \ldots \sqcap \exists r_m.C_m,$$
$$D = B_1 \sqcap \ldots \sqcap B_\ell \sqcap \exists s_1.D_1 \sqcap \ldots \sqcap \exists s_n.D_n,$$

where $A_1, \ldots, A_k, B_1, \ldots, B_\ell$ *are concept names. Then* $C \sqsubseteq D$ *iff*

- $\{B_1, \ldots, B_\ell\} \subseteq \{A_1, \ldots, A_k\}$ *and*
- *for every* $j, 1 \leq j \leq n$*, there exists* $i, 1 \leq i \leq m$*, s.t.* $r_i = s_j$ *and* $C_i \sqsubseteq D_j$*.*

When defining unification in $\mathcal{EL}$, we assume that the set of concepts names is partitioned into a set N_v of concept variables (which may be replaced by substitutions) and a set N_c of concept constants (which must not be replaced by substitutions). A *substitution* σ is a mapping from N_v into the set of all $\mathcal{EL}$-concept terms. This mapping is extended to concept terms in the usual way, i.e., by replacing all occurrences of variables in the term by their σ-images.

A substitution σ induces the following binary relation $>_\sigma$ on variables:

$$X >_\sigma Y \text{ iff there are } n \geq 1 \text{ role names } r_1, \ldots, r_n \in N_{role} \text{ such that}$$
$$\sigma(X) \sqsubseteq \sigma(\exists r_1. \cdots \exists r_n.Y).$$

The following lemma is an easy consequence of Lemma 1.

Lemma 2. *The relation* $>_\sigma$ *is a strict partial order.*

Unification tries to make concept terms equivalent by applying a substitution.

Definition 1. *An* $\mathcal{EL}$-unification problem *is of the form* $\Gamma = \{C_1 \equiv^? D_1, \ldots, C_n \equiv^? D_n\}$*, where* $C_1, D_1, \ldots C_n, D_n$ *are* $\mathcal{EL}$*-concept terms. The substitution* σ *is a* unifier *(or* solution*) of* Γ *iff* $\sigma(C_i) \equiv \sigma(D_i)$ *for* $i = 1, \ldots, n$*. In this case,* Γ *is called* solvable *or* unifiable*.*

Note that Lemma 2 implies that the variable X cannot unify with the concept term $\exists r_1. \cdots \exists r_n.X$ $(n \geq 1)$, i.e., the $\mathcal{EL}$-unification problem $\{X \equiv^? \exists r_1. \cdots \exists r_n.X\}$ does not have a solution. This means that an $\mathcal{EL}$-unification algorithm has to realize a kind of *occurs check*.

We will assume without loss of generality that our $\mathcal{EL}$-unification problems are flattened in the sense that they do not contain nested existential restrictions. To define this notion in more detail, we need to introduce the notion of an atom. An $\mathcal{EL}$-concept term is called an *atom* iff it is a concept name (i.e., concept constant or concept variable) or an existential restriction $\exists r.D$. A *non-variable atom* is an atom that is not a concept variable. The set of *atoms of an* $\mathcal{EL}$*-concept term* C consists of all the subterms of C that are atoms. For example, $A \sqcap \exists r.(B \sqcap \exists r.\top)$ has the atom set $\{A, \exists r.(B \sqcap \exists r.\top), B, \exists r.\top\}$.

Obviously, any $\mathcal{EL}$-concept term is (equivalent to) a conjunction of atoms, where the empty conjunction is $\top$. The following lemma is an easy consequence of Lemma 1.

Lemma 3. *Let C, D be $\mathcal{EL}$-concept terms such that $C = C_1 \sqcap \ldots \sqcap C_m$ and $D = D_1 \sqcap \ldots \sqcap D_n$, where $D_1, \ldots, D_n$ are atoms. Then $C \sqsubseteq D$ iff for every $j, 1 \leq j \leq n$, there exists an $i, 1 \leq i \leq m$, such that $C_i \sqsubseteq D_j$.*

In our reduction, we will restrict the attention (without loss of generality) to unification problems that are built from atoms without nested existential restrictions. To be more precise, concept names and existential restrictions $\exists r.D$ where D is a concept name are called *flat atoms*. An $\mathcal{EL}$-concept term is *flat* iff it is a conjunction of flat atoms (where the empty conjunction is $\top$). The $\mathcal{EL}$-unification problem Γ is *flat* iff it consists of equations between flat $\mathcal{EL}$-concept terms. By introducing new concept variables and eliminating $\top$, any $\mathcal{EL}$-unification problem Γ can be transformed in polynomial time into a flat $\mathcal{EL}$-unification problem Γ' such that Γ is solvable iff Γ' is solvable. Thus, we may assume without loss of generality that our input $\mathcal{EL}$-unification problems are flat. Given a flat $\mathcal{EL}$-unification problem $\Gamma = \{C_1 \equiv^? D_1, \ldots, C_n \equiv^? D_n\}$, we call the atoms of $C_1, D_1, \ldots, C_n, D_n$ the *atoms of* Γ.

3 The SAT Encoding

In the following, let Γ be a flat $\mathcal{EL}$-unification problem. We show how to translate Γ into a set of propositional clauses $C(\Gamma)$ such that (i) the size of $C(\Gamma)$ is polynomial in the size of Γ, and (ii) Γ is unifiable iff $C(\Gamma)$ is satisfiable. The main idea underlying this translation is that we want to guess, for every pair of atoms A, B of the flat unification problem Γ, whether or not A is subsumed by B after the application of the unifier σ to be computed. In addition, we need to guess a strict partial order $>$ on the variables of Γ, which corresponds to (a subset of) the strict partial order $>_\sigma$ induced by σ.

Thus, we use the following propositional variables:

- $[A \not\sqsubseteq B]$ for every pair A, B of atoms of Γ;
- $[X > Y]$ for every pair of variables occurring in Γ.

Note that we use non-subsumption rather than subsumption for the propositional variables of the first kind since this will allow us to translate the equations of the unification problem into Horn clauses (*à la* Kapur and Narendran [9]). However, we will have to "pay" for this since expressing transitivity of subsumption then requires the use of non-Horn clauses.

Given a flat $\mathcal{EL}$-unification problem Γ, the set $C(\Gamma)$ consists of the following clauses:

(1) Translation of the equations of Γ. For every equation $A_1 \sqcap \cdots \sqcap A_m \equiv^? B_1 \sqcap \cdots \sqcap B_n$ of Γ, we create the following Horn clauses, which express that any atom that occurs as a top-level conjunct on one side of an equivalence must subsume a top-level conjunct on the other side:[3]

[3] See Lemma 3.

1. For every non-variable atom $C \in \{A_1, \ldots, A_m\}$:
   $[B_1 \not\sqsubseteq C] \wedge \ldots \wedge [B_n \not\sqsubseteq C] \rightarrow$
2. For every non-variable atom $C \in \{B_1, \ldots, B_n\}$:
   $[A_1 \not\sqsubseteq C] \wedge \ldots \wedge [A_m \not\sqsubseteq C] \rightarrow$
3. For every non-variable atom C of Γ s.t. $C \notin \{A_1, \ldots, A_m, B_1, \ldots, B_n\}$:
   $[A_1 \not\sqsubseteq C] \wedge \ldots \wedge [A_m \not\sqsubseteq C] \rightarrow [B_j \not\sqsubseteq C]$ for $j = 1, \ldots, n$
   $[B_1 \not\sqsubseteq C] \wedge \ldots \wedge [B_n \not\sqsubseteq C] \rightarrow [A_i \not\sqsubseteq C]$ for $i = 1, \ldots, m$

(2) Translation of the relevant properties of subsumption in $\mathcal{EL}$.

1. For every pair of distinct concept constants A, B occurring in Γ, we say that A cannot be subsumed by B:
   $\rightarrow [A \not\sqsubseteq B]$
2. For every pair of distinct role names r, s and atoms $\exists r.A, \exists s.B$ of Γ, we say that $\exists r.A$ cannot be subsumed by $\exists s.B$:
   $\rightarrow [\exists r.A \not\sqsubseteq \exists s.B]$
3. For every pair $\exists r.A, \exists r.B$ of atoms of Γ, we say that $\exists r.A$ can only be subsumed by $\exists r.B$ if A is already subsumed by B:
   $[A \not\sqsubseteq B] \rightarrow [\exists r.A \not\sqsubseteq \exists r.B]$
4. For every concept constant A and every atom $\exists r.B$ of Γ, we say that A and $\exists r.B$ are not in a subsumption relationship
   $\rightarrow [A \not\sqsubseteq \exists r.B]$ and $\rightarrow [\exists r.B \not\sqsubseteq A]$
5. Transitivity of subsumption is expressed using the *non-Horn* clauses:
   $[C_1 \not\sqsubseteq C_3] \rightarrow [C_1 \not\sqsubseteq C_2] \vee [C_2 \not\sqsubseteq C_3]$ where C_1, C_2, C_3 are atoms of Γ.

Note that there are further properties that hold for subsumption in $\mathcal{EL}$ (e.g., the fact that $A \sqsubseteq B$ implies $\exists r.A \sqsubseteq \exists r.B$), but that are not needed to ensure soundness of our translation.

(3) Translation of the relevant properties of $>$.

1. Transitivity and irreflexivity of $>$ can be expressed using the Horn clauses:
   $[X{>}X] \rightarrow$ and $[X{>}Y] \wedge [Y{>}Z] \rightarrow [X{>}Z]$,
   where X, Y, Z are concept variables occurring in Γ.
2. The connection between this order and the order $>_\sigma$ is expressed using the *non-Horn* clauses:
   $\rightarrow [X{>}Y] \vee [X \not\sqsubseteq \exists r.Y]$,
   where X, Y are concept variables occurring in Γ and $\exists r.Y$ is an atom of Γ.

Since the number of atoms of Γ is linear in the size of Γ, it is easy to see that $C(\Gamma)$ is of size polynomial in the size of Γ, and that it can be computed in polynomial time. Note, however, that without additional optimizations, the polynomial can be quite big. If the size of Γ is n, then the number of atoms of Γ is in $O(n)$. The number of possible propositional variables is thus in $O(n^2)$. The size of $C(\Gamma)$ is dominated by the number of clauses expressing the transitivity of subsumption and the transitivity of the order on variables. Thus, the size of $C(\Gamma)$ is in $O((n^2)^3) = O(n^6)$.

Example 1. It is easy to see that the $\mathcal{EL}$-unification problem $\Gamma := \{X \sqcap \exists r.X \equiv^? X\}$ does not have a solution. The set of clauses $C(\Gamma)$ has the following elements:

(1) The only clause created in (1) is: $[X \not\sqsubseteq \exists r.X] \rightarrow$.
(2) Among the clauses introduced in (2) is the following:
    5. $[\exists r.X \not\sqsubseteq \exists r.X] \rightarrow [\exists r.X \not\sqsubseteq X] \vee [X \not\sqsubseteq \exists r.X]$
(3) The following clauses are created in (3):
    1. $[X > X] \rightarrow$
    2. $\rightarrow [X > X] \vee [X \not\sqsubseteq \exists r.X]$.

This set of clauses is unsatisfiable. In fact, $[X \not\sqsubseteq \exists r.X]$ needs to be assigned the truth value 0 because of (1). Consequently, (3)2. implies that $[X > X]$ needs to be assigned the truth value 1, which then falsifies (3)1.

The next example considers an equation where the right-hand side is the top concept, which is the empty conjunction of flat atoms.

Example 2. The $\mathcal{EL}$-unification problem $\Gamma := \{A \sqcap B \equiv^? \top\}$ has no solution.

In (1)1. we need to construct clauses for the atoms A and B on the left-hand side. Since the right-hand side of the equation is the empty conjunction (i.e., $n = 0$), the left-hand sides of the implications generated this way are empty, i.e., both atoms yield the implication $\rightarrow$, in which both the left-hand side and the right-hand side is empty. An empty left-hand side is read as true (1), whereas an empty right-hand side is read as false (0). Thus, this implication is unsatisfiable.

Theorem 1 (Soundness and completeness). *Let Γ be a flat $\mathcal{EL}$-unification problem. Then, Γ is solvable iff $C(\Gamma)$ is satisfiable.*

We prove this theorem in the next two subsections, one devoted to the proof of soundness and the other to the proof of completeness. After the formal proof, we will also explain the reduction on a more intuitive level. Since our translation into SAT is polynomial and SAT is in NP, Theorem 1 shows that $\mathcal{EL}$-unification is in NP. NP-hardness follows from the fact that $\mathcal{EL}$-matching is known to be NP-hard [10]: in fact, matching problems are special unification problems where the terms on the right-hand sides of the equations do not contain variables.

Corollary 1. *$\mathcal{EL}$-unification is NP-complete.*

3.1 Soundness

To prove soundness, we assume that $C(\Gamma)$ is satisfiable. We must show that this implies that Γ is solvable. In order to define a unifier of Γ, we take a propositional valuation τ that satisfies $C(\Gamma)$, and use τ to define an *assignment* of sets S_X of non-variable atoms of Γ to the variables X of Γ:

$$S_X := \{C \mid C \text{ non-variable atom of } \Gamma \text{ s.t. } \tau([X \not\sqsubseteq C]) = 0\}.$$

Given this assignment of sets of non-variable atoms to the variables in Γ, we say that the variable X *directly depends on* the variable Y if Y occurs in an atom of S_X. Let *depends on* be the transitive closure of *directly depends on*. We define the binary relation $>_d$ on variables as $X >_d Y$ iff X depends on Y.

Lemma 4. *Let X, Y be variables occurring in Γ.*

1. *If $X >_d Y$, then $\tau([X{>}Y]) = 1$.*
2. *The relation $>_d$ is irreflexive, i.e., $X \not>_d X$.*

Proof. (1) If X *directly depends on* the variable Y, then Y appears in a non-variable atom of S_X. This atom must be of the form $\exists r.Y$. By the construction of S_X, $\exists r.Y \in S_X$ can only be the case if $\tau([X \not\sqsubseteq \exists r.Y]) = 0$. Since $C(\Gamma)$ contains the clause $\rightarrow [X{>}Y] \vee [X \not\sqsubseteq \exists r.Y]$, this implies $\tau([X{>}Y]) = 1$.

Since the transitivity clauses introduced in (3)1. are satisfied by τ, we also have that $\tau([X{>}Y]) = 1$ whenever X *depends on* the variable Y.

(2) If X depends on itself, then $\tau([X{>}X]) = 1$ by the first part of this lemma. This is, however, impossible since τ satisfies the clause $[X{>}X] \rightarrow$. □

The second part of this lemma shows that the relation $>_d$, which is transitive by definition, is a strict partial order. We can now use the sets S_X to define a substitution σ along the strict partial order $>_d$:[4]

- If X is a minimal variable w.r.t. $>_d$, then $\sigma(X)$ is the conjunction of the elements of S_X, where the empty conjunction is $\top$.
- Assume that $\sigma(Y)$ is already defined for all variables Y such that $X >_d Y$, and let $S_X = \{D_1, \ldots, D_n\}$. We define $\sigma(X) := \sigma(D_1) \sqcap \ldots \sqcap \sigma(D_n)$, where again the empty conjunction (in case $n = 0$) is $\top$.

Note that the substitution σ defined this way is actually a *ground* substitution, i.e., for all variables X occurring in Γ we have that $\sigma(X)$ does not contain variables. In the following, we will say that this substitution is *induced by* the valuation τ. Before we can show that σ is a unifier of Γ, we must first prove the following lemma.

Lemma 5. *Let C_1, C_2 be atoms of Γ. If $\tau([C_1 \not\sqsubseteq C_2]) = 0$, then $\sigma(C_1) \sqsubseteq \sigma(C_2)$.*

Proof. Assume that $\tau([C_1 \not\sqsubseteq C_2]) = 0$. First, consider the case where C_1 is a variable. If C_2 is not a variable, then (by the construction of σ) $\tau([C_1 \not\sqsubseteq C_2]) = 0$ implies that $\sigma(C_2)$ is a conjunct of $\sigma(C_1)$, and hence $\sigma(C_1) \sqsubseteq \sigma(C_2)$. If C_2 is a variable, then $\tau([C_1 \not\sqsubseteq C_2]) = 0$, together with the transitivity clauses of (2)5., implies that every conjunct of $\sigma(C_2)$ is also a conjunct of $\sigma(C_1)$, which again yields $\sigma(C_1) \sqsubseteq \sigma(C_2)$. Second, consider the case where $\sigma(C_2) = \top$. Then $\sigma(C_1) \sqsubseteq \sigma(C_2)$ obviously holds.

Hence, it remains to prove the lemma for the cases when C_1 is not a variable (i.e., it is a concept constant or an existential restriction) and $\sigma(C_2)$ is not $\top$. We use induction on the role depth of $\sigma(C_1) \sqcap \sigma(C_2)$, where the role depth of an $\mathcal{EL}$-concept term is the maximal nesting of existential restrictions in this term. To be more precise, if D_1, D_2, C_1, C_2 are atoms of Γ, then we define $(D_1, D_2) \succ (C_1, C_2)$ iff the role depth of $\sigma(D_1) \sqcap \sigma(D_2)$ is greater than the role depth of $\sigma(C_1) \sqcap \sigma(C_2)$.

[4] $>_d$ is well-founded since Γ contains only finitely many variables.

We prove the lemma by induction on $\succ$. The base case for this induction is the case where $\sigma(C_1)$ and $\sigma(C_2)$ have role depth 0, i.e., both are conjunctions of concept constants. Since C_1 is not a variable, this implies that C_1 is a concept constant. The atom C_2 is either a concept constant or a concept variable. We consider these two cases:

- Let C_2 be a concept constant (and thus $C_2 = \sigma(C_2)$). Since $\tau([C_1 \not\sqsubseteq C_2]) = 0$ and the clauses introduced in (2)1. of the translation to SAT are satisfied by τ, we have $C_2 = C_1$, and thus $\sigma(C_1) \sqsubseteq \sigma(C_2)$.
- Assume that C_2 is a variable. Since the role depth of $\sigma(C_2)$ is 0 and $\sigma(C_2)$ is not $\top$, $\sigma(C_2)$ is a non-empty conjunction of concept constants, i.e., $\sigma(C_2) = B_1 \sqcap \cdots \sqcap B_n$ for $n \geq 1$ constants $B_1, \ldots, B_n$ such that $\tau([C_2 \not\sqsubseteq B_i]) = 0$ for $i = \{1, \ldots, n\}$. Then, since τ satisfies the transitivity clauses introduced in (2)5. of the translation to SAT, $\tau([C_1 \not\sqsubseteq B_i]) = 0$ for $i = \{1, \ldots, n\}$. Since τ satisfies the clauses introduced in (2)1. of the translation to SAT, B_i must be identical to C_1 for $i = \{1, \ldots, n\}$. Hence, $\sigma(C_2) = B_1 \sqcap \cdots \sqcap B_n \equiv C_1 = \sigma(C_1)$, which implies $\sigma(C_1) \sqsubseteq \sigma(C_2)$.

Now we assume by induction that the statement of the lemma holds for all pairs of atoms D_1, D_2 such that $(C_1, C_2) \succ (D_1, D_2)$. Notice that, if C_1 is a constant, then $\sigma(C_2)$ cannot contain an atom of the form $\exists r.D$ as a top-level conjunct. In fact, this could only be the case if either C_2 is an existential restriction, or C_2 is a variable and S_{C_2} contains an existential restriction. In the first case, $\tau([C_1 \not\sqsubseteq C_2]) = 0$ would then imply that one of the clauses introduced in (2)4. is not satisfied by τ. In the second case, τ would either need to violate one of the transitivity clauses introduced in (2)5. or one of the clauses introduced in (2)4. Thus, $\sigma(C_2)$ cannot contain an atom of the form $\exists r.D$ as a top-level conjunct. This implies that $\sigma(C_1) \sqcap \sigma(C_2)$ has role depth 0, which actually means that we are in the base case. Therefore, we can assume that C_1 is not a constant.

Since C_1 is not a variable, we have only one case to consider: C_1 is of the form $C_1 = \exists r.C$. Then, because of the clauses in (2)4. and the transitivity clauses in (2)5., $\sigma(C_2)$ cannot contain a constant as a conjunct. If C_2 is an existential restriction $C_2 = \exists s.D$, then $\tau([C_1 \not\sqsubseteq C_2]) = 0$, together with the clauses in (2)2. yields $r = s$. Consequently, $\tau([C_1 \not\sqsubseteq C_2]) = 0$, together with the clauses in (2)3., yields $\tau([C \not\sqsubseteq D] = 0$. By induction, this implies $\sigma(C) \sqsubseteq \sigma(D)$, and thus $\sigma(C_1) = \exists r.\sigma(C) \sqsubseteq \exists r.\sigma(D) = \sigma(C_2)$.

If C_2 is a variable, then (by the construction of σ and the clauses in (2)4.) $\sigma(C_2)$ must be a conjunction of atoms of the form $\exists r_1.\sigma(D_1), \ldots, \exists r_n.\sigma(D_n)$, where $\tau([C_2 \not\sqsubseteq \exists r_i.D_i]) = 0$ for $i = 1, \ldots, n$. The transitivity clauses in (2)5. yield $\tau([\exists r.C \not\sqsubseteq \exists r_1.D_1]) = \ldots = \tau([\exists r.C \not\sqsubseteq \exists r_n.D_n]) = 0$, and the clauses in (2)2. yield $r_1 = \cdots = r_n = r$. Using the clauses in (2)3., we thus obtain $\tau([C \not\sqsubseteq D_1]) = \ldots = \tau([C \not\sqsubseteq D_n]) = 0$. Induction yields $\sigma(C) \sqsubseteq \sigma(D_1), \ldots, \sigma(C) \sqsubseteq \sigma(D_n)$, which in turn implies $\sigma(C_1) = \exists r.\sigma(C) \sqsubseteq \exists r_1.\sigma(D_1) \sqcap \cdots \sqcap \exists r_n.\sigma(D_n) = \sigma(C_2)$. □

Now we can easily prove the soundness of the translation.

Proposition 1 (Soundness). *The substitution σ induced by a satisfying valuation of $C(\Gamma)$ is a unifier of Γ.*

Proof. We have to show, for each equation $A_1 \sqcap \ldots \sqcap A_m \equiv^? B_1 \sqcap \ldots \sqcap B_n$ in Γ, that $\sigma(A_1) \sqcap \ldots \sqcap \sigma(A_m) \equiv \sigma(B_1) \sqcap \ldots \sqcap \sigma(B_n)$. Both sides of this equivalence are conjunctions of ground atoms, i.e., $\sigma(A_1) \sqcap \ldots \sqcap \sigma(A_m) = E_1 \sqcap \ldots \sqcap E_l$ and $\sigma(B_1) \sqcap \ldots \sqcap \sigma(B_n) = F_1 \sqcap \ldots \sqcap F_k$. By Lemma 3, we can prove that the equivalence holds by showing that, for each F_i, there is an A_j such that $\sigma(A_j) \sqsubseteq F_i$, and for each E_j, there is a B_i such that $\sigma(B_i) \sqsubseteq E_j$. Here we show only the first part since the other one can be shown in the same way.

First, assume that $F_i = \sigma(B_\nu)$ for a non-variable atom $B_\nu \in \{B_1, \ldots, B_n\}$. Since the clauses introduced in (1)2. of the translation are satisfied by τ, there is an A_j such that $\tau([A_j \not\sqsubseteq B_\nu]) = 0$. By Lemma 5, this implies $\sigma(A_j) \sqsubseteq \sigma(B_\nu) = F_i$.

If there is no non-variable atom $B_\nu \in \{B_1, \ldots, B_n\}$ such that $\sigma(B_\nu) = F_i$, then there is a variable B_ν such that the atom F_i is a conjunct of $\sigma(B_\nu)$. By the construction of σ, we know that there is a non-variable atom C of Γ such that $F_i = \sigma(C)$ and $\tau([B_\nu \not\sqsubseteq C]) = 0$. By our assumption, C is not in $\{B_1, \ldots, B_n\}$. Since the clauses created in (1)3. are satisfied by τ, there is an A_j such that $\tau([A_j \not\sqsubseteq C]) = 0$. By Lemma 5, this implies $\sigma(A_j) \sqsubseteq \sigma(C) = F_i$. □

3.2 Completeness

To show *completeness*, assume that Γ is solvable, and let γ be a unifier Γ. We must show that there is a propositional valuation τ satisfying all the clauses in $C(\Gamma)$. We define the propositional valuation τ as follows:

- for all atoms C, D of Γ, we define $\tau([C \not\sqsubseteq D]) := 1$ if $\gamma(C) \not\sqsubseteq \gamma(D)$; and $\tau([C \not\sqsubseteq D]) := 0$ if $\gamma(C) \sqsubseteq \gamma(D)$.
- for all variables X, Y occurring in Γ, we define $\tau([X{>}Y]) := 1$ if $X >_\gamma Y$; and $\tau([X{>}Y]) := 0$ otherwise.

In the following, we call τ the valuation *induced by* γ. We show that τ satisfies all the clauses that are created by our translation:

(1) In (1) of the translation we create three types of Horn clauses for each equation $A_1 \sqcap \cdots \sqcap A_m \equiv^? B_1 \sqcap \cdots \sqcap B_n$.
   1. If $C \in \{A_1, \ldots, A_m\}$ is a non-variable atom, then $C(\Gamma)$ contains the clause $[B_1 \not\sqsubseteq C] \wedge \cdots \wedge [B_n \not\sqsubseteq C] \rightarrow$.

      The fact that C is a non-variable atom (i.e., a concept constant or an existential restriction) implies that $\gamma(C)$ is also a concept constant or an existential restriction. Since γ is a unifier of the equation, Lemma 3 implies there must be an atom B_i such that $\gamma(B_i) \sqsubseteq \gamma(C)$. Therefore $\tau([B_i \not\sqsubseteq C]) = 0$, and the clause is satisfied by τ.
   2. The clauses generated in (1)2. of the translation can be treated similarly.
   3. If C is a non-variable atom of Γ that does not belong to $\{A_1, \ldots, A_m, B_1, \ldots, B_n\}$, then $C(\Gamma)$ contains the clause $[A_1 \not\sqsubseteq C] \wedge \cdots \wedge [A_m \not\sqsubseteq C] \rightarrow [B_k \not\sqsubseteq C]$ for $k = 1, \ldots, n$. (The symmetric clauses also introduced in (1)3. can be treated similarly.)

      To show that this clause is satisfied by τ, assume that $\tau([B_k \not\sqsubseteq C]) = 0$, i.e., $\gamma(B_k) \sqsubseteq \gamma(C)$. We must show that this implies $\tau([A_j \not\sqsubseteq C]) = 0$ for some j.

Now, $\gamma(A_1) \sqcap \cdots \sqcap \gamma(A_m) \equiv \gamma(B_1) \sqcap \cdots \sqcap \gamma(B_n) \sqsubseteq \gamma(B_k) \sqsubseteq \gamma(C)$ implies that there is an A_j such that $\gamma(A_j) \sqsubseteq \gamma(C)$, by Lemma 3. Thus, or definition of τ yields $\tau([A_j \not\sqsubseteq C]) = 0$.

(2) Now we look at the clauses introduced in (2). Since two constants cannot be in a subsumption relationship, the clauses in (2)1. are satisfied by τ. Similarly, the clauses in (2)2. are satisfied by τ since no existential restriction can subsume another one built using a different role name. The clauses in (2)3. are satisfied because $\gamma(\exists r.A) \sqsubseteq \gamma(\exists r.B)$ implies $\gamma(A) \sqsubseteq \gamma(B)$, by Lemma 1. In a similar way we can show that all clauses in (2)4. and (2)5. are satisfied by our valuation τ. Indeed, these clauses just describe valid properties of the subsumption relation in $\mathcal{EL}$.

(3) The clauses introduced in (3) all describe valid properties of the strict partial order $>_\gamma$; hence they are satisfied by τ.

Proposition 2 (Completeness). *The valuation τ induced by a unifier of Γ satisfies $C(\Gamma)$.*

3.3 Some Comments Regarding the Reduction

We have shown above that our SAT reduction is sound and complete in the sense that the (flat) $\mathcal{EL}$-unification problem Γ is solvable iff its translation $C(\Gamma)$ into a SAT problem is satisfiable. This proof is, of course, a formal justification of our definition of this translation. Here, we want to explain some aspects of this translation on a more intuitive level.

Basically, the clauses generated in (1) enforce that "enough" subsumption relationships hold to have a unifier, i.e., solve each equation. What "enough" means is based on Lemma 3: once we have applied the unifier, every atom on one side of the (instantiated) equation must subsume an (instantiated) conjunct on the other side. Such an atom can either be an instance of a non-variable atom (i.e., an existential restriction or a concept constant) occurring on this side of the equation, or it is introduced by the instantiation of a variable. The first case is dealt with by the clauses in (1)1. and (1)2. whereas the second case is dealt with by (1)3. A valuation of the propositional variables of the form $[A \not\sqsubseteq B]$ guesses such subsumptions, and the clauses generated in (1) ensure that enough of them are guessed for solving all equations. However, it is not sufficient to guess enough subsumptions. We also must make sure that these subsumptions can really be made to hold by applying an appropriate substitution. This is the role of the clauses introduced in (2). Basically, they say that two existential restrictions can only subsume each other if they are built using the same role name, and their direct subterms subsume each other. Two concept constants subsume each other iff they are equal, and there cannot be a subsumption relation between a concept constant and an existential restriction. To ensure that all such consequences of the guessed subsumptions are really taken into account, transitivity of subsumption is needed. Otherwise, we would, for example, not detect the conflict caused by guessing that $[A \not\sqsubseteq X]$ and $[X \not\sqsubseteq B]$ should be evaluated to 0, i.e., that (for the unifier σ to be constructed) we have $\sigma(A) \sqsubseteq \sigma(X) \sqsubseteq \sigma(B)$ for distinct concept

constants A, B. These kinds of conflicts correspond to what is called a *clash failure* in syntactic unification [8].

Example 3. To see the clauses generated in (1) and (2) of the translation at work, let us consider a simple example, where we assume that A, B are distinct concept constants and X, Y are distinct concept variables. Consider the equation

$$\exists r.X \equiv^? \exists r.Y, \tag{1}$$

which in (1)1. and (1)2. yields the clauses

$$[\exists r.Y \not\sqsubseteq \exists r.X] \rightarrow \quad \text{and} \quad [\exists r.X \not\sqsubseteq \exists r.Y] \rightarrow \tag{2}$$

These clauses state that, for any unifier σ of the equation (1) we must have $\sigma(\exists r.Y) \sqsubseteq \sigma(\exists r.X)$ and $\sigma(\exists r.X) \sqsubseteq \sigma(\exists r.Y)$. However, stating just these two clauses is not sufficient: we must also ensure that the assignments for the variables X and Y really realize these subsumptions. To see this, assume that we have the additional equation

$$X \sqcap Y \equiv^? A \sqcap B, \tag{3}$$

which yields the clauses

$$[X \not\sqsubseteq A] \wedge [Y \not\sqsubseteq A] \rightarrow \quad \text{and} \quad [X \not\sqsubseteq B] \wedge [Y \not\sqsubseteq B] \rightarrow \tag{4}$$

One possible way of satisfying these two clauses is to set

$$\tau([X \not\sqsubseteq A]) = 0 = \tau([Y \not\sqsubseteq B]) \quad \text{and} \quad \tau([X \not\sqsubseteq B]) = 1 = \tau([Y \not\sqsubseteq A]). \tag{5}$$

The substitution σ induced by this valuation replaces X by A and Y by B, and thus clearly does *not* satisfy the subsumptions $\sigma(\exists r.Y) \sqsubseteq \sigma(\exists r.X)$ and $\sigma(\exists r.X) \sqsubseteq \sigma(\exists r.Y)$. Choosing the incorrect valuation (5) is prevented by the clauses introduced in (2) of the translation. In fact, in (2)3. we introduce the clauses

$$[X \not\sqsubseteq Y] \rightarrow [\exists r.X \not\sqsubseteq \exists r.Y] \quad \text{and} \quad [Y \not\sqsubseteq X] \rightarrow [\exists r.Y \not\sqsubseteq \exists r.X] \tag{6}$$

Together with the clauses (2), these clauses can be used to deduce the clauses

$$[X \not\sqsubseteq Y] \rightarrow \quad \text{and} \quad [Y \not\sqsubseteq X] \rightarrow \tag{7}$$

Together with the transitivity clauses introduced in (2)5.:

$$[X \not\sqsubseteq B] \rightarrow [X \not\sqsubseteq Y] \vee [Y \not\sqsubseteq B] \quad \text{and} \quad [Y \not\sqsubseteq A] \rightarrow [Y \not\sqsubseteq X] \vee [X \not\sqsubseteq A] \tag{8}$$

the clauses (7) prevent the valuation (5).

This example illustrates, among other things, why the clauses introduced in (2)3. of the translation are needed. In fact, without the clauses (6), the incorrect valuation (5) could not have been prevented.

One may wonder why we only construct the implications in (2)3., but *not* the implications in the other direction:

$$[\exists r.A \not\sqsubseteq \exists r.B] \rightarrow [A \not\sqsubseteq B]$$

The reason is that these implications are not needed to ensure soundness.

Example 4. Consider the unification problem

$$\{X \equiv^? A, \quad Y \equiv^? \exists r.X, \quad Z \equiv^? \exists r.A\},$$

which produces the clauses $[X \not\sqsubseteq A] \rightarrow$, $[Y \not\sqsubseteq \exists r.X] \rightarrow$, $[Z \not\sqsubseteq \exists r.A] \rightarrow$.

The clause $[X \not\sqsubseteq A] \rightarrow$ states that, in any unifier σ of the first equation, we must have $\sigma(X) \sqsubseteq \sigma(A)$. Though this does imply that $\sigma(\exists r.X) \sqsubseteq \sigma(\exists r.A)$, there is no need to state this with the clause $[\exists r.X \not\sqsubseteq \exists r.A] \rightarrow$ since this subsumption is not needed to solve the equation. Thus, it actually does not hurt if a valuation evaluates $[\exists r.X \not\sqsubseteq \exists r.A]$ with 1. In fact, this decision does not influence the substitution for X that is computed from the valuation.

Expressed on a more technical level, the crucial tool for proving soundness is Lemma 5, which says that $\tau([C_1 \not\sqsubseteq C_2]) = 0$ implies $\sigma(C_1) \sqsubseteq \sigma(C_2)$ for the substitution σ induced by τ. This lemma does not state, and our proof of soundness does not need, the implication in the other direction. As illustrated in the above example, it may well be the case that $\sigma(C_1) \sqsubseteq \sigma(C_2)$ although the satisfying valuation τ evaluates $[C_1 \not\sqsubseteq C_2]$ to 1. The proof of Lemma 5 is by induction on the role depth, and thus reduces the problem of showing a subsumption relationship for terms of a higher role depth to the problem of showing subsumption relationships for terms of a lower role depth. This is exactly what the clauses in (2)3. allow us to do. The implications in the other direction are not required for this. They would be needed for proving the other direction of the lemma, but this is not necessary for proving soundness.

Until now, we have not mentioned the clauses generated in (3). Intuitively, they are there to detect what are called *occurs check failures* in the terminology of syntactic unification [8]. To be more precise, the variables of the form $[X{>}Y]$ together with the clauses generated in (3)1. are used to guess a strict partial order on the variables occurring in the unification problem. The clauses generated in (3)2. are used to enforce that only variables Y smaller than X can occur in the set S_X defined by a satisfying valuation. This makes it possible to use the sets S_X to define a substitution σ by induction on the strict partial order. Thus, this order realizes what is called a *constant restriction* in the literature on combining unification algorithms [7]. We have already seen the clauses generated in (3) at work in Example 1.

4 Connection to the Original "in NP" Proof

It should be noted that, in the present paper, we give a proof of the fact that $\mathcal{EL}$-unification is in NP that is independent of the proof in [4]. The only result from [4] that we have used is the characterization of subsumption (Lemma 1), which is an easy consequence of known results for $\mathcal{EL}$ [10]. In [4], the "in NP" result is basically shown as follows:

1. define a well-founded partial order $\succ$ on substitutions and use this to show that any solvable $\mathcal{EL}$-unification problem has a ground unifier that is minimal w.r.t. this order;

2. show that minimal ground unifiers are local in the sense that they are built from atoms of Γ;
3. use the locality of minimal ground unifiers to devise a "guess and then test" NP-algorithm for generating a minimal ground unifier.

The proof of 2., which shows that a non-local unifier cannot be minimal, is quite involved. Compared to that proof, the proof of soundness and completeness given in the present paper is much simpler.

In order to give a closer comparison between the approach used in [4] and the one employed in the present paper, let us recall some of the definitions and results from [4] in more detail:

Definition 2. *Let Γ be a flat $\mathcal{EL}$-unification problem, and γ be a ground unifier of Γ. Then γ is called* local *if, for each variable X in Γ, there are $n \geq 0$ non-variable atoms $D_1, \ldots, D_n$ of Γ such that $\gamma(X) = \gamma(D_1) \sqcap \cdots \sqcap \gamma(D_n)$, where the empty conjunction is $\top$.*

The "guess and then test" algorithm in [4] crucially depends on the fact that any solvable $\mathcal{EL}$-unification problem has a local unifier. This result can be obtained as an easy consequence of our proof of soundness and completeness.

Corollary 2. *Let Γ be a flat $\mathcal{EL}$-unification problem that is solvable. Then Γ has a local unifier.*

Proof. Since Γ is solvable, our completeness result implies that $C(\Gamma)$ is satisfiable. Let τ be a valuation that satisfies $C(\Gamma)$, and let σ be the unifier of Γ induced by τ in our proof of soundness. Locality of σ is an immediate consequence of the definition of σ. □

This shows that one does not really need the notion of minimality, and the quite involved proof that minimal unifiers are local given in [4], to justify the completeness of the "guess and then test" algorithm from [4]. However, in [4] minimal unifiers are also used to show a stronger completeness result for the "guess and then test" algorithm: it is shown that (up to equivalence) every minimal ground unifier is computed by the algorithm. In the following, we show that this is also the case for the unification algorithm obtained through our reduction.

Definition 3. *Let σ and γ be substitutions, and Γ be an $\mathcal{EL}$-unification problem. We define*

- *$\gamma \succeq \sigma$ if, for each variable X in Γ, we have $\gamma(X) \sqsubseteq \sigma(X)$;*
- *$\gamma \equiv \sigma$ if $\gamma \succeq \sigma$ and $\sigma \succeq \gamma$, and $\gamma \succ \sigma$ if $\gamma \succeq \sigma$ and $\sigma \not\equiv \gamma$;*
- *γ is a* minimal *unifier of Γ if there is no unifier σ of Γ such that $\gamma \succ \sigma$.*

As a corollary to our soundness and completeness proof, we can show that any minimal ground unifier σ of Γ is computed by our reduction, in the sense that it is induced by a satisfying valuation of $C(\Gamma)$.

Corollary 3. *Let Γ be a flat $\mathcal{EL}$-unification problem. If γ is a minimal ground unifier of Γ, then there is a unifier σ, induced by a satisfying valuation τ of $C(\Gamma)$, such that $\sigma \equiv \gamma$.*

Proof. Let γ be a minimal ground unifier of Γ, and τ the satisfying valuation of $C(\Gamma)$ induced by γ. We show that the unifier σ of Γ induced by τ satisfies $\gamma \succeq \sigma$. Minimality of γ then implies $\gamma \equiv \sigma$.

We must show that, for each variable X occurring in Γ, we have $\gamma(X) \sqsubseteq \sigma(X)$. We prove this by well-founded induction on the strict partial order $>$ defined as $X > Y$ iff $\tau([X{>}Y]) = 1$.[5]

Let X be a minimal variable with respect to this order. Since τ satisfies the clauses in (3)2., the set S_X induced by τ (see the proof of soundness) contains only ground atoms. Let $S_X = \{C_1, \ldots, C_n\}$ for $n \geq 0$ ground atoms. If $n = 0$, then $\sigma(X) = \top$, and thus $\gamma(X) \sqsubseteq \sigma(X)$ is trivially satisfied. Otherwise, we have $\sigma(X) = \sigma(C_1) \sqcap \ldots \sqcap \sigma(C_n) = C_1 \sqcap \ldots \sqcap C_n$, and we know, for each $i \in \{1, \ldots, n\}$, that $\tau([X \not\sqsubseteq C_i]) = 0$ by the definition of S_X. Since τ is the valuation induced by the unifier γ, this implies that $\gamma(X) \sqsubseteq \gamma(C_i) = C_i$. Consequently, we have shown that $\gamma(X) \sqsubseteq C_1 \sqcap \ldots \sqcap C_n = \sigma(X)$.

Now we assume, by induction, that we have $\gamma(Y) \sqsubseteq \sigma(Y)$ for all variables Y such that $X > Y$. Let $S_X = \{C_1, \ldots, C_n\}$ for $n \geq 0$ non-variable atoms of Γ. If $n = 0$, then $\sigma(X) = \top$, and thus $\gamma(X) \sqsubseteq \sigma(X)$ is again trivially satisfied. Otherwise, we have $\sigma(X) = \sigma(C_1) \sqcap \cdots \sqcap \sigma(C_n)$, and we know, for each $i \in \{1, \ldots, n\}$, that $\tau([X \not\sqsubseteq C_i]) = 0$ by the definition of S_X. Since τ is the valuation induced by the unifier γ, this implies that $\gamma(X) \sqsubseteq \gamma(C_i)$. for each $i \in \{1, \ldots, n\}$. Since all variables occurring in $C_1, \ldots, C_n$ are smaller than X and since the concept constructors of $\mathcal{EL}$ are monotonic w.r.t. subsumption, we have by induction that $\gamma(C_i) \sqsubseteq \sigma(C_i)$ for each $i \in \{1, \ldots, n\}$. Consequently, we have $\gamma(X) \sqsubseteq \gamma(C_1) \sqcap \ldots \sqcap \gamma(C_n) \sqsubseteq \sigma(C_1) \sqcap \cdots \sqcap \sigma(C_n) = \sigma(X)$. □

Table 2. Experimental Results

Size	#InVars(#FlatVars)	#Atoms	#PropVars	#Clauses	OverallTime	MiniSatTime
10	2(5)	10	125	895	58 ms	0 ms
10	2(5)	11	146	1 184	79 ms	4 ms
22	2(10)	24	676	13 539	204 ms	4 ms
22	2(10)	25	725	15 254	202 ms	8 ms
22	2(10)	25	725	15 254	211 ms	8 ms
22	3(11)	26	797	17 358	222 ms	8 ms

5 Conclusion

The results presented in this paper are of interest both from a theoretical and a practical point of view. From the theoretical point of view, this paper gives a new proof of the fact that $\mathcal{EL}$-unification is in NP, which is considerably simpler

[5] The clauses in $C(\Gamma)$ make sure that this is indeed a strict partial order. It is trivially well-founded since Γ contains only finitely many variables.

than the original proof given in [4]. We have also shown that the stronger completeness result for the "guess and then test" NP algorithm of [4] (all minimal ground unifiers are computed) holds as well for the new algorithm presented in this paper. From the practical point of view, the translation into propositional satisfiability allows us to employ highly optimized state of the art SAT solvers when implementing an $\mathcal{EL}$-unification algorithm.

We have actually implemented the SAT translation described in this paper in Java, and have used MiniSat for the satisfiability check. Until now, we have not yet optimized the translation, and we have tested the algorithm only on relatively small (solvable) unification problems extracted from SNOMED CT. Table 1 shows the first experimental results obtained for these problems. The *first column* counts the size of the input problem (number of occurrences of concept and role names); the *second column* the number of concept variables before and after flattening; the *third column* the number of atoms in the flattened unification problem; the *fourth column* the number of propositional variables introduced by our translation; the *fifth column* the number of clauses introduced by our translation; the *sixth column* the overall run-time (in milliseconds) for deciding whether a unifier exists; and the *seventh column* the time (in milliseconds) needed by MiniSat for deciding the satisfiability of the generated clause set.

In [5] we have introduced a more goal-oriented variant of the brutal "guess and then test" algorithm of [4], which tries to transform a given flat unification problem into solved form. However, without any smart backtracking strategies, a first implementation of this algorithm cannot compete with the SAT translation presented in this paper.

References

1. Baader, F.: Terminological cycles in a description logic with existential restrictions. In: Proc. IJCAI 2003. Morgan Kaufmann, Los Altos (2003)
2. Baader, F., Brandt, S., Lutz, C.: Pushing the $\mathcal{EL}$ envelope. In: Proc. IJCAI 2005. Morgan Kaufmann, Los Altos (2005)
3. Baader, F., Calvanese, D., McGuinness, D., Nardi, D., Patel-Schneider, P.F. (eds.): The Description Logic Handbook: Theory, Implementation, and Applications. Cambridge University Press, Cambridge (2003)
4. Baader, F., Morawska, B.: Unification in the description logic $\mathcal{EL}$. In: Treinen, R. (ed.) RTA 2009. LNCS, vol. 5595, pp. 350–364. Springer, Heidelberg (2009)
5. Baader, F., Morawska, B.: Unification in the description logic $\mathcal{EL}$. In: Logical Methods in Computer Science (to appear, 2010)
6. Baader, F., Narendran, P.: Unification of concepts terms in description logics. J. of Symbolic Computation 31(3), 277–305 (2001)
7. Baader, F., Schulz, K.: Unification in the union of disjoint equational theories: Combining decision procedures. J. of Symbolic Computation 21(2), 211–243 (1996)
8. Baader, F., Snyder, W.: Unification theory. In: Handbook of Automated Reasoning, vol. I. Elsevier Science Publishers, Amsterdam (2001)
9. Kapur, D., Narendran, P.: Complexity of unification problems with associative-commutative operators. J. Automated Reasoning 9, 261–288 (1992)
10. Küsters, R. (ed.): Non-Standard Inferences in Description Logics. LNCS (LNAI), vol. 2100, p. 33. Springer, Heidelberg (2001)

Logical Methods in Computer Science
Vol. 6 (3:17) 2010, pp. 1–31
www.lmcs-online.org

Submitted Oct. 21, 2009
Published Sep. 4, 2010

UNIFICATION IN THE DESCRIPTION LOGIC $\mathcal{EL}$

FRANZ BAADER AND BARBARA MORAWSKA

Theoretical Computer Science, TU Dresden, Germany
e-mail address: {baader,morawska}@tcs.inf.tu-dresden.de

Abstract. The Description Logic $\mathcal{EL}$ has recently drawn considerable attention since, on the one hand, important inference problems such as the subsumption problem are polynomial. On the other hand, $\mathcal{EL}$ is used to define large biomedical ontologies. Unification in Description Logics has been proposed as a novel inference service that can, for example, be used to detect redundancies in ontologies. The main result of this paper is that unification in $\mathcal{EL}$ is decidable. More precisely, $\mathcal{EL}$-unification is NP-complete, and thus has the same complexity as $\mathcal{EL}$-matching. We also show that, w.r.t. the unification type, $\mathcal{EL}$ is less well-behaved: it is of type zero, which in particular implies that there are unification problems that have no finite complete set of unifiers.

1. Introduction

Description logics (DLs) [6] are a family of logic-based knowledge representation formalisms, which can be used to represent the conceptual knowledge of an application domain in a structured and formally well-understood way. They are employed in various application domains, such as natural language processing, configuration of technical systems, databases, and biomedical ontologies, but their most notable success so far is the adoption of the DL-based language OWL [20] as standard ontology language for the semantic web.

In DLs, concepts are formally described by *concept terms*, i.e., expressions that are built from concept names (unary predicates) and role names (binary predicates) using concept constructors. The expressivity of a particular DL is determined by which concept constructors are available in it. From a semantic point of view, concept names and concept terms represent sets of individuals, whereas roles represent binary relations between individuals. For example, using the concept name Woman, and the role name child, the concept of *women having a daughter* can be represented by the concept term

$$\mathsf{Woman} \sqcap \exists\, \mathsf{child}.\mathsf{Woman},$$

and the concept of *women having only daughters* by

$$\mathsf{Woman} \sqcap \forall\, \mathsf{child}.\mathsf{Woman}.$$

1998 ACM Subject Classification: F.4.1, I.2.3, I.2.4.
Key words and phrases: knowledge representation, unification, Description Logics, complexity.
Supported by DFG under grant BA 1122/14–1.

DOI:10.2168/LMCS-6 (3:17) 2010

Knowledge representation systems based on DLs provide their users with various inference services that allow them to deduce implicit knowledge from the explicitly represented knowledge. An important inference problem solved by DL systems is the subsumption problem: the subsumption algorithm allows one to determine subconcept-superconcept relationships. For example, the concept term Woman subsumes the concept term $\mathsf{Woman} \sqcap \exists\, \mathsf{child}.\mathsf{Woman}$ since all instances of the second term are also instances of the first term, i.e., the second term is always interpreted as a subset of the first term. With the help of the subsumption algorithm, a newly introduced concept term can automatically be placed at the correct position in the hierarchy of the already existing concept terms.

Two concept terms C, D are *equivalent* ($C \equiv D$) if they subsume each other, i.e., if they always represent the same set of individuals. For example, the terms $\forall\, \mathsf{child}.\mathsf{Rich} \sqcap \forall\, \mathsf{child}.\mathsf{Woman}$ and $\forall\, \mathsf{child}.(\mathsf{Rich} \sqcap \mathsf{Woman})$ are equivalent since the value restriction operator ($\forall\, r.C$) distributes over the conjunction operator ($\sqcap$). If we replace the value restriction operator by the existential restriction operator ($\exists\, r.C$), then this equivalence no longer holds. However, for this operator, we still have the equivalence

$$\exists\, \mathsf{child}.\mathsf{Rich} \sqcap \exists\, \mathsf{child}.(\mathsf{Woman} \sqcap \mathsf{Rich}) \equiv \exists\, \mathsf{child}.(\mathsf{Woman} \sqcap \mathsf{Rich}).$$

The equivalence test can, for example, be used to find out whether a concept term representing a particular notion has already been introduced, thus avoiding multiple introduction of the same concept into the concept hierarchy. This inference capability is very important if the knowledge base containing the concept terms is very large, evolves during a long time period, and is extended and maintained by several knowledge engineers. However, testing for equivalence of concepts is not always sufficient to find out whether, for a given concept term, there already exists another concept term in the knowledge base describing the same notion. On the one hand, different knowledge engineers may use different names for concepts, like Male versus $\mathsf{Masculine}$. On the other hand, they may model on different levels of granularity. For example, assume that one knowledge engineer has defined the concept of *men loving fast cars* by the concept term

$$\mathsf{Human} \sqcap \mathsf{Male} \sqcap \exists\, \mathsf{loves}.\mathsf{Sports\_car}.$$

A second knowledge engineer might represent this notion in a somewhat different way, e.g., by using the concept term

$$\mathsf{Man} \sqcap \exists\, \mathsf{loves}.(\mathsf{Car} \sqcap \mathsf{Fast}).$$

These two concept terms are not equivalent, but they are meant to represent the same concept. The two terms can obviously be made equivalent by substituting the concept name $\mathsf{Sports\_car}$ in the first term by the concept term $\mathsf{Car} \sqcap \mathsf{Fast}$ and the concept name Man in the second term by the concept term $\mathsf{Human} \sqcap \mathsf{Male}$. This leads us to *unification of concept terms*, i.e., the question whether two concept terms can be made equivalent by applying an appropriate substitution, where a substitution replaces (some of the) concept names by concept terms. Of course, it is not necessarily the case that unifiable concept terms are meant to represent the same notion. A unifiability test can, however, suggest to the knowledge engineer possible candidate terms. A *unifier* (i.e., a substitution whose application makes the two terms equivalent) then proposes appropriate definitions for the concept names. In our example, we know that, if we define Man as $\mathsf{Human} \sqcap \mathsf{Male}$ and $\mathsf{Sports\_car}$ as $\mathsf{Car} \sqcap \mathsf{Fast}$, then the concept terms $\mathsf{Human} \sqcap \mathsf{Male} \sqcap \exists\, \mathsf{loves}.\mathsf{Sports\_car}$ and $\mathsf{Man} \sqcap \exists\, \mathsf{loves}.(\mathsf{Car} \sqcap \mathsf{Fast})$ are equivalent w.r.t. these definitions.

Unification in DLs was first considered in [12] for a DL called $\mathcal{FL}_0$, which has the concept constructors *conjunction* ($\sqcap$), *value restriction* ($\forall r.C$), and the *top concept* ($\top$). It was shown that unification in $\mathcal{FL}_0$ is decidable and ExpTime-complete, i.e., given an $\mathcal{FL}_0$-unification problem, we can effectively decide whether it has a solution or not, but in the worst-case, any such decision procedure needs exponential time. This result was extended in [8] to a more expressive DL, which additionally has the role constructor *transitive closure*. Interestingly, the *unification type* of $\mathcal{FL}_0$ had been determined almost a decade earlier in [2]. In fact, as shown in [12], unification in $\mathcal{FL}_0$ corresponds to unification modulo the equational theory of idempotent Abelian monoids with several homomorphisms. In [2] it was shown that, already for a single homomorphism, unification modulo this theory has unification type zero, i.e., there are unification problems for this theory that do not have a minimal complete set of unifiers. In particular, such unification problems cannot have a finite complete set of unifiers.

In this paper, we consider unification in the DL $\mathcal{EL}$. The $\mathcal{EL}$-family consists of inexpressive DLs whose main distinguishing feature is that they provide their users with *existential restrictions* ($\exists r.C$) rather than value restrictions ($\forall r.C$) as the main concept constructor involving roles. The core language of this family is $\mathcal{EL}$, which has the top concept, conjunction, and existential restrictions as concept constructors. This family has recently drawn considerable attention since, on the one hand, the subsumption problem stays tractable (i.e., decidable in polynomial time) in situations where $\mathcal{FL}_0$, the corresponding DL with value restrictions, becomes intractable: subsumption between concept terms is tractable for both $\mathcal{FL}_0$ and $\mathcal{EL}$ [25, 10], but allowing the use of concept definitions or even more expressive terminological formalisms makes $\mathcal{FL}_0$ intractable [26, 3, 23, 5], whereas it leaves $\mathcal{EL}$ tractable [4, 17, 5]. On the other hand, although of limited expressive power, $\mathcal{EL}$ is nevertheless used in applications, e.g., to define biomedical ontologies. For example, both the large medical ontology SNOMED CT[1] and the Gene Ontology[2] can be expressed in $\mathcal{EL}$, and the same is true for large parts of the medical ontology GALEN [27]. The importance of $\mathcal{EL}$ can also be seen from the fact that the new OWL 2 standard[3] contains a sub-profile OWL 2 EL, which is based on (an extension of) $\mathcal{EL}$.

Unification in $\mathcal{EL}$ has, to the best of our knowledge, not been investigated before, but matching (where one side of the equation(s) to be solved does not contain variables) has been considered in [7, 24]. In particular, it was shown in [24] that the decision problem, i.e., the problem of deciding whether a given $\mathcal{EL}$-matching problem has a matcher or not, is NP-complete. Interestingly, $\mathcal{FL}_0$ behaves better w.r.t. matching than $\mathcal{EL}$: for $\mathcal{FL}_0$, the decision problem is tractable [9]. In this paper, we show that, w.r.t. the unification type, $\mathcal{FL}_0$ and $\mathcal{EL}$ behave the same: just as $\mathcal{FL}_0$, the DL $\mathcal{EL}$ has unification type zero. However, w.r.t. the decision problem, $\mathcal{EL}$ behaves much better than $\mathcal{FL}_0$: $\mathcal{EL}$-unification is NP-complete, and thus has the same complexity as $\mathcal{EL}$-matching.

Regarding unification in DLs that are more expressive than $\mathcal{EL}$ and $\mathcal{FL}_0$, one must look at the literature on unification in modal logics. It is well-known that there is a close connection between modal logics and DLs [6]. For example, the DL $\mathcal{ALC}$, which can be obtained by adding negation to $\mathcal{EL}$ or $\mathcal{FL}_0$, corresponds to the basic (multi-)modal logic K. Decidability of unification in K is a long-standing open problem. Recently, undecidability of unification in some extensions of K (for example, by the universal modality) was shown

[1] http://www.ihtsdo.org/snomed-ct/

[2] http://www.geneontology.org/

[3] See http://www.w3.org/TR/owl2-profiles/

in [29]. The undecidability results in [29] also imply undecidability of unification in some expressive DLs (e.g., $\mathcal{SHIQ}$ [21]). The unification types of some modal (and related) logics have been determined by Ghilardi; for example in [19] he shows that K4 and S4 have unification type finitary. Unification in sub-Boolean modal logics (i.e., modal logics that are not closed under all Boolean operations, such as the modal logics corresponding to $\mathcal{EL}$ and $\mathcal{FL}_0$) has, to the best of our knowledge, not been considered in the modal logic literature.

In addition to unification of concept terms as introduced until now, we will also consider unification w.r.t. a so-called acyclic TBox in this article. Until now, we have only talked about concept terms, i.e., complex descriptions of concepts that are built from concept and role names using the concept constructors of the given DL. In applications of DLs, it is, of course, inconvenient to always use such complex descriptions when referring to concepts. For this reason, DLs are usually also equipped with a terminological formalism. In its simplest form, this formalism allows to introduce abbreviations for concept terms. For example, the two concept definitions

$$\mathsf{Mother} \equiv \mathsf{Woman} \sqcap \exists\, \mathsf{child}.\mathsf{Human} \quad \text{and} \quad \mathsf{Woman} \equiv \mathsf{Human} \sqcap \mathsf{Female}$$

introduce the abbreviation Woman for the concept term Human ⊓ Female and the abbreviation Mother for the concept term Human ⊓ Female ⊓ ∃ child.Human. A finite set of such concept definitions is called an *acyclic* TBox if it is unambiguous (i.e., every concept name occurs at most once as left-hand side) and acyclic (i.e., there are no cyclic dependencies between concept definitions). These restrictions ensure that every defined concept (i.e., concept name occurring on the left-hand side of a definition) has a unique expansion to a concept term that it abbreviates. Inference problems like subsumption and unification can also be considered w.r.t. such acyclic TBoxes. As mentioned above, the complexity of the subsumption problem increases for the DL $\mathcal{FL}_0$ if acyclic TBoxes are taken into account [26]. In contrast, for $\mathcal{EL}$, the complexity of the subsumption problem stays polynomial in the presence of acyclic TBoxes. We show that, for unification in $\mathcal{EL}$, adding acyclic TBoxes is also harmless, i.e., unification in $\mathcal{EL}$ *w.r.t. acyclic TBoxes* is also NP-complete.

This article is structured as follows. In the next section, we define the DL $\mathcal{EL}$ and unification in $\mathcal{EL}$ more formally. In Section 3, we recall the characterization of subsumption and equivalence in $\mathcal{EL}$ from [24], and in Section 4 we use this to show that unification in $\mathcal{EL}$ has type zero. In Section 5, we show that unification in $\mathcal{EL}$ is NP-complete. The unification algorithm establishing the complexity upper bound is a typical "guess and then test" NP-algorithm, and thus it is unlikely that a direct implementation of this algorithm will perform well in practice. In Section 6, we introduce a more goal-oriented unification algorithm for $\mathcal{EL}$, in which non-deterministic decisions are only made if they are triggered by "unsolved parts" of the unification problem. In Section 7, we point out that our results for $\mathcal{EL}$-unification imply that unification modulo the equational theory of semilattices with monotone operators [28] is NP-complete and of unification type zero.

More information about Description Logics can be found in [6], and about unification theory in [16]. This article is an extended version of a paper [11] published in the proceedings of the 20th international Conference on Rewriting Techniques and applications (RTA'09). In addition to giving more detailed proofs, we have added the goal-oriented unification algorithm (Section 6) and the treatment of unification modulo acyclic TBoxes (Subsection 2.3).

Name	Syntax	Semantics
concept name	A	$A^{\mathcal{I}} \subseteq \mathcal{D}_{\mathcal{I}}$
role name	r	$r^{\mathcal{I}} \subseteq \mathcal{D}_{\mathcal{I}} \times \mathcal{D}_{\mathcal{I}}$
top-concept	$\top$	$\top^{\mathcal{I}} = \mathcal{D}_{\mathcal{I}}$
conjunction	$C \sqcap D$	$(C \sqcap D)^{\mathcal{I}} = C^{\mathcal{I}} \cap D^{\mathcal{I}}$
existential restriction	$\exists\, r.C$	$(\exists\, r.C)^{\mathcal{I}} = \{x \mid \exists\, y : (x,y) \in r^{\mathcal{I}} \wedge y \in C^{\mathcal{I}}\}$
subsumption	$C \sqsubseteq D$	$C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$
equivalence	$C \equiv D$	$C^{\mathcal{I}} = D^{\mathcal{I}}$

Table 1: Syntax and semantics of $\mathcal{EL}$

2. Unification in $\mathcal{EL}$

In this section, we first define the syntax and semantics of $\mathcal{EL}$-concept terms as well as the subsumption and the equivalence relation on these terms. Then, we introduce unification of $\mathcal{EL}$-concept terms, and finally extend this notion to unification modulo an acyclic TBox.

2.1. **The Description Logic $\mathcal{EL}$.** Starting with a set N_{con} of concept names and a set N_{role} of role names, *$\mathcal{EL}$-concept terms* are built using the following concept constructors: the nullary constructor *top-concept* ($\top$), the binary constructor *conjunction* ($C \sqcap D$), and for every role name $r \in N_{role}$, the unary constructor *existential restriction* ($\exists\, r.C$). The semantics of $\mathcal{EL}$ is defined in the usual way, using the notion of an interpretation $\mathcal{I} = (\mathcal{D}_{\mathcal{I}}, \cdot^{\mathcal{I}})$, which consists of a nonempty domain $\mathcal{D}_{\mathcal{I}}$ and an interpretation function $\cdot^{\mathcal{I}}$ that assigns binary relations on $\mathcal{D}_{\mathcal{I}}$ to role names and subsets of $\mathcal{D}_{\mathcal{I}}$ to concept terms, as shown in the semantics column of Table 1.

The concept term C *is subsumed by* the concept term D (written $C \sqsubseteq D$) iff $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ holds for all interpretations $\mathcal{I}$. We say that C *is equivalent to* D (written $C \equiv D$) iff $C \sqsubseteq D$ and $D \sqsubseteq C$, i.e., iff $C^{\mathcal{I}} = D^{\mathcal{I}}$ holds for all interpretations $\mathcal{I}$. The concept term C *is strictly subsumed by* the concept term D (written $C \sqsubset D$) iff $C \sqsubseteq D$ and $C \not\equiv D$. It is well-known that subsumption (and thus also equivalence) of $\mathcal{EL}$-concept terms can be decided in polynomial time [10].

2.2. **Unification of concept terms.** In order to define unification of concept terms, we first introduce the notion of a substitution operating on concept terms. To this purpose, we partition the set of concepts names into a set N_v of concept variables (which may be replaced by substitutions) and a set N_c of concept constants (which must not be replaced by substitutions). Intuitively, N_v are the concept names that have possibly been given another name or been specified in more detail in another concept term describing the same notion. The elements of N_c are the ones of which it is assumed that the same name is used by all knowledge engineers (e.g., standardized names in a certain domain).

A *substitution* σ is a mapping from N_v into the set of all $\mathcal{EL}$-concept terms. This mapping is extended to concept terms in the obvious way, i.e.,

- $\sigma(A) := A$ for all $A \in N_c$,
- $\sigma(\top) := \top$,
- $\sigma(C \sqcap D) := \sigma(C) \sqcap \sigma(D)$, and

- $\sigma(\exists r.C) := \exists r.\sigma(C)$.

Definition 2.1. An *$\mathcal{EL}$-unification problem* is of the form $\Gamma = \{C_1 \equiv^? D_1, \ldots, C_n \equiv^? D_n\}$, where $C_1, D_1, \ldots, C_n, D_n$ are $\mathcal{EL}$-concept terms. The substitution σ is a *unifier* (or *solution*) of Γ iff $\sigma(C_i) \equiv \sigma(D_i)$ for $i = 1, \ldots, n$. In this case, Γ is called *solvable* or *unifiable*.

When we say that $\mathcal{EL}$-unification is *decidable*, then we mean that the following decision problem is decidable: given an $\mathcal{EL}$-unification problem Γ, decide whether Γ is solvable or not. Accordingly, we say that $\mathcal{EL}$-unification is *NP-complete* if this decision problem is NP-complete.

In the following, we introduce some standard notions from unification theory [16], but formulated for the special case of $\mathcal{EL}$-unification rather than for an arbitrary equational theory. Unifiers can be compared using the instantiation preorder $\preceq$. Let Γ be an $\mathcal{EL}$-unification problem, V the set of variables occurring in Γ, and σ, θ two unifiers of this problem. We define

$$\sigma \preceq \theta \text{ iff there is a substitution } \lambda \text{ such that } \theta(X) \equiv \lambda(\sigma(X)) \text{ for all } X \in V.$$

If $\sigma \preceq \theta$, then we say that θ is an *instance* of σ.

Definition 2.2. Let Γ be an $\mathcal{EL}$-unification problem. The set of substitutions M is called a *complete set of unifiers* for Γ iff it satisfies the following two properties:

(1) every element of M is a unifier of Γ;
(2) if θ is a unifier of Γ, then there exists a unifier $\sigma \in M$ such that $\sigma \preceq \theta$.

The set M is called a *minimal complete set of unifiers* for Γ iff it additionally satisfies

(3) if $\sigma, \theta \in M$, then $\sigma \preceq \theta$ implies $\sigma = \theta$.

The unification type of a given unification problem is determined by the existence and cardinality[4] of such a minimal complete set.

Definition 2.3. Let Γ be an $\mathcal{EL}$-unification problem. This problem has type

- *unitary* iff it has a minimal complete set of unifiers of cardinality 1;
- *finitary* iff it has a finite minimal complete set of unifiers;
- *infinitary* iff it has an infinite minimal complete set of unifiers;
- *zero* iff it does not have a minimal complete set of unifiers.

Note that the set of all unifiers of a given $\mathcal{EL}$-unification problem is always a complete set of unifiers. However, this set is usually infinite and redundant (in the sense that some unifiers are instances of others). For a unitary or finitary $\mathcal{EL}$-unification problem, all unifiers can be represented by a finite complete set of unifiers, whereas for problems of type infinitary or zero this is no longer possible. In fact, if a problem has a finite complete set of unifiers M, then it also has a finite *minimal* complete set of unifiers, which can be obtained by iteratively removing redundant elements from M. For an infinite complete set of unifiers, this approach of removing redundant unifiers may be infinite, and the set reached in the limit need no longer be complete. This is what happens for problems of type zero. The difference between infinitary and type zero is that a unification problem of type zero cannot even have a non-redundant complete set of unifiers, i.e., every complete set of unifiers must contain different unifiers σ, θ such that $\sigma \preceq \theta$. More information on unification type zero can be found in [1].

[4]It is easy to see that the cardinality of a minimal complete set of unifiers is uniquely determined by the unification problem.

When we say that $\mathcal{EL}$ *has unification type zero*, we mean that there exists an $\mathcal{EL}$-unification problem that has type zero. Before we can prove in Section 4 that this is indeed the case, we must have a closer look at equivalence in $\mathcal{EL}$ in Section 3. But first, we consider unification modulo acyclic TBoxes.

2.3. **Unification modulo acyclic TBoxes.** A *concept definition* is of the form $A \doteq C$ where A is a concept name and C is a concept term. A *TBox* $\mathcal{T}$ is a finite set of concept definitions such that no concept name occurs more than once on the left-hand side of a concept definition in $\mathcal{T}$. The TBox $\mathcal{T}$ is called *acyclic* if there are no cyclic dependencies between its concept definitions. To be more precise, we say that the concept name A *directly depends on* the concept name B in a TBox $\mathcal{T}$ if $\mathcal{T}$ contains a concept definition $A \doteq C$ and B occurs in C. Let *depends on* be the transitive closure of the relation *directly depends on*. Then $\mathcal{T}$ contains a *terminological cycle* if there is a concept name A that depends on itself. Otherwise, $\mathcal{T}$ is called *acyclic*. Given a TBox $\mathcal{T}$, we call a concept name A a *defined concept* if it occurs as the left-side of a concept definition $A \doteq C$ in $\mathcal{T}$. All other concept names are called *primitive concepts*.

The interpretation $\mathcal{I}$ is a model of the TBox $\mathcal{T}$ iff $A^{\mathcal{I}} = C^{\mathcal{I}}$ holds for all concept definitions $A \doteq C$ in $\mathcal{T}$. Subsumption and equivalence w.r.t. a TBox are defined as follows: $C \sqsubseteq_{\mathcal{T}} D$ ($C \equiv_{\mathcal{T}} D$) iff $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ ($C^{\mathcal{I}} = D^{\mathcal{I}}$) holds for all models $\mathcal{I}$ of $\mathcal{T}$.

Subsumption and equivalence w.r.t. an acyclic TBox can be reduced to subsumption and equivalence of concept terms (without TBox) by *expanding* the concept terms w.r.t. the TBox: given a concept term C, its expansion $C^{\mathcal{T}}$ w.r.t. the acyclic TBox $\mathcal{T}$ is obtained by exhaustively replacing all defined concept names A occurring on the left-hand side of concept definitions $A \doteq C$ in $\mathcal{T}$ by their defining concept terms C. Given concept terms C, D, we have $C \sqsubseteq_{\mathcal{T}} D$ iff $C^{\mathcal{T}} \sqsubseteq D^{\mathcal{T}}$ [14]. The same is true for equivalence, i.e., $C \equiv_{\mathcal{T}} D$ iff $C^{\mathcal{T}} \equiv D^{\mathcal{T}}$. This expansion process may, however, result in an exponential blow-up [26, 14], and thus this reduction of subsumption and equivalence w.r.t. an acyclic TBox to subsumption and equivalence without a TBox is not polynomial. Nevertheless, in $\mathcal{EL}$, subsumption (and thus also equivalence) w.r.t. acyclic TBoxes can be decided in polynomial time [4].

In our definition of unification modulo acyclic TBoxes, we assume that all defined concepts are concept constants. In fact, defined concepts already have a definition in the given TBox, and thus it does not make sense to introduce new ones for them by unification. In this setting, a *substitution* σ is a mapping from N_v into the set of all $\mathcal{EL}$-concept terms *not containing any defined concepts*.[5] The extension of σ to concept terms is defined as in the previous subsection, and its application to $\mathcal{T}$ is defined as

$$\sigma(\mathcal{T}) := \{A \doteq \sigma(C) \mid A \doteq C \in \mathcal{T}\}.$$

Definition 2.4. An *$\mathcal{EL}$-unification problem modulo an acyclic TBox* is of the form $\Gamma = \{C_1 \equiv^?_{\mathcal{T}} D_1, \ldots, C_n \equiv^?_{\mathcal{T}} D_n\}$, where $C_1, D_1, \ldots, C_n, D_n$ are $\mathcal{EL}$-concept terms, and $\mathcal{T}$ is an acyclic $\mathcal{EL}$-TBox. The substitution σ is a *unifier* (or *solution*) of Γ *modulo* $\mathcal{T}$ iff $\sigma(C_i) \equiv_{\sigma(\mathcal{T})} \sigma(D_i)$ for $i = 1, \ldots, n$. In this case, Γ is called *solvable modulo* $\mathcal{T}$ or *unifiable modulo* $\mathcal{T}$.

Coming back to our example from the introduction, assume that one knowledge engineer has written the concept definition

$$\mathsf{Real\_man} \doteq \mathsf{Human} \sqcap \mathsf{Male} \sqcap \exists\, \mathsf{loves.Sports\_car}.$$

[5]This restriction prevents the unifier from introducing cycles into the TBox.

to the TBox, whereas a second one has written the definition

$$\mathsf{Stupid\_man} \doteq \mathsf{Man} \sqcap \exists\, \mathsf{loves}.(\mathsf{Car} \sqcap \mathsf{Fast}),$$

where all the concept names occurring on the left-hand side of these definitions are primitive concepts. Then the substitution that replaces $\mathsf{Sports\_car}$ by $\mathsf{Car} \sqcap \mathsf{Fast}$ and Man by $\mathsf{Human} \sqcap \mathsf{Male}$ is a unifier of $\{\mathsf{Real\_man} \equiv^?_{\mathcal{T}} \mathsf{Stupid\_man}\}$ w.r.t. the TBox $\mathcal{T}$ consisting of these two definitions.

Using expansion, we can reduce unification modulo an acyclic TBox to unification without a TBox. In fact, the following lemma is an easy consequence of the fact that $\sigma(C^{\mathcal{T}}) = \sigma(C)^{\sigma(\mathcal{T})}$ holds for all $\mathcal{EL}$-concept terms C.

Lemma 2.5. *The substitution σ is a unifier of $\{C_1 \equiv^?_{\mathcal{T}} D_1, \ldots, C_n \equiv^?_{\mathcal{T}} D_n\}$ modulo $\mathcal{T}$ iff it is a unifier of $\{C_1^{\mathcal{T}} \equiv^? D_1^{\mathcal{T}}, \ldots, C_n^{\mathcal{T}} \equiv^? D_n^{\mathcal{T}}\}$.*

Since expansion can cause an exponential blow-up, this is not a polynomial reduction. In the remainder of this subsection, we show that there actually exists a polynomial-time reduction of unification modulo an acyclic TBox to unification without a TBox.

We say that the $\mathcal{EL}$-unification problem Γ is in *dag-solved form* if it can be written as $\Gamma = \{X_1 \equiv^? C_1, \ldots, X_n \equiv^? C_n\}$, where $X_1, \ldots, X_n$ are distinct concept variables such that, for all $i \leq n$, X_i does not occur in $C_i, \ldots, C_n$. For $i = 1, \ldots, n$, let σ_i be the substitution that maps X_i to C_i and leaves all other variables unchanged. We define the substitution σ_Γ as

$$\sigma_\Gamma(X_i) := \sigma_n(\cdots(\sigma_i(X_i))\cdots)$$

for $i = 1, \ldots, n$, and $\sigma_\Gamma(X) := X$ for all other variables X. The following is an instance of a well-known fact from unification theory [22].

Lemma 2.6. *Let $\Gamma = \{X_1 \equiv^? C_1, \ldots, X_n \equiv^? C_n\}$ be an $\mathcal{EL}$-unification problem in dag-solved form. Then, the set $\{\sigma_\Gamma\}$ is a complete set of unifiers for Γ.*

There is a close relationship between acyclic TBoxes and unification problems in dag-solved form. In fact, if $\mathcal{T}$ is an acyclic TBox, then there is an enumeration $A_1, \ldots, A_n$ of the defined concepts in $\mathcal{T}$ such that $\mathcal{T} = \{A_1 \doteq C_1, \ldots, A_n \doteq C_n\}$ and A_i does not occur in $C_i, \ldots, C_n$. Consequently, the corresponding unification problem

$$\Gamma(\mathcal{T}) := \{A_1 \equiv^? C_1, \ldots, A_n \equiv^? C_n\}$$

(where $A_1, \ldots, A_n$ are now viewed as concept variables) is in dag-solved form. In addition, it is easy to see that, for any $\mathcal{EL}$-concept term C, we have $C^{\mathcal{T}} = \sigma_{\Gamma(\mathcal{T})}(C)$.

Lemma 2.7. *The $\mathcal{EL}$-unification problem $\Gamma = \{C_1 \equiv^?_{\mathcal{T}} D_1, \ldots, C_n \equiv^?_{\mathcal{T}} D_n\}$ is solvable modulo the acyclic TBox $\mathcal{T}$ iff $\{C_1 \equiv^? D_1, \ldots, C_n \equiv^? D_n\} \cup \Gamma(\mathcal{T})$ is solvable.*[6]

Proof. Assume that θ is a unifier of $\Gamma = \{C_1 \equiv^?_{\mathcal{T}} D_1, \ldots, C_n \equiv^?_{\mathcal{T}} D_n\}$ modulo $\mathcal{T}$. Then it is a unifier of $\widehat{\Gamma} := \{C_1^{\mathcal{T}} \equiv^? D_1^{\mathcal{T}}, \ldots, C_n^{\mathcal{T}} \equiv^? D_n^{\mathcal{T}}\}$, by Lemma 2.5. Since $C_i^{\mathcal{T}} = \sigma_{\Gamma(\mathcal{T})}(C_i)$ and $D_i^{\mathcal{T}} = \sigma_{\Gamma(\mathcal{T})}(D_i)$, we have $\widehat{\Gamma} = \{\sigma_{\Gamma(\mathcal{T})}(C_i) \equiv^? \sigma_{\Gamma(\mathcal{T})}(D_i) \mid 1 \leq i \leq n\}$. Consequently, if we define the substitution τ by setting $\tau(X) := \theta(\sigma_{\Gamma(\mathcal{T})}(X))$ for all concept variables and defined concepts X, then τ is a unifier of $\{C_1 \equiv^? D_1, \ldots, C_n \equiv^? D_n\}$. In addition, since $\sigma_{\Gamma(\mathcal{T})}$ is a unifier of $\Gamma(\mathcal{T})$, τ is also a unifier of $\{C_1 \equiv^? D_1, \ldots, C_n \equiv^? D_n\} \cup \Gamma(\mathcal{T})$.

[6]Note that the defined concepts of $\mathcal{T}$ are treated as concept constants in Γ, and as concept variables in $\{C_1 \equiv^? D_1, \ldots, C_n \equiv^? D_n\} \cup \Gamma(\mathcal{T})$.

Conversely, assume that τ is a unifier of $\{C_1 \equiv^? D_1, \ldots, C_n \equiv^? D_n\} \cup \Gamma(\mathcal{T})$. In particular, this implies that τ is a unifier of $\Gamma(\mathcal{T})$. By Lemma 2.6, $\{\sigma_{\Gamma(\mathcal{T})}\}$ is a complete set of unifiers for $\Gamma(\mathcal{T})$, and thus there is a substitution θ such that $\tau(X) = \theta(\sigma_{\Gamma(\mathcal{T})}(X))$ for all concept variables occurring in the unification problem $\{C_1 \equiv^? D_1, \ldots, C_n \equiv^? D_n\} \cup \Gamma(\mathcal{T})$. Since $C_i^{\mathcal{T}} = \sigma_{\Gamma(\mathcal{T})}(C_i)$ and $D_i^{\mathcal{T}} = \sigma_{\Gamma(\mathcal{T})}(D_i)$, this implies that θ is a unifier of $\{C_1^{\mathcal{T}} \equiv^? D_1^{\mathcal{T}}, \ldots, C_n^{\mathcal{T}} \equiv^? D_n^{\mathcal{T}}\}$, and thus of $\Gamma = \{C_1 \equiv^?_{\mathcal{T}} D_1, \ldots, C_n \equiv^?_{\mathcal{T}} D_n\}$ modulo $\mathcal{T}$, by Lemma 2.5. □

Since the size of $\Gamma(\mathcal{T})$ is basically the same as the size of $\mathcal{T}$, the size of $\Gamma \cup \Gamma(\mathcal{T})$ is linear in the size of Γ and $\mathcal{T}$. Thus, the above lemma provides us with a polynomial-time reduction of $\mathcal{EL}$-unification w.r.t. acyclic TBoxes to $\mathcal{EL}$-unification.

Theorem 2.8. *$\mathcal{EL}$-unification w.r.t. acyclic TBoxes can be reduced in polynomial time to $\mathcal{EL}$-unification.*

3. Equivalence and subsumption in $\mathcal{EL}$

In order to characterize equivalence of $\mathcal{EL}$-concept terms, the notion of a reduced $\mathcal{EL}$-concept term is introduced in [24]. A given $\mathcal{EL}$-concept term can be transformed into an equivalent reduced term by applying the following rules modulo associativity and commutativity of conjunction:

$C \sqcap \top \to C$	for all $\mathcal{EL}$-concept terms C
$A \sqcap A \to A$	for all concept names $A \in N_{con}$
$\exists r.C \sqcap \exists r.D \to \exists r.C$	for all $\mathcal{EL}$-concept terms C, D with $C \sqsubseteq D$

Obviously, these rules are equivalence preserving. We say that the $\mathcal{EL}$-concept term D *is reduced* if none of the above rules is applicable to it (modulo associativity and commutativity of $\sqcap$), and that C *can be reduced to* D if D can be obtained from C by applying the above rules (modulo associativity and commutativity of $\sqcap$). The $\mathcal{EL}$-concept term D is a *reduced form* of C if C can be reduced to D and D is reduced. The following theorem is an easy consequence of Theorem 6.3.1 on page 181 of [24].

Theorem 3.1. *Let C, D be $\mathcal{EL}$-concept terms, and $\widehat{C}, \widehat{D}$ reduced forms of C, D, respectively. Then $C \equiv D$ iff $\widehat{C}$ is identical to $\widehat{D}$ up to associativity and commutativity of $\sqcap$.*

This theorem can also be used to derive a recursive characterization of subsumption in $\mathcal{EL}$. In fact, if $C \sqsubseteq D$, then $C \sqcap D \equiv C$, and thus C and $C \sqcap D$ have the same reduced form. Thus, during reduction, all concept names and existential restrictions of D must be "eaten up" by corresponding concept names and existential restrictions of C.

Corollary 3.2. *Let $C = A_1 \sqcap \ldots \sqcap A_k \sqcap \exists r_1.C_1 \sqcap \ldots \sqcap \exists r_m.C_m$ and $D = B_1 \sqcap \ldots \sqcap B_\ell \sqcap \exists s_1.D_1 \sqcap \ldots \sqcap \exists s_n.D_n$, where $A_1, \ldots, A_k, B_1, \ldots, B_\ell$ are concept names. Then $C \sqsubseteq D$ iff $\{B_1, \ldots, B_\ell\} \subseteq \{A_1, \ldots, A_k\}$ and for every $j, 1 \leq j \leq n$, there exists an $i, 1 \leq i \leq m$, such that $r_i = s_j$ and $C_i \sqsubseteq D_j$.*

Note that this corollary also covers the cases where some of the numbers k, ℓ, m, n are zero. The empty conjunction should then be read as $\top$. The following lemma, which is an immediate consequence of this corollary, will be used in our proof that $\mathcal{EL}$ has unification type zero.

Lemma 3.3. *If C, D are reduced $\mathcal{EL}$-concept terms such that $\exists r.D \sqsubseteq C$, then C is either $\top$, or of the form $C = \exists r.C_1 \sqcap \ldots \sqcap \exists r.C_n$ where $n \geq 1$; $C_1, \ldots, C_n$ are reduced and pairwise incomparable w.r.t. subsumption; and $D \sqsubseteq C_1, \ldots, D \sqsubseteq C_n$. Conversely, if C, D are $\mathcal{EL}$-concept terms such that $C = \exists r.C_1 \sqcap \ldots \sqcap \exists r.C_n$ and $D \sqsubseteq C_1, \ldots, D \sqsubseteq C_n$, then $\exists r.D \sqsubseteq C$.*

The following lemma states several other obvious consequences of Corollary 3.2.

Lemma 3.4.

(1) *The existential restriction $\exists r.C$ is reduced iff C is reduced.*
(2) *Let $C_1 \sqcap \ldots \sqcap C_n$ be concept names or existential restrictions. Then the conjunction $C_1 \sqcap \ldots \sqcap C_n$ is reduced iff $C_1, \ldots, C_n$ are reduced and pairwise incomparable w.r.t. subsumption.*
(3) *Let $C = C_1 \sqcap \ldots \sqcap C_m$ and $D = D_1 \sqcap \ldots \sqcap D_n$ be conjunctions of $\mathcal{EL}$-concept terms. If, for all $i, 1 \leq i \leq n$, there exists $j, 1 \leq j \leq m$, such that $C_j \sqsubseteq D_i$, then $C \sqsubseteq D$. If $D_1, \ldots, D_n$ are concept names or existential restrictions, then the implication in the other direction also holds.*

In the proof of decidability of $\mathcal{EL}$-unification, we will make use of the fact that the inverse strict subsumption order is well-founded.

Proposition 3.5. *There is no infinite sequence $C_0, C_1, C_2, C_3, \ldots$ of $\mathcal{EL}$-concept terms such that $C_0 \sqsubset C_1 \sqsubset C_2 \sqsubset C_3 \sqsubset \cdots$.*

Proof. We define the *role depth* of an $\mathcal{EL}$-concept term C as the maximal nesting of existential restrictions in C. Let n_0 be the role depth of C_0. Since $C_0 \sqsubseteq C_i$ for $i \geq 1$, it is an easy consequence of Corollary 3.2 that the role depth of C_i is bounded by n_0, and that C_i contains only concept and role names occurring in C_0. In addition, it is known that, for a given natural number n_0 and finite sets of concept names N_{con} and role names N_{role}, there are, up to equivalence, only finitely many $\mathcal{EL}$-concept terms built using concept names from $\mathcal{C}$ and role names from $\mathcal{R}$ and of a role depth bounded by n_0 [15]. Consequently, there are indices $i < j$ such that $C_i \equiv C_j$. This contradicts our assumption that $C_i \sqsubset C_j$. □

4. An $\mathcal{EL}$-unification problem of type zero

To show that $\mathcal{EL}$ has unification type zero, we exhibit an $\mathcal{EL}$-unification problem that has this type.

Theorem 4.1. *Let X, Y be variables. The $\mathcal{EL}$-unification problem $\Gamma := \{X \sqcap \exists r.Y \equiv^? \exists r.Y\}$ has unification type zero.*

Proof. It is enough to show that any complete set of unifiers for this problem is redundant, i.e., contains two different unifiers that are comparable w.r.t. the instantiation preorder. Thus, let M be a complete set of unifiers for Γ.

First, note that M must contain a unifier that maps X to an $\mathcal{EL}$-concept term not equivalent to $\top$ or $\exists r.\top$. In fact, consider a substitution τ such that $\tau(X) = \exists r.A$ and $\tau(Y) = A$. Obviously, τ is a unifier of Γ. Thus, M must contain a unifier σ such that $\sigma \preceq \tau$. In particular, this means that there is a substitution λ such that $\exists r.A = \tau(X) \equiv \lambda(\sigma(X))$. Obviously, $\sigma(X) \equiv \top$ would imply $\lambda(\sigma(X)) \equiv \top$, and thus $\exists r.A \equiv \top$, which is, however,

not the case. Similarly, $\sigma(X) \equiv \exists r.\top$ would imply $\lambda(\sigma(X)) \equiv \exists r.\top$, and thus $\exists r.A \equiv \exists r.\top$, which is also not the case.

Thus, let $\sigma \in M$ be such that $\sigma(X) \not\equiv \top$ and $\sigma(X) \not\equiv \exists r.\top$. Without loss of generality, we assume that $C := \sigma(X)$ and $D := \sigma(Y)$ are reduced. Since σ is a unifier of Γ, we have $\exists r.D \sqsubseteq C$. Consequently, Lemma 3.3 yields that C is of the form $C = \exists r.C_1 \sqcap \ldots \sqcap \exists r.C_n$ where $n \geq 1$, $C_1, \ldots, C_n$ are reduced and pairwise incomparable w.r.t. subsumption, and $D \sqsubseteq C_1, \ldots, D \sqsubseteq C_n$.

We use σ to construct a new unifier $\widehat{\sigma}$ as follows:

$$\begin{aligned} \widehat{\sigma}(X) &:= \exists r.C_1 \sqcap \ldots \sqcap \exists r.C_n \sqcap \exists r.Z \\ \widehat{\sigma}(Y) &:= D \sqcap Z \end{aligned}$$

where Z is a new variable (i.e., one not occurring in C, D). The second part of Lemma 3.3 implies that $\widehat{\sigma}$ is indeed a unifier of Γ.

Next, we show that $\widehat{\sigma} \preceq \sigma$. To this purpose, we consider the substitution λ that maps Z to C_1, and does not change any of the other variables. Then we have $\lambda(\widehat{\sigma}(X)) = \exists r.C_1 \sqcap \ldots \sqcap \exists r.C_n \sqcap \exists r.C_1 \equiv \exists r.C_1 \sqcap \ldots \sqcap \exists r.C_n = \sigma(X)$ and $\lambda(\widehat{\sigma}(Y)) = D \sqcap C_1 \equiv D = \sigma(Y)$. Note that the second equivalence holds since we have $D \sqsubseteq C_1$.

Since M is complete, there exists a unifier $\theta \in M$ such that $\theta \preceq \widehat{\sigma}$. Transitivity of the relation $\preceq$ thus yields $\theta \preceq \sigma$. Since σ and θ both belong to M, we have completed the proof of the theorem once we have shown that $\sigma \neq \theta$. Assume to the contrary that $\sigma = \theta$. Then we have $\sigma \preceq \widehat{\sigma}$, and thus there exists a substitution μ such that $\mu(\sigma(X)) \equiv \widehat{\sigma}(X)$, i.e.,

$$\exists r.\mu(C_1) \sqcap \ldots \sqcap \exists r.\mu(C_n) \equiv \exists r.C_1 \sqcap \ldots \sqcap \exists r.C_n \sqcap \exists r.Z. \tag{4.1}$$

Recall that the concept terms $C_1, \ldots, C_n$ are reduced and pairwise incomparable w.r.t. subsumption. In addition, since $\sigma(X) = \exists r.C_1 \sqcap \ldots \sqcap \exists r.C_n$ is reduced and not equivalent to $\exists r.\top$, none of the concept terms $C_1, \ldots, C_n$ can be equivalent to $\top$. Finally, Z is a concept name that does not occur in $C_1, \ldots, C_n$. All this implies that $\exists r.C_1 \sqcap \ldots \sqcap \exists r.C_n \sqcap \exists r.Z$ is reduced. Obviously, any reduced form for $\exists r.\mu(C_1) \sqcap \ldots \sqcap \exists r.\mu(C_n)$ is a conjunction of at most n existential restrictions. Thus, Theorem 3.1 shows that the above equivalence (4.1) actually cannot hold.

To sum up, we have shown that M contains two distinct unifiers σ, θ such that $\theta \preceq \sigma$. Since M was an arbitrary complete set of unifiers for Γ, this shows that this unification problem cannot have a minimal complete set of unifiers. □

5. The decision problem

Before we can describe our decision procedure for $\mathcal{EL}$-unification, we must introduce some notation. An $\mathcal{EL}$-concept term is called an *atom* iff it is a concept name (i.e., concept constant or concept variable) or an existential restriction $\exists r.D$.[7] Obviously, any $\mathcal{EL}$-concept term is (equivalent to) a conjunction of atoms, where the empty conjunction is $\top$. The set $At(C)$ of *atoms of an $\mathcal{EL}$-concept term C* is defined inductively: if $C = \top$, then $At(C) := \emptyset$; if C is a concept name, then $At(C) := \{C\}$; if $C = \exists r.D$ then $At(C) := \{C\} \cup At(D)$; if $C = C_1 \sqcap C_2$, then $At(C) := At(C_1) \cup At(C_2)$.

Concept names and existential restrictions $\exists r.D$ where D is a concept name or $\top$ are called *flat atoms*. An $\mathcal{EL}$-concept term is *flat* iff it is a conjunction of flat atoms (where the

[7] Note that $\top$ is *not* an atom.

empty conjunction is $\top$). The $\mathcal{EL}$-unification problem Γ is *flat* iff it consists of equations between flat $\mathcal{EL}$-concept terms. By introducing new concept variables and eliminating $\top$, any $\mathcal{EL}$-unification problem Γ can be transformed in polynomial time into a flat $\mathcal{EL}$-unification problem Γ' such that Γ is solvable iff Γ' is solvable. Thus, we may assume without loss of generality that our input $\mathcal{EL}$-unification problems are flat. Given a flat $\mathcal{EL}$-unification problem $\Gamma = \{C_1 \equiv^? D_1, \ldots, C_n \equiv^? D_n\}$, we call the atoms of $C_1, D_1, \ldots, C_n, D_n$ the *atoms of* Γ. Atoms of Γ that are not variables (i.e., not elements of N_v) are called *non-variable atoms of* Γ.

The unifier σ of Γ is called *reduced* iff, for all concept variables X occurring in Γ, the $\mathcal{EL}$-concept term $\sigma(X)$ is reduced. It is *ground* iff, for all concept variables X occurring in Γ, the $\mathcal{EL}$-concept term $\sigma(X)$ does not contain variables. Obviously, Γ is solvable iff it has a reduced ground unifier. Given a ground unifier σ of Γ, the *atoms of* σ are the atoms of all the concept terms $\sigma(X)$, where X ranges over all variables occurring in Γ.

Remark 5.1. In the following, we consider situations where all occurrences of a given reduced atom D in a reduced concept term C are replaced by a more general concept term, i.e., by a concept term D' with $D \sqsubset D'$. However, when we say *occurrence of D in C*, we mean occurrence modulo equivalence ($\equiv$) rather than syntactic occurrence. For example, if $C = \exists r.(A \sqcap B) \sqcap \exists r.(B \sqcap A)$, $D = \exists r.(A \sqcap B)$, and $D' = \exists r.A$, then the term obtained by replacing all occurrences of D in C by D' should be $\exists r.A \sqcap \exists r.A$, and not $\exists r.A \sqcap \exists r.(B \sqcap A)$. Since C and D are reduced, equivalence is actually the same as being identical up to associativity and commutativity of $\sqcap$. In particular, this means that any concept term that (syntactically) occurs in C and is equivalent to the atom D is also an atom, i.e., only atoms can be replaced by D'. In order to make this meaning of occurrence explicit we will call it *occurrence modulo AC* in the following. We will write $D_1 =_{AC} D_2$ to express that the atoms D_1 and D_2 are identical up to associativity and commutativity of $\sqcap$. Obviously, $D_1 =_{AC} D_2$ implies $D_1 \equiv D_2$.

Lemma 5.2. *Let C, D, D' be $\mathcal{EL}$-concept terms such that D is a reduced atom, $D \sqsubset D'$, and C is reduced and contains at least one occurrence of D modulo AC. If C' is obtained from C by replacing all occurrences of D by D', then $C \sqsubset C'$.*

Proof. We prove the lemma by induction on the size of C. If $C =_{AC} D$, then $C' = D'$, and thus $C \equiv D \sqsubset D' = C'$, which yields $C \sqsubset C'$. Thus, assume that $C \neq_{AC} D$. In this case, C cannot be a concept name since it contains the atom D. If $C = \exists r.C_1$, then D occurs in C_1 modulo AC. By induction, we can assume that $C_1 \sqsubset C_1'$, where C_1' is obtained from C_1 by replacing all occurrences of D (modulo AC) by D'. Thus, we have $C = \exists r.C_1 \sqsubset \exists r.C_1' = C'$ by Corollary 3.2. Finally, assume that $C = C_1 \sqcap \ldots \sqcap C_n$ for $n > 1$ atoms $C_1, \ldots, C_n$. Since C is reduced, these atoms are incomparable w.r.t. subsumption, and since the atom D occurs in C modulo AC we can assume without loss of generality that D occurs in C_1 modulo AC. Let $C_1', \ldots, C_n'$ be respectively obtained from $C_1, \ldots, C_n$ by replacing every occurrence of D (modulo AC) by D', and then reducing the concept term obtained this way. By induction, we have $C_1 \sqsubset C_1'$. Assume that $C \not\sqsubset C'$. Since the concept constructors of $\mathcal{EL}$ are monotone w.r.t. subsumption $\sqsubseteq$, we have $C \sqsubseteq C'$, and thus $C \not\sqsubset C'$ means that $C \equiv C'$. Consequently, $C = C_1 \sqcap \ldots \sqcap C_n$ and the reduced form of $C_1' \sqcap \ldots \sqcap C_n'$ must be equal up to associativity and commutativity of $\sqcap$. If $C_1' \sqcap \ldots \sqcap C_n'$ is not reduced, then its reduced form is actually a conjunction of $m < n$ atoms, which contradicts $C \equiv C'$. If $C_1' \sqcap \ldots \sqcap C_n'$ is reduced, then $C_1 \sqsubset C_1'$ implies that there is an $i \neq 1$ such that

$C_i \equiv C_1'$. However, then $C_i \equiv C_1' \sqsupset C_1$ contradicts the fact that the atoms $C_1, \ldots, C_n$ are incomparable w.r.t. subsumption. □

Proposition 3.5 says that the inverse strict subsumption order on concept terms is well-founded. We use this fact to obtain a well-founded strict order $\succ$ on ground unifiers.

Definition 5.3. Let σ, θ be ground unifiers of Γ. We define

(1) $\sigma \succeq \theta$ iff $\sigma(X) \sqsubseteq \theta(X)$ holds for all variables X occurring in Γ.
(2) $\sigma \succ \theta$ iff $\sigma \succeq \theta$ and $\theta \not\succeq \sigma$, i.e., iff $\sigma(X) \sqsubseteq \theta(X)$ holds for all variables X occurring in Γ, and $\sigma(X) \sqsubset \theta(X)$ holds for at least one variable X occurring in Γ.

If Γ contains n variables, then $\succeq$ is the n-fold product of the order $\sqsubseteq$ with itself. Since the strict part $\sqsubset$ of the inverse subsumption order $\sqsubseteq$ is well-founded by Proposition 3.5, the strict part $\succ$ of $\succeq$ is also well-founded [13]. The ground unifier σ of Γ is called *is-minimal* iff there is no ground unifier θ of Γ such that $\sigma \succ \theta$. The following proposition is an easy consequence of the fact that $\succ$ is well-founded.

Proposition 5.4. *Let Γ be an $\mathcal{EL}$-unification problem. Then Γ is solvable iff it has an is-minimal reduced ground unifier.*

In the following, we show that is-minimal reduced ground unifiers of flat $\mathcal{EL}$-unification problems satisfy properties that make it easy to check (with an NP-algorithm) whether such a unifier exists or not.

Lemma 5.5. *Let Γ be a flat $\mathcal{EL}$-unification problem and γ an is-minimal reduced ground unifier of Γ. If C is an atom of γ, then there is a non-variable atom D of Γ such that $C \equiv \gamma(D)$.*

The main idea underlying the proof of this crucial lemma is that an atom C of a unifier σ that violates the condition of the lemma (i.e., that is not of the form $C \equiv \gamma(D)$ for a non-variable atom D of Γ) can be replaced by a concept term $\widehat{D}$ such that $C \sqsubset \widehat{D}$, which yields a unifier of Γ that is smaller than σ w.r.t. $\succ$.

Before proving the lemma formally, let us illustrate this idea by two examples.

Example 5.6. First, consider the unification problem

$$\Gamma_1 := \{\exists r.X \sqcap \exists r.A \equiv^? \exists r.X\}.$$

The substitution $\sigma_1 := \{X \mapsto A \sqcap B\}$ is a unifier of Γ_1 that does not satisfy the condition of Lemma 5.5. In fact, B is an atom of σ_1, but none of the non-variable atoms D of Γ_1 (which are A, $\exists r.A$, and $\exists r.X$) satisfy $B \equiv \sigma_1(D)$. The unifier σ_1 is not is-minimal since $\gamma_1 := \{X \mapsto A\}$, which can be obtained from σ_1 by replacing the offending atom B with $\top$, is a unifier of Γ_1 that is smaller than σ_1 w.r.t. $\succ$. The unifier γ_1 is is-minimal, and it clearly satisfies the condition of Lemma 5.5.

Second, consider the unification problem

$$\Gamma_2 := \{X \sqcap \exists r.A \sqcap \exists r.B \equiv^? X\}.$$

The substitution $\sigma_2 := \{X \mapsto \exists r.(A \sqcap B)\}$ is a unifier of Γ_2 that does not satisfy the condition of Lemma 5.5. In fact, $\exists r.(A \sqcap B)$ is an atom of σ_2, but none of the non-variable atoms D of Γ_2 (which are A, B, $\exists r.A$, and $\exists r.B$) satisfy $\exists r.(A \sqcap B) \equiv \sigma_2(D)$. The unifier σ_2 is not is-minimal since $\gamma_2 := \{X \mapsto \exists r.A \sqcap \exists r.B\}$, which can be obtained from σ_2 by replacing the offending atom $\exists r.(A \sqcap B)$ with $\exists r.A \sqcap \exists r.B$, is a unifier of Γ_2 that is

smaller than σ_2 w.r.t. $\succ$. The unifier γ_2 is is-minimal, and it clearly satisfies the condition of Lemma 5.5.

Proof of Lemma 5.5. Assume that γ is an is-minimal reduced ground unifier of Γ. Since γ is reduced, all atoms of γ are reduced. In particular, this implies that C is reduced, and since γ is ground, we know that C is either a concept constant or an existential restriction.

First, assume that C *is of the form* A *for a concept constant* A, but there is no non-variable atom D of Γ such that $A \equiv \gamma(D)$. This simply means that A does not appear in Γ. Let γ' be the substitution obtained from γ by replacing every occurrence of A by $\top$. Since equivalence in $\mathcal{EL}$ is preserved under replacing concept names by $\top$, and since A does not appear in Γ, it is easy to see that γ' is also a unifier of Γ. However, since $\gamma \succ \gamma'$, this contradicts our assumption that γ is is-minimal.

Second, assume that C *is an existential restriction of the form* $\exists r.C_1$, but there is no non-variable atom D of Γ such that $C \equiv \gamma(D)$. We assume that C is maximal (w.r.t. subsumption) with this property, i.e., for every atom C' of γ with $C \sqsubset C'$, there is a non-variable atom D' of Γ such that $C' \equiv \gamma(D')$. Let $D_1, \ldots, D_\ell$ be all the non-variable atoms of Γ with $C \sqsubseteq \gamma(D_i)$ $(i = 1, \ldots, \ell)$. By our assumptions on C, we actually have $C \sqsubset \gamma(D_i)$ and, by Lemma 3.3, the atom D_i is also an existential restriction $D_i = \exists r.D_i'$ $(i = 1, \ldots, \ell)$. We consider the conjunction

$$\widehat{D} := \gamma(D_1) \sqcap \ldots \sqcap \gamma(D_\ell),$$

which is $\top$ in case $\ell = 0$.

Definition 5.7. Given an $\mathcal{EL}$-concept term F, the concept term $F^{[C/\widehat{D}]}$ is obtained from F by replacing every occurrence of C (modulo AC) by $\widehat{D}$. The substitution $\gamma^{[C/\widehat{D}]}$ is obtained from γ by replacing every occurrence of C (modulo AC) by $\widehat{D}$, i.e., $\gamma^{[C/\widehat{D}]}(X) := \gamma(X)^{[C/\widehat{D}]}$ for all variables X.

We will show in the following that $\gamma^{[C/\widehat{D}]}$ is a unifier of Γ that is smaller than γ w.r.t. $\succ$. This will then again contradict our assumption that γ is is-minimal.

Lemma 5.8. $\gamma \succ \gamma^{[C/\widehat{D}]}$.

Proof. Obviously, $\widehat{D}$ subsumes C. We claim that this subsumption relationship is actually strict. In fact, if $\ell = 0$, then $\widehat{D} = \top$, and since C is an atom, it is not equivalent to $\top$. If $\ell \geq 1$, then $C = \exists r.C_1 \sqsupseteq \exists r.\gamma(D_1') \sqcap \ldots \sqcap \exists r.\gamma(D_\ell')$ would imply (by Corollary 3.2) that there is an $i, 1 \leq i \leq \ell$, with $C_1 \sqsupseteq \gamma(D_i')$. However, this would yield $C = \exists r.C_1 \sqsupseteq \exists r.\gamma(D_i') = \gamma(D_i)$, which contradicts the fact that $C \sqsubset \gamma(D_i)$. Thus, we have shown that $C \sqsubset \widehat{D}$. Lemma 5.2 implies that $\gamma \succ \gamma'$. □

To complete the proof of Lemma 5.5, it remains to show the next lemma.

Lemma 5.9. $\gamma^{[C/\widehat{D}]}$ *is a unifier of* Γ.

Proof. Consider an equation in Γ of the form $L_1 \sqcap \ldots \sqcap L_m \equiv^? R_1 \sqcap \ldots \sqcap R_n$ where $L_1, \ldots, L_m$ and $R_1, \ldots, R_n$ are flat atoms, and define $L := \gamma(L_1 \sqcap \ldots \sqcap L_m)$ and $R := \gamma(R_1 \sqcap \ldots \sqcap R_n)$. We know that L, R are conjunctions of atoms of the form $L = A_1 \sqcap \ldots \sqcap A_\mu$ and $R = B_1 \sqcap \ldots \sqcap B_\nu$, where each conjunct $A_1, \ldots, A_\mu, B_1, \ldots, B_\nu$ is a reduced ground atom that is either an atom of γ or equal to $\gamma(E)$ for a non-variable atom E of Γ. Since γ is a unifier of Γ, we have $L \equiv R$.

(1) Since C is an atom, we obviously have $L^{[C/\widehat{D}]} = A_1^{[C/\widehat{D}]} \sqcap \ldots \sqcap A_\mu^{[C/\widehat{D}]}$ and $R^{[C/\widehat{D}]} = B_1^{[C/\widehat{D}]} \sqcap \ldots \sqcap B_\nu^{[C/\widehat{D}]}$. Now, we show that $L^{[C/\widehat{D}]} = \gamma^{[C/\widehat{D}]}(L_1 \sqcap \ldots \sqcap L_m)$ and $R^{[C/\widehat{D}]} = \gamma^{[C/\widehat{D}]}(R_1 \sqcap \ldots \sqcap R_n)$. We concentrate on proving the first identity since the second one can be shown analogously. To show the first identity, it is enough to prove that $\gamma(L_j)^{[C/\widehat{D}]} = \gamma^{[C/\widehat{D}]}(L_j)$ holds for all $j, 1 \leq j \leq m$.
   (a) If L_j is a variable X, then $\gamma^{[C/\widehat{D}]}(X) = \gamma(X)^{[C/\widehat{D}]}$ holds by the definition of $\gamma^{[C/\widehat{D}]}$.
   (b) If L_j is a concept constant A, then $A^{[C/\widehat{D}]} = A$ since C is an existential restriction. Thus, we have $\gamma^{[C/\widehat{D}]}(A) = A = A^{[C/\widehat{D}]} = \gamma(A)^{[C/\widehat{D}]}$.
   (c) Otherwise, L_j is an existential restriction $\exists\, r_j.L'_j$. By our assumption on C, we have $C \not\equiv \gamma(L_j)$, and thus $\gamma(L_j)^{[C/\widehat{D}]} = \exists\, r_j.\left(\gamma(L_j)^{[C/\widehat{D}]}\right)$. In addition, we have $\gamma^{[C/\widehat{D}]}(L_j) = \exists\, r_j.\gamma^{[C/\widehat{D}]}(L'_j)$. Thus, it is enough to show $\gamma(L'_j)^{[C/\widehat{D}]} = \gamma^{[C/\widehat{D}]}(L'_j)$. Since L_j is a flat atom, we know that L'_j is either a concept constant, the top-concept $\top$, or a concept variable. In the first to cases, we can show $\gamma(L'_j)^{[C/\widehat{D}]} = \gamma^{[C/\widehat{D}]}(L'_j)$ as in (1b), and in the third case we can show this identity as in (1a).

(2) Because of (1), if we can prove that $L^{[C/\widehat{D}]} \equiv R^{[C/\widehat{D}]}$, then we have shown that $\gamma^{[C/\widehat{D}]}$ solves the equation $L_1 \sqcap \ldots \sqcap L_m \equiv^? R_1 \sqcap \ldots \sqcap R_n$.

   Without loss of generality, we concentrate on showing that $L^{[C/\widehat{D}]} \sqsubseteq R^{[C/\widehat{D}]}$. Since $L^{[C/\widehat{D}]} = A_1^{[C/\widehat{D}]} \sqcap \ldots \sqcap A_\mu^{[C/\widehat{D}]}$ and $R^{[C/\widehat{D}]} = B_1^{[C/\widehat{D}]} \sqcap \ldots \sqcap B_\nu^{[C/\widehat{D}]}$, it is thus sufficient to show that, for every $i, 1 \leq i \leq \nu$, there exists a $j, 1 \leq j \leq \mu$, such that $A_j^{[C/\widehat{D}]} \sqsubseteq B_i^{[C/\widehat{D}]}$ (see (3) of Lemma 3.4). Since $L = A_1 \sqcap \ldots \sqcap A_\mu \sqsubseteq B_1 \sqcap \ldots \sqcap B_\nu = R$ and $A_1, \ldots, A_\mu, B_1, \ldots, B_\nu$ are atoms, we actually know that, for every $i, 1 \leq i \leq \nu$, there exists a $j, 1 \leq j \leq \mu$, such that $A_j \sqsubseteq B_i$. Thus, it is sufficient to show that $A_j \sqsubseteq B_i$ implies $A_j^{[C/\widehat{D}]} \sqsubseteq B_i^{[C/\widehat{D}]}$. This is an easy consequence of the next lemma since A_i, B_j satisfy the conditions of this lemma. □

Lemma 5.10. *Let A, B be reduced ground atoms such that B is an atom of γ or of the form $\gamma(D)$ for a non-variable atom D of Γ. If $A \sqsubseteq B$, then $A^{[C/\widehat{D}]} \sqsubseteq B^{[C/\widehat{D}]}$.*

Proof. We show $A^{[C/\widehat{D}]} \sqsubseteq B^{[C/\widehat{D}]}$ by induction on the size of A.

(1) First, assume that $A =_{AC} C$, which implies that $A^{[C/\widehat{D}]} = \widehat{D} = \gamma(D_1) \sqcap \ldots \sqcap \gamma(D_n)$.
   (a) If B is of the form $B \equiv \gamma(D)$ for a non-variable atom D of Γ, then there is an $h, 1 \leq h \leq n$, such that $D = D_h$, which shows that $A^{[C/\widehat{D}]} \sqsubseteq B$. Since $C \sqsubseteq \widehat{D}$ and the constructors of $\mathcal{EL}$ are monotone w.r.t. subsumption, we also have $B \sqsubseteq B^{[C/\widehat{D}]}$, and thus $A^{[C/\widehat{D}]} \sqsubseteq B^{[C/\widehat{D}]}$.
   (b) Assume that B is an atom of γ. If $B =_{AC} C$, then $B^{[C/\widehat{D}]} = \widehat{D}$, and thus $A^{[C/\widehat{D}]} = B^{[C/\widehat{D}]}$, which implies $A^{[C/\widehat{D}]} \sqsubseteq B^{[C/\widehat{D}]}$. Otherwise, since C, B are reduced atoms, $B \neq_{AC} C$ implies $B \not\equiv C$. Together with $C \equiv A \sqsubseteq B$, this shows that $C \sqsubset B$. Thus, the maximality of C implies that there is a non-variable atom D of Γ such that $B \equiv \gamma(D)$. Thus, we are actually in case (a), which yields $A^{[C/\widehat{D}]} \sqsubseteq B^{[C/\widehat{D}]}$.

(2) Now, assume that $A \neq_{AC} C$. If there is no occurrence (modulo AC) of C in A, then we have $A^{[C/\widehat{D}]} = A \sqsubseteq B \sqsubseteq B^{[C/\widehat{D}]}$.

Otherwise, A is of the form $A = \exists\, s.E$ and C occurs in E (modulo AC). Obviously, $A \sqsubseteq B$ then implies that B is of the form $B = \exists\, s.F$ with $E \sqsubseteq F$. The concept terms E, F are conjunctions of reduced ground atoms, i.e., $E = E_1 \sqcap \ldots \sqcap E_\kappa$ and $F = F_1 \sqcap \ldots \sqcap F_\lambda$ where $E_1, \ldots, E_\kappa, F_1, \ldots, F_\lambda$ are reduced ground atoms. By Corollary 3.2, for every $h, 1 \leq h \leq \lambda$, there exists $k, 1 \leq k \leq \kappa$ such that $E_k \sqsubseteq F_h$.

In order to be able to assume, by induction, that $E_k \sqsubseteq F_h$ implies $E_k^{[C/\widehat{D}]} \sqsubseteq F_h^{[C/\widehat{D}]}$, we must show that the conditions in the statement of the lemma hold for the concept terms E_k, F_h, where E_k plays the rôle of A and F_h plays the rôle of B. Since we already know that $E_1, \ldots, E_\kappa, F_1, \ldots, F_\lambda$ are reduced ground atoms, it is sufficient to show that each of the atoms $F_1, \ldots, F_\lambda$ is an atom of γ or of the form $\gamma(D)$ for a non-variable atom D of Γ. We know that $B = \exists\, s.(F_1 \sqcap \ldots \sqcap F_\lambda)$ is an atom of γ or an instance (w.r.t. γ) of a non-variable atom of Γ. In the first case, the atoms $F_1, \ldots, F_\lambda$ are clearly also atoms of γ. In the second case, $B = \gamma(D')$ for a non-variable atom D' of Γ. If D' is a ground atom, then $F_1, \ldots, F_\lambda$ are also ground atoms that are atoms of Γ, and thus they are instances (w.r.t. γ) of non-variable atoms of Γ. Otherwise, since Γ is flat, D' is of the form $\exists\, s.X$ for a variable X and $\gamma(X) = F_1 \sqcap \ldots \sqcap F_\lambda$. In this case, $F_1, \ldots, F_\lambda$ are clearly atoms of γ.

Thus, we can assume by induction:

($*$) for every $h, 1 \leq h \leq \lambda$, there exists $k, 1 \leq k \leq \kappa$ such that $E_k^{[C/\widehat{D}]} \sqsubseteq F_h^{[C/\widehat{D}]}$

It remains to show that this implies $A^{[C/\widehat{D}]} \sqsubseteq B^{[C/\widehat{D}]}$.

(a) If $B \neq_{AC} C$, then $A^{[C/\widehat{D}]} = \exists\, s.(E_1^{[C/\widehat{D}]} \sqcap \ldots \sqcap E_\kappa^{[C/\widehat{D}]})$ and $B^{[C/\widehat{D}]} = \exists\, s.(F_1^{[C/\widehat{D}]} \sqcap \ldots \sqcap F_\lambda^{[C/\widehat{D}]})$, and thus property ($*$) yields $A^{[C/\widehat{D}]} \sqsubseteq B^{[C/\widehat{D}]}$.

(b) Assume that $B =_{AC} C$. In this case, C cannot occur (modulo AC) in any of the concept terms $F_1, \ldots, F_h$, which implies that $B = \exists\, s.(F_1 \sqcap \ldots \sqcap F_\lambda) = \exists\, s.(F_1^{[C/\widehat{D}]} \sqcap \ldots \sqcap F_\lambda^{[C/\widehat{D}]})$. Since we have $A^{[C/\widehat{D}]} = \exists\, s.(E_1^{[C/\widehat{D}]} \sqcap \ldots \sqcap E_\kappa^{[C/\widehat{D}]})$, property ($*$) yields $A^{[C/\widehat{D}]} \sqsubseteq B$. Since we also have $B \sqsubseteq B^{[C/\widehat{D}]}$, this yields $A^{[C/\widehat{D}]} \sqsubseteq B^{[C/\widehat{D}]}$.

Thus, we have shown in all cases that $A^{[C/\widehat{D}]} \sqsubseteq B^{[C/\widehat{D}]}$, which completes the proof of Lemma 5.10. □

Overall, we have thus completed the proof of Lemma 5.5. The next proposition is an easy consequence of this lemma.

Proposition 5.11. *Let Γ be a flat $\mathcal{EL}$-unification problem and γ an is-minimal reduced ground unifier of Γ. If X is a concept variable occurring in Γ, then $\gamma(X) \equiv \top$ or there are non-variable atoms $D_1, \ldots, D_n$ ($n \geq 1$) of Γ such that $\gamma(X) \equiv \gamma(D_1) \sqcap \ldots \sqcap \gamma(D_n)$.*

Proof. If $\gamma(X) \not\equiv \top$, then it is a non-empty conjunction of atoms, i.e., there are atoms $C_1, \ldots, C_n$ ($n \geq 1$) such that $\gamma(X) = C_1 \sqcap \ldots \sqcap C_n$. Then $C_1, \ldots, C_n$ are atoms of γ, and thus Lemma 5.5 yields non-variable atoms $D_1, \ldots, D_n$ of Γ such that $C_i \equiv \gamma(D_i)$ for $i = 1, \ldots n$. Consequently, $\gamma(X) \equiv \gamma(D_1) \sqcap \ldots \sqcap \gamma(D_n)$. □

This proposition suggests the following non-deterministic algorithm for deciding solvability of a given flat $\mathcal{EL}$-unification problem.

Algorithm 5.12. Let Γ be a flat $\mathcal{EL}$-unification problem.

(1) For every variable X occurring in Γ, guess a finite, possibly empty, set S_X of non-variable atoms of Γ.
(2) We say that the variable X *directly depends on* the variable Y if Y occurs in an atom of S_X. Let *depends on* be the transitive closure of *directly depends on*. If there is a variable that depends on itself, then the algorithm returns "fail." Otherwise, there exists a strict linear order $>$ on the variables occurring in Γ such that $X > Y$ if X depends on Y.
(3) We define the substitution σ along the linear order $>$:
   - If X is the least variable w.r.t. $>$, then S_X does not contain any variables. We define $\sigma(X)$ to be the conjunction of the elements of S_X, where the empty conjunction is $\top$.
   - Assume that $\sigma(Y)$ is defined for all variables $Y < X$. Then S_X only contains variables Y for which $\sigma(Y)$ is already defined. If S_X is empty, then we define $\sigma(X) := \top$. Otherwise, let $S_X = \{D_1, \dots, D_n\}$. We define $\sigma(X) := \sigma(D_1) \sqcap \dots \sqcap \sigma(D_n)$.
(4) Test whether the substitution σ computed in the previous step is a unifier of Γ. If this is the case, then return σ; otherwise, return "fail."

This algorithm is trivially *sound* since it only returns substitutions that are unifiers of Γ. In addition, it obviously always terminates. Thus, to show correctness of our algorithm, it is sufficient to show that it is complete.

Lemma 5.13 (Completeness)**.** *If Γ is solvable, then there is a way of guessing in Step 1 subsets S_X of the non-variable atoms of Γ such that the* depends on *relation determined in Step 2 is acyclic and the substitution σ computed in Step 3 is a unifier of Γ.*

Proof. If Γ is solvable, then it has an is-minimal reduced ground unifier γ. By Proposition 5.11, for every variable X occurring in Γ we have $\gamma(X) \equiv \top$ or there are non-variable atoms $D_1, \dots, D_n$ ($n \geq 1$) of Γ such that $\gamma(X) \equiv \gamma(D_1) \sqcap \dots \sqcap \gamma(D_n)$. If $\gamma(X) \equiv \top$, then we define $S_X := \emptyset$. Otherwise, we define $S_X := \{D_1, \dots, D_n\}$.

We show that the relation *depends on* induced by these sets S_X is acyclic, i.e., there is no variable X such that X depends on itself. If X directly depends on Y, then Y occurs in an element of S_X. Since S_X consists of non-variable atoms of the flat unification problem Γ, this means that there is a role name r such that $\exists r.Y \in S_X$. Consequently, we have $\gamma(X) \sqsubseteq \exists r.\gamma(Y)$. Thus, if X depends on X, then there are $k \geq 1$ role names $r_1, \dots, r_k$ such that $\gamma(X) \sqsubseteq \exists r_1. \cdots \exists r_k.\gamma(X)$. This is clearly not possible since $\gamma(X)$ cannot be subsumed by an $\mathcal{EL}$-concept term whose role depth is larger than the role depth of $\gamma(X)$.

To show that the substitution σ induced by the sets S_X is a unifier of Γ, we prove that σ is equivalent to γ, i.e., $\sigma(X) \equiv \gamma(X)$ holds for all variables X occurring in Γ. The substitution σ is defined along the linear order $>$. If X is the least variable w.r.t. $>$, then the elements of S_X do not contain any variables. If S_X is empty, then $\sigma(X) = \top \equiv \gamma(X)$. Otherwise, let $S_X = \{D_1, \dots, D_n\}$. Since the atoms D_i do not contain variables, we have $D_i = \gamma(D_i)$. Thus, the definitions of S_X and of σ yield $\sigma(X) = D_1 \sqcap \dots \sqcap D_n = \gamma(D_1) \sqcap \dots \sqcap \gamma(D_n) \equiv \gamma(X)$.

Assume that $\sigma(Y) \equiv \gamma(Y)$ holds for all variables $Y < X$. If $S_X = \emptyset$, then we have again $\sigma(X) = \top \equiv \gamma(X)$. Otherwise, let $S_X = \{D_1, \dots, D_n\}$. Since the atoms D_i contain only variables that are smaller than X, we have $\sigma(D_i) \equiv \gamma(D_i)$ by induction. Thus, the definitions of S_X and of σ yield $\sigma(X) = \sigma(D_1) \sqcap \dots \sqcap \sigma(D_n) \equiv \gamma(D_1) \sqcap \dots \sqcap \gamma(D_n) \equiv \gamma(X)$. □

Note that our proof of completeness actually shows that, up to equivalence, the algorithm returns all is-minimal reduced ground unifiers of Γ.

Theorem 5.14. *$\mathcal{EL}$-unification is NP-complete.*

Proof. NP-hardness follows from the fact that $\mathcal{EL}$-matching is NP-complete [24].[8] To show that the problem can be decided by a non-deterministic polynomial-time algorithm, we analyze the complexity of our algorithm. Obviously, guessing the sets S_X (Step 1) can be done within NP. Computing the *depends on* relation and checking it for acyclicity (Step 2) is clearly polynomial.

Steps 3 and 4 are more problematic. In fact, since a variable may occur in different atoms of Γ, the substitution σ computed in Step 3 may be of exponential size. This is actually the same reason that makes a naive algorithm for syntactic unification compute an exponentially large most general unifier [16]. As in the case of syntactic unification, the solution to this problem is basically structure sharing. Instead of computing the substitution σ explicitly, we view its definition as an acyclic TBox. To be more precise, for every concept variable X occurring in Γ, the TBox $\mathcal{T}_\sigma$ contains the concept definition $X \doteq \top$ if $S_X = \emptyset$ and $X \doteq D_1 \sqcap \ldots \sqcap D_n$ if $S_X = \{D_1, \ldots, D_n\}$ $(n \geq 1)$. Instead of computing σ in Step 3, we compute $\mathcal{T}_\sigma$. Because of the acyclicity test in Step 2, we know that $\mathcal{T}_\sigma$ is an acyclic TBox. The size of $\mathcal{T}_\sigma$ is obviously polynomial in the size of Γ, and thus this modified Step 3 is polynomial.

It is easy to see that applying the substitution σ to a concept term C is the same as expanding C w.r.t. the TBox $\mathcal{T}_\sigma$, i.e., $\sigma(C) = C^{\mathcal{T}_\sigma}$. This implies that, for every equation $C \equiv^? D$ in Γ, we have $C \equiv_{\mathcal{T}_\sigma} D$ iff $\sigma(C) \equiv \sigma(D)$. Thus, testing in Step 4 whether σ is a unifier of Γ can be reduced to testing whether $C \equiv_{\mathcal{T}_\sigma} D$ holds for every equation $C \equiv^? D$ in Γ. Since subsumption (and thus equivalence) in $\mathcal{EL}$ w.r.t. acyclic TBoxes can be decided in polynomial time [4], this completes the proof of the theorem. □

In Subsection 2.3, we have shown that there exists a polynomial-time reduction of unification modulo an acyclic TBox to unification without a TBox. Thus, Theorem 5.14 also yields the exact complexity for $\mathcal{EL}$-unification w.r.t. acyclic TBoxes.

Corollary 5.15. *$\mathcal{EL}$-unification w.r.t. acyclic TBoxes is NP-complete.*

Proof. The problem is in NP since Theorem 2.8 states that there is a polynomial-time reduction of $\mathcal{EL}$-unification w.r.t. acyclic TBoxes to $\mathcal{EL}$-unification, and we have just shown that $\mathcal{EL}$-unification is in NP.

NP-hardness for $\mathcal{EL}$-unification w.r.t. acyclic TBoxes follows from NP-hardness of $\mathcal{EL}$-unification since $\mathcal{EL}$-unification can be viewed as the special case of $\mathcal{EL}$-unification w.r.t. acyclic TBoxes where the TBox is empty. □

6. A goal-oriented algorithm

The NP-algorithm introduced in the previous section is a typical "guess and then test" NP-algorithm, and thus it is unlikely that a direct implementation of this algorithm will perform well in practice. Here, we introduce a more goal-oriented unification algorithm for $\mathcal{EL}$, in which non-deterministic decisions are only made if they are triggered by "unsolved parts" of the unification problem.

As in the previous section, we assume without loss of generality that our input unification problem Γ_0 is flat. For a given flat equation $C \equiv^? D$, the concept terms C, D are thus

[8] The NP-hardness proof in [24] is by reduction of SAT. This reduction employs two concept constants and four role names. However, the roles are mainly used to encode several (matching) equations into a single one. When using a set of equations rather than a single equation, one role name is sufficient.

conjunctions of flat atoms. We will often view such an equation as consisting of four sets: the left-hand side C is given by the set of variables occurring in the top-level conjunction of C, together with the set of non-variable atoms occurring in this top-level conjunction; the right-hand side D is given by the set of variables occurring in the top-level conjunction of D, together with the set of non-variable atoms occurring in this top-level conjunction. To be more precise, let e denote the equation $C \equiv^? D$, where $C = X_1 \sqcap \ldots \sqcap X_m \sqcap A_1 \sqcap \ldots \sqcap A_k$ and $D = Y_1 \sqcap \ldots \sqcap Y_n \sqcap B_1 \sqcap \ldots \sqcap B_\ell$ for concept variables $X_1, \ldots, X_m, Y_1, \ldots, Y_n$ and non-variable atoms $A_1, \ldots, A_k, B_1, \ldots B_\ell$. Then we define

$$\begin{array}{ll} \mathit{LVar}(e) := \{X_1, \ldots, X_m\}, & \mathit{RVar}(e) := \{Y_1, \ldots, Y_n\}, \\ \mathit{LAto}(e) := \{A_1, \ldots, A_k\}, & \mathit{RAto}(e) := \{B_1, \ldots B_\ell\}. \end{array}$$

Obviously, the equation $e : C \equiv^? D$ is uniquely determined (up to associativity, commutativity, and idempotency of conjunction) by the four sets $\mathit{LVar}(e), \mathit{LAto}(e), \mathit{RVar}(e), \mathit{RAto}(e)$. Instead of viewing an equation e as being given by a pair of concept terms, we can thus also view it as being given by these four sets. In the following, it will often be convenient to employ this representation of equations. If, with this point of view, we say that we *add an atom* to the set $\mathit{LAto}(e)$ or $\mathit{RAto}(e)$, then this means, for the other point of view, that we conjoin this atom to the top-level conjunction of the left-hand side or right-hand side of the equation. In addition, if we say that the equation e *contains* the variable X, then we mean that $X \in \mathit{LVar}(e) \cup \mathit{RVar}(e)$. Similarly, if we say that the left-hand side of e contains X, then we mean that $X \in \mathit{LVar}(e)$, and if we say that the right-hand side of e contains X, then we mean that $X \in \mathit{RVar}(e)$).[9]

In addition to the unification problem itself, the algorithm also maintains, for every variable X occurring in the input problem Γ_0, a set S_X of non-variable atoms of Γ_0. Initially, all the sets S_X are empty. We call the set S_X the current assignment for X, and the collection of all these sets the *current assignment*. Throughout the run of our goal-oriented algorithm, we will ensure that the current assignment is *acyclic* in the sense that no variable depends on itself w.r.t. this assignment (see (2) of Algorithm 5.12). An acyclic assignment induces a substitution σ, as defined in (3) of Algorithm 5.12. We call this substitution the *current substitution*. Initially, the current substitution maps all variables to $\top$.

The algorithm applies rules that can

(1) change an equation of the unification problem by adding non-variable atoms of the input problem Γ_0 to one side of the equation;
(2) introduce a new flat equation of the form $C \sqcap B \equiv B$, where C, B are atoms of the input problem Γ_0 or $\top$;
(3) add non-variable atoms of the input problem Γ_0 to the sets S_X.

Another property that is maintained throughout the run of our algorithm is that all equations e are *expanded* w.r.t. the current assignment in the following sense: for all variables X we have

$$A \in S_X \wedge X \in \mathit{LVar}(e) \Rightarrow A \in \mathit{LAto}(e) \quad \text{and} \quad A \in S_X \wedge X \in \mathit{RVar}(e) \Rightarrow A \in \mathit{RAto}(e).$$

Given a flat equation e that contains the variable X, the *expansion* of e w.r.t. the assignment S_X for X is defined as follows: if $X \in \mathit{LVar}(e)$ then all elements of S_X are added to $\mathit{LAto}(e)$, and if $X \in \mathit{RVar}(e)$ then all elements of S_X are added to $\mathit{RAto}(e)$.

[9] Note that occurrences of X inside non-variable atoms $\exists r.X \in \mathit{LAto}(e) \cup \mathit{RAto}(e)$ are not taken into consideration here.

The L-variant of the Eager-Assignment rule applies to the equation e if there is an unfinished variable $X \in LVar(e)$ such that

- all variables $Z \in (LVar(e) \setminus \{X\}) \cup RVar(e)$ are finished;
- $LAto(e) = \emptyset$.

Its application sets $S_X := RAto(e)$.

(1) If this makes the current assignment cyclic, then return "fail."
(2) Otherwise, label X as finished and expand all equations containing X w.r.t. the new assignment for X.

Figure 1: The Eager-Assignment rule in its L-variant. The R-variant is obtained by exchanging the rôles of the two sides of the equation.

The following lemma is an immediate consequence of the definition of expanded equations and of the construction of the current substitution.

Lemma 6.1. *If the equation $C \equiv^? D$ is expanded w.r.t. the current assignment, then $LAto(C \equiv^? D) = RAto(C \equiv^? D)$ implies that the current substitution σ solves this equation, i.e., $\sigma(C) \equiv \sigma(D)$.*

We say that an equation e is *solved* if $LAto(e) = RAto(e)$. An atom $A \in LAto(e) \cap RAto(e)$ is called *solved in* e; atoms $A \in LAto(e) \cup RAto(e)$ that are not solved in e are called *unsolved in* e. Obviously, an equation e is solved iff all atoms $A \in LAto(e) \cup RAto(e)$ are solved in e.

Basically, in each step, the goal-oriented algorithm considers an unsolved equation and an unsolved atom in this equation, and tries to solve it. Picking the unsolved equation and the unsolved atom in it is don't care non-deterministic, i.e., there is no need to backtrack over such a choice. Once an unsolved equation and an unsolved atom in it was picked, don't know non-determinism comes in since there may be several possibilities for how to solve this atom in the equation, some of which may lead to overall success whereas others won't. In some cases, however, a given equation uniquely determines the assignment for a certain variable X. In this case, we make this assignment and then label the variable X as *finished*. This has the effect that the set S_X can no longer be extended. Initially, none of the variables occurring in the input unification problem is labeled as finished. We say that the variable X is *unfinished* if it is not labeled as finished.

Algorithm 6.2. Let Γ_0 be a flat $\mathcal{EL}$-unification problem. We define $\Gamma := \Gamma_0$ and $S_X := \emptyset$ for all variables X occurring in Γ_0. None of these variables is labeled as finished.

As long as Γ contains an unsolved equation, do the following:

(1) If the Eager-Assignment rule applies to some equation e, then apply it to this equation (see Figure 1).
(2) Otherwise, let e be an unsolved equation and A an unsolved atom in e. If neither of the rules Decomposition (see Figure 2) and Extension (see Figure 3) applies to A in e, then return "fail." If one of these rules applies to A in e, then (don't know) non-deterministically choose one of these rules and apply it.

Once all equations of Γ are solved, return the substitution σ that is induced by the current assignment.

The *Eager-Assignment rule* is described in Figure 1. Note that, after a non-failing application of this rule, the equation it was applied to is solved since the expansion of this

The L-variant of the Decomposition rule applies to the unsolved atom A in the equation e if

- $A \in LAto(e) \setminus RAto(e)$;
- A is of the form $A = \exists r.C$;
- there is at least one atom of the form $\exists r.B \in RAto(e)$.

Its application chooses (don't know) non-deterministically an atom of the form $\exists r.B \in RAto(e)$ and

- adds $\exists r.C$ to $RAto(e)$;
- creates a new equation $C \sqcap B \equiv^? B$ and expands it w.r.t. the assignments of all variables contained in this equation, unless this equation has already been generated before. If the equation has already been generated before, it is not generated again.

Figure 2: The Decomposition rule in its L-variant. The R-variant is obtained by exchanging the rôles of the two sides of the equation.

equation w.r.t. the new assignment for X adds all elements of $RAto(e)$ to $LAto(e)$. As an example, consider the equations

$$Y \equiv^? \top, \quad Z \equiv^? \exists r.\top, \quad X \sqcap Y \equiv^? Z,$$

and assume that $S_X = S_Y = S_Z = \emptyset$ and none of the three variables X, Y, Z is finished. An application of the Eager-Assignment rule to the first equation labels Y as finished, but does not change anything else. The subsequent application of the Eager-Assignment rule to the second equation changes the assignment for Z to $S_Z = \{\exists r.\top\}$, labels Z as finished, and expands the second and the third equation w.r.t. the new assignment for Z. Thus, we now have the equations

$$Y \equiv^? \top, \quad Z \sqcap \exists r.\top \equiv^? \exists r.\top, \quad X \sqcap Y \equiv^? Z \sqcap \exists r.\top.$$

Since Y, Z are finished, the Eager-Assignment rule can now be applied to the third equation. This changes the assignment for X to $S_X = \{\exists r.\top\}$, labels X as finished, and adds $\exists r.\top$ to the left-hand side of the third equation. Now all equations are solved. The current assignment induces a substitution σ with $\sigma(X) = \exists r.\top = \sigma(Z)$ and $\sigma(Y) = \top$, which is a unifier of the original set of equations.

The *Decomposition rule* is described in Figure 2. This rule solves the unsolved atom $A = \exists r.C$ by adding it to the other side. For this to be admissible, one needs a more specific atom $\exists r.B$ on that side, where the "more specific" is meant to hold after application of the unifier. Thus, to ensure that the unifier σ computed by the algorithm satisfies $\sigma(\exists r.B) \sqsubseteq \sigma(\exists r.C)$, the rule adds the new equation $C \sqcap B \equiv^? B$. Obviously, if the substitution σ solves this equation, then it satisfies $\sigma(B) \sqsubseteq \sigma(C)$, and thus $\sigma(\exists r.B) \sqsubseteq \sigma(\exists r.C)$. As an example, consider the equation

$$\exists r.X \sqcap \exists r.A \equiv^? \exists r.A,$$

and assume that $S_X = \emptyset$ and that X is unfinished. An application of the L-variant of the Decomposition rule to this equation adds $\exists r.X$ to the right-hand side of this equation, and thus solves it. In addition, it generates the new equation $X \sqcap A \equiv^? A$, which is solved. The current assignment induces a substitution σ with $\sigma(X) = \top$, which solves the original equation.

The L-variant of the Extension rule applies to the unsolved atom A of the equation e if

- $A \in LAto(e) \setminus RAto(e)$;
- there is at least one unfinished variable $X \in RAto(e)$

Its application chooses (don't know) non-deterministically an unfinished variable $X \in RAto(e)$ and adds A to S_X.

- If this makes the current assignment cyclic, then return "fail."
- Otherwise, expand all equations containing X w.r.t. the new assignment for X.

Figure 3: The Extension rule in its L-variant. The R-variant is obtained by exchanging the rôles of the two sides of the equation.

The *Extension rule* is described in Figure 3. Basically, this rule solves the unsolved atom A by extending with this atom the assignment of an unfinished variable contained in the other side of the equation. As an example, consider the equation

$$A \sqcap \exists r.\top \equiv^? \exists r.\top \sqcap X,$$

where A is a concept constant, $S_X = \emptyset$, and X is unfinished. An application of the Extension rule to A in this equation extends the assignment for X to $S_X = \{A\}$, and expands this equation by adding A to the right-hand side. The equation obtained this way is solved. The substitution σ induced by the current assignment replaces X by A, and solves the original equation.

Theorem 6.3. *Algorithm 6.2 is an NP-algorithm for testing solvability of flat $\mathcal{EL}$-unification problems.*

First, we show that the algorithm is indeed an NP-algorithm. For this, we consider all *runs* of the algorithm, where for every (don't care) non-deterministic choice exactly one alternative is taken. Since a single rule application can obviously be realized in polynomial time, it is sufficient to show the following lemma.

Lemma 6.4 (Termination)**.** *Every run of the algorithm terminates after a polynomial number of rule applications.*

Proof. Each application of the Eager-Assignment rule finishes an unfinished variable. Thus, since finished variables never become unfinished again, it can only be applied k times, where k is the number of variables occurring in the input unification problem Γ_0. This number is clearly linearly bounded by the size of Γ_0.

Every application of the Decomposition rule or the Extension rule turns an unsolved atom in an equation into a solved one, and a solved atom in an equation never becomes unsolved again in this equation. For a fixed equation, in the worst case every atom of Γ_0 may become an unsolved atom of the equation that needs to be solved. There is, however, only a linear number of atoms of Γ_0. Each equation considered during the run of the algorithm is either descended from an original equation of Γ_0, or from an equation of the form $C \sqcap B \equiv^? B$ for atoms $\exists r.B$ and $\exists r.C$ of Γ_0. Thus, the number of equations is also polynomially bounded by the size of Γ_0. Overall, this shows that the Decomposition rule and the Extension rule can only be applied a polynomial number of times. □

Next, we show soundness of Algorithm 6.2. We call a run of this algorithm *non-failing* if it terminates with a unification problem containing only solved equations.

Lemma 6.5 (Soundness)**.** *Let Γ_0 be a flat $\mathcal{EL}$-unification problem. The substitution σ returned after a successful run of Algorithm 6.2 on input Γ_0 is an $\mathcal{EL}$-unifier of Γ_0.*

Proof. First, note that the rules employed by Algorithm 6.2 indeed preserve the two invariants mentioned before:

(1) the current assignment is always acyclic;
(2) all equations are expanded.

In fact, whenever the current assignment is extended, the rules test acyclicity (and return "fail," if it is not satisfied). In addition, they expand all equations w.r.t. the new assignment.

Now, assume that the run of the algorithm has terminated with the $\mathcal{EL}$-unification problem $\widehat{\Gamma}$, in which all equations are solved. The first invariant ensures that the final assignment constructed by the run is acyclic, and thus indeed induces a substitution σ. Because of the second invariant, Lemma 6.1 applies, and thus we know that σ is a solution of $\widehat{\Gamma}$.

It remains to show that the substitution σ is also a solution of the input problem Γ_0. To this purpose, we take all the equations that were considered during the run of the algorithm, i.e., present in Γ_0 or in any of the other unification problems generated during the run. Let $\mathcal{E}$ denote the set of these equations. We define the relation $\to$ on $\mathcal{E}$ as follows: $e \to e'$ if e was transformed into e' using one of the rules of Algorithm 6.2. To be more precise, the Eager-Assignment rule transforms equations containing X from the current unification problem Γ by expanding them w.r.t. the new assignment for X. The same is true for the Extension rule. The decomposition rule transforms an equation e containing the unsolved atom $A = \exists r.C$ by adding this atom to the other side, which needs to contain an atom of the form $\exists r.B$. For this new equation e', we have $e \to e'$. The decomposition rule may also generate a new equation e'' of the form $C \sqcap B \equiv^? B$ (if this equation was not generated before). However, we do not view this equation as a successor of e w.r.t. $\to$, i.e., we do *not* have $e \to e''$. Equations $C \sqcap B \equiv^? B$ that are generated by an application of the decomposition rule are called *D-equations*. Equations that are elements of the input problem Γ_0 are called *I-equations*. Any equation e' that is not an I-equation or a D-equation has a unique predecessor w.r.t. $\to$, i.e., there is an equation $e \in \mathcal{E}$ such that $e \to e'$.

Starting with the set $\mathcal{F} := \widehat{\Gamma}$ we will now step by step extend $\mathcal{F}$ by a predecessor of an equation in $\mathcal{F}$ until no new predecessors can be added. Since $\mathcal{E}$ is finite, this process terminates after a finite number of steps. After termination we have $\mathcal{E} = \mathcal{F}$, and thus in particular $\Gamma_0 \subseteq \mathcal{F}$. This is due to the fact that, for every element e_0 of $\mathcal{E}$, there are $n \geq 0$ elements $e_1, \ldots, e_n \in \mathcal{E}$ such that $e_0 \to e_1 \to \ldots \to e_n$ and $e_n \in \widehat{\Gamma}$. Thus, it is enough to show that the set $\mathcal{F}$ satisfies the following *invariant*:

$(*)$ *the substitution σ solves every equation in $\mathcal{F}$.*

Since σ is a solution of $\widehat{\Gamma}$, this invariant is initially satisfied. To prove that it is preserved under adding predecessors of equations in $\mathcal{F}$, we start with the equations of minimal role depth. To be more precise, if the equation e is of the form $C \equiv^? D$, we define the *role depth of e w.r.t. σ* to be the role depth[10] of the concept term $\sigma(C) \sqcap \sigma(D)$. The strict order $\succ$ on $\mathcal{E}$ is defined as follows: $e \succ_\sigma e'$ iff the role depth of e w.r.t. σ is larger than the role depth

[10] see the proof of Proposition 3.5 for a definition.

of e' w.r.t. σ. We write $e \approx_\sigma e'$ if e and e' have the same role depth w.r.t. σ. The following is an easy consequence of the definition of σ and of our rules:

$$(**) \quad e_1 \to e_2 \to \ldots \to e_n \text{ implies } e_1 \approx_\sigma e_2 \approx_\sigma \ldots \approx_\sigma e_n.$$

Assume that we have already constructed a set $\mathcal{F}$ such that the invariant $(*)$ is satisfied. Let e' be an equation in $\mathcal{F}$ such that

- there is an $e \in \mathcal{E} \setminus \mathcal{F}$ with $e \to e'$;
- e' is of minimal role depth with this property, i.e., if $f' \in \mathcal{F}$ is such that $e' \succ f'$ and f' has a predecessor f w.r.t. $\to$, then $f \in \mathcal{F}$.

If no such equation e' exists, then we are finished, and we have $\mathcal{E} = \mathcal{F}$. Otherwise, let e' be such an equation and e its predecessor w.r.t. $\to$. We add e to $\mathcal{F}$. In order to show that the invariant $(*)$ is still satisfied, we make a case distinction according to which rule was applied to e to produce e':

(1) *Eager-Assignment.* By an application of this rule, the assignment for X is modified from $S_X = \emptyset$ to $S_X = \{A_1, \ldots, A_n\}$, where $A_1, \ldots, A_n$ are non-variable atoms. In addition, X is labeled as finished. Since the assignment of a finished variable cannot be changed anymore, we know that we also have $S_X = \{A_1, \ldots, A_n\}$ in the final assignment, and thus $\sigma(X) = \sigma(A_1) \sqcap \ldots \sqcap \sigma(A_n)$. The rule modifies equations as follows: all equations containing X are expanded w.r.t. the assignment $S_X = \{A_1, \ldots, A_n\}$. Since e is transformed into e' using this rule, it must contain X. We assume for the sake of simplicity that X is contained in the left-hand side of e, but not in the right-hand side, i.e., e is of the form $C \sqcap X \equiv^? D$ and the new equation $e' \in \Gamma'$ obtained from e is $C \sqcap X \sqcap A_1 \sqcap \ldots \sqcap A_n \equiv^? D$. Since σ solves e', we have $\sigma(D) \equiv \sigma(C \sqcap X \sqcap A_1 \sqcap \ldots \sqcap A_n) \equiv \sigma(C) \sqcap \sigma(A_1) \sqcap \ldots \sqcap \sigma(A_n) \sqcap \sigma(A_1) \sqcap \ldots \sqcap \sigma(A_n) \equiv \sigma(C) \sqcap \sigma(A_1) \sqcap \ldots \sqcap \sigma(A_n) \equiv \sigma(C \sqcap X)$, which shows that σ also solves e.

(2) *Decomposition.* Without loss of generality, we consider the L-variant of this rule. Thus, the equation e is of the form $D \sqcap \exists\, r.C \equiv^? E \sqcap \exists\, r.B$, and e' is obtained from e by adding $\exists\, r.C$ to the right-hand side, i.e., e' is of the form $D \sqcap \exists\, r.C \equiv^? E \sqcap \exists\, r.B \sqcap \exists\, r.C$. We know that σ solves e'. Thus, if we can show $\sigma(B) \sqsubseteq \sigma(C)$, then we have $\sigma(D \sqcap \exists\, r.C) \equiv \sigma(E) \sqcap \sigma(\exists\, r.B) \sqcap \sigma(\exists\, r.C) \equiv \sigma(E) \sqcap \sigma(\exists\, r.B)$, which shows that σ solves e.

Consequently, it is sufficient to prove $\sigma(B) \sqsubseteq \sigma(C)$. The Decomposition rule also generates the equation $C \sqcap B \equiv^? B$ and expands it w.r.t. the assignments of all the variables contained in this equation, unless this equation has already been generated before. Thus, either this application or a previous one of the Decomposition rule has generated the equation $C \sqcap B \equiv^? B$, and then expanded it (w.r.t. the current assignment at that time) to an equation e_1. Since atoms are never removed from an assignment, the atoms present in the assignment at the time when the Decomposition rule generated the equation $C \sqcap B \equiv^? B$ are also present in the final assignment used to define the substitution σ. Thus, if we can show that σ solves e_1, then we have also shown that σ solves $C \sqcap B \equiv^? B$, and thus satisfies $\sigma(B) \sqsubseteq \sigma(C)$.

Since equations are never completely removed by our rules, but only modified, there is a sequence of equations $e_1 \to e_2 \to \ldots \to e_n$ such that $e_n \in \widehat{\Gamma}$. Property $(**)$ thus yields $e_1 \approx_\sigma e_2 \approx_\sigma \ldots \approx_\sigma e_n$. In addition, the role depth of $C \sqcap B \equiv^? B$ w.r.t. σ is the same as the role depth of e_1 w.r.t. σ. Consequently, we have $e' \succ e_i$ for all $i, 1 \leq i \leq n$. Now, assume that $e_1 \notin \mathcal{F}$. Then there is an $i > 1$ such that $e_i \in \mathcal{F}$, but $e_{i-1} \in \mathcal{E} \setminus \mathcal{F}$.

This contradicts our assumption that e' is minimal. Thus, we have shown that $e_1 \in \mathcal{F}$, and this implies that σ solves e_1.

Overall, this finishes the proof that σ solves e.

(3) *Extension.* By an application of this rule, the assignment for X is modified by adding a non-variable atom A to it. Since atoms are never removed from an assignment, we know that we also have $A \in S_X$ in the final assignment, and thus $\sigma(X) \sqsubseteq \sigma(A)$. The rule modifies equations as follows: all equations containing X are expanded w.r.t. the new assignment for X. Since e is transformed into e' using this rule, it must contain X. We assume for the sake of simplicity that X is contained in the left-hand side of e, but not in the right-hand side, i.e., e is of the form $C \sqcap X \equiv^? D$ and the new equation e' obtained from e is $C \sqcap X \sqcap A \equiv^? D$. Since σ solves e', we have $\sigma(D) \equiv \sigma(C \sqcap X \sqcap A) \equiv \sigma(C) \sqcap \sigma(X) \sqcap \sigma(A) \equiv \sigma(C) \sqcap \sigma(X) \equiv \sigma(C \sqcap X)$, which shows that σ also solves e.

To sum up, we have shown that the invariant $(*)$ is still satisfied after adding e to $\mathcal{F}$. This completes the proof of soundness of our procedure. □

It remains to show completeness of Algorithm 6.2. Thus, assume that the input unification problem Γ_0 is solvable. Proposition 5.4 tells us that Γ_0 then has an is-minimal reduced ground unifier γ, and Proposition 5.11 implies that, for every variable X occurring in Γ_0, there is a set S_X^γ of non-variable atoms of Γ_0 such that

$$\gamma(X) \equiv \gamma(\bigsqcap S_X^\gamma),$$

where, for a set of non-variable atoms S of Γ_0, the expression $\bigsqcap S$ denotes the conjunction of the elements of S (where the empty conjunction is $\top$).

Lemma 6.6 (Completeness)**.** *Let Γ_0 be a flat $\mathcal{EL}$-unification problem, and assume that γ is an is-minimal reduced ground unifier of Γ_0. Then there is a successful run of Algorithm 6.2 on input Γ_0 that returns a unifier σ that is equivalent to γ, i.e., satisfies $\sigma(X) \equiv \gamma(X)$ for all variables X occurring in Γ_0.*

Proof. The algorithm starts with $\Gamma := \Gamma_0$ and the initial assignment $S_X := \emptyset$ for all variables X occurring in Γ_0. It then applies rules that change Γ and the current assignment as long as the problem Γ contains an unsolved equation.

We use γ to guide the (don't know) non-deterministic choices to be made during the algorithm. We show that this ensures that the run of the algorithm generated this way *does not fail* and that the following *invariants* are satisfied throughout this run:

(I_1) γ is a unifier of Γ;
(I_2) for all atoms $B \in S_X$ there exists an atom $A \in S_X^\gamma$ such that $\gamma(A) \sqsubseteq \gamma(B)$;
(I_3) for all finished variables X we have $\gamma(X) \equiv \gamma(\bigsqcap S_X)$.

Before constructing a run that satisfies these invariants, let us point out two interesting consequences that they have:

(C_1) *The current assignment is always acyclic.* In fact, if X directly depends on Y, then there is an atom $B \in S_X$ that has the form $B = \exists r.Y$ for some role name r. Invariant I_2 then implies that there is an $A \in S_X^\gamma$ such that $\gamma(X) \sqsubseteq \gamma(A) \sqsubseteq \gamma(B) = \exists r.\gamma(Y)$. Thus, if X depends on X, then there are $k \geq 1$ role names $r_1, \ldots, r_k$ such that $\gamma(X) \sqsubseteq \exists r_1. \cdots \exists r_k.\gamma(X)$, which is impossible.

(C_2) *For each variable X occurring in Γ_0, we have $\gamma(X) \sqsubseteq \sigma(X)$, where σ is the current substitution induced by the current assignment.* This is again a consequence of invariant I_2. Indeed, recall that the fact that the current assignment is acyclic implies that there is a strict linear order $>$ on the variables occurring in Γ such that $X > Y$ if X depends on Y. The current substitution σ is defined along this order. We prove $\gamma(X) \sqsubseteq \sigma(X)$ by induction on this order.

Consider the least variable X. If $S_X = \emptyset$, then $\sigma(X) = \top$, and thus $\gamma(X) \sqsubseteq \sigma(X)$ is trivially satisfied. Otherwise, we know, for every $B \in S_X$, that it does not contain any variables, which implies that $\sigma(B) = B = \gamma(B) \sqsupseteq \gamma(A)$ for some atom $A \in S_X^\gamma$. Obviously, this yields $\sigma(X) = \sigma(\bigsqcap S_X) \sqsupseteq \gamma(\bigsqcap S_X^\gamma) = \gamma(X)$.

Now, assume that $\gamma(Y) \sqsubseteq \sigma(Y)$ holds for all variables $Y < X$. Since the concept constructors of $\mathcal{EL}$ are monotone w.r.t. subsumption, this implies $\gamma(C) \sqsubseteq \sigma(C)$ for all concept terms C containing only variables smaller than X. If S_X is empty, then $\sigma(X) = \top \sqsupseteq \gamma(X)$ is trivially satisfied. Otherwise, we know, for every $B \in S_X$, that it contains only variables smaller than X. This yields $\sigma(B) \sqsupseteq \gamma(B) \sqsupseteq \gamma(A)$ for some atom $A \in S_X^\gamma$. Again, this implies $\sigma(X) = \sigma(\bigsqcap S_X) \sqsupseteq \gamma(\bigsqcap S_X^\gamma) = \gamma(X)$.

Since γ was assumed to be an is-minimal unifier of Γ_0, the consequence C_2 implies that σ can only be a unifier of Γ_0 if σ is equivalent to γ. If the run has terminated successfully, then the final substitution σ obtained by the run is a unifier of Γ_0 (due to soundness). Thus, in this case the computed unifier σ is indeed equivalent to γ. Consequently, to prove the lemma, it is sufficient to *construct a non-failing run of the algorithm that satisfies the above invariants.*

The invariants are initially satisfied since γ is a unifier of Γ_0, the initial assignment for all variables X occurring in Γ_0 is $S_X = \emptyset$, and there are no finished variables. Now, assume that, by application of the rules of Algorithm 6.2, we have constructed a unification problem Γ and a current assignment such that the invariants are satisfied.

(1) If all equations in Γ are solved, then the run terminates successfully, and we are done.
(2) If there is an unsolved equation to which the *Eager-Assignment rule* applies, then the algorithm picks such an equation e and applies this rule to it. Without loss of generality, we assume that the L-variant of the rule is applied. The selected equation e is of the form

$$X \sqcap Z_1 \sqcap \ldots \sqcap Z_k \equiv^? A_1 \sqcap \ldots \sqcap A_n \sqcap Y_1 \sqcap \ldots \sqcap Y_m,$$

where $A_1, \ldots, A_n$ are non-variable atoms, and $Y_1, \ldots, Y_m, Z_1, \ldots Z_k$ are finished variables. Because the left-hand side of the equation does not contain any non-variable atoms, we know that $S_X = S_{Z_1} = \ldots = S_{Z_k} = \emptyset$ (since the algorithm keeps all equations expanded). Since $Z_1, \ldots, Z_k$ are finished, we thus have $\gamma(Z_1) = \ldots = \gamma(Z_k) = \top$ (by invariant I_3). We also know that $S_{Y_i} \subseteq \{A_1, \ldots, A_n\}$ for all $i, 1 \leq i \leq m$. Since the variables Y_i are finished, invariant I_3 implies that $\gamma(Y_i) \sqsupseteq \gamma(A_1) \sqcap \ldots \sqcap \gamma(A_n)$.

The new assignment for X is $S_X = \{A_1, \ldots, A_n\}$, all equations containing X are expanded w.r.t. this assignment, and X becomes a finished variable. First, we show that I_3 is satisfied. Nothing has changed for the variables that were already finished before the application of the rule. However, X is now also finished. Thus, we must show that $\gamma(X) \equiv \gamma(\bigsqcap S_X)$. We know that γ solves the equation e (by I_1). This yields $\gamma(X) \equiv \gamma(X) \sqcap \gamma(Z_1) \sqcap \ldots \sqcap \gamma(Z_k) \equiv \gamma(A_1) \sqcap \ldots \sqcap \gamma(A_n) \sqcap \gamma(Y_1) \sqcap \ldots \sqcap \gamma(Y_m) \equiv \gamma(A_1) \sqcap \ldots \sqcap \gamma(A_n) = \gamma(\bigsqcap S_X)$. Regarding I_2, the only assignment that was changed is the one for X. Since the new assignment for X is $S_X = \{A_1, \ldots, A_n\}$, and we have

already shown that $\gamma(X) \equiv \gamma(A_1) \sqcap \ldots \sqcap \gamma(A_n)$, the invariant I_2 holds by Corollary 3.2. Note that this also implies that the new assignment is acyclic, and thus the application of the Eager-Assignment rule does not fail. Finally, consider the invariant I_1. The rule application modifies equations containing X by adding the atoms $A_1, \ldots, A_n$. Since $\gamma(X) \equiv \gamma(A_1) \sqcap \ldots \sqcap \gamma(A_n)$, an equation that was solved by γ before this modification, is also solved by γ after this modification. To sum up, we have shown that the application of the Eager-Assignment rule does not fail and preserves the invariants.

(3) If there is *no* unsolved equation to which the *Eager-Assignment rule* applies, then the algorithm picks an unsolved equation e and an unsolved atom A occurring in this equation. We must show that we can apply either the Decomposition or the Extension rule to A in e such that the invariants stay satisfied. Without loss of generality, we assume that the unsolved atom A occurs on the left-hand side of the equation e.

(a) First, assume that A *is an existential restriction* $A = \exists\, r.C$. The selected unsolved equation e is thus of the form

$$\exists\, r.C \sqcap A_1 \sqcap \ldots \sqcap A_m \equiv B_1 \sqcap \ldots \sqcap B_n,$$

where $A_1, \ldots, A_m$ and $B_1, \ldots, B_n$ are (variable or non-variable) atoms and $\exists\, r.C \notin \{B_1, \ldots, B_n\}$. Since γ solves this equation (by invariant I_1), Corollary 3.2 implies that there must be an $i, 1 \leq i \leq n$, such that $\gamma(B_i) \sqsubseteq \exists\, r.\gamma(C)$.

(i) If B_i is an existential restriction $B_i = \exists\, r.B$, then we have $\gamma(B) \sqsubseteq \gamma(C)$. We apply the Decomposition rule to A and B_i. The application of this rule modifies the equation e to an equation e' by adding the atom A to the right-hand side. In addition, it generates the equation $C \sqcap B \equiv^? B$ and expands it w.r.t. the assignments of all variables contained in this equation (unless this equation has been generated before). After the application of this rule, the invariants I_2 and I_3 are still satisfied since the current assignments and the set of finished variables remain unchanged. Regarding invariant I_1, since γ solves e, it obviously also solves e' due to the fact that $\gamma(B_i) \sqsubseteq \gamma(A)$ and B_i is a conjunct on the right-hand side of e. In addition, $\gamma(B) \sqsubseteq \gamma(C)$ implies that γ also solves the equation $C \sqcap B \equiv^? B$. Since invariant I_2 is satisfied, this implies that γ also solves the equation obtained from $C \sqcap B \equiv^? B$ by expanding it w.r.t. the assignments of all variables contained in it.

(ii) Assume that there is no $i, 1 \leq i \leq n$, such that B_i is an existential restriction satisfying $\gamma(B_i) \sqsubseteq \gamma(A)$. Thus, if B_i is such that $\gamma(B_i) \sqsubseteq \gamma(A)$, then we know that $B_i = X$ is a variable. We want to apply the Extension rule to A and X. To be able to do this, we must first show that X is not a finished variable.
Thus, assume that X is finished, and let $S_X = \{C_1, \ldots, C_\ell\}$. Invariant I_3 yields $\gamma(C_1) \sqcap \ldots \sqcap \gamma(C_\ell) = \gamma(X) = \gamma(B_i) \sqsubseteq \gamma(A) = \exists\, r.\gamma(C)$, and thus there is a $j, 1 \leq j \leq \ell$, such that $\gamma(C_j) \sqsubseteq \gamma(A)$. Since A is an existential restriction, the non-variable atom C_j must also be an existential restriction, and since the equation e is expanded, $C_j \in S_X$ occurs on the right-hand side of this equation. This contradicts our assumption that there is no such existential restriction on the right-hand side. Thus, we have shown that X is not finished, which means that we can apply the Extension rule to A and X.
The application of this rule adds the atom A to the assignment for X, and it expands all equations containing X w.r.t. this new assignment, i.e., it adds A to the left-hand side and/or right-hand side of an equation whenever X is

contained in this side. Since we know that $\gamma(X) \sqsubseteq \gamma(A)$, it is easy to see that, if γ solves an equation before this expansion, it also solves it after the expansion. Thus invariant I_1 is satisfied. Invariant I_2 also remains satisfied. In fact, if $S_X^\gamma = \{D_1, \ldots, D_k\}$, then $\gamma(D_1) \sqcap \ldots \sqcap \gamma(D_k) = \gamma(X) \sqsubseteq \gamma(A)$ implies that there is a $j, 1 \leq j \leq \ell$, such that $\gamma(D_j) \sqsubseteq \gamma(A)$. The fact that I_2 is satisfied by the new assignment also implies that this new assignment is acyclic, and thus the application of the Extension rule does not fail. Invariant I_3 is still satisfied since X is not finished, and the assignments of variables different from X were not changed.

(b) Second, assume that A *is a concept name*. The selected unsolved equation e is thus of the form

$$A \sqcap A_1 \sqcap \ldots \sqcap A_m \equiv B_1 \sqcap \ldots \sqcap B_n,$$

where $A_1, \ldots, A_m$ and $B_1, \ldots, B_n$ are (variable or non-variable) atoms, and $A \notin \{B_1, \ldots, B_n\}$. Since γ solves this equation (by invariant I_1), Corollary 3.2 implies that there must be an $i, 1 \leq i \leq n$, such that $\gamma(B_i) \sqsubseteq \gamma(A) = A$. Since $A \notin \{B_1, \ldots, B_n\}$, we know that $B_i = X$ is a variable. We want to apply the Extension rule to A and X. To be able to do this, we must first show that X is not a finished variable.

Thus, assume that X is finished, and let $S_X = \{C_1, \ldots, C_\ell\}$. Invariant I_3 yields $\gamma(C_1) \sqcap \ldots \sqcap \gamma(C_\ell) = \gamma(X) = \gamma(B_i) \sqsubseteq \gamma(A) = A$, and thus there is a $j, 1 \leq j \leq \ell$, such that $\gamma(C_j) \sqsubseteq A$. Since A is a concept name, the non-variable atom C_j must actually be equal to A, and since the equation e is expanded, $C_j = A \in S_X$ occurs on the right-hand side of this equation. This contradicts our assumption that A is an unsolved atom. Thus, we have shown that X is not finished, which means that we can apply the Extension rule to A and X. The application of this rule adds the atom A to the assignment for X, and it expands all equations containing X w.r.t. this new assignment. The proof that this rule application does not fail and preserves the invariants is identical to the one for the case where A was an existential restriction. □

To sum up, we have shown that Algorithm 6.2 always terminates (in non-deterministic polynomial time) and that it is sound and complete. This finishes the proof of Theorem 6.3.

7. Unification in semilattices with monotone operators

Unification problems and their types were originally not introduced for Description Logics, but for equational theories [16]. In this section, we show that the above results for unification in $\mathcal{EL}$ can actually be viewed as results for an equational theory. As shown in [28], the equivalence problem for $\mathcal{EL}$-concept terms corresponds to the word problem for the equational theory of semilattices with monotone operators. In order to define this theory, we consider a signature Σ_{SLmO} consisting of a binary function symbol $\wedge$, a constant symbol 1, and finitely many unary function symbols $f_1, \ldots, f_n$. Terms can then be built using these symbols and additional variable symbols and free constant symbols.

Definition 7.1. The equational theory of *semilattices with monotone operators* is defined by the following identities:

$$SLmO := \{x \wedge (y \wedge z) = (x \wedge y) \wedge z,\ x \wedge y = y \wedge x,\ x \wedge x = x,\ x \wedge 1 = x\} \cup$$
$$\{f_i(x \wedge y) \wedge f_i(y) = f_i(x \wedge y) \mid 1 \leq i \leq n\}$$

A given $\mathcal{EL}$-concept term C using only roles $r_1, \ldots, r_n$ can be translated into a term t_C over the signature Σ_{SLmO} by replacing each concept constant A by a corresponding free constant a, each concept variable X by a corresponding variable x, $\top$ by 1, $\sqcap$ by $\wedge$, and $\exists\, r_i$ by f_i. For example, the $\mathcal{EL}$-concept term $C = A \sqcap \exists\, r_1.\top \sqcap \exists\, r_3.(X \sqcap B)$ is translated into $t_C = a \wedge f_1(1) \wedge f_3(x \wedge b)$. Conversely, any term over the signature Σ_{SLmO} can be translated back into an $\mathcal{EL}$-concept term.

Lemma 7.2. *Let C, D be $\mathcal{EL}$-concept term using only roles $r_1, \ldots, r_n$. Then $C \equiv D$ iff $t_C =_{SLmO} t_D$.*

As an immediate consequence of this lemma, we have that unification in the DL $\mathcal{EL}$ corresponds to unification modulo the equational theory *SLmO*. Thus, Theorem 4.1 implies that *SLmO* has unification type zero, and Theorem 5.14 implies that *SLmO*-unification is NP-complete.

Corollary 7.3. *The equational theory SLmO of semilattices with monotone operators has unification type zero, and deciding solvability of an SLmO-unification problem is an NP-complete problem.*

Since the unification problem introduced in Theorem 4.1 contains only one role r, this is already true in the presence of a single monotone operator.

8. Conclusion

In this paper, we have shown that unification in the DL $\mathcal{EL}$ is of type zero and NP-complete. There are interesting differences between the behavior of $\mathcal{EL}$ and the closely related DL $\mathcal{FL}_0$ w.r.t. unification and matching. Though the unification types coincide for these two DLs, the complexities of the decision problems differ: $\mathcal{FL}_0$-unification is ExpTime-complete, and thus considerably harder than $\mathcal{EL}$-unification. In contrast, $\mathcal{FL}_0$-matching is polynomial, and thus considerably easier than $\mathcal{EL}$-matching, which is NP-complete. In addition to showing the complexity upper bound for $\mathcal{EL}$-unification by a simple "guess and then test" NP-algorithm, we have also developed a more goal-oriented NP-algorithm that makes (don't know) non-deterministic decisions (i.e., ones that require backtracking) only if they are triggered by unsolved atoms in the unification problem.

As future work, we will consider also unification of concept terms for other members of the $\mathcal{EL}$-family of DLs [5]. In addition, we will investigate unification modulo more expressive terminological formalisms. On the practical side, we will optimize and implement the goal-oriented $\mathcal{EL}$-unification algorithm developed in Section 6. We intend to test the usefulness of this algorithm for the purpose on finding redundancies in $\mathcal{EL}$-based ontologies by considering extensions of the medical ontology Snomed ct. For example, in [18], two different extensions of Snomed ct by so-called post-coordinated concepts were considered. The authors used an (incomplete) equivalence test to find out how large the overlap between the two extensions is (i.e., how many of the new concepts belonged to both extensions). As pointed out in the introduction, the equivalence test cannot deal with situations where

different knowledge engineers use different names for concepts, or model on different levels of granularity. We want to find out whether using unifiability rather than equivalence finds more cases of overlapping concepts. Of course, in the case of unification one may also obtain false positives, i.e., pairs of concepts that are unifiable, but are not meant to represent the same (intuitive) concept. It is also important to find out how often this happens. Another problem to be dealt with in this application is the development of heuristics for choosing the pairs of concepts to be tested for unifiability and for deciding which concept names are turned into variables.

References

[1] Franz Baader. Characterizations of unification type zero. In N. Dershowitz, editor, *Proceedings of the 3rd International Conference on Rewriting Techniques and Applications*, volume 355 of *Lecture Notes in Computer Science*, pages 2–14, Chapel Hill, North Carolina, 1989. Springer-Verlag.

[2] Franz Baader. Unification in commutative theories. *J. Symbolic Computation*, 8(5):479–497, 1989.

[3] Franz Baader. Terminological cycles in KL-ONE-based knowledge representation languages. In *Proc. of the 8th Nat. Conf. on Artificial Intelligence (AAAI'90)*, pages 621–626, Boston (Ma, USA), 1990.

[4] Franz Baader. Terminological cycles in a description logic with existential restrictions. In Georg Gottlob and Toby Walsh, editors, *Proc. of the 18th Int. Joint Conf. on Artificial Intelligence (IJCAI 2003)*, pages 325–330, Acapulco, Mexico, 2003. Morgan Kaufmann, Los Altos.

[5] Franz Baader, Sebastian Brandt, and Carsten Lutz. Pushing the $\mathcal{EL}$ envelope. In Leslie Pack Kaelbling and Alessandro Saffiotti, editors, *Proc. of the 19th Int. Joint Conf. on Artificial Intelligence (IJCAI 2005)*, pages 364–369, Edinburgh (UK), 2005. Morgan Kaufmann, Los Altos.

[6] Franz Baader, Diego Calvanese, Deborah McGuinness, Daniele Nardi, and Peter F. Patel-Schneider, editors. *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press, 2003.

[7] Franz Baader and Ralf Küsters. Matching in description logics with existential restrictions. In *Proc. of the 7th Int. Conf. on Principles of Knowledge Representation and Reasoning (KR 2000)*, pages 261–272, 2000.

[8] Franz Baader and Ralf Küsters. Unification in a description logic with transitive closure of roles. In Robert Nieuwenhuis and Andrei Voronkov, editors, *Proc. of the 8th Int. Conf. on Logic for Programming and Automated Reasoning (LPAR 2001)*, volume 2250 of *Lecture Notes in Artificial Intelligence*, pages 217–232, Havana, Cuba, 2001. Springer-Verlag.

[9] Franz Baader, Ralf Küsters, Alex Borgida, and Deborah L. McGuinness. Matching in description logics. *J. of Logic and Computation*, 9(3):411–447, 1999.

[10] Franz Baader, Ralf Küsters, and Ralf Molitor. Computing least common subsumers in description logics with existential restrictions. In *Proc. of the 16th Int. Joint Conf. on Artificial Intelligence (IJCAI'99)*, pages 96–101, 1999.

[11] Franz Baader and Barbara Morawska. Unification in the description logic $\mathcal{EL}$. In Ralf Treinen, editor, *Proc. of the 20th Int. Conf. on Rewriting Techniques and Applications (RTA 2009)*, volume 5595 of *Lecture Notes in Computer Science*, pages 350–364. Springer-Verlag, 2009.

[12] Franz Baader and Paliath Narendran. Unification of concepts terms in description logics. *J. of Symbolic Computation*, 31(3):277–305, 2001.

[13] Franz Baader and Tobias Nipkow. *Term Rewriting and All That*. Cambridge University Press, United Kingdom, 1998.

[14] Franz Baader and Werner Nutt. Basic description logics. In [6], pages 43–95. 2003.

[15] Franz Baader, Baris Sertkaya, and Anni-Yasmin Turhan. Computing the least common subsumer w.r.t. a background terminology. *J. of Applied Logic*, 5(3):392–420, 2007.

[16] Franz Baader and Wayne Snyder. Unification theory. In J.A. Robinson and A. Voronkov, editors, *Handbook of Automated Reasoning*, volume I, pages 447–533. Elsevier Science Publishers, 2001.

[17] Sebastian Brandt. Polynomial time reasoning in a description logic with existential restrictions, GCI axioms, and—what else? In Ramon López de Mántaras and Lorenza Saitta, editors, *Proc. of the 16th Eur. Conf. on Artificial Intelligence (ECAI 2004)*, pages 298–302, 2004.

[18] James R. Campbell, Alejandro Lopez Osornio, Fernan de Quiros, Daniel Luna, and Guillermo Reynoso. Semantic interoperability and SNOMED CT: A case study in clinical problem lists. In K.A. Kuhn, J.R. Warren, and T.-Y. Leong, editors, *Proc. of the 12th World Congress on Health (Medical) Informatics (MEDINFO 2007)*, pages 2401–2402. IOS Press, 2007.

[19] Silvio Ghilardi. Best solving modal equations. *Ann. Pure Appl. Logic*, 102(3):183–198, 2000.

[20] Ian Horrocks, Peter F. Patel-Schneider, and Frank van Harmelen. From SHIQ and RDF to OWL: The making of a web ontology language. *Journal of Web Semantics*, 1(1):7–26, 2003.

[21] Ian Horrocks, Ulrike Sattler, and Stefan Tobies. Practical reasoning for very expressive description logics. *J. of the Interest Group in Pure and Applied Logic*, 8(3):239–264, 2000.

[22] Jean-Pierre Jouannaud and Claude Kirchner. Solving equations in abstract algebras: A rule-based survey of unification. In J.-L. Lassez and G. Plotkin, editors, *Computational Logic: Essays in Honor of A. Robinson*. MIT Press, Cambridge, MA, 1991.

[23] Yevgeny Kazakov and Hans de Nivelle. Subsumption of concepts in $\mathcal{FL}_0$ for (cyclic) terminologies with respect to descriptive semantics is PSPACE-complete. In *Proc. of the 2003 Description Logic Workshop (DL 2003)*. CEUR Electronic Workshop Proceedings, http://CEUR-WS.org/Vol-81/, 2003.

[24] Ralf Küsters. *Non-standard Inferences in Description Logics*, volume 2100 of *Lecture Notes in Artificial Intelligence*. Springer-Verlag, 2001.

[25] Hector J. Levesque and Ron J. Brachman. A fundamental tradeoff in knowledge representation and reasoning. In Ron J. Brachman and Hector J. Levesque, editors, *Readings in Knowledge Representation*, pages 41–70. Morgan Kaufmann, Los Altos, 1985.

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

[27] Alan Rector and Ian Horrocks. Experience building a large, re-usable medical ontology using a description logic with transitivity and concept inclusions. In *Proceedings of the Workshop on Ontological Engineering, AAAI Spring Symposium (AAAI'97)*, Stanford, CA, 1997. AAAI Press.

[28] Viorica Sofronie-Stokkermans. Locality and subsumption testing in $\mathcal{EL}$ and some of its extensions. In *Proc. Advances in Modal Logic (AiML'08)*, 2008.

[29] Frank Wolter and Michael Zakharyaschev. Undecidability of the unification and admissibility problems for modal and description logics. *ACM Trans. Comput. Log.*, 9(4), 2008.

Unification in the Description Logic $\mathcal{EL}$ without the Top Concept

Franz Baader[1,⋆], Nguyen Thanh Binh[2], Stefan Borgwardt[1,⋆], and Barbara Morawska[1,⋆]

[1] TU Dresden, Germany
{baader,stefborg,morawska}@tcs.inf.tu-dresden.de
[2] ETH Zürich, Switzerland
thannguy@inf.ethz.ch

Abstract. Unification in Description Logics has been proposed as a novel inference service that can, for example, be used to detect redundancies in ontologies. The inexpressive Description Logic $\mathcal{EL}$ is of particular interest in this context since, on the one hand, several large biomedical ontologies are defined using $\mathcal{EL}$. On the other hand, unification in $\mathcal{EL}$ has recently been shown to be NP-complete, and thus of considerably lower complexity than unification in other DLs of similarly restricted expressive power. However, $\mathcal{EL}$ allows the use of the top concept ($\top$), which represents the whole interpretation domain, whereas the large medical ontology SNOMED CT makes no use of this feature. Surprisingly, removing the top concept from $\mathcal{EL}$ makes the unification problem considerably harder. More precisely, we will show in this paper that unification in $\mathcal{EL}$ without the top concept is PSPACE-complete.

1 Introduction

Description logics (DLs) [4] are a well-investigated family of logic-based knowledge representation formalisms. They can be used to represent the relevant concepts of an application domain using concept terms, which are built from concept names and role names using certain concept constructors. The DL $\mathcal{EL}$ offers the constructors conjunction ($\sqcap$), existential restriction ($\exists r.C$), and the top concept ($\top$). From a semantic point of view, concept names and concept terms represent sets of individuals, whereas roles represent binary relations between individuals. The top concept is interpreted as the set of all individuals. For example, using the concept names Male, Female, Person and the role names child, job, the concept of *persons having a son, a daughter, and a job* can be represented by the $\mathcal{EL}$-concept term $\mathsf{Person} \sqcap \exists \mathsf{child}.\mathsf{Male} \sqcap \exists \mathsf{child}.\mathsf{Female} \sqcap \exists \mathsf{job}.\top$.

In this example, the availability of the top concept in $\mathcal{EL}$ allows us to state that the person has some job, without specifying any further to which concept this job belongs. Knowledge representation systems based on DLs provide their users with various inference services that allow them to deduce implicit knowledge from the explicitly represented knowledge. For instance, the subsumption

⋆ Supported by DFG under grant BA 1122/14-1.

N. Bjørner and V. Sofronie-Stokkermans (Eds.): CADE 2011, LNAI 6803, pp. 70–84, 2011.

algorithm allows one to determine subconcept-superconcept relationships. For example, the concept term $\exists \mathsf{job}.\top$ subsumes (i.e., is a superconcept of) the concept term $\exists \mathsf{job}.\mathsf{Boring}$ since anyone that has a boring job at least has some job. Two concept terms are called *equivalent* if they subsume each other, i.e., if they are always interpreted as the same set of individuals.

The DL $\mathcal{EL}$ has recently drawn considerable attention since, on the one hand, important inference problems such as the subsumption problem are polynomial in $\mathcal{EL}$ [1,3]. On the other hand, though quite inexpressive, $\mathcal{EL}$ can be used to define biomedical ontologies. For example, the large medical ontology SNOMED CT[1] can be expressed in $\mathcal{EL}$. Actually, if one takes a closer look at the concept definitions in SNOMED CT, then one sees that they do not contain the top concept.

Unification in DLs has been proposed in [8] as a novel inference service that can, for example, be used to detect redundancies in ontologies. For example, assume that one knowledge engineer defines the concept of *female professors* as

$$\mathsf{Person} \sqcap \mathsf{Female} \sqcap \exists \mathsf{job}.\mathsf{Professor},$$

whereas another knowledge engineer represent this notion in a somewhat different way, e.g., by using the concept term

$$\mathsf{Woman} \sqcap \exists \mathsf{job}.(\mathsf{Teacher} \sqcap \mathsf{Researcher}).$$

These two concept terms are not equivalent, but they are nevertheless meant to represent the same concept. They can obviously be made equivalent by substituting the concept name $\mathsf{Professor}$ in the first term by the concept term $\mathsf{Teacher} \sqcap \mathsf{Researcher}$ and the concept name Woman in the second term by the concept term $\mathsf{Person} \sqcap \mathsf{Female}$. We call a substitution that makes two concept terms equivalent a *unifier* of the two terms. Such a unifier proposes definitions for the concept names that are used as variables. In our example, we know that, if we define Woman as $\mathsf{Person} \sqcap \mathsf{Female}$ and $\mathsf{Professor}$ as $\mathsf{Teacher} \sqcap \mathsf{Researcher}$, then the two concept terms from above are equivalent w.r.t. these definitions.

In [8] it was shown that, for the DL $\mathcal{FL}_0$, which differs from $\mathcal{EL}$ by offering value restrictions ($\forall r.C$) in place of existential restrictions, deciding unifiability is an ExpTime-complete problem. In [5], we were able to show that unification in $\mathcal{EL}$ is of considerably lower complexity: the decision problem is "only" NP-complete. The original unification algorithm for $\mathcal{EL}$ introduced in [5] was a brutal "guess and then test" NP-algorithm, but we have since then also developed more practical algorithms. On the one hand, in [7] we describe a goal-oriented unification algorithm for $\mathcal{EL}$, in which non-deterministic decisions are only made if they are triggered by "unsolved parts" of the unification problem. On the other hand, in [6], we present an algorithm that is based on a reduction to satisfiability in propositional logic (SAT), and thus allows us to employ highly optimized state-of-the-art SAT solvers for implementing an $\mathcal{EL}$-unification algorithm.

As mentioned above, however, SNOMED CT is not formulated in $\mathcal{EL}$, but rather in its sub-logic $\mathcal{EL}^{-\top}$, which differs from $\mathcal{EL}$ in that the use of the top

[1] See http://www.ihtsdo.org/snomed-ct/

Table 1. Syntax and semantics of $\mathcal{EL}$ and $\mathcal{EL}^{-\top}$

Name	Syntax	Semantics	$\mathcal{EL}$	$\mathcal{EL}^{-\top}$
concept name	A	$A^{\mathcal{I}} \subseteq \mathcal{D}_{\mathcal{I}}$	×	×
role name	r	$r^{\mathcal{I}} \subseteq \mathcal{D}_{\mathcal{I}} \times \mathcal{D}_{\mathcal{I}}$	×	×
top-concept	$\top$	$\top^{\mathcal{I}} = \mathcal{D}_{\mathcal{I}}$	×	
conjunction	$C \sqcap D$	$(C \sqcap D)^{\mathcal{I}} = C^{\mathcal{I}} \cap D^{\mathcal{I}}$	×	×
existential restriction	$\exists r.C$	$(\exists r.C)^{\mathcal{I}} = \{x \mid \exists y : (x, y) \in r^{\mathcal{I}} \wedge y \in C^{\mathcal{I}}\}$	×	×
subsumption	$C \sqsubseteq D$	$C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$	×	×
equivalence	$C \equiv D$	$C^{\mathcal{I}} = D^{\mathcal{I}}$	×	×

concept is disallowed. If we employ $\mathcal{EL}$-unification to detect redundancies in (extensions of) SNOMED CT, then a unifier may introduce concept terms that contain the top concept, and thus propose definitions for concept names that are of a form that is not used in SNOMED CT. Apart from this practical motivation for investigating unification in $\mathcal{EL}^{-\top}$, we also found it interesting to see how such a small change in the logic influences the unification problem. Surprisingly, it turned out that the complexity of the problem increases considerably (from NP to PSPACE). In addition, compared to $\mathcal{EL}$-unification, quite different methods had to be developed to actually solve $\mathcal{EL}^{-\top}$-unification problems. In particular, we will show in this paper, that—similar to the case of $\mathcal{FL}_0$-unification—$\mathcal{EL}^{-\top}$-unification can be reduced to solving certain language equations. In contrast to the case of $\mathcal{FL}_0$-unification, these language equations can be solved in PSPACE rather than EXPTIME, which we show by a reduction to the emptiness problem for alternating automata on finite words. Complete proofs of the results presented in this paper can be found in [2].

2 The Description Logics $\mathcal{EL}$ and $\mathcal{EL}^{-\top}$

Starting with a set N_C of concept names and a set N_R of role names, *$\mathcal{EL}$-concept terms* are built using the concept constructors *top-concept* ($\top$), *conjunction* ($C \sqcap D$), and *existential restriction* ($\exists r.C$ for every $r \in N_R$). The $\mathcal{EL}$-concept term C is an *$\mathcal{EL}^{-\top}$-concept term* if $\top$ does not occur in C. Since $\mathcal{EL}^{-\top}$-concept terms are special $\mathcal{EL}$-concept terms, many definitions and results transfer from $\mathcal{EL}$ to $\mathcal{EL}^{-\top}$, and thus we only formulate them for $\mathcal{EL}$. We will explicitly mention it if this is not the case.

The semantics of $\mathcal{EL}$ and $\mathcal{EL}^{-\top}$ is defined in the usual way, using the notion of an interpretation $\mathcal{I} = (\mathcal{D}_{\mathcal{I}}, \cdot^{\mathcal{I}})$, which consists of a nonempty domain $\mathcal{D}_{\mathcal{I}}$ and an interpretation function $\cdot^{\mathcal{I}}$ that assigns binary relations on $\mathcal{D}_{\mathcal{I}}$ to role names and subsets of $\mathcal{D}_{\mathcal{I}}$ to concept terms, as shown in the semantics column of Table 1.

The concept term C *is subsumed by* the concept term D (written $C \sqsubseteq D$) iff $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ holds for all interpretations $\mathcal{I}$. We say that C *is equivalent to* D (written $C \equiv D$) iff $C \sqsubseteq D$ and $D \sqsubseteq C$, i.e., iff $C^{\mathcal{I}} = D^{\mathcal{I}}$ holds for all interpretations $\mathcal{I}$.

An $\mathcal{EL}$-concept term is called an *atom* iff it is a concept name $A \in N_C$ or an existential restriction $\exists r.D$. Concept names and existential restrictions $\exists r.D$, where D is a concept name or $\top$, are called *flat atoms*. The set $\mathrm{At}(C)$ of *atoms of an $\mathcal{EL}$-concept term C* consists of all the subterms of C that are atoms. For example, $C = A \sqcap \exists r.(B \sqcap \exists r.\top)$ has the atom set $\mathrm{At}(C) = \{A, \exists r.(B \sqcap \exists r.\top), B, \exists r.\top\}$. Obviously, any $\mathcal{EL}$-concept term C is a conjunction $C = C_1 \sqcap \ldots \sqcap C_n$ of atoms and $\top$. We call the atoms among $C_1, \ldots, C_n$ the *top-level atoms* of C. The $\mathcal{EL}$-concept term C is called *flat* if all its top-level atoms are flat. Subsumption in $\mathcal{EL}$ and $\mathcal{EL}^{-\top}$ can be characterized as follows [7]:

Lemma 1. *Let $C = A_1 \sqcap \ldots \sqcap A_k \sqcap \exists r_1.C_1 \sqcap \ldots \sqcap \exists r_m.C_m$ and $D = B_1 \sqcap \ldots \sqcap B_l \sqcap \exists s_1.D_1 \sqcap \ldots \sqcap \exists s_n.D_n$ be two $\mathcal{EL}$-concept terms, where $A_1, \ldots, A_k, B_1, \ldots, B_l$ are concept names. Then $C \sqsubseteq D$ iff $\{B_1, \ldots, B_l\} \subseteq \{A_1, \ldots, A_k\}$ and for every $j \in \{1, \ldots, n\}$ there exists an $i \in \{1, \ldots, m\}$ such that $r_i = s_j$ and $C_i \sqsubseteq D_j$.*

In particular, this means that $C \sqsubseteq D$ iff for every top-level atom D' of D there is a top-level atom C' of C such that $C' \sqsubseteq D'$.

Modulo equivalence, the subsumption relation is a partial order on concept terms. In $\mathcal{EL}$, the top concept $\top$ is the greatest element w.r.t. this order. In $\mathcal{EL}^{-\top}$, there are many incomparable maximal concept terms. We will see below that these are exactly the $\mathcal{EL}^{-\top}$-concept terms of the form $\exists r_1. \cdots \exists r_n.A$ for $n \geq 0$ role names $r_1, \ldots, r_n$ and a concept name A. We call such concept terms *particles* . The set $\mathrm{Part}(C)$ of all particles of a given $\mathcal{EL}^{-\top}$-concept term C is defined as

- $\mathrm{Part}(C) := \{C\}$ if C is a concept name,
- $\mathrm{Part}(C) := \{\exists r.E \mid E \in \mathrm{Part}(D)\}$ if $C = \exists r.D$,
- $\mathrm{Part}(C) := \mathrm{Part}(C_1) \cup \mathrm{Part}(C_2)$ if $C = C_1 \sqcap C_2$.

For example, the particles of $C = A \sqcap \exists r.(A \sqcap \exists r.B)$ are $A, \exists r.A, \exists r.\exists r.B$. Such particles will play an important role in our $\mathcal{EL}^{-\top}$-unification algorithm. The next lemma states that particles are indeed the maximal concept terms w.r.t. to subsumption in $\mathcal{EL}^{-\top}$, and that the particles subsuming an $\mathcal{EL}^{-\top}$-concept term C are exactly the particles of C.

Lemma 2. *Let C be an $\mathcal{EL}^{-\top}$-concept term and B a particle.*

1. *If $B \sqsubseteq C$, then $B \equiv C$.*
2. *$B \in \mathrm{Part}(C)$ iff $C \sqsubseteq B$.*

3 Unification in $\mathcal{EL}$ and $\mathcal{EL}^{-\top}$

To define unification in $\mathcal{EL}$ and $\mathcal{EL}^{-\top}$ simultaneously, let $\mathcal{L} \in \{\mathcal{EL}, \mathcal{EL}^{-\top}\}$. When defining unification in $\mathcal{L}$, we assume that the set of concepts names is

partitioned into a set N_v of concept variables (which may be replaced by substitutions) and a set N_c of concept constants (which must not be replaced by substitutions). An $\mathcal{L}$-*substitution* σ is a mapping from N_v into the set of all $\mathcal{L}$-concept terms. This mapping is extended to concept terms in the usual way, i.e., by replacing all occurrences of variables in the term by their σ-images. An $\mathcal{L}$-concept term is called *ground* if it contains no variables, and an $\mathcal{L}$-substitution σ is called *ground* if the concept terms $\sigma(X)$ are ground for all $X \in N_v$.

Unification tries to make concept terms equivalent by applying a substitution.

Definition 1. *An* $\mathcal{L}$-unification problem *is of the form* $\Gamma = \{C_1 \equiv^? D_1, \ldots, C_n \equiv^? D_n\}$, *where* $C_1, D_1, \ldots C_n, D_n$ *are* $\mathcal{L}$*-concept terms. The* $\mathcal{L}$*-substitution* σ *is an* $\mathcal{L}$-unifier *of* Γ *iff it solves all the equations* $C_i \equiv^? D_i$ *in* Γ, *i.e., iff* $\sigma(C_i) \equiv \sigma(D_i)$ *for* $i = 1, \ldots, n$. *In this case,* Γ *is called* $\mathcal{L}$-unifiable.

In the following, we will use the subsumption $C \sqsubseteq^? D$ as an abbreviation for the equation $C \sqcap D \equiv^? C$. Obviously, σ solves this equation iff $\sigma(C) \sqsubseteq \sigma(D)$.

Clearly, every $\mathcal{EL}^{-\top}$-unification problem Γ is also an $\mathcal{EL}$-unification problem. Whether Γ is $\mathcal{L}$-unifiable or not may depend, however, on whether $\mathcal{L} = \mathcal{EL}$ or $\mathcal{L} = \mathcal{EL}^{-\top}$. As an example, consider the problem $\Gamma := \{A \sqsubseteq^? X, B \sqsubseteq^? X\}$, where A, B are distinct concept constants and X is a concept variable. Obviously, the substitution that replaces X by $\top$ is an $\mathcal{EL}$-unifier of Γ. However, Γ does not have an $\mathcal{EL}^{-\top}$-unifier. In fact, for such a unifier σ, the $\mathcal{EL}^{-\top}$-concept term $\sigma(X)$ would need to satisfy $A \sqsubseteq \sigma(X)$ and $B \sqsubseteq \sigma(X)$. Since A and B are particles, Lemma 2 would imply $A \equiv \sigma(X) \equiv B$ and thus $A \equiv B$, which is not the case.

It is easy to see that, for both $\mathcal{L} = \mathcal{EL}$ and $\mathcal{L} = \mathcal{EL}^{-\top}$, an $\mathcal{L}$-unification problem Γ has an $\mathcal{L}$-unifier iff it has a ground $\mathcal{L}$-unifier σ that uses only concept and role names occurring in Γ,[2] i.e., for all variables X, the $\mathcal{L}$-concept term $\sigma(X)$ is a ground term that contains only such concept and role names. In addition, we may without loss of generality restrict our attention to *flat* $\mathcal{L}$*-unification problems*, i.e., unification problems in which the left- and right-hand sides of equations are flat $\mathcal{L}$-concept terms (see, e.g., [7]).

Given a flat $\mathcal{L}$-unification problem Γ, we denote by $\mathrm{At}(\Gamma)$ the set of all atoms of Γ, i.e., the union of all sets of atoms of the concept terms occurring in Γ. By $\mathrm{Var}(\Gamma)$ we denote the variables that occur in Γ, and by $\mathrm{NV}(\Gamma) := \mathrm{At}(\Gamma) \setminus \mathrm{Var}(\Gamma)$ the set of all *non-variable atoms* of Γ.

$\mathcal{EL}$-unification by guessing acyclic assignments

The NP-algorithm for $\mathcal{EL}$-unification introduced in [5] guesses, for every variable X occurring in Γ, a set $S(X)$ of non-variable atoms of Γ. Given such an *assignment* of sets of non-variable atoms to the variables in Γ, we say that the variable X *directly depends on* the variable Y if Y occurs in an atom of $S(X)$. Let *depends on* be the transitive closure of *directly depends on*. If there is no variable that depends on itself, then we call this assignment *acyclic*. In case the guessed assignment is not acyclic, this run of the NP-algorithm returns "fail."

[2] Without loss of generality, we assume that Γ contains at least one concept name.

Otherwise, there exists a strict linear order $>$ on the variables occurring in Γ such that $X > Y$ if X depends on Y. One can then define the substitution γ^S induced by the assignment S along this linear order:

- If X is the least variable w.r.t. $>$, then $\gamma^S(X)$ is the conjunction of the elements of $S(X)$, where the empty conjunction is $\top$.
- Assume $\gamma^S(Y)$ is defined for all variables $Y < X$. If $S(X) = \{D_1, \ldots, D_n\}$, then $\gamma^S(X) := \gamma^S(D_1) \sqcap \ldots \sqcap \gamma^S(D_n)$.

The algorithm then tests whether the substitution γ^S computed this way is a unifier of Γ. If this is the case, then this run returns γ^S; otherwise, it returns "fail." In [5] it is shown that Γ is unifiable iff there is a run of this algorithm on input Γ that returns a substitution (which is then an $\mathcal{EL}$-unifier of Γ).

Why this does not work for $\mathcal{EL}^{-\top}$

The $\mathcal{EL}$-unifiers returned by the $\mathcal{EL}$-unification algorithm sketched above need not be $\mathcal{EL}^{-\top}$-unifiers since some of the sets $S(X)$ in the guessed assignment may be empty, in which case $\gamma^S(X) = \top$. This suggests the following simple modification of the above algorithm: require that the guessed assignment is such that all sets $S(X)$ are nonempty. If such an assignment S is acyclic, then the induced substitution γ^S is actually an $\mathcal{EL}^{-\top}$-substitution, and thus the substitutions returned by the modified algorithm are indeed $\mathcal{EL}^{-\top}$-unifiers. However, this modified algorithm does not always detect $\mathcal{EL}^{-\top}$-unifiability, i.e., it may return no substitution although the input problem is $\mathcal{EL}^{-\top}$-unifiable.

As an example, consider the $\mathcal{EL}^{-\top}$-unification problem

$$\Gamma := \{A \sqcap B \equiv^? Y,\ B \sqcap C \equiv^? Z,\ \exists r.Y \sqsubseteq^? X,\ \exists r.Z \sqsubseteq^? X\},$$

where X, Y, Z are concept variables and A, B, C are distinct concept constants. We claim that, up to equivalence, the substitution that maps X to $\exists r.B$, Y to $A \sqcap B$, and Z to $B \sqcap C$ is the only $\mathcal{EL}^{-\top}$-unifier of Γ. In fact, any $\mathcal{EL}^{-\top}$-unifier γ of Γ must map Y to $A \sqcap B$ and Z to $B \sqcap C$, and thus satisfy $\exists r.(A \sqcap B) \sqsubseteq \gamma(X)$ and $\exists r.(B \sqcap C) \sqsubseteq \gamma(X)$. Lemma 1 then yields that the only possible top-level atom of $\gamma(X)$ is $\exists r.B$. However, there is no non-variable atom $D \in \mathrm{NV}(\Gamma)$ such that $\gamma(D)$ is equivalent to $\exists r.B$. This shows that Γ has an $\mathcal{EL}^{-\top}$-unifier, but this unifier cannot be computed by the modified algorithm sketched above.

The main idea underlying the $\mathcal{EL}^{-\top}$-unification algorithm introduced in the next section is that one starts with an $\mathcal{EL}$-unifier, and then conjoins "appropriate" particles to the images of the variables that are replaced by $\top$ by this unifier. It is, however, not so easy to decide which particles can be added this way without turning the $\mathcal{EL}$-unifier into an $\mathcal{EL}^{-\top}$-substitution that no longer solves the unification problem.

4 An $\mathcal{EL}^{-\top}$-Unification Algorithm

In the following, let Γ be a flat $\mathcal{EL}^{-\top}$-unification problem. Without loss of generality we assume that Γ consists of subsumptions of the form $C_1 \sqcap \ldots \sqcap C_n \sqsubseteq^? D$

for atoms $C_1, \ldots, C_n, D$. Our decision procedure for $\mathcal{EL}^{-\top}$-unifiability proceeds in four steps.

Step 1. If S is an acyclic assignment guessed by the $\mathcal{EL}$-unification algorithm sketched above, then $D \in S(X)$ implies that the subsumption $\gamma^S(X) \sqsubseteq \gamma^S(D)$ holds for the substitution γ^S induced by S. Instead of guessing just subsumptions between variables and non-variable atoms, our $\mathcal{EL}^{-\top}$-unification algorithm starts with guessing subsumptions between arbitrary atoms of Γ. To be more precise, it guesses a mapping $\tau : \mathrm{At}(\Gamma)^2 \to \{0, 1\}$, which specifies which subsumptions between atoms of Γ should hold for the $\mathcal{EL}^{-\top}$-unifier that it tries to generate: if $\tau(D_1, D_2) = 1$ for $D_1, D_2 \in \mathrm{At}(\Gamma)$, then this means that the search for a unifier is restricted (in this branch of the search tree) to substitutions γ satisfying $\gamma(D_1) \sqsubseteq \gamma(D_2)$. Obviously, any such mapping τ also yields an assignment

$$S^\tau(X) := \{D \in \mathrm{NV}(\Gamma) \mid \tau(X, D) = 1\},$$

and we require that this assignment is acyclic and induces an $\mathcal{EL}$-unifier of Γ.

Definition 2. *The mapping $\tau : \mathrm{At}(\Gamma)^2 \to \{0, 1\}$ is called a* subsumption mapping for Γ *if it satisfies the following three conditions:*

1. *It respects the properties of subsumption in $\mathcal{EL}$:*
   (a) $\tau(D, D) = 1$ *for each* $D \in \mathrm{At}(\Gamma)$.
   (b) $\tau(A_1, A_2) = 0$ *for distinct concept constants* $A_1, A_2 \in \mathrm{At}(\Gamma)$.
   (c) $\tau(\exists r.C_1, \exists s.C_2) = 0$ *for distinct* $r, s \in N_R$ *with* $\exists r.C_1, \exists s.C_2 \in \mathrm{At}(\Gamma)$.
   (d) $\tau(A, \exists r.C) = \tau(\exists r.C, A) = 0$ *for each constant* $A \in \mathrm{At}(\Gamma)$, *role name* r *and variable or constant* C *with* $\exists r.C \in \mathrm{At}(\Gamma)$.
   (e) *If* $\exists r.C_1, \exists r.C_2 \in \mathrm{At}(\Gamma)$, *then* $\tau(\exists r.C_1, \exists r.C_2) = \tau(C_1, C_2)$.
   (f) *For all atoms* $D_1, D_2, D_3 \in \mathrm{At}(\Gamma)$, *if* $\tau(D_1, D_2) = \tau(D_2, D_3) = 1$, *then* $\tau(D_1, D_3) = 1$.
2. *It induces an $\mathcal{EL}$-substitution, i.e., the assignment S^τ is acyclic and thus induces a substitution γ^{S^τ}, which we will simply denote by γ^τ.*
3. *It respects the subsumptions of Γ, i.e., it satisfies the following conditions for each subsumption $C_1 \sqcap \ldots \sqcap C_n \sqsubseteq^? D$ in Γ:*
   (a) *If D is a non-variable atom, then there is at least one C_i such that $\tau(C_i, D) = 1$.*
   (b) *If D is a variable and $\tau(D, C) = 1$ for a non-variable atom $C \in \mathrm{NV}(\Gamma)$, then there is at least one C_i with $\tau(C_i, C) = 1$.*

Though this is not really necessary for the proof of correctness of our $\mathcal{EL}^{-\top}$-unification algorithm, it can be shown that the substitution γ^τ induced by a subsumption mapping τ for Γ is indeed an $\mathcal{EL}$-unifier of Γ. It should be noted that γ^τ need not be an $\mathcal{EL}^{-\top}$-unifier of Γ. In addition, γ^τ need not agree with τ on every subsumption between atoms of Γ. The reason for this is that τ specifies subsumptions which should hold in the $\mathcal{EL}^{-\top}$-unifier of Γ to be constructed. To turn γ^τ into such an $\mathcal{EL}^{-\top}$-unifier, we may have to add certain particles, and these additions may invalidate subsumptions that hold for γ^τ. However, we will ensure that no subsumption claimed by τ is invalidated.

Step 2. In this step, we use τ to turn Γ into a unification problem that has only variables on the right-hand sides of subsumptions. More precisely, we define $\Delta_{\Gamma,\tau} := \Delta_\Gamma \cup \Delta_\tau$, where

$$\Delta_\Gamma := \{C_1 \sqcap \ldots \sqcap C_n \sqsubseteq^? X \in \Gamma \mid X \text{ is a variable of } \Gamma\},$$
$$\Delta_\tau := \{C \sqsubseteq^? X \mid X \text{ is a variable and } C \text{ an atom of } \Gamma \text{ with } \tau(C, X) = 1\}.$$

For an arbitrary $\mathcal{EL}^{-\top}$-substitution σ, we define

$$S^\sigma(X) := \{D \in \mathrm{NV}(\Gamma) \mid \sigma(X) \sqsubseteq \sigma(D)\},$$

and write $S^\tau \leq S^\sigma$ if $S^\tau(X) \subseteq S^\sigma(X)$ for every variable X. The following lemma states the connection between $\mathcal{EL}^{-\top}$-unifiability of Γ and of $\Delta_{\Gamma,\tau}$, using the notation that we have just introduced.

Lemma 3. *Let Γ be a flat $\mathcal{EL}^{-\top}$-unification problem. Then the following statements are equivalent for any $\mathcal{EL}^{-\top}$-substitution σ:*

1. *σ is an $\mathcal{EL}^{-\top}$-unifier of Γ.*
2. *There is a subsumption mapping $\tau : \mathrm{At}(\Gamma)^2 \to \{0, 1\}$ for Γ such that σ is an $\mathcal{EL}^{-\top}$-unifier of $\Delta_{\Gamma,\tau}$ and $S^\tau \leq S^\sigma$.*

Step 3. In this step, we characterize which particles can be added in order to turn γ^τ into an $\mathcal{EL}^{-\top}$-unifier σ of $\Delta_{\Gamma,\tau}$ satisfying $S^\tau \leq S^\sigma$. Recall that particles are of the form $\exists r_1. \cdots \exists r_n.A$ for $n \geq 0$ role names $r_1, \ldots, r_n$ and a concept name A. We write such a particle as $\exists w.A$, where $w = r_1 \cdots r_n$ is viewed as a word over the alphabet N_R of all role names. If $n = 0$, then w is the empty word ε and $\exists \varepsilon.A$ is just A.

Admissible particles are determined by solutions of a system of linear language inclusions. These *linear inclusions* are of the form

$$X_i \subseteq L_0 \cup L_1 X_1 \cup \ldots \cup L_n X_n, \tag{1}$$

where $X_1, \ldots, X_n$ are indeterminates, $i \in \{1, \ldots, n\}$, and each L_i ($i \in \{0, \ldots, n\}$) is a subset of $N_R \cup \{\varepsilon\}$. A *solution* θ of such an inclusion assigns sets of words $\theta(X_i) \subseteq N_R^*$ to the indeterminates X_i such that $\theta(X_i) \subseteq L_0 \cup L_1\theta(X_1) \cup \ldots \cup L_n\theta(X_n)$.

The unification problem $\Delta_{\Gamma,\tau}$ induces a finite system $\mathcal{I}_{\Gamma,\tau}$ of such inclusions. The indeterminates of $\mathcal{I}_{\Gamma,\tau}$ are of the form X_A, where $X \in N_v$ and $A \in N_c$. For each constant $A \in N_c$ and each subsumption of the form $C_1 \sqcap \ldots \sqcap C_n \sqsubseteq^? X \in \Delta_{\Gamma,\tau}$, we add the following inclusion to $\mathcal{I}_{\Gamma,\tau}$:

$$X_A \subseteq f_A(C_1) \cup \ldots \cup f_A(C_n), \text{ where}$$
$$f_A(C) := \begin{cases} \{r\} f_A(C') & \text{if } C = \exists r.C' \\ Y_A & \text{if } C = Y \text{ is a variable} \\ \{\varepsilon\} & \text{if } C = A \\ \emptyset & \text{if } C \in N_c \setminus \{A\} \end{cases}$$

Since $\Delta_{\Gamma,\tau}$ contains only flat atoms, these inclusion are indeed of the form (1).

We call a solution θ of $\mathcal{I}_{\Gamma,\tau}$ *admissible* if, for every variable $X \in N_v$, there is a constant $A \in N_c$ such that $\theta(X_A)$ is nonempty. This condition will ensure that we can add enough particles to turn γ^τ into an $\mathcal{EL}^{-\top}$-substitution. In order to obtain a substitution at all, only finitely many particles can be added. Thus, we are interested in *finite* solutions of $\mathcal{I}_{\Gamma,\tau}$, i.e., solutions θ such that all the sets $\theta(X_A)$ are finite.

Lemma 4. *Let Γ be a flat $\mathcal{EL}^{-\top}$-unification problem and τ a subsumption mapping for Γ. Then $\Delta_{\Gamma,\tau}$ has an $\mathcal{EL}^{-\top}$-unifier σ with $S^\tau \leq S^\sigma$ iff $\mathcal{I}_{\Gamma,\tau}$ has a finite, admissible solution.*

Proof sketch. Given a ground $\mathcal{EL}^{-\top}$-unifier σ of $\Delta_{\Gamma,\tau}$ with $S^\tau \leq S^\sigma$, we define for each concept variable X and concept constant A occurring in Γ:

$$\theta(X_A) := \{w \in N_R^* \mid \exists w.A \in \mathrm{Part}(\sigma(X))\}.$$

It can then be shown that θ is a solution of $\mathcal{I}_{\Gamma,\tau}$. This solution is finite since any concept term has only finitely many particles, and it is admissible since σ is an $\mathcal{EL}^{-\top}$-substitution.

Conversely, let θ be a finite, admissible solution of $\mathcal{I}_{\Gamma,\tau}$. We define the substitution σ by induction on the dependency order $>$ induced by S^τ as follows. Let X be a variable of Γ and assume that $\sigma(Y)$ has already been defined for all variables Y with $X > Y$. Then we set

$$\sigma(X) := \prod_{D \in S^\tau(X)} \sigma(D) \sqcap \prod_{A \in N_c} \prod_{w \in \theta(X_A)} \exists w.A.$$

Since θ is finite and admissible, σ is a well-defined $\mathcal{EL}^{-\top}$-substitution. It can be shown that $\sigma(X)$ is indeed an $\mathcal{EL}^{-\top}$-unifier of $\Delta_{\Gamma,\tau}$ with $S^\tau \leq S^\sigma$. □

Step 4. In this step we show how to test whether the system $\mathcal{I}_{\Gamma,\tau}$ of linear language inclusions constructed in the previous step has a finite, admissible solution or not. The main idea is to consider the greatest solution of $\mathcal{I}_{\Gamma,\tau}$.

To be more precise, given a system of linear language inclusions $\mathcal{I}$, we can order the solutions of $\mathcal{I}$ by defining $\theta_1 \subseteq \theta_2$ iff $\theta_1(X) \subseteq \theta_2(X)$ for all indeterminates X of $\mathcal{I}$. Since $\theta_\emptyset$, which assigns the empty set to each indeterminate of $\mathcal{I}$, is a solution of $\mathcal{I}$ and solutions are closed under argument-wise union, the following clearly defines the (unique) greatest solution θ^* of $\mathcal{I}$ w.r.t. this order:

$$\theta^*(X) := \bigcup_{\theta \text{ solution of } \mathcal{I}} \theta(X).$$

Lemma 5. *Let X be an indeterminate in $\mathcal{I}$ and θ^* the maximal solution of $\mathcal{I}$. If $\theta^*(X)$ is nonempty, then there is a finite solution θ of $\mathcal{I}$ such that $\theta(X)$ is nonempty.*

Proof. Let $w \in \theta^*(X)$. We construct the finite solution θ of $\mathcal{I}$ by keeping only the words of length $|w|$: for all indeterminates Y occurring in $\mathcal{I}$ we define

$$\theta(Y) := \{u \in \theta^*(Y) \mid |u| \leq |w|\}.$$

By definition, we have $w \in \theta(X)$. To show that θ is indeed a solution of $\mathcal{I}$, consider an arbitrary inclusion $Y \subseteq L_0 \cup L_1X_1 \cup \ldots \cup L_nX_n$ in $\mathcal{I}$, and assume that $u \in \theta(Y)$. We must show that $u \in L_0 \cup L_1\theta(X_1) \cup \ldots \cup L_n\theta(X_n)$. Since $u \in \theta^*(Y)$ and θ^* is a solution of $\mathcal{I}$, we have (i) $u \in L_0$ or (ii) $u \in L_i\theta^*(X_i)$ for some $i, 1 \leq i \leq n$. In the first case, we are done. In the second case, $u = \alpha u'$ for some $\alpha \in L_i \subseteq N_R \cup \{\varepsilon\}$ and $u' \in \theta^*(X_i)$. Since $|u'| \leq |u| \leq |w|$, we have $u' \in \theta(X_i)$, and thus $u \in L_i\theta(X_i)$. □

Lemma 6. *There is a finite, admissible solution of $\mathcal{I}_{\Gamma,\tau}$ iff the maximal solution θ^* of $\mathcal{I}_{\Gamma,\tau}$ is admissible.*

Proof. If $\mathcal{I}_{\Gamma,\tau}$ has a finite, admissible solution θ, then the maximal solution of $\mathcal{I}_{\Gamma,\tau}$ contains this solution, and is thus also admissible.

Conversely, if θ^* is admissible, then (by Lemma 5) for each $X \in \mathrm{Var}(\Gamma)$ there is a constant $A(X)$ and a finite solution θ_X of $\mathcal{I}_{\Gamma,\tau}$ such that $\theta_X(X_{A(X)}) \neq \emptyset$. The union of these solutions θ_X for $X \in \mathrm{Var}(\Gamma)$ is the desired finite, admissible solution. □

Given this lemma, it remains to show how we can test admissibility of the maximal solution θ^* of $\mathcal{I}_{\Gamma,\tau}$. For this purpose, it is obviously sufficient to be able to test, for each indeterminate X_A in $\mathcal{I}_{\Gamma,\tau}$, whether $\theta^*(X_A)$ is empty or not. This can be achieved by representing the languages $\theta^*(X_A)$ using *alternating finite automata with ε-transitions (ε-AFA)*, which are a special case of two-way alternating finite automata. In fact, as shown in [11], the emptiness problem for two-way alternating finite automata (and thus also for ε-AFA) is in PSPACE.

Lemma 7. *For each indeterminate X_A in $\mathcal{I}_{\Gamma,\tau}$, we can construct in polynomial time in the size of $\mathcal{I}_{\Gamma,\tau}$ an ε-AFA $\mathcal{A}(X, A)$ such that the language $L(\mathcal{A}(X, A))$ accepted by $\mathcal{A}(X, A)$ is equal to $\theta^*(X_A)$, where θ^* denotes the maximal solution of $\mathcal{I}_{\Gamma,\tau}$.*

This finishes the description of our $\mathcal{EL}^{-\top}$-unification algorithm. It remains to argue why it is a PSPACE decision procedure for $\mathcal{EL}^{-\top}$-unifiability.

Theorem 1. *The problem of deciding unifiability in $\mathcal{EL}^{-\top}$ is in* PSPACE.

Proof. We show that the problem is in NPSPACE, which is equal to PSPACE by Savitch's theorem [14].

Let Γ be a flat $\mathcal{EL}^{-\top}$-unification problem. By Lemma 3, Lemma 4, and Lemma 6, we know that Γ is $\mathcal{EL}^{-\top}$-unifiable iff there is a subsumption mapping τ for Γ such that the maximal solution θ^* of $\mathcal{I}_{\Gamma,\tau}$ is admissible.

Thus, we first guess a mapping $\tau : \mathrm{At}(\Gamma)^2 \to \{0, 1\}$ and test whether τ is a subsumption mapping for Γ. Guessing τ can clearly be done in NPSPACE. For

a given mapping τ, the test whether it is a subsumption mapping for Γ can be done in polynomial time.

From τ we can first construct $\Delta_{\Gamma,\tau}$ and then $\mathcal{I}_{\Gamma,\tau}$ in polynomial time. Given $\mathcal{I}_{\Gamma,\tau}$, we then construct the (polynomially many) ε-AFA $\mathcal{A}(X, A)$, and test them for emptiness. Since the emptiness problem for ε-AFA is in PSPACE, this can be achieved within PSPACE. Given the results of these emptiness tests, we can then check in polynomial time whether, for each concept variable X of Γ there is a concept constant A of Γ such that $\theta^*(X_A) = L(\mathcal{A}(X, A)) \neq \emptyset$. If this is the case, then θ^* is admissible, and thus Γ is $\mathcal{EL}^{-\top}$-unifiable. □

5 PSpace-Hardness of $\mathcal{EL}^{-\top}$-Unification

We show PSPACE-hardness of $\mathcal{EL}^{-\top}$-unification by reducing the PSPACE-hard intersection emptiness problem for deterministic finite automata (DFA) [12,9] to the problem of deciding whether a given $\mathcal{EL}^{-\top}$-unification problem has an $\mathcal{EL}^{-\top}$-unifier or not.

First, we define a translation from a given DFA $\mathcal{A} = (Q, \Sigma, q_0, \delta, F)$ to a set of subsumptions $\Gamma_\mathcal{A}$. In the following, we only consider automata that accept a nonempty language. For such DFAs we can assume without loss of generality that there is no state $q \in Q$ that cannot be reached from q_0 or from which F cannot be reached. In fact, such states can be removed from $\mathcal{A}$ without changing the accepted language.

For every state $q \in Q$, we introduce a concept variable X_q. We use only one concept constant, A, and define $N_R := \Sigma$. The set $\Gamma_\mathcal{A}$ is defined as follows:

$$\Gamma_\mathcal{A} := \{L_q \sqsubseteq^? X_q \mid q \in Q \setminus F\} \cup \{A \sqcap L_q \sqsubseteq^? X_q \mid q \in F\}, \text{ where}$$
$$L_q := \prod_{\substack{\alpha \in \Sigma \\ \delta(q,\alpha) \text{ is defined}}} \exists \alpha.X_{\delta(q,\alpha)}.$$

Note that the left-hand sides of the subsumptions in $\Gamma_\mathcal{A}$ are indeed $\mathcal{EL}^{-\top}$-concept terms, i.e., the conjunctions on the left-hand sides are nonempty. In fact, every state $q \in Q$ is either a final state or a final state is reachable by a nonempty path from q. In the first case, A occurs in the conjunction, and in the second, there must be an $\alpha \in \Sigma$ such that $\delta(q, \alpha)$ is defined, in which case $\exists \alpha.X_{\delta(q,\alpha)}$ occurs in the conjunction.

The following lemma, which can easily be proved by induction on $|w|$, connects particles occurring in $\mathcal{EL}^{-\top}$-unifiers of $\Gamma_\mathcal{A}$ to words accepted by states of the DFA $\mathcal{A}$.

Lemma 8. *Let $q \in Q$, $w \in \Sigma^*$, and γ be a ground $\mathcal{EL}^{-\top}$-unifier of $\Gamma_\mathcal{A}$ with $\gamma(X_q) \sqsubseteq \exists w.A$. Then $w \in L(\mathcal{A}_q)$, where $\mathcal{A}_q := (Q, \Sigma, q, \delta, F)$ is obtained from $\mathcal{A}$ by making q the initial state.*

Together with Lemma 2, this lemma implies that, for every ground $\mathcal{EL}^{-\top}$-unifier γ of $\Gamma_\mathcal{A}$, the language $\{w \in \Sigma^* \mid \exists w.A \in \text{Part}(\gamma(X_{q_0}))\}$ is contained in $L(\mathcal{A})$.

Conversely, we will show that for every word w accepted by $\mathcal{A}$ we can construct a unifier γ_w such that $\exists w.A \in \mathrm{Part}(\gamma_w(X_{q_0}))$.

For the construction of γ_w, we first consider every $q \in Q$ and try to find a word u_q of minimal length that is accepted by $\mathcal{A}_q$. Such a word always exists since we have assumed that we can reach F from every state. Taking arbitrary such words is not sufficient, however. They need to be related in the following sense.

Lemma 9. *There exists a mapping from the states $q \in Q$ to words $u_q \in L(\mathcal{A}_q)$ such that that either $q \in F$ and $u_q = \varepsilon$ or there is a symbol $\alpha \in \Sigma$ such that $\delta(q, \alpha)$ is defined and $u_q = \alpha u_{\delta(q,\alpha)}$.*

Proof. We construct the words u_q by induction on the length n of a shortest word accepted by $\mathcal{A}_q$.

If $n = 0$, then q must be a final state. In this case, we set $u_q := \varepsilon$.

Now, let q be a state such that a shortest word w_q accepted by $\mathcal{A}_q$ has length $n > 0$. Then $w_q = \alpha w'$ for $\alpha \in \Sigma$ and $w' \in \Sigma^*$ and the transition $\delta(q, \alpha) = q'$ is defined. The length of a shortest word accepted by $\mathcal{A}_{q'}$ must be smaller than n, since w' is accepted by $\mathcal{A}_{q'}$. By induction, $u_{q'} \in L(\mathcal{A}_{q'})$ has already been defined and we have $\alpha u_{q'} \in L(\mathcal{A}_q)$. Since $\alpha u_{q'}$ cannot be shorter than $w_q = \alpha w'$, it must also be of length n. We now define $u_q := \alpha u_{q'}$. □

We can now proceed with the definition of γ_w for a word $w \in L(\mathcal{A})$. The (unique) accepting run of $\mathcal{A}$ on $w = w_1 \dots w_n$ yields a sequence of states $q_0, q_1, \dots, q_n$ with $q_n \in F$ and $\delta(q_i, w_{i+1}) = q_{i+1}$ for every $i \in \{0, \dots, n-1\}$. We define the substitution γ_w as follows:

$$\gamma_w(X_q) := \exists u_q.A \sqcap \bigsqcap_{i \in I_q} \exists w_{i+1} \dots w_n.A,$$

where $I_q := \{i \in \{0, \dots, n-1\} \mid q_i = q\}$. For every $q \in Q$, we include at least the conjunct $\exists u_q.A$ in $\gamma_w(X_q)$, and thus γ_w is in fact an $\mathcal{EL}^{-\top}$-substitution.

Lemma 10. *If $w \in L(\mathcal{A})$, then γ_w is an $\mathcal{EL}^{-\top}$-unifier of $\Gamma_\mathcal{A}$ and $\gamma_w(X_{q_0}) \sqsubseteq \exists w.A$.*

Proof. Let the unique accepting run of $\mathcal{A}$ on $w = w_1 \dots w_n$ be given by the sequence $q_0 q_1 \dots q_n$ of states with $q_n \in F$ and $\delta(q_i, w_{i+1}) = q_{i+1}$ for every $i \in \{0, \dots, n-1\}$, and let γ_w be defined as above.

We must show that γ_w satisfies the subsumption constraints introduced in $\Gamma_\mathcal{A}$ for every state $q \in Q$: $L_q \sqsubseteq^? X_q$ if $q \in Q \setminus F$ and $A \sqcap L_q \sqsubseteq^? X_q$ if $q \in F$, where

$$L_q := \bigsqcap_{\substack{\alpha \in \Sigma \\ \delta(q,\alpha) \text{ is defined}}} \exists \alpha.X_{\delta(q,\alpha)}.$$

To do this, we consider every top-level atom of $\gamma_w(X_q)$ and show that it subsumes the left-hand side of the above subsumption.

- Consider the conjunct $\exists u_q.A$. If $u_q = \varepsilon$, then $q \in F$ and the left-hand side contains the conjunct A. In this case, the subsumption is satisfied. Otherwise, there is a symbol $\alpha \in \Sigma$ such that $q' := \delta(q, \alpha)$ is defined and $u_q = \alpha u_{q'}$. Since $\exists u_{q'}.A$ is a top-level atom of $\gamma_w(X_{q'})$, we have $\gamma(X_{q'}) \sqsubseteq \exists u_{q'}.A$, and thus $\gamma_w(L_q) \sqsubseteq \exists \alpha.\gamma_w(X_{q'}) \sqsubseteq \exists u_q.A$.
- Let $i \in I_q$, i.e., $q_i = q$, and consider the conjunct $\exists w_{i+1} \dots w_n.A$. Since we have $\delta(q_i, w_{i+1}) = q_{i+1}$ and $\exists w_{i+2} \dots w_n.A$ is a conjunct of $\gamma_w(X_{q_{i+1}})$,[3] we obtain $\gamma_w(L_q) \sqsubseteq \exists w_{i+1}.\gamma_w(X_{q_{i+1}}) \sqsubseteq \exists w_{i+1} \exists w_{i+2} \dots w_n.A = \exists w_{i+1} \dots w_n.A$.

This shows that γ_w is a ground $\mathcal{EL}^{-\top}$-unifier of $\Gamma_{\mathcal{A}}$. Furthermore, since $0 \in I_{q_0}$, the particle $\exists w_1 \dots w_n.A = \exists w.A$ is a top-level atom of $\gamma_w(X_{q_0})$, and thus $\gamma_w(X_{q_0}) \sqsubseteq \exists w.A$. □

For the *intersection emptiness problem* one considers finitely many DFAs $\mathcal{A}_1, \dots, \mathcal{A}_k$, and asks whether $L(\mathcal{A}_1) \cap \dots \cap L(\mathcal{A}_k) \neq \emptyset$. Since this problem is trivially solvable in polynomial time in case $L(\mathcal{A}_i) = \emptyset$ for some $i, 1 \leq i \leq k$, we can assume that the languages $L(\mathcal{A}_i)$ are all nonempty. Thus, we can also assume without loss of generality that the automata $\mathcal{A}_i = (Q_i, \Sigma, q_{0,i}, \delta_i, F_i)$ have pairwise disjoint sets of states Q_i and are reduced in the sense introduced above, i.e., there is no state that cannot be reached from the initial state or from which no final state can be reached. The flat $\mathcal{EL}^{-\top}$-unification problem Γ is now defined as follows:

$$\Gamma := \bigcup_{i \in \{1,\dots,k\}} \left(\Gamma_{\mathcal{A}_i} \cup \{X_{q_{0,i}} \sqsubseteq^? Y\}\right) \ ,$$

where Y is a new variable not contained in $\Gamma_{\mathcal{A}_i}$ for $i = 1, \dots, k$.

Lemma 11. *Γ is unifiable in $\mathcal{EL}^{-\top}$ iff $L(\mathcal{A}_1) \cap \dots \cap L(\mathcal{A}_k) \neq \emptyset$.*

Proof. If Γ is unifiable in $\mathcal{EL}^{-\top}$, then it has a ground $\mathcal{EL}^{-\top}$-unifier γ and there must be a particle $\exists w.A$ with $w \in \Sigma^*$ and $\gamma(Y) \sqsubseteq \exists w.A$. Since $\gamma(X_{q_{0,i}}) \sqsubseteq \gamma(Y) \sqsubseteq \exists w.A$, Lemma 8 yields $w \in L(\mathcal{A}_{i,q_{0,i}}) = L(\mathcal{A}_i)$ for each $i \in \{1, \dots, k\}$. Thus, the intersection of the languages $L(\mathcal{A}_i)$ is nonempty.

Conversely, let $w \in \Sigma^*$ be a word with $w \in L(\mathcal{A}_1) \cap \dots \cap L(\mathcal{A}_k)$. By Lemma 10, we have for each of the unification problems $\Gamma_{\mathcal{A}_i}$ an $\mathcal{EL}^{-\top}$-unifier $\gamma_{w,i}$ such that $\gamma_{w,i}(X_{q_{0,i}}) \sqsubseteq \exists w.A$. Since the automata have disjoint state sets, the unification problems $\Gamma_{\mathcal{A}_i}$ do not share variables. Thus, we can combine the unifiers $\gamma_{w,i}$ into an $\mathcal{EL}^{-\top}$-substitution γ by defining $\gamma(Y) := \exists w.A$ and $\gamma(X_q) := \gamma_{w,i}(X_q)$ for each $i \in \{1, \dots, k\}$ and $q \in Q_i$. Obviously, this is an $\mathcal{EL}^{-\top}$-unifier of Γ since it satisfies the additional subsumptions $X_{q_{0,i}} \sqsubseteq^? Y$. □

Since the intersection emptiness problem for DFAs is PSPACE-hard [12,9], this lemma immediately yields our final theorem:

[3] If $i = n - 1$, then $\exists w_{i+2} \dots w_n.A = A$.

Theorem 2. *The problem of deciding unifiability in* $\mathcal{EL}^{-\top}$ *is* PSPACE*-hard.*

6 Conclusion

Unification in $\mathcal{EL}$ was introduced in [5] as an inference service that can support the detection of redundancies in large biomedical ontologies, which are frequently written in this DL. Motivated by the fact that the large medical ontology SNOMED CT actually does not use the top concept available in $\mathcal{EL}$, we have in this paper investigated unification in $\mathcal{EL}^{-\top}$, which is obtained from $\mathcal{EL}$ by removing the top concept. More precisely, SNOMED CT is a so-called acyclic $\mathcal{EL}^{-\top}$-TBox,[4] rather than a collection of $\mathcal{EL}^{-\top}$-concept terms. However, as shown in [7], acyclic TBoxes can be easily handled by a unification algorithm for concept terms.

Surprisingly, it has turned out that the complexity of unification in $\mathcal{EL}^{-\top}$ (PSPACE) is considerably higher than of unification in $\mathcal{EL}$ (NP). From a theoretical point of view, this result is interesting since it provides us with a natural example where reducing the expressiveness of a given DL (in a rather minor way) results in a drastic increase of the complexity of the unifiability problem. Regarding the complexity of unification in more expressive DLs, not much is known. If we add negation to $\mathcal{EL}$, then we obtain the well-known DL $\mathcal{ALC}$, which corresponds to the basic (multi-)modal logic K [15]. Decidability of unification in K is a long-standing open problem. Recently, undecidability of unification in some extensions of K (for example, by the universal modality) was shown in [18]. These undecidability results also imply undecidability of unification in some expressive DLs (e.g., in $\mathcal{SHIQ}$ [10]).

Apart from its theoretical interest, the result of this paper also has practical implications. Whereas practically rather efficient unification algorithm for $\mathcal{EL}$ can readily be obtained by a translation into SAT [6], it is not so clear how to turn the PSPACE algorithm for $\mathcal{EL}^{-\top}$-unification introduced in this paper into a practically useful algorithm. One possibility could be to use a SAT modulo theories (SMT) approach [13]. The idea is that the SAT solver is used to generate all possible subsumption mappings for Γ, and that the theory solver tests the system $\mathcal{I}_{\Gamma,\tau}$ induced by τ for the existence of a finite, admissible solution. How well this works will mainly depend on whether we can develop such a theory solver that satisfies well all the requirements imposed by the SMT approach.

Another topic for future research is how to actually compute $\mathcal{EL}^{-\top}$-unifiers for a unifiable $\mathcal{EL}^{-\top}$-unification problem. In principle, our decision procedure is constructive in the sense that, from appropriate successful runs of the ε-AFA $\mathcal{A}(X, A)$, one can construct a finite, admissible solution of $\mathcal{I}_{\Gamma,\tau}$, and from this an $\mathcal{EL}^{-\top}$-unifier of Γ. However, this needs to be made more explicit, and we need to investigate what kind of $\mathcal{EL}^{-\top}$-unifiers can be computed this way.

[4] Note that the right-identity rules in SNOMED CT [16] are actually not expressed using complex role inclusion axioms, but through the SEP-triplet encoding [17]. Thus, complex role inclusion axioms are not relevant here.

References

1. Baader, F.: Terminological cycles in a description logic with existential restrictions. In: Proc. of the 18th Int. Joint Conf. on Artificial Intelligence (IJCAI 2003), pp. 325–330. Morgan Kaufmann, Los Alamitos (2003)
2. Baader, F., Binh, N.T., Borgwardt, S., Morawska, B.: Unification in the description logic $\mathcal{EL}$ without the top concept. LTCS-Report 11-01, TU Dresden, Dresden, Germany (2011), `http://lat.inf.tu-dresden.de/research/reports.html`
3. Baader, F., Brandt, S., Lutz, C.: Pushing the $\mathcal{EL}$ envelope. In: Proc. of the 19th Int. Joint Conf. on Artificial Intelligence (IJCAI 2005), pp. 364–369. Morgan Kaufmann, Los Alamitos (2005)
4. Baader, F., Calvanese, D., McGuinness, D., Nardi, D., Patel-Schneider, P.F. (eds.): The Description Logic Handbook: Theory, Implementation, and Applications. Cambridge University Press, Cambridge (2003)
5. Baader, F., Morawska, B.: Unification in the Description Logic $\mathcal{EL}$. In: Treinen, R. (ed.) RTA 2009. LNCS, vol. 5595, pp. 350–364. Springer, Heidelberg (2009)
6. Baader, F., Morawska, B.: SAT Encoding of Unification in $\mathcal{EL}$. In: Fermüller, C.G., Voronkov, A. (eds.) LPAR-17. LNCS, vol. 6397, pp. 97–111. Springer, Heidelberg (2010)
7. Baader, F., Morawaska, B.: Unification in the description logic $\mathcal{EL}$. Logical Methods in Computer Science 6(3) (2010)
8. Baader, F., Narendran, P.: Unification of concept terms in description logics. J. of Symbolic Computation 31(3), 277–305 (2001)
9. Garey, M.R., Johnson, D.S.: Computers and Intractability — A guide to NP-completeness. W.H. Freeman and Company, San Francisco (1979)
10. Horrocks, I., Sattler, U., Tobies, S.: Practical reasoning for very expressive description logics. Logic Journal of the IGPL 8(3), 239–264 (2000)
11. Jiang, T., Ravikumar, B.: A note on the space complexity of some decision problems for finite automata. Information Processing Letters 40, 25–31 (1991)
12. Kozen, D.: Lower bounds for natural proof systems. In: Annual IEEE Symposium on Foundations of Computer Science, pp. 254–266 (1977)
13. Nieuwenhuis, R., Oliveras, A., Tinelli, C.: Solving SAT and SAT modulo theories: From an abstract Davis-Putnam-Logemann-Loveland procedure to DPLL(T). J. ACM 53(6), 937–977 (2006)
14. Savitch, W.J.: Relationships between nondeterministic and deterministic tape complexities. Journal of Computer and System Sciences 4(2), 177–192 (1970)
15. Schild, K.: A correspondence theory for terminological logics: Preliminary report. In: Proc. of the 12th Int. Joint Conf. on Artificial Intelligence (IJCAI 1991), pp. 466–471 (1991)
16. Spackman., K.A.: Spackman. Managing clinical terminology hierarchies using algorithmic calculation of subsumption: Experience with SNOMED-RT. Journal of the American Medical Informatics Association (2000); Fall Symposium Special Issue
17. Suntisrivaraporn, B., Baader, F., Schulz, S., Spackman, K.: Replacing SEP-Triplets in SNOMED CT Using Tractable Description Logic Operators. In: Bellazzi, R., Abu-Hanna, A., Hunter, J. (eds.) AIME 2007. LNCS (LNAI), vol. 4594, pp. 287–291. Springer, Heidelberg (2007)
18. Wolter, F., Zakharyaschev, M.: Undecidability of the unification and admissibility problems for modal and description logics. ACM Trans. Comput. Log. 9(4) (2008)

Matching with Respect to General Concept Inclusions in the Description Logic $\mathcal{EL}$

Franz Baader and Barbara Morawska*

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

Abstract. Matching concept descriptions against concept patterns was introduced as a new inference task in Description Logics (DLs) almost 20 years ago, motivated by applications in the Classic system. For the DL $\mathcal{EL}$, it was shown in 2000 that matching without a TBox is NP-complete. In this paper we show that matching in $\mathcal{EL}$ w.r.t. general TBoxes (i.e., finite sets of general concept inclusions, GCIs) is in NP by introducing a goal-oriented matching algorithm that uses non-deterministic rules to transform a given matching problem into a solved form by a polynomial number of rule applications. We also investigate some tractable variants of the matching problem w.r.t. general TBoxes.

1 Introduction

The DL $\mathcal{EL}$, which offers the constructors conjunction ($\sqcap$), existential restriction ($\exists r.C$), and the top concept ($\top$), has recently drawn considerable attention since, on the one hand, important inference problems such as the subsumption problem are polynomial in $\mathcal{EL}$, even in the presence of general concept inclusions (GCIs) [12]. On the other hand, though quite inexpressive, $\mathcal{EL}$ can be used to define biomedical ontologies, such as the large medical ontology SNOMED CT.[1]

Matching of concept descriptions against concept patterns is a non-standard inference task in Description Logics, which was originally motivated by applications of the Classic system [9]. In [11], Borgida and McGuinness proposed matching as a means to filter out the unimportant aspects of large concept descriptions appearing in knowledge bases of Classic. Subsequently, matching (as well as the more general problem of unification) was also proposed as a tool for detecting redundancies in knowledge bases [8] and to support the integration of knowledge bases by prompting interschema assertions to the integrator [10].

All three applications have in common that one wants to search the knowledge base for concepts having a certain (not completely specified) form. This "form" can be expressed with the help of so-called *concept patterns*, i.e., concept descriptions containing variables (which stand for descriptions). For example, assume that we want to find concepts that are concerned with individuals having a son and a daughter sharing some characteristic. This can be expressed

* Supported by DFG under grant BA 1122/14-2.

[1] See http://www.ihtsdo.org/snomed-ct/

C. Lutz and M. Thielscher (Eds.): KI 2014, LNCS 8736, pp. 135–146, 2014.

by the pattern $D := \exists \mathsf{has\text{-}child}.(\mathsf{Male} \sqcap X) \sqcap \exists \mathsf{has\text{-}child}.(\mathsf{Female} \sqcap X)$, where X is a variable standing for the common characteristic. The concept description $C := \exists \mathsf{has\text{-}child}.(\mathsf{Tall} \sqcap \mathsf{Male}) \sqcap \exists \mathsf{has\text{-}child}.(\mathsf{Tall} \sqcap \mathsf{Female})$ matches this pattern in the sense that, if we replace the variable X by the description Tall, the pattern becomes *equivalent* to the description. Thus, the substitution $\sigma := \{X \mapsto \mathsf{Tall}\}$ is a *matcher modulo equivalence* of the matching problem $C \equiv^? D$ since $C \equiv \sigma(D)$.

The original paper by Borgida and McGuinness actually considered matching modulo subsumption rather than matching modulo equivalence: such a problem is of the form $C \sqsubseteq^? D$, and a matcher is a substitution σ satisfying $C \sqsubseteq \sigma(D)$. Obviously, any matcher modulo equivalence is also a matcher modulo subsumption, but not vice versa. For example, the substitution $\sigma_\top := \{X \mapsto \top\}$ is a *matcher modulo subsumption* of the matching problem $C \sqsubseteq^? D$, but it is not a matcher modulo equivalence of $C \equiv^? D$. For both cases of matching, the original definitions were formulated for concept descriptions without any TBox, i.e., the subsumption or equivalence that has to be achieved by an application of the matcher does not take a TBox into account. The reason was that at that time TBoxes were usually acyclic, and thus could be reduced away by unfolding.

The first results on matching in DLs were concerned with sublanguages of the Classic description language, which does not allow for existential restrictions of the kind used above. A polynomial-time algorithm for computing matchers modulo subsumption for a rather expressive DL was introduced in [11]. The main drawback of this algorithm was that it required the concept patterns to be in structural normal form, and thus it was not able to handle arbitrary matching problems. In addition, the algorithm was incomplete, i.e., it did not always find a matcher, even if one existed. For the DL $\mathcal{ALN}$, a polynomial-time algorithm for matching modulo subsumption and equivalence was presented in [6]. This algorithm is complete and it applies to arbitrary patterns. In [5], matching in DLs with existential restrictions was investigated for the first time. In particular, it was shown that in $\mathcal{EL}$ the matching problem (i.e., the problem of deciding whether a given matching problem has a matcher or not) is polynomial for matching modulo subsumption, but NP-complete for matching modulo equivalence.

Unification is a generalization of matching where both sides of the problem are patterns and thus the substitution needs to be applied to both sides. In [8] it was shown that the unification problem in the DL $\mathcal{FL}_0$, which offers the constructors conjunction ($\sqcap$), value restriction ($\forall r.C$), and the top concept ($\top$), is ExpTime-complete. In contrast, unification in $\mathcal{EL}$ is "only" NP-complete [7]. In the results for matching and unification mentioned until now, there was no TBox involved, i.e., equivalence and subsumption was considered with respect to the empty TBox. For unification in $\mathcal{EL}$, first attempts were made to take *general TBoxes*, i.e., finite sets of general concept inclusions (GCIs), into account. However, the results obtained so far, which are again NP-completeness results, are restricted to general TBoxes that satisfy a certain restriction on cyclic dependencies between concepts [2,3].

For matching, we solve the general case in this paper: we show that matching in $\mathcal{EL}$ w.r.t. general TBoxes is NP-complete by introducing a goal-oriented matching algorithm that uses non-deterministic rules to transform a given matching problem into a solved form by a polynomial number of rule applications. The matching problems considered in this paper are actually generalizations of matching modulo equivalence and matching modulo subsumption. For the special case of matching modulo subsumption, we show that the problem is tractable also in the presence of GCIs. The same is true for the dual problem where the pattern is on the side of the subsumee rather than on the side of the subsumer.

Due to space constraints, we cannot provide complete proofs of our results. They can be found in [1].

2 The Description Logics $\mathcal{EL}$

The expressiveness of a DL is determined both by the formalism for describing concepts (the concept description language) and the terminological formalism, which can be used to state additional constraints on the interpretation of concepts and roles in a so-called TBox.

The *concept description language* considered in this paper is called $\mathcal{EL}$. Starting with a finite set N_C of *concept names* and a finite set N_R of *role names*, *$\mathcal{EL}$-concept descriptions* are built from concept names using the constructors *conjunction* ($C \sqcap D$), *existential restriction* ($\exists r.C$ for every $r \in N_R$), and *top* ($\top$). Since in this paper we only consider $\mathcal{EL}$-concept descriptions, we will sometimes dispense with the prefix $\mathcal{EL}$.

On the *semantic side*, concept descriptions are interpreted as sets. To be more precise, an *interpretation* $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ consists of a non-empty domain $\Delta^{\mathcal{I}}$ and an interpretation function $\cdot^{\mathcal{I}}$ that maps concept names to subsets of $\Delta^{\mathcal{I}}$ and role names to binary relations over $\Delta^{\mathcal{I}}$. This function is inductively extended to concept descriptions as follows:

$$\top^{\mathcal{I}} := \Delta^{\mathcal{I}}, \quad (C \sqcap D)^{\mathcal{I}} := C^{\mathcal{I}} \cap D^{\mathcal{I}}, \quad (\exists r.C)^{\mathcal{I}} := \{x \mid \exists y : (x, y) \in r^{\mathcal{I}} \wedge y \in C^{\mathcal{I}}\}$$

A *general concept inclusion axiom (GCI)* is of the form $C \sqsubseteq D$ for concept descriptions C, D. An interpretation $\mathcal{I}$ *satisfies* such an axiom $C \sqsubseteq D$ iff $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$. A *general $\mathcal{EL}$-TBox* is a finite set of GCIs. An interpretation is a *model* of a general $\mathcal{EL}$-TBox if it satisfies all its GCIs.

A concept description C is *subsumed* by a concept description D w.r.t. a general TBox $\mathcal{T}$ (written $C \sqsubseteq_{\mathcal{T}} D$) if every model of $\mathcal{T}$ satisfies the GCI $C \sqsubseteq D$. We say that C is *equivalent* to D w.r.t. $\mathcal{T}$ ($C \equiv_{\mathcal{T}} D$) if $C \sqsubseteq_{\mathcal{T}} D$ and $D \sqsubseteq_{\mathcal{T}} C$. If $\mathcal{T}$ is empty, we also write $C \sqsubseteq D$ and $C \equiv D$ instead of $C \sqsubseteq_{\mathcal{T}} D$ and $C \equiv_{\mathcal{T}} D$, respectively. As shown in [12], subsumption w.r.t. general $\mathcal{EL}$-TBoxes is decidable in polynomial time.

An $\mathcal{EL}$-concept description is an *atom* if it is an existential restriction or a concept name. The atoms of an $\mathcal{EL}$-concept description C are the subdescriptions of C that are atoms, and the top-level atoms of C are the atoms occurring in the top-level conjunction of C. Obviously, any $\mathcal{EL}$-concept description is the

conjunction of its top-level atoms, where the empty conjunction corresponds to $\top$. The atoms of a general $\mathcal{EL}$-TBox $\mathcal{T}$ are the atoms of all the concept descriptions occurring in GCIs of $\mathcal{T}$.

We say that a subsumption between two atoms is *structural* if their top-level structure is compatible. To be more precise, following [2] we define structural subsumption between atoms as follows: the atom C is *structurally subsumed* by the atom D w.r.t. $\mathcal{T}$ ($C \sqsubseteq^{\mathsf{s}}_{\mathcal{T}} D$) iff one of the following holds:

1. $C = D$ is a concept name,
2. $C = \exists r.C'$, $D = \exists r.D'$, and $C' \sqsubseteq_{\mathcal{T}} D'$.

It is easy to see that subsumption w.r.t. $\emptyset$ between two atoms implies structural subsumption w.r.t. $\mathcal{T}$, which in turn implies subsumption w.r.t. $\mathcal{T}$. The matching algorithms presented below crucially depend on the following characterization of subsumption w.r.t. general $\mathcal{EL}$-TBoxes first stated in [2]:

Lemma 1. *Let $\mathcal{T}$ be an $\mathcal{EL}$-TBox and $C_1, \ldots, C_n$, $D_1, \ldots, D_m$ be atoms. Then $C_1 \sqcap \cdots \sqcap C_n \sqsubseteq_{\mathcal{T}} D_1 \sqcap \cdots \sqcap D_m$ iff for every $j \in \{1, \ldots, m\}$*

1. *there is an index $i \in \{1, \ldots, n\}$ such that $C_i \sqsubseteq^{\mathsf{s}}_{\mathcal{T}} D_j$ or*
2. *there are atoms $A_1, \ldots, A_k, B$ of $\mathcal{T}$ ($k \geq 0$) such that*
   (a) *$A_1 \sqcap \cdots \sqcap A_k \sqsubseteq_{\mathcal{T}} B$,*
   (b) *for every $\eta \in \{1, \ldots, k\}$ there is $i \in \{1, \ldots, n\}$ with $C_i \sqsubseteq^{\mathsf{s}}_{\mathcal{T}} A_\eta$, and*
   (c) *$B \sqsubseteq^{\mathsf{s}}_{\mathcal{T}} D_j$.*

3 Matching in $\mathcal{EL}$

In addition to the set N_C of concept names (which must not be replaced by substitutions), we introduce a set N_V of concept variables (which may be replaced by substitutions). *Concept patterns* are now built from concept names and concept variables by applying the constructors of $\mathcal{EL}$. A *substitution* σ maps every concept variable to an $\mathcal{EL}$-concept description. It is extended to concept patterns in the usual way:

- $\sigma(A) := A$ for all $A \in N_C \cup \{\top\}$,
- $\sigma(C \sqcap D) := \sigma(C) \sqcap \sigma(D)$ and $\sigma(\exists r.C) := \exists r.\sigma(C)$.

An $\mathcal{EL}$-concept pattern C is *ground* if it does not contain variables, i.e., if it is a concept description. Obviously, a ground concept pattern is not modified by applying a substitution.

Definition 2. *Let $\mathcal{T}$ be a general $\mathcal{EL}$-TBox.*[2] *An* $\mathcal{EL}$-matching problem *w.r.t. $\mathcal{T}$ is a finite set $\Gamma = \{C_1 \sqsubseteq^? D_1, \ldots, C_n \sqsubseteq^? D_n\}$ of subsumptions between $\mathcal{EL}$-concept patterns, where for each $i, 1 \leq i \leq n$, C_i or D_i is ground. A substitution σ is a* matcher *of Γ w.r.t. $\mathcal{T}$ if σ* solves *all the subsumptions in Γ, i.e. if $\sigma(C_1) \sqsubseteq_{\mathcal{T}} \sigma(D_1), \ldots, \sigma(C_n) \sqsubseteq_{\mathcal{T}} \sigma(D_n)$. We say that Γ is* matchable *w.r.t. $\mathcal{T}$ if it has a matcher.*

[2] Note that the GCIs in $\mathcal{T}$ are built using concept descriptions, and thus do not contain variables.

Matching problems modulo equivalence and subsumption are special cases of the matching problems introduced above:

- The $\mathcal{EL}$-matching problem Γ is a *matching problem modulo equivalence* if $C \sqsubseteq^? D \in \Gamma$ implies $D \sqsubseteq^? C \in \Gamma$. This coincides with the notion of matching modulo equivalence considered in [6,5], but extended to a non-empty general TBox.
- The $\mathcal{EL}$-matching problem Γ is a *left-ground matching problem modulo subsumption* if $C \sqsubseteq^? D \in \Gamma$ implies that C is ground. This coincides with the notion of matching modulo subsumption considered in [6,5], but again extended to a non-empty general TBox.
- The $\mathcal{EL}$-matching problem Γ is a *right-ground matching problem modulo subsumption* if $C \sqsubseteq^? D \in \Gamma$ implies that D is ground. To the best of our knowledge, this notion of matching has not been investigated before.

We will show in the following that the general case of matching, as introduced in Definition 2, and thus also matching modulo equivalence, is NP-complete, whereas the two notions of matching modulo subsumption are tractable, even in the presence of GCIs.

4 Matching Modulo Subsumption

The case of *left-ground matching problems modulo subsumption* can be treated as sketched in [5] for the case without a TBox. Given a general $\mathcal{EL}$-TBox $\mathcal{T}$ and two substitutions σ, τ, we define: $\sigma \sqsubseteq_\mathcal{T} \tau$ iff $\sigma(X) \sqsubseteq_\mathcal{T} \tau(X)$ for all $X \in N_V$.

Consequently, if $\sigma_\top$ denotes the substitution satisfying $\sigma_\top(X) = \top$ for all $X \in N_V$, then $\sigma \sqsubseteq_\mathcal{T} \sigma_\top$ holds for all substitutions σ. Since the concept constructors of $\mathcal{EL}$ are monotonic w.r.t. subsumption, this implies $\sigma(D) \sqsubseteq_\mathcal{T} \sigma_\top(D)$ for all concept patterns D.

Lemma 3. *Let $\Gamma = \{C_1 \sqsubseteq^? D_1, \ldots, C_n \sqsubseteq^? D_n\}$ be a left-ground matching problem modulo subsumption. Then Γ has a matcher w.r.t. $\mathcal{T}$ iff $\sigma_\top$ is a matcher of Γ w.r.t. $\mathcal{T}$.*

Proof. The "if" direction is trivial. Conversely, assume that σ is a matcher of Γ w.r.t. $\mathcal{T}$. Then we have, for all $i, 1 \leq i \leq n$, that $\sigma_\top(C_i) = C_i = \sigma(C_i) \sqsubseteq_\mathcal{T} \sigma(D_i) \sqsubseteq_\mathcal{T} \sigma_\top(D_i)$, which shows that $\sigma_\top$ is a matcher of Γ w.r.t. $\mathcal{T}$. □

The lemma shows that it is sufficient to test whether the substitution $\sigma_\top$ is a matcher of Γ, i.e., whether $\sigma_\top(C_i) \sqsubseteq_\mathcal{T} \sigma_\top(D_i)$ holds for all $i, 1 \leq i \leq n$. Since in $\mathcal{EL}$ subsumption w.r.t. general TBoxes is decidable in polynomial time, this yields a polynomial-time algorithm for left-ground matching modulo subsumption in $\mathcal{EL}$.

Theorem 4. *Let Γ be a left-ground $\mathcal{EL}$-matching problem modulo subsumption and $\mathcal{T}$ a general $\mathcal{EL}$-TBox. Then we can decide in polynomial time whether Γ has a matcher w.r.t. $\mathcal{T}$ or not.*

The case of *right-ground matching problems modulo subsumption* can be treated similarly. However, since $\mathcal{EL}$ does not have the bottom concept $\bot$ as a concept constructor, we cannot simply define $\sigma_\bot$ as the substitution satisfying $\sigma_\bot(X) = \bot$ for all $X \in N_V$, and then show that that the right-ground matching problems modulo subsumption, Γ, has a matcher w.r.t. $\mathcal{T}$ iff $\sigma_\bot$ is a matcher of Γ w.r.t. $\mathcal{T}$. Instead, we need to define $\sigma_\bot$ in a more complicated manner.

Given a general $\mathcal{EL}$-TBox $\mathcal{T}$ and a right-ground matching problems modulo subsumption $\Gamma = \{C_1 \sqsubseteq^? D_1, \ldots, C_n \sqsubseteq^? D_n\}$, we use $\bot(\Gamma, \mathcal{T})$ to denote the $\mathcal{EL}$-concept description that is the conjunction of all the atoms of $\mathcal{T}$ and of $D_1, \ldots, D_n$. We now define $\sigma_{\bot(\Gamma,\mathcal{T})}$ as the substitution satisfying $\sigma_{\bot(\Gamma,\mathcal{T})}(X) = \bot(\Gamma, \mathcal{T})$ for all $X \in N_V$

Lemma 5. *Let $\Gamma = \{C_1 \sqsubseteq^? D_1, \ldots, C_n \sqsubseteq^? D_n\}$ be a right-ground matching problem modulo subsumption. Then Γ has a matcher w.r.t. $\mathcal{T}$ iff $\sigma_{\bot(\Gamma,\mathcal{T})}$ is a matcher of Γ w.r.t. $\mathcal{T}$.*

Proof. The "if" direction is trivial. To see the "only-if" direction, assume that σ is a matcher of Γ w.r.t. $\mathcal{T}$. We need to show that this implies the $\sigma_{\bot(\Gamma,\mathcal{T})}$ is also a matcher of Γ w.r.t. $\mathcal{T}$, i.e., that it satisfies $\sigma_{\bot(\Gamma,\mathcal{T})}(C) \sqsubseteq_\mathcal{T} D$ for every subsumption $C \sqsubseteq^? D \in \Gamma$.

More generally, we consider subsumptions $C \sqsubseteq^? D$ where C is a subpattern of a pattern occurring in Γ or $\mathcal{T}$ and D is an atom of $\mathcal{T}$ or $D_1, \ldots, D_n$. We show the following claim:

Claim: *For every such subsumption $C \sqsubseteq^? D$, it holds that $\sigma(C) \sqsubseteq_\mathcal{T} D$ implies $\sigma_{\bot(\Gamma,\mathcal{T})}(C) \sqsubseteq_\mathcal{T} D$.*

Before proving the claim, let us show that this implies that $\sigma_{\bot(\Gamma,\mathcal{T})}$ solves Γ w.r.t. $\mathcal{T}$. In fact, any subsumption in Γ is of the form $C \sqsubseteq^? E_1 \sqcap \ldots \sqcap E_k$ where C is a subpattern of a pattern occurring in Γ, and $E_1, \ldots, E_k$ are atoms of one of the D_i. In addition, a substitution solves $C \sqsubseteq^? E_1 \sqcap \ldots \sqcap E_k$ w.r.t. $\mathcal{T}$ iff it solves all the subsumptions $C \sqsubseteq^? E_i$ for $i = 1, \ldots, k$.

We prove the claim by induction on the size $|C|$ of the left-hand side C of the subsumption $C \sqsubseteq^? D$. Let $C = F_1 \sqcap \ldots \sqcap F_\ell$, where $F_1, \ldots, F_\ell$ are atoms. We distinguish the following three cases:

1. If there is an index $i \in \{1, \ldots, \ell\}$ such that F_i is a variable, then $\sigma_{\bot(\Gamma,\mathcal{T})}(F_i) \sqsubseteq D$ since D occurs as a conjunct in $\bot(\Gamma, \mathcal{T})$. This implies $\sigma_{\bot(\Gamma,\mathcal{T})}(C) \sqsubseteq_\mathcal{T} D$.
2. If there is an index $i \in \{1, \ldots, \ell\}$ such that F_i is ground and $\sigma(F_i) \sqsubseteq_\mathcal{T} D$, then $\sigma_{\bot(\Gamma,\mathcal{T})}(F_i) = F_i = \sigma(F_i) \sqsubseteq_\mathcal{T} D$. This again implies $\sigma_{\bot(\Gamma,\mathcal{T})}(C) \sqsubseteq_\mathcal{T} D$.
3. Assume that the above two cases do not hold. Using Lemma 1, we can distinguish two more cases, depending on whether the first or the second condition of the lemma applies.
   (a) If the first condition applies, then there is an index $i \in \{1, \ldots, \ell\}$ such that $F_i \sqsubseteq^{\mathsf{s}}_\mathcal{T} D$. Since F_i is neither ground nor a variable, we know that F_i is a non-ground existential restriction. Thus, $F_i = \exists r.F'$, $D = \exists r.(D_1 \sqcap \ldots \sqcap D_m)$ with $D_1, \ldots, D_m$ atoms, and $\sigma(F') \sqsubseteq_\mathcal{T} D_i$ for all

$i \in \{1, \ldots, m\}$. Since F' is a subpattern of C, D_i are atoms of D, and $|F'| < |C|$, we can apply the induction hypothesis to the subsumptions $F' \sqsubseteq^? D_i$. This yields $\sigma_{\bot(\Gamma,\mathcal{T})}(F') \sqsubseteq_\mathcal{T} D_i$ for all $i \in \{1, \ldots, m\}$, and thus $\sigma_{\bot(\Gamma,\mathcal{T})}(C) \sqsubseteq_\mathcal{T} D$.

(b) If the second condition applies, then there are atoms $A_1, \ldots, A_k, B$ of $\mathcal{T}$ such that $A_1 \sqcap \cdots \sqcap A_k \sqsubseteq_\mathcal{T} B \sqsubseteq_\mathcal{T} D$ and for each $\eta \in \{1, \ldots, k\}$, there is $j \in \{1, \ldots, \ell\}$ such that
   i. F_j is a concept variable and $\sigma(F_j) \sqsubseteq_\mathcal{T} A_\eta$, or
   ii. F_j is ground and $F_j \sqsubseteq_\mathcal{T} A_\eta$, or
   iii. $F_j = \exists r.F'$, $A_\eta = \exists r.A'$ and $\sigma(F') \sqsubseteq_\mathcal{T} A'$.

   It is sufficient to show that the subsumption relationships in 3(b)i and 3(b)iii also hold if we replace σ by $\sigma_{\bot(\Gamma,\mathcal{T})}$. For 3(b)i this can be shown as in 1 and for 3(b)iii as in 3a.

This completes the proof of the claim, and thus of the lemma. □

Since the size of $\bot(\Gamma, \mathcal{T})$ is polynomial in the size of Γ and $\mathcal{T}$, this lemma yields a polynomial-time decision procedure for right-ground matching modulo subsumption.

Theorem 6. *Let Γ be a right-ground $\mathcal{EL}$-matching problem modulo subsumption and $\mathcal{T}$ a general $\mathcal{EL}$-TBox. Then we can decide in polynomial time whether Γ has a matcher w.r.t. $\mathcal{T}$ or not.*

5 The General Case

NP-hardness for the general case follows from the known NP-hardness result for matching modulo equivalence without a TBox [5]. In the following, we show that matching in $\mathcal{EL}$ w.r.t. general TBoxes is in NP by introducing a goal-oriented matching algorithm that uses non-deterministic rules to transform a given matching problem into a solved form by a polynomial number of rule applications.

Let $\mathcal{T}$ be a general $\mathcal{EL}$-TBox and Γ_0 an $\mathcal{EL}$-matching problem. We can assume without loss of generality that all the subsumptions $C \sqsubseteq^? D$ in Γ_0 are such that either C or D is non-ground. In fact, if both C and D are ground, then the following holds:

- If $C \sqsubseteq_\mathcal{T} D$, then Γ_0 has a matcher w.r.t. $\mathcal{T}$ iff $\Gamma_0 \setminus \{C \sqsubseteq^? D\}$ has a matcher w.r.t. $\mathcal{T}$.
- If $C \not\sqsubseteq_\mathcal{T} D$, then Γ_0 does not have a matcher w.r.t. $\mathcal{T}$.

Consequently, we can either remove all the offending ground subsumptions without changing the solvability status of the problem, or immediately decide non-solvability. Using the fact that $C \sqsubseteq_\mathcal{T} D_1 \sqcap D_2$ iff $C \sqsubseteq_\mathcal{T} D_1$ and $C \sqsubseteq_\mathcal{T} D_2$, we can additionally normalize Γ_0 such that the right-hand side of each subsumption in Γ_0 is an atom. We call an $\mathcal{EL}$-matching problem *normalized* if $C \sqsubseteq^? D \in \Gamma_0$ implies that (i) either C or D is non-ground, and (ii) D is an atom.

Eager Solving (variable on the right):

Condition: A subsumption $C \sqsubseteq^? X \in \Gamma$ where $X \in N_V$.
Action:
- If there is some subsumption of the form $X \sqsubseteq^? D \in \Gamma$ such that $C \not\sqsubseteq_{\mathcal{T}} D$, then the rule application fails.
- Otherwise, mark $C \sqsubseteq^? X$ as "solved."

Eager Solving (variable on the left):

Condition: A subsumption $X \sqsubseteq^? D \in \Gamma$ where $X \in N_V$.
Action:
- If there is some subsumption of the form $C \sqsubseteq^? X \in \Gamma$ such that $C \not\sqsubseteq_{\mathcal{T}} D$, then the rule application fails.
- Otherwise, mark $X \sqsubseteq^? D$ as "solved."

Fig. 1. Eager Rules

Thus, assume that Γ_0 is a normalized $\mathcal{EL}$-matching problem. Our algorithm starts with $\Gamma := \Gamma_0$, and then applies non-deterministic rules to Γ. A non-failing application of a rule may add subsumptions to Γ. Note, however, that a subsumption is only added if it is not yet present. New subsumptions that are added are marked as "unsolved," as are initially all the subsumptions of Γ_0. A rule application may *fail*, which means that this attempt of solving the matching problem was not successful. A non-failing rule application marks one of the subsumptions in the matching problem as "solved." Rules are applied until all subsumptions are marked "solved" or an attempt to apply a rule has failed.

Our definition of the rules uses a function $Dec(\dots)$ on subsumptions of the form $C \sqsubseteq^? D$, where C and D are atoms and D is not a variable. A call of $Dec(C \sqsubseteq^? D)$ returns a (possibly empty) set of subsumptions or it fails:

1. $Dec(C \sqsubseteq^? D) := \{C \sqsubseteq^? D\}$, if C is a variable.
2. If $D_1, \dots, D_n$ are atoms, then $Dec(\exists r.C' \sqsubseteq^? \exists r.(D_1 \sqcap \dots \sqcap D_n))$ fails if there is an $i \in \{1, \dots, n\}$ such that both sides of $C' \sqsubseteq^? D_i$ are ground and $C' \not\sqsubseteq_{\mathcal{T}} D_i$. Otherwise, $Dec(\exists r.C' \sqsubseteq^? \exists r.(D_1 \sqcap \dots \sqcap D_n)) := \{C' \sqsubseteq^? D_i \mid 1 \leq i \leq n \text{ and } C' \text{ or } D_i \text{ is non-ground}\}$.
3. If $C = \exists r.C'$ and $D = \exists s.D'$ for roles $s \neq r$, then $Dec(C \sqsubseteq^? D)$ fails.
4. If $C = A$ is a concept name and $D = \exists r.D'$ an existential restriction, then $Dec(C \sqsubseteq^? D)$ fails.
5. If $D = A$ is a concept name and $C = \exists r.C'$ an existential restriction, then $Dec(C \sqsubseteq^? D)$ fails.
6. If both C and D are ground and $C \not\sqsubseteq_{\mathcal{T}} D$ then $Dec(C \sqsubseteq^? D)$ fails, and otherwise returns $\emptyset$.

Algorithm 7. Let Γ_0 be a normalized $\mathcal{EL}$-matching problem. Starting with $\Gamma := \Gamma_0$, apply the rules of Figure 1 and Figure 2 exhaustively in the following order:

Decomposition:

Condition: This rule applies to $\mathfrak{s} = C_1 \sqcap \cdots \sqcap C_n \sqsubseteq^? D \in \Gamma$.
Action: Its application chooses an index $i \in \{1, \ldots, n\}$ and calls $Dec(C_i \sqsubseteq^? D)$. If this call does not fail, then it adds the returned subsumptions to Γ, and marks $\mathfrak{s}$ as *solved*. If $Dec(C_i \sqsubseteq^? D)$ fails, it returns "failure."

Mutation :

Condition: This rule applies to $\mathfrak{s} = C_1 \sqcap \cdots \sqcap C_n \sqsubseteq^? D$ in Γ.
Action: Its application chooses atoms $A_1, \ldots, A_k, B$ of $\mathcal{T}$. If $A_1 \sqcap \cdots \sqcap A_k \sqsubseteq_{\mathcal{T}} B$ does not hold, then it returns "failure." Otherwise, it performs the following two steps:

- Choose for each $\eta \in \{1, \ldots, k\}$ an $i \in \{1, \ldots, n\}$ and call $Dec(C_i \sqsubseteq^? A_\eta)$. If this call does not fail, it adds the returned subsumptions to Γ. Otherwise, if $Dec(C_i \sqsubseteq^? A_\eta)$ fails, the rule returns "failure."
- If it has not failed before and $Dec(B \sqsubseteq^? D)$ does not fail, it adds the returned subsumptions to Γ. Otherwise, if $Dec(B \sqsubseteq^? D)$ fails, it returns "failure."

If these steps did not fail, then the rule marks $\mathfrak{s}$ as *solved*.

Fig. 2. Non-deterministic rules

(1) **Eager rule application:** If an eager rule from Figure 1 applies to an unsolved subsumption, apply it. If the rule application fails, stop and return "failure."
(2) **Non-deterministic rule application:** If no eager rule is applicable, let $\mathfrak{s}$ be an unsolved subsumption in Γ. Choose one of the non-deterministic rules of Figure 2, and apply it to $\mathfrak{s}$. If this rule application fails, then stop and return "failure."

If no more rule applies and the algorithm has not stopped returning "failure," then return "success."

In (2), the choice which unsolved subsumption to consider next is don't care non-deterministic. However, choosing which rule to apply to the chosen subsumption is don't know non-deterministic. Additionally, the application of a non-deterministic rules may require don't know non-deterministic choices to be made. If a non-deterministic rule is applied to a subsumption $\mathfrak{s}$, then neither its left-hand side nor its right-hand side is a variable. In fact, a subsumption that has a variable on one of its sides is solved by one of the eager rules, which have precedence over the non-deterministic rules.

It is easy to see that the subsumptions added by the non-deterministic rules satisfy the normalization conditions (i) and (ii), and thus all the sets Γ generated during a run of the algorithm are normalized $\mathcal{EL}$-matching problems. The next lemma states an important property ensured by the presence of the eager rules.

Lemma 8. *If Γ is a matching problem generated during a non-failing run of the algorithm, and both $C \sqsubseteq^? X \in \Gamma$ and $X \sqsubseteq^? D \in \Gamma$ are solved, then $C \sqsubseteq_{\mathcal{T}} D$.*

Proof. Obviously, one of the two subsumptions was solved after the other. This means that, when it was solved by the application of an eager rule, the other one was already present. Since we consider a non-failing run, the application of the eager rule did not fail, which yields $C \sqsubseteq_{\mathcal{T}} D$. □

Any run of the algorithm *terminates after a polynomial number of steps*. The main reason for this is that there are only *polynomially many subsumptions* that can occur in the matching problems Γ generated during a run.

Lemma 9. *Let Γ be a matching problem generated during a run of Algorithm 7. Then any subsumption occurring in Γ is of one of the following forms:*

1. *A subsumption contained in the original input matching problem Γ_0.*
2. *A subsumption of the form $C \sqsubseteq^? D$ where C, D are subpatterns of concept patterns occurring in Γ_0.*
3. *A subsumption of the form $C \sqsubseteq^? A$ or $A \sqsubseteq^? C$ where A is an atom of $\mathcal{T}$ and C is a subpattern of a concept pattern occurring in Γ_0.*

Since any rule application either fails while trying to solve an unsolved subsumption (in which case the algorithm stops immediately) or actually solves an unsolved subsumption, there can be only polynomially many rule applications during a run. In addition, it is easy to see that each rule application can be realized in polynomial time, with a polynomial number of possible non-deterministic choices. This shows that Algorithm 7 is indeed an NP-algorithm. It remains to show that it is sound and complete.

To show *soundness*, assume that Γ is a matching problem obtained after termination of a non-failing run of the algorithm. Since the run terminated without failure, all the subsumptions in Γ are solved. We use the subsumptions of the form $X \sqsubseteq^? C \in \Gamma$ to define a substitution σ_Γ. Note that the fact that Γ is a normalized $\mathcal{EL}$-matching problem implies that C is a ground pattern, i.e., a concept description. For each variable $X \in N_V$, we define

$$S_X^\Gamma := \{C \mid X \sqsubseteq^? C \in \Gamma\},$$

and denote the conjunction of all the elements of S_X^Γ as $\sqcap S_X^\Gamma$, where the empty conjunction is $\top$. The substitution σ_Γ is now defined as

$$\sigma_\Gamma(X) := \sqcap S_X^\Gamma \quad \text{for all } X \in N_V.$$

Lemma 10. *σ_Γ is a matcher of Γ w.r.t. $\mathcal{T}$.*

Since the input matching problem Γ_0 is contained in Γ, this lemma shows that σ_Γ is a matcher also of Γ_0 w.r.t. $\mathcal{T}$. This completes the proof of soundness.

Regarding *completeness*, we can use a given matcher of Γ_0 w.r.t. $\mathcal{T}$ to guide the application of the non-deterministic rules such that a non-failing run is generated (see [1] for details).

Lemma 11. *Let σ be a matcher of Γ_0 w.r.t. $\mathcal{T}$. Then there is a non-failing and terminating run of Algorithm 7 producing a matching problem Γ such that σ is a matcher of Γ w.r.t. $\mathcal{T}$.*

This lemma provides the final step towards showing that Algorithm 7 is an NP-decision procedure for matching w.r.t. general TBoxes in $\mathcal{EL}$.

Theorem 12. *The problem of deciding whether a given $\mathcal{EL}$-matching problem has a matcher w.r.t. a given general $\mathcal{EL}$-TBox or not is NP-complete.*

Let us illustrate the working of the algorithm with a small example. We consider the TBox $\mathcal{T} := \{C \sqsubseteq A, C \sqsubseteq \exists s.C, \exists s.B \sqsubseteq \exists s.C\}$ and the matching problem $\Gamma := \{X \sqcap B \sqsubseteq^? \exists s.A, \exists s.B \sqsubseteq^? \exists s.X\}$. Obviously, this problem is neither left- nor right-ground, and thus we need to use Algorithm 7 to solve it. In the beginning, all the subsumptions in Γ are unsolved, and no eager rule is applicable.

In order to apply a non-deterministic rule, the algorithm chooses one of the unsolved subsumptions. Let us assume that this is the first one, i.e., $X \sqcap B \sqsubseteq^? \exists s.A$. Now, we have a (don't know non-deterministic) choice between applying *Decomposition* or *Mutation*. Consider the case where *Decomposition* is applied in such a way that it produces $Dec(X \sqsubseteq^? \exists s.A) = \{X \sqsubseteq^? \exists s.A\}$. The unsolved subsumption $X \sqsubseteq^? \exists s.A$ is then added to Γ, while $X \sqcap B \sqsubseteq^? \exists s.A$ is marked as "solved."

Now, the algorithm applies *Eager Solving (variable on the left)* to $X \sqsubseteq^? \exists s.A$. Since there are no subsumptions with right-hand side X, the rule application does not fail and $X \sqsubseteq^? \exists s.A$ is marked as "solved."

The algorithm then chooses the only unsolved subsumption left: $\exists s.B \sqsubseteq^? \exists s.X$. Again, there is the choice between applying *Decomposition* and *Mutation*. Let us assume that *Decomposition* is chosen, which yields $Dec(\exists s.B \sqsubseteq^? \exists s.X) = \{B \sqsubseteq^? X\}$. The subsumption $\exists s.B \sqsubseteq^? \exists s.X$ is marked as "solved" and the unsolved subsumption $B \sqsubseteq^? X$ is added to Γ.

Now *Eager Solving (variable on the right)* is applied to this subsumption, which leads to failure since $B \not\sqsubseteq_{\mathcal{T}} \exists s.A$.

Backtracking to the last choice point, the algorithm applies *Mutation* to $\exists s.B \sqsubseteq^? \exists s.X$. Let us assume that it chooses the atoms $\exists s.B, \exists s.C$ of $\mathcal{T}$, which is a good choice since $\exists s.B \sqsubseteq_{\mathcal{T}} \exists s.C$. *Mutation* then yields $Dec(\exists s.B \sqsubseteq^? \exists s.B) = \emptyset$ and $Dec(\exists s.C \sqsubseteq^? \exists s.X) = \{C \sqsubseteq^? X\}$. The subsumption $\exists s.B \sqsubseteq^? \exists s.X$ is then marked as "solved" and the unsolved subsumption $C \sqsubseteq^? X$ is added to Γ.

Finally, *Eager Solving (variable on the right)* is applied to this subsumption, which does not fail since $C \sqsubseteq_{\mathcal{T}} \exists s.A$.

Since now all subsumptions are solved, no more rules apply, and the algorithm returns "success." The matcher computed by this run of the algorithm (as defined in the proof of soundness) is $\{X \mapsto \exists s.A\}$.

6 Conclusion

We have extended the known results for matching in $\mathcal{EL}$ [5] to the case where subsumption and equivalence is considered w.r.t. a non-empty general TBox, i.e., a non-empty set of GCIs. For the DL $\mathcal{FL}_0$, matching without GCIs is polynomial, and this remains true even in the extension $\mathcal{ALN}$ of $\mathcal{FL}_0$. It would be interesting

to see how one can solve matching problems w.r.t. general TBoxes in these DLs. Since already subsumption in $\mathcal{FL}_0$ w.r.t. general TBoxes is ExpTime-complete [4], the complexity of solving such matching problems is at least ExpTime-hard. Another interesting open problem is unification in $\mathcal{EL}$ w.r.t. general TBoxes.

References

1. Baader, F., Morawska, B.: Matching with respect to general concept inclusions in the description logic $\mathcal{EL}$. LTCS-Report 14-03, Chair of Automata Theory, Institute of Theoretical Computer Science, Technische Universität Dresden, Dresden, Germany (2014), `http://lat.inf.tu-dresden.de/research/reports.html`
2. Baader, F., Borgwardt, S., Morawska, B.: Extending unification in $\mathcal{EL}$ towards general TBoxes. In: Proc. of the 13th Int. Conf. on Principles of Knowledge Representation and Reasoning (KR 2012), pp. 568–572. AAAI Press (2012)
3. Baader, F., Borgwardt, S., Morawska, B.: A goal-oriented algorithm for unification in $\mathcal{EL}\mathcal{H}_{R^+}$ w.r.t. cycle-restricted ontologies. In: Thielscher, M., Zhang, D. (eds.) AI 2012. LNCS, vol. 7691, pp. 493–504. Springer, Heidelberg (2012)
4. Baader, F., Brandt, S., Lutz, C.: Pushing the $\mathcal{EL}$ envelope. In: Kaelbling, L.P., Saffiotti, A. (eds.) Proc. of the 19th Int. Joint Conf. on Artificial Intelligence (IJCAI 2005), pp. 364–369. Morgan Kaufmann, Los Altos (2005)
5. Baader, F., Küsters, R.: Matching in description logics with existential restrictions. In: Proc. of the 7th Int. Conf. on Principles of Knowledge Representation and Reasoning (KR 2000), pp. 261–272 (2000)
6. Baader, F., Küsters, R., Borgida, A., McGuinness, D.L.: Matching in description logics. J. of Logic and Computation 9(3), 411–447 (1999)
7. Baader, F., Morawska, B.: Unification in the description logic $\mathcal{EL}$. Logical Methods in Computer Science 6(3) (2010)
8. Baader, F., Narendran, P.: Unification of concept terms in description logics. J. of Symbolic Computation 31(3), 277–305 (2001)
9. Borgida, A., Brachman, R.J., McGuinness, D.L., Alperin Resnick, L.: CLASSIC: A structural data model for objects. In: Proc. of the ACM SIGMOD Int. Conf. on Management of Data, pp. 59–67 (1989)
10. Borgida, A., Küsters, R.: What's not in a name? Initial explorations of a structural approach to integrating large concept knowledge-bases. Tech. Rep. DCS-TR-391, Rutgers University (1999)
11. Borgida, A., McGuinness, D.L.: Asking queries about frames. In: Proc. of the 5th Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR 1996), pp. 340–349 (1996)
12. Brandt, S.: Polynomial time reasoning in a description logic with existential restrictions, GCI axioms, and—what else? In: de Mántaras, R.L., Saitta, L. (eds.) Proc. of the 16th Eur. Conf. on Artificial Intelligence (ECAI 2004), pp. 298–302 (2004)

Proceedings of the Thirteenth International Conference on Principles of Knowledge Representation and Reasoning

Extending Unification in $\mathcal{EL}$ towards General TBoxes*

Franz Baader and **Stefan Borgwardt** and **Barbara Morawska**
Theoretical Computer Science, TU Dresden, Germany
{*baader,stefborg,morawska*}*@tcs.inf.tu-dresden.de*

Abstract

Unification in Description Logics (DLs) has been proposed as an inference service that can, for example, be used to detect redundancies in ontologies. The inexpressive Description Logic $\mathcal{EL}$ is of particular interest in this context since, on the one hand, several large biomedical ontologies are defined using $\mathcal{EL}$. On the other hand, unification in $\mathcal{EL}$ has recently been shown to be NP-complete, and thus of significantly lower complexity than unification in other DLs of similarly restricted expressive power. However, the unification algorithms for $\mathcal{EL}$ developed so far cannot deal with general concept inclusion axioms (GCIs). This paper makes a considerable step towards addressing this problem, but the GCIs our new unification algorithm can deal with still need to satisfy a certain cycle restriction.

1 Introduction

The DL $\mathcal{EL}$, which offers the constructors conjunction ($\sqcap$), existential restriction ($\exists r.C$), and the top concept ($\top$), has recently drawn considerable attention since, on the one hand, important inference problems such as the subsumption problem are polynomial in $\mathcal{EL}$, even in the presence of GCIs (Brandt 2004; Baader, Brandt, and Lutz 2005). On the other hand, though quite inexpressive, $\mathcal{EL}$ can be used to define biomedical ontologies, such as the large medical ontology SNOMED CT.[1]

Unification in DLs has been proposed in (Baader and Narendran 2001) (for the DL $\mathcal{FL}_0$, which differs from $\mathcal{EL}$ by offering value restrictions ($\forall r.C$) in place of existential restrictions) as a novel inference service that can, for instance, be used to detect redundancies in ontologies. For example, assume that one developer of a medical ontology defines the concept of a *finding of severe head injury* as

$$\exists \mathsf{finding}.(\mathsf{Head\_injury} \sqcap \exists \mathsf{severity}.\mathsf{Severe}), \quad (1)$$

whereas another one represents it as

$$\exists \mathsf{finding}.(\mathsf{Severe\_injury} \sqcap \exists \mathsf{finding\_site}.\mathsf{Head}). \quad (2)$$

These two concept descriptions are not equivalent, but they are nevertheless meant to represent the same concept. They can obviously be made equivalent by treating the concept names Head_injury and Severe_injury as variables, and substituting the first one by Injury $\sqcap$ ∃finding_site.Head and the second one by Injury $\sqcap$ ∃severity.Severe. In this case, we say that the descriptions are unifiable, and call the substitution that makes them equivalent a *unifier*. Intuitively, such a unifier proposes definitions for the concept names that are used as variables: in our example, we know that, if we define Head_injury as Injury $\sqcap$ ∃finding_site.Head and Severe_injury as Injury $\sqcap$ ∃severity.Severe, then the two concept descriptions (1) and (2) are equivalent w.r.t. these definitions. Here equivalence holds without additional GCIs.

To motivate our interest in unification w.r.t. GCIs, assume that the second developer uses the description

$$\exists \mathsf{status}.\mathsf{Emergency} \quad \sqcap \quad (3)$$
$$\exists \mathsf{finding}.(\mathsf{Severe\_injury} \sqcap \exists \mathsf{finding\_site}.\mathsf{Head})$$

instead of (2). The descriptions (1) and (3) are not unifiable without additional GCIs, but they are unifiable, with the same unifier as above, if the GCI

$$\exists \mathsf{finding}.\exists \mathsf{severity}.\mathsf{Severe} \sqsubseteq \exists \mathsf{status}.\mathsf{Emergency}$$

is present in a background ontology.

In (Baader and Morawska 2009), we were able to show that unification in $\mathcal{EL}$ is of considerably lower complexity than in $\mathcal{FL}_0$: the decision problem in $\mathcal{EL}$ is NP-complete rather than EXPTIME-complete in $\mathcal{FL}_0$. In addition to a brute-force "guess and then test" NP-algorithm (Baader and Morawska 2009), we were able to develop a goal-oriented unification algorithm for $\mathcal{EL}$, in which nondeterministic decisions are only made if they are triggered by "unsolved parts" of the unification problem (Baader and Morawska 2010b), and an algorithm that is based on a reduction to satisfiability in propositional logic (SAT) (Baader and Morawska 2010a), which enables the use of highly-optimized SAT solvers. In (Baader and Morawska 2010b) it was also shown that the approaches for unification of $\mathcal{EL}$-concept descriptions (without any background ontology) can easily be extended to the case of an acyclic TBox as background ontology without really changing the algorithms or increasing their complexity. Basically, by viewing defined concepts as variables, an acyclic TBox can be turned into a unification problem that has as its unique unifier the substitution that replaces the defined concepts by unfolded ver-

*Supported by DFG under grant BA 1122/14-1

[1] see http://www.ihtsdo.org/snomed-ct/

Name	Syntax	Semantics
concept name	A	$A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$
role name	r	$r^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$
top-concept	$\top$	$\top^{\mathcal{I}} = \Delta^{\mathcal{I}}$
conjunction	$C \sqcap D$	$(C \sqcap D)^{\mathcal{I}} = C^{\mathcal{I}} \cap D^{\mathcal{I}}$
existential restr.	$\exists r.C$	$(\exists r.C)^{\mathcal{I}} = \{x \mid \exists y : (x,y) \in r^{\mathcal{I}} \wedge y \in C^{\mathcal{I}}\}$
GCI	$C \sqsubseteq D$	$C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$

Table 1: Syntax and semantics of $\mathcal{EL}$.

sions of their definitions. For GCIs, this simple trick is not possible.

In the present paper, we extend the brute-force "guess and then test" NP-algorithm from (Baader and Morawska 2009) to the case of GCIs, which requires the development of a new characterization of subsumption w.r.t. GCIs in $\mathcal{EL}$. Unfortunately, the algorithm is complete only for general TBoxes (i.e., finite sets of GCIs) that satisfy a certain restriction on cycles, which, however, does not prevent all cycles. For example, the cyclic GCI $\exists \mathsf{child}.\mathsf{Human} \sqsubseteq \mathsf{Human}$ satisfies this restriction, whereas the cyclic GCI $\mathsf{Human} \sqsubseteq \exists \mathsf{parent}.\mathsf{Human}$ does not.

Due to space constraints, we cannot present and prove all our results in detail here. Full proofs and a goal-oriented algorithm for unification in $\mathcal{EL}$ w.r.t. cycle-restricted general TBoxes can be found in (Baader, Borgwardt, and Morawska 2011).

2 The Description Logic $\mathcal{EL}$

Starting with a finite set N_C of *concept names* and a finite set N_R of *role names*, *$\mathcal{EL}$-concept descriptions* are built using the concept constructors *top-concept* ($\top$), *conjunction* ($C \sqcap D$), and *existential restriction* ($\exists r.C$ for every $r \in N_R$). *Nested existential restrictions* $\exists r_1.\exists r_2.\cdots\exists r_n.C$ will sometimes also be written as $\exists r_1 r_2 \dots r_n.C$, where $r_1 r_2 \dots r_n$ is viewed as a word over the alphabet of role names, i.e., an element of N_R^*.

An *interpretation* $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ consists of a nonempty domain $\Delta^{\mathcal{I}}$ and an interpretation function $\cdot^{\mathcal{I}}$ that assigns binary relations on $\Delta^{\mathcal{I}}$ to role names and subsets of $\Delta^{\mathcal{I}}$ to concept descriptions, as shown in the semantics column of Table 1.

A *general concept inclusion (GCI)* is of the form $C \sqsubseteq D$ for concept descriptions C, D, and a general TBox is a finite set of GCIs. An interpretation $\mathcal{I}$ *satisfies* such a GCI if $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$, and it is a *model* of the general TBox $\mathcal{T}$ if it satisfies all GCIs in $\mathcal{T}$.

Subsumption asks whether a given GCI $C \sqsubseteq D$ follows from a general TBox $\mathcal{T}$, i.e. whether every model of $\mathcal{T}$ satisfies $C \sqsubseteq D$. In this case we say C is *subsumed* by D w.r.t. $\mathcal{T}$ and write $C \sqsubseteq_{\mathcal{T}} D$. Subsumption in $\mathcal{EL}$ w.r.t. a general TBox is known to be decidable in polynomial time (Brandt 2004). Our unification algorithm will use a polynomial-time subsumption algorithm as a subprocedure. In order to develop the unification algorithm itself, however, we need a structural characterization of subsumption w.r.t. a general TBox. Before we can present this characterization, we need to introduce some new notions.

An $\mathcal{EL}$-concept description is an *atom* if it is an existential restriction or a concept name. The atoms of an $\mathcal{EL}$-concept description C are the subdescriptions of C that are atoms, and the top-level atoms of C are the atoms occurring in the top-level conjunction of C. Obviously, any $\mathcal{EL}$-concept description is the conjunction of its top-level atoms, where the empty conjunction corresponds to $\top$. The atoms of a general TBox $\mathcal{T}$ are the atoms of all the concept descriptions occurring in $\mathcal{T}$.

We say that a subsumption between two atoms is *structural* if their top-level structure is compatible. To be more precise, we define structural subsumption between atoms as follows: the atom C is *structurally subsumed* by the atom D w.r.t. $\mathcal{T}$ ($C \sqsubseteq^{\mathsf{s}}_{\mathcal{T}} D$) iff either

- $C = D$ is a concept name, or
- $C = \exists r.C'$, $D = \exists r.D'$, and $C' \sqsubseteq_{\mathcal{T}} D'$.

It is easy to see that subsumption w.r.t. $\emptyset$ between two atoms implies structural subsumption w.r.t. $\mathcal{T}$, which in turn implies subsumption w.r.t. $\mathcal{T}$. The unification algorithm presented in this paper crucially depends on the following characterization of subsumption w.r.t. general TBoxes:

Lemma 1. *Let $\mathcal{T}$ be a general TBox and $C_1, \dots, C_n$, $D_1, \dots, D_m$ atoms. Then $C_1 \sqcap \cdots \sqcap C_n \sqsubseteq_{\mathcal{T}} D_1 \sqcap \cdots \sqcap D_m$ iff for every $j \in \{1, \dots, m\}$*

1. *there is an index $i \in \{1, \dots, n\}$ such that $C_i \sqsubseteq^{\mathsf{s}}_{\mathcal{T}} D_j$, or*
2. *there are atoms $A_1, \dots, A_k, B$ of $\mathcal{T}$ ($k \geq 0$) such that*
   a) *$A_1 \sqcap \cdots \sqcap A_k \sqsubseteq_{\mathcal{T}} B$,*
   b) *for every $\eta \in \{1, \dots, k\}$ there is $i \in \{1, \dots, n\}$ with $C_i \sqsubseteq^{\mathsf{s}}_{\mathcal{T}} A_\eta$, and*
   c) *$B \sqsubseteq^{\mathsf{s}}_{\mathcal{T}} D_j$.*

Our proof of this lemma in (Baader, Borgwardt, and Morawska 2011) is based on a new Gentzen-style proof calculus for subsumption w.r.t. a general TBox, which is similar to the one developed in (Hofmann 2005) for subsumption w.r.t. cyclic and general TBoxes.

As mentioned in the introduction, our unification algorithm is complete only for general TBoxes that satisfy a certain restriction on cycles.

Definition 2. The general TBox $\mathcal{T}$ is called *cycle-restricted* iff there is no nonempty word $w \in N_R^+$ and $\mathcal{EL}$-concept description C such that $C \sqsubseteq_{\mathcal{T}} \exists w.C$.

In (Baader, Borgwardt, and Morawska 2011) we show that a given general TBox can easily be tested for cycle-restrictedness. The main idea is that it is sufficient to consider the cases where C is a concept name or $\top$.

Lemma 3. *Let $\mathcal{T}$ be a general TBox. It can be decided in time polynomial in the size of $\mathcal{T}$ whether $\mathcal{T}$ is cycle-restricted or not.*

3 Unification in $\mathcal{EL}$ w.r.t. General TBoxes

We partition the set N_C of concepts names into a set N_v of concept variables (which may be replaced by substitutions) and a set N_c of concept constants (which must not

be replaced by substitutions). A *substitution* σ maps every concept variable to an $\mathcal{EL}$-concept description. It can be extended to concept descriptions in the usual way:

- $\sigma(A) := A$ for all $A \in N_c \cup \{\top\}$,
- $\sigma(C \sqcap D) := \sigma(C) \sqcap \sigma(D)$ and $\sigma(\exists r.C) := \exists r.\sigma(C)$.

An $\mathcal{EL}$-concept description C is *ground* if it does not contain variables. Obviously, a ground concept description is not modified by applying a substitution. A general TBox is *ground* if it does not contain variables.

Definition 4. Let $\mathcal{T}$ be a general TBox that is ground. An *$\mathcal{EL}$-unification problem* w.r.t. $\mathcal{T}$ is a finite set $\Gamma = \{C_1 \sqsubseteq^? D_1, \ldots, C_n \sqsubseteq^? D_n\}$ of subsumptions between $\mathcal{EL}$-concept descriptions. A substitution σ is a *unifier* of Γ w.r.t. $\mathcal{T}$ if σ *solves* all the subsumptions in Γ, i.e., if $\sigma(C_1) \sqsubseteq_{\mathcal{T}} \sigma(D_1), \ldots, \sigma(C_n) \sqsubseteq_{\mathcal{T}} \sigma(D_n)$. We say that Γ is *unifiable* w.r.t. $\mathcal{T}$ if it has a unifier.

Two remarks regarding this definition are in order. First, note that the previous papers on unification in DLs used equivalences $C \equiv^? D$ instead of subsumptions $C \sqsubseteq^? D$. This difference is, however, irrelevant since $C \equiv^? D$ can be seen as a shorthand for the two subsumptions $C \sqsubseteq^? D$ and $D \sqsubseteq^? C$, and $C \sqsubseteq^? D$ has the same unifiers as $C \sqcap D \equiv^? C$.

Second, note that we have restricted the background general TBox $\mathcal{T}$ to be ground. This is not without loss of generality. In fact, if $\mathcal{T}$ contained variables, then we would need to apply the substitution also to its GCIs, and instead of requiring $\sigma(C_i) \sqsubseteq_{\mathcal{T}} \sigma(D_i)$ we would thus need to require $\sigma(C_i) \sqsubseteq_{\sigma(\mathcal{T})} \sigma(D_i)$, which would change the nature of the problem considerably. The treatment of unification w.r.t. acyclic TBoxes in (Baader and Morawska 2010b) actually considers a more general setting, where some of the primitive concepts occurring in the TBox may be variables. The restriction to ground general TBoxes is, however, appropriate for the application scenario sketched in the introduction. In this scenario, there is a fixed background ontology, given by a general TBox, which is extended with definitions of new concepts by several knowledge engineers. Unification w.r.t. the background ontology is used to check whether some of these new definitions actually are redundant, i.e., define the same intuitive concept. Here, some of the primitive concepts newly introduced by one knowledge engineer may be further defined by another one, but we assume that the knowledge engineers use the vocabulary from the background ontology unchanged, i.e., they define *new* concepts rather than adding definitions for concepts that already occur in the background ontology. An instance of this scenario can, e.g., be found in (Campbell et al. 2007), where different extensions of SNOMED CT are checked for overlaps, albeit not by using unification, but by simply testing for equivalence.

In the remainder of this section we will show that $\mathcal{EL}$-unification w.r.t. cycle-restricted TBoxes is NP-complete. NP-hardness is an immediate consequence of the fact that $\mathcal{EL}$-unification is NP-complete w.r.t. the empty TBox (Baader and Morawska 2009). Thus, it is enough to show that $\mathcal{EL}$-unification is still in NP w.r.t. cycle-restricted TBoxes.

Preprocessing To simplify the description of the NP-algorithm, it is convenient to first normalize the TBox and the unification problem appropriately.

An atom is called *flat* if it is a concept name or an existential restriction of the form $\exists r.A$ for a concept name A. The general TBox $\mathcal{T}$ is called *flat* if it contains only GCIs of the form $A \sqcap B \sqsubseteq C$, where A, B are flat atoms or $\top$ and C is a flat atom. The unification problem Γ is called *flat* if it contains only flat subsumptions of the form $C_1 \sqcap \cdots \sqcap C_n \sqsubseteq^? D$, where $n \geq 0$ and $C_1, \ldots, C_n, D$ are flat atoms.[2]

Let Γ be a unification problem and $\mathcal{T}$ a general TBox. By introducing auxiliary variables and concept names, respectively, Γ and $\mathcal{T}$ can be transformed in polynomial time into a flat unification problem Γ' and a flat general TBox $\mathcal{T}'$ such that the unifiability status remains unchanged, i.e., Γ has a unifier w.r.t. $\mathcal{T}$ iff Γ' has a unifier w.r.t. $\mathcal{T}'$. In addition, if $\mathcal{T}$ was cycle-restricted, then so is $\mathcal{T}'$ (see (Baader, Borgwardt, and Morawska 2011) for details). Thus, we can assume without loss of generality that the input unification problem and general TBox are flat.

Local Unifiers The main idea underlying the "in NP" result in (Baader and Morawska 2009) is to show that any $\mathcal{EL}$-unification problem that is unifiable w.r.t. the empty TBox has a so-called local unifier. Here, we generalize the notion of a local unifier to the case of unification w.r.t. cycle-restricted TBoxes, and show that a similar locality result holds in this case.

Let $\mathcal{T}$ be a flat cycle-restricted TBox and Γ a flat unification problem. The atoms of Γ are the atoms of all the concept descriptions occurring in Γ. We define

$$\begin{aligned} \mathrm{At} &:= \{C \mid C \text{ is an atom of } \mathcal{T} \text{ or of } \Gamma\} \text{ and} \\ \mathrm{At}_{\mathsf{nv}} &:= \mathrm{At} \setminus N_v \quad \text{(non-variable atoms).} \end{aligned}$$

Every assignment S of subsets S_X of $\mathrm{At}_{\mathsf{nv}}$ to the variables X in N_v induces the following relation $>_S$ on N_v: $>_S$ is the transitive closure of

$$\{(X, Y) \in N_v \times N_v \mid Y \text{ occurs in an element of } S_X\}.$$

We call the assignment S *acyclic* if $>_S$ is irreflexive (and thus a strict partial order). Any acyclic assignment S induces a unique substitution σ_S, which can be defined by induction along $>_S$:

- If X is a minimal element of N_v w.r.t. $>_S$, then we define $\sigma_S(X) := \bigsqcap_{D \in S_X} D$.
- Assume that $\sigma(Y)$ is already defined for all Y such that $X >_S Y$. Then we define $\sigma_S(X) := \bigsqcap_{D \in S_X} \sigma_S(D)$.

We call a substitution σ *local* if it is of this form, i.e., if there is an acyclic assignment S such that $\sigma = \sigma_S$. If the unifier σ of Γ w.r.t. $\mathcal{T}$ is a local substitution, then we call it a *local unifier* of Γ w.r.t. $\mathcal{T}$.

Theorem 5. *Let $\mathcal{T}$ be a flat cycle-restricted TBox and Γ a flat unification problem. If Γ has a unifier w.r.t. $\mathcal{T}$, then it also has a local unifier w.r.t. $\mathcal{T}$.*

[2] If $n = 0$, then we have an empty conjunction on the left-hand side, which as usual stands for $\top$.

This theorem immediately implies that unification in $\mathcal{EL}$ w.r.t. cycle-restricted TBoxes is decidable within NP. In fact, one can guess an acyclic assignment S in polynomial time. To check whether the induced local substitution σ_S is a unifier of Γ w.r.t. $\mathcal{T}$, we build the general TBox

$$\mathcal{T}_S := \{X \sqsubseteq \prod_{D \in S_X} D,\ \prod_{D \in S_X} D \sqsubseteq X \mid X \in N_v\},$$

and then check in polynomial time whether $C \sqsubseteq_{\mathcal{T} \cup \mathcal{T}_S} D$ holds for all $C \sqsubseteq^? D \in \Gamma$. It is easy to show that this is the case iff $\sigma_S(C) \sqsubseteq_{\mathcal{T}} \sigma_S(D)$ for all $C \sqsubseteq^? D \in \Gamma$.

Corollary 6. *Unification in $\mathcal{EL}$ w.r.t. cycle-restricted TBoxes is in NP.*

Proof of Theorem 5 Assume that γ is a unifier of Γ w.r.t. $\mathcal{T}$. We define the assignment S^γ induced by γ as

$$S^\gamma_X := \{D \in \mathrm{At}_{\mathsf{nv}} \mid \gamma(X) \sqsubseteq_{\mathcal{T}} \gamma(D)\}.$$

The following lemma is the only place in the proof of Theorem 5 where cycle-restrictedness of $\mathcal{T}$ is needed. Later we will give an example (Example 9) that demonstrates that the theorem actually does not hold if this restriction is removed.

Lemma 7. *The assignment S^γ is acyclic.*

Proof. Assume that S^γ is cyclic. Then there are variables $X_1, \ldots, X_n$ and role names $r_1, \ldots, r_{n-1}$ ($n \geq 2$) such that $X_1 = X_n$ and $\exists r_i.X_{i+1} \in S^\gamma(X_i)$ ($i = 1, \ldots, n-1$). But then we have $\gamma(X_i) \sqsubseteq_{\mathcal{T}} \exists r_i.\gamma(X_{i+1})$ for $i = 1, \ldots, n-1$, which yields $\gamma(X_1) \sqsubseteq_{\mathcal{T}} \exists r_1.\gamma(X_2) \sqsubseteq_{\mathcal{T}} \exists r_1.\exists r_2.\gamma(X_3) \sqsubseteq_{\mathcal{T}} \cdots \sqsubseteq_{\mathcal{T}} \exists r_1.\cdots\exists r_{n-1}.\gamma(X_n)$. Since $X_1 = X_n$ and $n \geq 2$, this contradicts our assumption that $\mathcal{T}$ is cycle-restricted. Thus, S^γ must be acyclic. □

Since S^γ is acyclic, it induces a substitution σ_{S^γ}. To simplify the notation, we call this substitution in the following σ^γ. The following lemma implies that σ^γ is a unifier of Γ w.r.t. $\mathcal{T}$, and thus proves Theorem 5.

Lemma 8. *Let $C_1, \ldots, C_n, D \in$ At. Then $\gamma(C_1) \sqcap \ldots \sqcap \gamma(C_n) \sqsubseteq_{\mathcal{T}} \gamma(D)$ implies $\sigma^\gamma(C_1) \sqcap \ldots \sqcap \sigma^\gamma(C_n) \sqsubseteq_{\mathcal{T}} \sigma^\gamma(D)$.*

Proof. We prove the lemma by induction over

$$max\{rd(\sigma^\gamma(E)) \mid E \in \{C_1, \ldots, C_n, D\} \wedge E \text{ not ground}\},$$

where the *role depth* $rd(C)$ of a concept description C is defined as follows: $rd(A) = rd(\top) = 0$ for $A \in N_C$, $rd(C \sqcap D) = \max\{rd(C), rd(D)\}$, $rd(\exists r.C) = 1 + rd(C)$.

First, assume that $D = Y \in N_v$, and let $S^\gamma_Y = \{D_1, \ldots, D_m\}$. By the definition of S^γ, this implies $\gamma(Y) \sqsubseteq_{\mathcal{T}} \gamma(D_1) \sqcap \ldots \sqcap \gamma(D_m)$, and thus

$$\gamma(C_1) \sqcap \ldots \sqcap \gamma(C_n) \sqsubseteq_{\mathcal{T}} \gamma(D_1) \sqcap \ldots \sqcap \gamma(D_m).$$

We apply Lemma 1 to this subsumption. Consider $\gamma(D_j)$ for some $j, 1 \leq j \leq m$. Since D_j is a non-variable atom, $\gamma(D_j)$ is an atom, and thus the first or the second case of the lemma holds.

1. In the first case, there is an $i, 1 \leq i \leq n$, such that one of the following two cases holds:
   (i) *C_i is a non-variable atom and $\gamma(C_i) \sqsubseteq^s_{\mathcal{T}} \gamma(D_j)$.*
   By the definition of $\sqsubseteq^s_{\mathcal{T}}$, there are two possible cases. Either both concept descriptions are the same concept name A, or both are existential restrictions for the same role name r. In the first case, $C_i = A = D_j$, and thus $\sigma^\gamma(C_i) = A = \sigma^\gamma(D_j)$. In the second case, $C_i = \exists r.C'_i$, $D_j = \exists r.D'_j$, and $\gamma(C'_i) \sqsubseteq_{\mathcal{T}} \gamma(D'_j)$. Both C'_i and D'_j are elements of At. The role depth of $\sigma^\gamma(C'_i)$ is obviously smaller than the role depth of $\sigma^\gamma(C_i)$. For the same reason, the role depth of $\sigma^\gamma(D'_j)$ is smaller that the one of $\sigma^\gamma(D_j)$. Since $\sigma^\gamma(D_j)$ is a top-level conjunct in $\sigma^\gamma(D)$, the role depth of $\sigma^\gamma(D'_j)$ is also smaller than the one of $\sigma^\gamma(D)$. Consequently, if C_i or D_j is non-ground, induction yields $\sigma^\gamma(C'_i) \sqsubseteq_{\mathcal{T}} \sigma^\gamma(D'_j)$, and thus also $\sigma^\gamma(C_i) = \exists r.\sigma^\gamma(C'_i) \sqsubseteq_{\mathcal{T}} \exists r.\sigma^\gamma(D'_j) = \sigma^\gamma(D_j)$. If C_i, D_j are both ground, then $\sigma^\gamma(C_i) = C_i = \gamma(C_i) \sqsubseteq_{\mathcal{T}} \gamma(D_j) = D_j = \sigma^\gamma(D_j)$.
   (ii) *$C_i = X$ is a variable and the top-level conjunction of $\gamma(X)$ contains an atom E such that $E \sqsubseteq^s_{\mathcal{T}} \gamma(D_j)$.*
   Then we have $\gamma(X) \sqsubseteq_{\mathcal{T}} E \sqsubseteq_{\mathcal{T}} \gamma(D_j)$, and thus $D_j \in S^\gamma_X$. By the definition of σ^γ, this implies $\sigma^\gamma(C_i) = \sigma^\gamma(X) \sqsubseteq_{\mathcal{T}} \sigma^\gamma(D_j)$.
   Both (i) and (ii) yield $\sigma^\gamma(C_1) \sqcap \ldots \sqcap \sigma^\gamma(C_n) \sqsubseteq_{\mathcal{T}} \sigma^\gamma(D_j)$.
2. In the second case, there are atoms $A_1, \ldots, A_k, B$ of $\mathcal{T}$ such that
   a) $A_1 \sqcap \ldots \sqcap A_k \sqsubseteq_{\mathcal{T}} B$,
   b) for every $\eta, 1 \leq \eta \leq k$, there is $i, 1 \leq i \leq n$, such that one of the following two cases holds:
      (i) C_i is a non-variable atom and $\gamma(C_i) \sqsubseteq^s_{\mathcal{T}} A_\eta$,
      (ii) $C_i = X$ is a variable and the top-level conjunction of $\gamma(X)$ contains an atom E such that $E \sqsubseteq^s_{\mathcal{T}} A_\eta$;
   c) $B \sqsubseteq^s_{\mathcal{T}} \gamma(D_j)$.
   In case (i) we have that either $C_i = A = A_\eta$ is a concept name, or both concept descriptions are existential restrictions $C_i = \exists r.C'_i$ and $A_\eta = \exists r.A'_\eta$ with $\gamma(C'_i) \sqsubseteq_{\mathcal{T}} A'_\eta$. In the first case, we have $\sigma^\gamma(C_i) = A = A_\eta$. In the second case, the case where C_i is ground is again trivial. Otherwise, we can apply induction since $C'_i, A'_\eta \in$ At, the role depth of $\sigma^\gamma(C'_i)$ is smaller than the one of $\sigma^\gamma(C_i)$, and the role depth of A'_η is not counted since it is ground. Thus, we have $\sigma^\gamma(C'_i) \sqsubseteq_{\mathcal{T}} A'_\eta$, which yields $\sigma^\gamma(C_i) \sqsubseteq_{\mathcal{T}} A_\eta$.
   In case (ii), we again have $\gamma(X) \sqsubseteq_{\mathcal{T}} E \sqsubseteq_{\mathcal{T}} A_\eta$, and thus $A_\eta \in S^\gamma_X$. This yields $\sigma^\gamma(C_i) = \sigma^\gamma(X) \sqsubseteq_{\mathcal{T}} A_\eta$.
   For similar reasons as before, we can again show that $B \sqsubseteq^s_{\mathcal{T}} \gamma(D_j)$ implies $B \sqsubseteq_{\mathcal{T}} \sigma^\gamma(D_j)$.
   To sum up, we thus have also in this case $\sigma^\gamma(C_1) \sqcap \ldots \sqcap \sigma^\gamma(C_n) \sqsubseteq_{\mathcal{T}} A_1 \sqcap \ldots \sqcap A_k \sqsubseteq_{\mathcal{T}} B \sqsubseteq_{\mathcal{T}} \sigma^\gamma(D_j)$.

Hence, we have shown that, for all $j, 1 \leq j \leq m$, we have $\sigma^\gamma(C_1) \sqcap \ldots \sqcap \sigma^\gamma(C_n) \sqsubseteq_{\mathcal{T}} \sigma^\gamma(D_j)$, which yields $\sigma^\gamma(C_1) \sqcap \ldots \sqcap \sigma^\gamma(C_n) \sqsubseteq_{\mathcal{T}} \sigma^\gamma(D_1) \sqcap \ldots \sqcap \sigma^\gamma(D_m) = \sigma(D)$. The last identity holds since $D = Y$ and $S^\gamma_Y = \{D_1, \ldots, D_m\}$.

It remains to consider the case where D is a non-variable atom. But then we have

$$\gamma(C_1) \sqcap \ldots \sqcap \gamma(C_n) \sqsubseteq_{\mathcal{T}} \gamma(D),$$

and $\gamma(D)$ is an atom. As for $\gamma(D_j)$ above, we can use Lemma 1 to show that this implies $\sigma^\gamma(C_1) \sqcap \ldots \sqcap \sigma^\gamma(C_n) \sqsubseteq_\mathcal{T} \sigma^\gamma(D)$. □

Example 9 (Cycle-restrictedness is needed). We show that Theorem 5 does not hold for arbitrary general TBoxes. To this purpose, consider the general TBox $\mathcal{T} = \{B \sqsubseteq \exists s.D,\ D \sqsubseteq B\}$, which is not cycle-restricted, and the unification problem

$$\Gamma = \{A_1 \sqcap B \equiv^? Y_1,\ A_2 \sqcap B \equiv^? Y_2,\ \exists s.Y_1 \sqsubseteq^? X,\ \exists s.Y_2 \sqsubseteq^? X,\ X \sqsubseteq^? \exists s.X\}.$$

This problem has the unifier $\gamma := \{Y_1 \mapsto A_1 \sqcap B, Y_2 \mapsto A_2 \sqcap B, X \mapsto \exists s.B\}$. However, the induced assignment S^γ is cyclic since $\gamma(X) = \exists s.B \sqsubseteq_\mathcal{T} \exists s.\exists s.B = \gamma(\exists s.X)$ yields $\exists s.X \in S^\gamma_X$. Thus, γ does not induce a local unifier.

We claim that Γ actually does not have any local unifier w.r.t. $\mathcal{T}$. Assume to the contrary that σ is a local unifier of Γ w.r.t. $\mathcal{T}$. Then $\sigma(X)$ cannot be $\top$ since $\top \not\sqsubseteq_\mathcal{T} \exists s.\top$. Thus, $\sigma(X)$ must contain a top-level atom of the form $\sigma(E)$ for $E \in \mathrm{At}_{\mathsf{nv}}$. This atom cannot be $\sigma(\exists s.Y_i) \equiv_\mathcal{T} \exists s.(A_i \sqcap B)$ for $i \in \{1,2\}$ since then $\sigma(\exists s.Y_j) \sqsubseteq_\mathcal{T} \sigma(E)$ for $j \in \{1,2\} \setminus \{i\}$ would not hold, contradicting the assumption that σ solves $\exists s.Y_j \sqsubseteq^? X$ w.r.t. $\mathcal{T}$. Since local unifiers are induced by acyclic assignments, E cannot be $\sigma(\exists s.X)$, and thus E must be an atom of $\mathcal{T}$. However, none of the atoms $B, D, \exists s.D$ subsume $\exists s.(A_j \sqcap B)$ w.r.t. $\mathcal{T}$, again contradicting the assumption that σ solves $\exists s.Y_j \sqsubseteq^? X$ w.r.t. $\mathcal{T}$.

4 Conclusions

We have shown that unification in $\mathcal{EL}$ stays in NP in the presence of a cycle-restricted general TBox, by giving a brute-force NP-algorithm that tries to guess a local unifier. This algorithm is interesting since it provides a quite simple, self-contained proof for the complexity upper-bound. Indeed, it is much simpler than the original proof (Baader and Morawska 2009; 2010b) of the NP-upper bound for $\mathcal{EL}$ without TBoxes.

In (Baader, Borgwardt, and Morawska 2011), we also introduce a goal-oriented algorithm for unification in $\mathcal{EL}$ in the presence of a cycle-restricted TBox, in which nondeterministic decisions are only made if they are triggered by "unsolved parts" of the unification problem. Another advantage of the goal-oriented algorithm is that it only generates substitutions that are unifiers, whereas the brute-force algorithm generates all local substitutions, and requires a subsequent test of whether this substitution is a unifier. Nevertheless, this algorithms still requires a considerable amount of additional optimization work to be useful in practice.

On the theoretical side, the main topic for future research is to consider unification w.r.t. unrestricted general TBoxes. In order to generalize the brute-force algorithm in this direction, we need to find a more general notion of locality. Starting with the goal-oriented algorithm (Baader, Borgwardt, and Morawska 2011), the idea would be not to fail when a cyclic assignment is generated, but rather to add rules that can break such cycles, similar to what is done in procedures for general E-unification (Morawska 2007).

References

Baader, F., and Morawska, B. 2009. Unification in the description logic $\mathcal{EL}$. In Treinen, R., ed., *Proc. of the 20th Int. Conf. on Rewriting Techniques and Applications (RTA 2009)*, volume 5595 of *Lecture Notes in Computer Science*, 350–364. Springer-Verlag.

Baader, F., and Morawska, B. 2010a. SAT encoding of unification in $\mathcal{EL}$. In Fermüller, C. G., and Voronkov, A., eds., *Proc. of the 17th Int. Conf. on Logic for Programming, Artifical Intelligence, and Reasoning (LPAR-17)*, volume 6397 of *Lecture Notes in Computer Science*, 97–111. Springer-Verlag.

Baader, F., and Morawska, B. 2010b. Unification in the description logic $\mathcal{EL}$. *Logical Methods in Computer Science* 6(3). Special Issue: 20th Int. Conf. on Rewriting Techniques and Applications (RTA'09).

Baader, F., and Narendran, P. 2001. Unification of concept terms in description logics. *J. of Symbolic Computation* 31(3):277–305.

Baader, F.; Borgwardt, S.; and Morawska, B. 2011. Unification in the description logic $\mathcal{EL}$ w.r.t. cycle-restricted TBoxes. LTCS-Report 11-05, Chair of Automata Theory, Institute of Theoretical Computer Science, Technische Universität Dresden, Dresden, Germany. See http://lat.inf.tu-dresden.de/research/reports.html.

Baader, F.; Brandt, S.; and Lutz, C. 2005. Pushing the $\mathcal{EL}$ envelope. In Kaelbling, L. P., and Saffiotti, A., eds., *Proc. of the 19th Int. Joint Conf. on Artificial Intelligence (IJCAI 2005)*, 364–369. Morgan-Kaufmann Publishers.

Brandt, S. 2004. Polynomial time reasoning in a description logic with existential restrictions, GCI axioms, and—what else? In de Mántaras, R. L., and Saitta, L., eds., *Proc. of the 16th Eur. Conf. on Artificial Intelligence (ECAI 2004)*, 298–302. IOS Press.

Campbell, J. R.; Lopez Osornio, A.; de Quiros, F.; Luna, D.; and Reynoso, G. 2007. Semantic interoperability and SNOMED CT: A case study in clinical problem lists. In Kuhn, K.; Warren, J.; and Leong, T.-Y., eds., *Proc. of the 12th World Congress on Health (Medical) Informatics (MEDINFO 2007)*, 2401–2402. IOS Press.

Hofmann, M. 2005. Proof-theoretic approach to description-logic. In *Proc. of the 20th IEEE Symp. on Logic in Computer Science (LICS 2005)*, 229–237.

Morawska, B. 2007. General E-unification with eager variable elimination and a nice cycle rule. *J. of Automated Reasoning* 39(1):77–106.

A Goal-Oriented Algorithm for Unification in $\mathcal{ELH}_{R^+}$ w.r.t. Cycle-Restricted Ontologies*

Franz Baader, Stefan Borgwardt, and Barbara Morawska

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

Abstract. Unification in Description Logics (DLs) has been proposed as an inference service that can, for example, be used to detect redundancies in ontologies. For the DL $\mathcal{EL}$, which is used to define several large biomedical ontologies, unification is NP-complete. A goal-oriented NP unification algorithm for $\mathcal{EL}$ that uses nondeterministic rules to transform a given unification problem into solved form has recently been presented. In this paper, we extend this goal-oriented algorithm in two directions: on the one hand, we add general concept inclusion axioms (GCIs), and on the other hand, we add role hierarchies ($\mathcal{H}$) and transitive roles (R^+). For the algorithm to be complete, however, the ontology consisting of the GCIs and role axioms needs to satisfy a certain cycle restriction.

1 Introduction

The DL $\mathcal{EL}$, which offers the constructors conjunction ($\sqcap$), existential restriction ($\exists r.C$), and the top concept ($\top$), has recently drawn considerable attention since, on the one hand, important inference problems such as the subsumption problem are polynomial in $\mathcal{EL}$, even in the presence of general concept inclusions (GCIs) [12]. On the other hand, though quite inexpressive, $\mathcal{EL}$ can be used to define biomedical ontologies, such as the large medical ontology SNOMED CT.[1] A tractable extension of $\mathcal{EL}$ [7], which includes role hierarchy and transitivity axioms, is the basis of the OWL 2 EL profile of the new Web Ontology Language OWL 2.[2]

Unification in DLs has been proposed in [11] as a novel inference service that can, for instance, be used to detect redundancies in ontologies. For example, assume that one developer of a medical ontology defines the concept of a *patient with severe injury of the frontal lobe* as

$$\exists \mathsf{finding}.(\mathsf{Frontal\_lobe\_injury} \sqcap \exists \mathsf{severity}.\mathsf{Severe}), \tag{1}$$

whereas another one represents it as

$$\exists \mathsf{finding}.(\mathsf{Severe\_injury} \sqcap \exists \mathsf{finding\_site}.\exists \mathsf{part\_of}.\mathsf{Frontal\_lobe}). \tag{2}$$

* Supported by DFG under grant BA 1122/14-1.

[1] see http://www.ihtsdo.org/snomed-ct/

[2] See http://www.w3.org/TR/owl2-profiles/

M. Thielscher and D. Zhang (Eds.): AI 2012, LNCS 7691, pp. 493–504, 2012.

These two concept descriptions are not equivalent, but they are nevertheless meant to represent the same concept. They can obviously be made equivalent by treating the concept names Frontal_lobe_injury and Severe_injury as variables, and substituting the first one by Injury ⊓ ∃finding_site.∃part_of.Frontal_lobe and the second one by Injury ⊓ ∃severity.Severe. In this case, we say that the descriptions are unifiable, and call the substitution that makes them equivalent a *unifier*.

Our interest in unification w.r.t. GCIs, role hierarchies, and transitive roles stems from the fact that these features are important for expressing medical knowledge. For example, assume that the developers use the descriptions (3) and (4) instead of (1) and (2):

$$\exists \mathsf{finding}.\exists \mathsf{finding\_site}.\exists \mathsf{part\_of}.\mathsf{Brain} \sqcap \exists \mathsf{finding}.(\mathsf{Frontal\_lobe\_injury} \sqcap \exists \mathsf{severity}.\mathsf{Severe}) \tag{3}$$

$$\exists \mathsf{status}.\mathsf{Emergency} \sqcap \exists \mathsf{finding}.(\mathsf{Severe\_injury} \sqcap \exists \mathsf{finding\_site}.\exists \mathsf{part\_of}.\mathsf{Frontal\_lobe}) \tag{4}$$

The descriptions (3) and (4) are not unifiable without additional background knowledge, but they are unifiable, with the same unifier as above, if the GCIs

$$\exists \mathsf{finding}.\exists \mathsf{severity}.\mathsf{Severe} \sqsubseteq \exists \mathsf{status}.\mathsf{Emergency},$$
$$\mathsf{Frontal\_lobe} \sqsubseteq \exists \mathsf{proper\_part\_of}.\mathsf{Brain}$$

are present in a background ontology and this ontology additionally states that part_of is transitive and proper_part_of is a subrole of part_of.

In [8], we were able to show that unification in the DL $\mathcal{EL}$ (without GCIs and role axioms) is NP-complete. In addition to a brute-force "guess and then test" NP-algorithm [8], we have developed a goal-oriented unification algorithm for $\mathcal{EL}$, in which nondeterministic decisions are only made if they are triggered by "unsolved parts" of the unification problem [10], and an algorithm that is based on a reduction to satisfiability in propositional logic (SAT) [9], which enables the use of highly-optimized SAT solvers [14]. Whereas both approaches are clearly better than the brute-force algorithm, none of them is uniformly better than the other. First experiments with our system UEL [1] show that the SAT translation is usually faster in deciding unifiability, but it needs more space than the goal-oriented algorithm and it produces more uninteresting and large unifiers. In fact, the SAT translation generates all so-called local unifiers, whereas the goal-oriented algorithm produces all so-called minimal unifiers, though it may also produce some non-minimal ones. The set of minimal unifiers is a subset of the set of local unifiers, and in our experiments the minimal unifiers usually made more sense in the application.

In [10] it was shown that the approaches for unification of $\mathcal{EL}$-concept descriptions (without any background ontology) mentioned above can easily be extended to the case of a so-called acyclic TBox (a simple form of GCIs, which basically introduce abbreviations for concept descriptions) as background ontology without really changing the algorithms or increasing their complexity.

For more general GCIs, such a simple solution is no longer possible. In [2], we extended the brute-force "guess and then test" NP-algorithm from [8] to the case of GCIs, which required the development of a new characterization of subsumption w.r.t. GCIs in $\mathcal{EL}$. Unfortunately, the algorithm is complete only for general TBoxes (i.e., finite sets of GCIs) that satisfy a certain restriction on cycles, which, however, does not prevent all cycles. For example, the cyclic GCI $\exists \mathsf{child}.\mathsf{Human} \sqsubseteq \mathsf{Human}$ satisfies this restriction, whereas the cyclic GCI $\mathsf{Human} \sqsubseteq \exists \mathsf{parent}.\mathsf{Human}$ does not. In [5] we provide a more practical unification algorithm that is based on a translation into SAT, and can also deal with role hierarchies and transitive roles, but still needs the ontology (now consisting of GCIs and role axioms) to be cycle-restricted. In the presence of role hierarchies ($\mathcal{H}$) and transitive roles (R^+), we use the name $\mathcal{ELH}_{R^+}$ rather than $\mathcal{EL}$ for the logic.

Motivated by our experience that, for the case of $\mathcal{EL}$ without background ontology, the goal-oriented algorithm sometimes behaves better than the one based on a translation into SAT, we introduce in this paper a goal-oriented algorithm for unification in $\mathcal{ELH}_{R^+}$ w.r.t. cycle-restricted ontologies.[3] Full proofs of the presented results can be found in [3].

2 The Description Logics $\mathcal{EL}$ and $\mathcal{ELH}_{R^+}$

The expressiveness of a DL is determined both by the formalism for describing concepts (the concept description language) and the terminological formalism, which can be used to state additional constraints on the interpretation of concepts and roles in a so-called ontology.

The *concept description language* considered in this paper is called $\mathcal{EL}$. Starting with a finite set N_C of *concept names* and a finite set N_R of *role names*, *$\mathcal{EL}$-concept descriptions* are built from concept names using the constructors *conjunction* ($C \sqcap D$), *existential restriction* ($\exists r.C$ for every $r \in N_R$), and *top* ($\top$). Since in this paper we only consider $\mathcal{EL}$-concept descriptions, we will sometimes dispense with the prefix $\mathcal{EL}$.

On the *semantic side*, concept descriptions are interpreted as sets. To be more precise, an *interpretation* $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ consists of a non-empty domain $\Delta^{\mathcal{I}}$ and an interpretation function $\cdot^{\mathcal{I}}$ that maps concept names to subsets of $\Delta^{\mathcal{I}}$ and role names to binary relations over $\Delta^{\mathcal{I}}$. This function is inductively extended to concept descriptions as follows:

$$\top^{\mathcal{I}} := \Delta^{\mathcal{I}}, \quad (C \sqcap D)^{\mathcal{I}} := C^{\mathcal{I}} \cap D^{\mathcal{I}}, \quad (\exists r.C)^{\mathcal{I}} := \{x \mid \exists y : (x, y) \in r^{\mathcal{I}} \wedge y \in C^{\mathcal{I}}\}$$

A *general concept inclusion axiom (GCI)* is of the form $C \sqsubseteq D$ for concept descriptions C, D, a *role hierarchy axiom* is of the form $r \sqsubseteq s$ for role names

[3] A previous version of this paper, which considers unification in $\mathcal{EL}$ w.r.t. cycle-restricted ontologies, but without role hierarchies and transitive roles, has been presented in 2012 at the Description Logic workshop (see [4]).

r, s, and a *transitivity axiom* is of the form $r \circ r \sqsubseteq r$ for a role name r. An interpretation $\mathcal{I}$ *satisfies* such an axiom $C \sqsubseteq D$, $r \sqsubseteq s$, $r \circ r \sqsubseteq r$, respectively, iff

$$C^{\mathcal{I}} \subseteq D^{\mathcal{I}}, \quad r^{\mathcal{I}} \subseteq s^{\mathcal{I}}, \quad \text{and} \quad r^{\mathcal{I}} \circ r^{\mathcal{I}} \subseteq r^{\mathcal{I}},$$

where $\circ$ stands for composition of binary relations. An $\mathcal{EL}\mathcal{H}_{R^+}$*-ontology* is a finite set of such axioms. It is an $\mathcal{EL}$*-ontology* if it contains only GCIs. An interpretation is a *model* of an ontology if it satisfies all its axioms.

A concept description C is *subsumed* by a concept description D w.r.t. an ontology $\mathcal{O}$ (written $C \sqsubseteq_{\mathcal{O}} D$) if every model of $\mathcal{O}$ satisfies the GCI $C \sqsubseteq D$. We say that C is *equivalent* to D w.r.t. $\mathcal{O}$ ($C \equiv_{\mathcal{O}} D$) if $C \sqsubseteq_{\mathcal{O}} D$ and $D \sqsubseteq_{\mathcal{O}} C$. If $\mathcal{O}$ is empty, we also write $C \sqsubseteq D$ and $C \equiv D$ instead of $C \sqsubseteq_{\mathcal{O}} D$ and $C \equiv_{\mathcal{O}} D$, respectively. As shown in [12,7], subsumption w.r.t. $\mathcal{EL}\mathcal{H}_{R^+}$-ontologies (and thus also w.r.t. $\mathcal{EL}$-ontologies) is decidable in polynomial time.

Since conjunction is interpreted as intersection, the concept descriptions $(C \sqcap D) \sqcap E$ and $C \sqcap (D \sqcap E)$ are always equivalent. Thus, we dispense with parentheses and write nested conjunctions in flat form $C_1 \sqcap \cdots \sqcap C_n$. Nested existential restrictions $\exists r_1.\exists r_2. \ldots \exists r_n.C$ will sometimes also be written as $\exists r_1 r_2 \ldots r_n.C$, where $r_1 r_2 \ldots r_n$ is viewed as a word over the alphabet of role names, i.e. an element of N_R^*.

The *role hierarchy* induced by $\mathcal{O}$ is a binary relation $\trianglelefteq_{\mathcal{O}}$ on N_R, which is defined as the reflexive-transitive closure of the relation $\{(r, s) \mid r \sqsubseteq s \in \mathcal{O}\}$. Using elementary reachability algorithms, the role hierarchy can be computed in polynomial time in the size of $\mathcal{O}$. It is easy to see that $r \trianglelefteq_{\mathcal{O}} s$ implies that $r^{\mathcal{I}} \subseteq s^{\mathcal{I}}$ for all models $\mathcal{I}$ of $\mathcal{O}$. Given an $\mathcal{EL}\mathcal{H}_{R^+}$-ontology $\mathcal{O}$, we call the role t *transitive w.r.t.* $\mathcal{O}$ if $\mathcal{O}$ contains the axiom $t \circ t \sqsubseteq t$. If $\mathcal{O}$ is clear from the context, we often omit the suffix "w.r.t. $\mathcal{O}$" and call t a *transitive* role.

An $\mathcal{EL}$-concept description is an *atom* if it is an existential restriction or a concept name. The atoms of an $\mathcal{EL}$-concept description C are the subdescriptions of C that are atoms, and the top-level atoms of C are the atoms occurring in the top-level conjunction of C. Obviously, any $\mathcal{EL}$-concept description is the conjunction of its top-level atoms, where the empty conjunction corresponds to $\top$. The atoms of an $\mathcal{EL}\mathcal{H}_{R^+}$-ontology $\mathcal{O}$ are the atoms of all the concept descriptions occurring in GCIs of $\mathcal{O}$.

We say that a subsumption between two atoms is *structural* if their top-level structure is compatible. To be more precise, following [5] we define structural subsumption between atoms as follows: the atom C is *structurally subsumed* by the atom D w.r.t. $\mathcal{O}$ ($C \sqsubseteq^{\mathsf{s}}_{\mathcal{O}} D$) iff one of the following holds:

1. $C = D$ is a concept name,
2. $C = \exists r.C'$, $D = \exists s.D'$, $r \trianglelefteq_{\mathcal{O}} s$, and $C' \sqsubseteq_{\mathcal{O}} D'$.
3. $C = \exists r.C'$, $D = \exists s.D'$, and $C' \sqsubseteq_{\mathcal{O}} \exists t.D'$ for a transitive role t such that $r \trianglelefteq_{\mathcal{O}} t \trianglelefteq_{\mathcal{O}} s$.

It is easy to see that subsumption w.r.t. $\emptyset$ between two atoms implies structural subsumption w.r.t. $\mathcal{O}$, which in turn implies subsumption w.r.t. $\mathcal{O}$. The unification algorithm presented below crucially depends on the following characterization of subsumption:

Lemma 1. *Let $\mathcal{O}$ be an $\mathcal{ELH}_{R^+}$-ontology and $C_1, \ldots, C_n$, $D_1, \ldots, D_m$ be atoms. Then $C_1 \sqcap \cdots \sqcap C_n \sqsubseteq_{\mathcal{O}} D_1 \sqcap \cdots \sqcap D_m$ iff for every $j \in \{1, \ldots, m\}$*

1. *there is an index $i \in \{1, \ldots, n\}$ such that $C_i \sqsubseteq^{\mathsf{s}}_{\mathcal{O}} D_j$ or*
2. *there are atoms $A_1, \ldots, A_k, B$ of $\mathcal{O}$ ($k \geq 0$) such that*
   (a) $A_1 \sqcap \cdots \sqcap A_k \sqsubseteq_{\mathcal{O}} B$,
   (b) for every $\eta \in \{1, \ldots, k\}$ there is $i \in \{1, \ldots, n\}$ with $C_i \sqsubseteq^{\mathsf{s}}_{\mathcal{O}} A_\eta$, and
   (c) $B \sqsubseteq^{\mathsf{s}}_{\mathcal{O}} D_j$.

Our proof of this lemma in [3] is based on a Gentzen-style proof calculus for subsumption w.r.t. $\mathcal{ELH}_{R^+}$-ontologies, which is similar to the one developed in [15] for subsumption w.r.t. $\mathcal{EL}$-ontologies. Although this characterization looks identical to the one given in [2] for the case of $\mathcal{EL}$-ontologies it differs from that characterization in that it uses a more general notion of structural subsumption. Also note that the characterization of subsumption w.r.t. $\mathcal{ELH}_{R^+}$-ontologies employed in [5] to show correctness of the the SAT translation is different from the one given above, and it is proved using a rewriting approach rather than a Gentzen-style proof calculus.

As mentioned in the introduction, our unification algorithm is complete only for $\mathcal{ELH}_{R^+}$-ontologies that satisfy a certain restriction on cycles.

Definition 2. *The $\mathcal{ELH}_{R^+}$-ontology $\mathcal{O}$ is called* cycle-restricted *iff there is no nonempty word $w \in N_R^+$ and $\mathcal{EL}$-concept description C such that $C \sqsubseteq_{\mathcal{O}} \exists w.C$.*

In [5] we show that a given $\mathcal{ELH}_{R^+}$-ontology can be tested for cycle-restrictedness in polynomial time. The main idea is that it is sufficient to consider the cases where C is a concept name or $\top$.

3 Unification in $\mathcal{ELH}_{R^+}$

We partition the set N_C into a set N_v of concept variables (which may be replaced by substitutions) and a set N_c of concept constants (which must not be replaced by substitutions). A *substitution* σ maps every concept variable to an $\mathcal{EL}$-concept description. It is extended to concept descriptions in the usual way:

- $\sigma(A) := A$ for all $A \in N_c \cup \{\top\}$,
- $\sigma(C \sqcap D) := \sigma(C) \sqcap \sigma(D)$ and $\sigma(\exists r.C) := \exists r.\sigma(C)$.

An $\mathcal{EL}$-concept description C is *ground* if it does not contain variables. Obviously, a ground concept description is not modified by applying a substitution. An $\mathcal{ELH}_{R^+}$-ontology is *ground* if it does not contain variables.

Definition 3. *Let $\mathcal{O}$ be an $\mathcal{ELH}_{R^+}$-ontology that is ground. An $\mathcal{ELH}_{R^+}$-unification problem w.r.t. $\mathcal{O}$ is a finite set $\Gamma = \{C_1 \sqsubseteq^? D_1, \ldots, C_n \sqsubseteq^? D_n\}$ of subsumptions between $\mathcal{EL}$-concept descriptions. A substitution σ is a* unifier *of Γ w.r.t. $\mathcal{O}$ if σ* solves *all the subsumptions in Γ, i.e. if $\sigma(C_1) \sqsubseteq_{\mathcal{O}} \sigma(D_1), \ldots, \sigma(C_n) \sqsubseteq_{\mathcal{O}} \sigma(D_n)$. We say that Γ is* unifiable *w.r.t. $\mathcal{O}$ if it has a unifier.*

Note that some of the previous papers on unification in DLs use equivalences $C \equiv^? D$ instead of subsumptions $C \sqsubseteq^? D$. This difference is, however, irrelevant since $C \equiv^? D$ can be seen as a shorthand for the two subsumptions $C \sqsubseteq^? D$ and $D \sqsubseteq^? C$, and $C \sqsubseteq^? D$ has the same unifiers as $C \sqcap D \equiv^? C$. Also note that we have restricted the background ontology $\mathcal{O}$ to be ground. This is not without loss of generality. If $\mathcal{O}$ contained variables, then we would need to apply the substitution also to its GCIs, and instead of requiring $\sigma(C_i) \sqsubseteq_{\mathcal{O}} \sigma(D_i)$ we would thus need to require $\sigma(C_i) \sqsubseteq_{\sigma(\mathcal{O})} \sigma(D_i)$, which would change the nature of the problem considerably (see [6] for a more detailed discussion).

Preprocessing. To simplify the description of the algorithm, it is convenient to first normalize the ontology and the unification problem appropriately. An atom is called *flat* if it is a concept name or an existential restriction of the form $\exists r.A$ for a concept name A. The $\mathcal{ELH}_{R^+}$-ontology $\mathcal{O}$ is called *flat* if it contains only GCIs of the form $A \sqcap B \sqsubseteq C$, where A, B are flat atoms or $\top$ and C is a flat atom. The unification problem Γ is called *flat* if it contains only flat subsumptions of the form $C_1 \sqcap \cdots \sqcap C_n \sqsubseteq^? D$, where $n \geq 0$ and $C_1, \ldots, C_n, D$ are flat atoms.[4] Let Γ be a unification problem and $\mathcal{O}$ an $\mathcal{ELH}_{R^+}$-ontology. By introducing auxiliary variables and concept names, respectively, Γ and $\mathcal{O}$ can be transformed in polynomial time into a flat unification problem Γ' and a flat $\mathcal{ELH}_{R^+}$-ontology $\mathcal{O}'$ such that the unifiability status remains unchanged, i.e., Γ has a unifier w.r.t. $\mathcal{O}$ iff Γ' has a unifier w.r.t. $\mathcal{O}'$. In addition, if $\mathcal{O}$ was cycle-restricted, then so is $\mathcal{O}'$ (see [6] for details). Thus, we can assume without loss of generality that the input unification problem and ontology are flat.

Local Unifiers. The main idea underlying the "in NP" results in [8,2] is to show that any unification problem that is unifiable has a so-called local unifier.

We denote by At the set of atoms occurring as subdescriptions in subsumptions in Γ or axioms in $\mathcal{O}$ and define

$$\mathrm{At}_{\mathsf{tr}} := \mathrm{At} \cup \{\exists t.D' \mid \exists s.D' \in \mathrm{At},\ t \trianglelefteq_{\mathcal{O}} s,\ t \text{ transitive}\}.$$

Furthermore, we define the set of *non-variable atoms* by $\mathrm{At}_{\mathsf{nv}} := \mathrm{At}_{\mathsf{tr}} \setminus N_v$. Though the elements of $\mathrm{At}_{\mathsf{nv}}$ cannot be variables, they may contain variables if they are of the form $\exists r.X$ for some role r and a variable X.

We call a function S that associates every variable $X \in N_v$ with a set $S_X \subseteq \mathrm{At}_{\mathsf{nv}}$ an *assignment*. Such an assignment induces the following relation $>_S$ on N_v: $>_S$ is the transitive closure of

$$\{(X, Y) \in N_v \times N_v \mid Y \text{ occurs in an element of } S_X\}.$$

We call the assignment S *acyclic* if $>_S$ is irreflexive (and thus a strict partial order). Any acyclic assignment S induces a unique substitution σ_S, which can be defined by induction along $>_S$:

[4] If $n = 0$, then we have an empty conjunction on the left-hand side, which as usual stands for $\top$.

- If $X \in N_v$ is minimal w.r.t. $>_S$, then we define $\sigma_S(X) := \bigsqcap_{D \in S_X} D$.
- Assume that $\sigma(Y)$ is already defined for all Y such that $X >_S Y$. Then we define $\sigma_S(X) := \bigsqcap_{D \in S_X} \sigma_S(D)$.

We call a substitution σ *local* if it is of this form, i.e., if there is an acyclic assignment S such that $\sigma = \sigma_S$. If the unifier σ of Γ w.r.t. $\mathcal{O}$ is a local substitution, then we call it a *local unifier* of Γ w.r.t. $\mathcal{O}$.

The main technical result shown in [2] is that any unifiable $\mathcal{EL}$-unification problem w.r.t. a cycle-restricted ontology has a local unifier. This yields the following brute-force unification algorithm for $\mathcal{EL}$ w.r.t. cycle-restricted ontologies: first guess an acyclic assignment S, and then check whether the induced local substitution σ_S solves Γ. As shown in [2], this algorithm runs in nondeterministic polynomial time. NP-hardness follows from the fact that already unification in $\mathcal{EL}$ w.r.t. the empty ontology is NP-hard [8]. In [2] it is also shown why cycle-restrictedness is needed: there is a non-cycle-restricted $\mathcal{EL}$-ontology $\mathcal{O}$ and an $\mathcal{EL}$-unification problem Γ such that Γ has a unifier w.r.t. $\mathcal{O}$, but it does not have a local unifier.

4 A Goal-Oriented Unification Algorithm

The brute-force algorithm is not practical since it blindly guesses an acyclic assignment and only afterwards checks whether the guessed assignment induces a unifier. We now introduce a more goal-oriented unification algorithm, in which nondeterministic decisions are only made if they are triggered by "unsolved parts" of the unification problem. In addition, failure due to wrong guesses can be detected early. Any non-failing run of the algorithm produces a unifier, i.e., there is no need for checking whether the assignment computed by this run really induces a unifier. This goal-oriented algorithm generalizes the algorithm for unification in $\mathcal{EL}$ (without background ontology) introduced in [10], though the rules look quite different because in the present paper we consider unification problems that consist of subsumptions whereas in [10] we considered equivalences. We assume without loss of generality that the cycle-restricted $\mathcal{ELH}_{R^+}$-ontology $\mathcal{O}$ and the unification problem Γ_0 are flat. Given $\mathcal{O}$ and Γ_0, the sets At, $\mathrm{At}_{\mathsf{tr}}$, and $\mathrm{At}_{\mathsf{nv}}$ are defined as above. Starting with Γ_0, the algorithm maintains a current unification problem Γ and a current acyclic assignment S, which initially assigns the empty set to all variables. In addition, for each subsumption in Γ it maintains the information on whether it is *solved* or not. Initially, all subsumptions are unsolved, except those with a variable on the right-hand side. Rules are applied only to unsolved subsumptions. A (non-failing) rule application does the following:

- it solves exactly one unsolved subsumption,
- it may extend the current assignment S, and
- it may introduce new flat subsumptions built from elements of $\mathrm{At}_{\mathsf{tr}}$.

Each rule application that extends S_X additionally *expands* Γ *w.r.t.* X as follows: every subsumption $\mathfrak{s} \in \Gamma$ of the form $C_1 \sqcap \cdots \sqcap C_n \sqsubseteq^? X$ is *expanded* by adding the subsumption $C_1 \sqcap \cdots \sqcap C_n \sqsubseteq^? A$ to Γ for every $A \in S_X$.

Eager Ground Solving:

Condition: This rule applies to $\mathfrak{s} = C_1 \sqcap \cdots \sqcap C_n \sqsubseteq^? D$ if it is ground.
Action: If $C_1 \sqcap \cdots \sqcap C_n \sqsubseteq_{\mathcal{O}} D$ does not hold, the rule application fails. Otherwise, $\mathfrak{s}$ is marked as *solved*.

Eager Solving:

Condition: This rule applies to $\mathfrak{s} = C_1 \sqcap \cdots \sqcap C_n \sqsubseteq^? D$ if either
- there is $i \in \{1, \ldots, n\}$ such that $C_i = D$ or $C_i = X \in N_v$ and $D \in S_X$, or
- D is ground and $\bigsqcap \mathcal{G} \sqsubseteq_{\mathcal{O}} D$ holds, where $\mathcal{G}$ is the set of all ground atoms in $\{C_1, \ldots, C_n\} \cup \bigcup_{X \in \{C_1, \ldots, C_n\} \cap N_v} S_X$.

Action: Its application marks $\mathfrak{s}$ as *solved*.

Eager Extension:

Condition: This rule applies to $\mathfrak{s} = C_1 \sqcap \cdots \sqcap C_n \sqsubseteq^? D$ if there is $i \in \{1, \ldots, n\}$ with $C_i = X \in N_v$ and $\{C_1, \ldots, C_n\} \setminus \{X\} \subseteq S_X$.
Action: Its application adds D to S_X. If this makes S cyclic, the rule application fails. Otherwise, Γ is expanded w.r.t. X and $\mathfrak{s}$ is marked as *solved*.

Fig. 1. The eager rules of the unification algorithm

Subsumptions are only added if they are not already present in Γ. If a new subsumption is added to Γ, either by a rule application or by expansion of Γ, then it is initially designated unsolved, except if it has a variable on the right-hand side. Once a subsumption is in Γ, it will not be removed. Likewise, if a subsumption in Γ is marked as solved, then it will not become unsolved later.

If a subsumption is marked as solved, this does not mean that it is already solved by the substitution induced by the current assignment. It may be the case that the task of satisfying the subsumption was deferred to solving other subsumptions which are "smaller" than the given subsumption in a well-defined sense. The task of solving a subsumption whose right-hand side is a variable is deferred to solving the subsumptions introduced by expansion.

The rules of the algorithm consist of the three *eager* rules Eager Ground Solving, Eager Solving, and Eager Extension (see Figure 1), and several *nondeterministic* rules (see Figures 2 and 3). Eager rules are applied with higher priority than nondeterministic rules. Among the eager rules, Eager Ground Solving has the highest priority, then comes Eager Solving, and then Eager Extension.

Algorithm 4. Let Γ_0 be a flat $\mathcal{EL}$-unification problem. We set $\Gamma := \Gamma_0$ and $S_X := \emptyset$ for all $X \in N_v$. While Γ contains an unsolved subsumption, apply the steps (1), (2), and (3).

(1) **Eager rule application:** If some eager rules apply to an unsolved subsumption $\mathfrak{s}$ in Γ, apply one of highest priority. If the rule application fails, then return "not unifiable".
(2) **Nondeterministic rule application:** If no eager rule is applicable, let $\mathfrak{s}$ be an unsolved subsumption in Γ. If one of the nondeterministic rules applies to $\mathfrak{s}$, nondeterministically choose one of these rules and apply it. If none of these rules apply to $\mathfrak{s}$ or the rule application fails, then return "not unifiable".

Decomposition 1:

Condition: This rule applies to $\mathfrak{s} = C_1 \sqcap \cdots \sqcap C_n \sqsubseteq^? \exists s.D'$ if there is an index $i \in \{1, \ldots, n\}$ with $C_i = \exists r.C'$ and $r \trianglelefteq_{\mathcal{O}} s$.
Action: Its application chooses such an index i, adds the subsumption $C' \sqsubseteq^? D'$ to Γ, expands it w.r.t. D' if D' is a variable, and marks $\mathfrak{s}$ as *solved*.

Decomposition 2:

Condition: This rule applies to $\mathfrak{s} = C_1 \sqcap \cdots \sqcap C_n \sqsubseteq^? \exists s.D'$ if there is an index $i \in \{1, \ldots, n\}$ and a transitive role t with $C_i = \exists r.C'$ and $r \trianglelefteq_{\mathcal{O}} t \trianglelefteq_{\mathcal{O}} s$.
Action: Its application chooses such an index i, adds the subsumption $C' \sqsubseteq^? \exists t.D'$ to Γ and marks $\mathfrak{s}$ as *solved*.

Extension:

Condition: This rule applies to $\mathfrak{s} = C_1 \sqcap \cdots \sqcap C_n \sqsubseteq^? D$ if there is an index $i \in \{1, \ldots, n\}$ with $C_i \in N_v$.
Action: Its application chooses such an i and adds D to S_{C_i}. If this makes S cyclic, the rule application fails. Otherwise, Γ is expanded w.r.t. C_i and $\mathfrak{s}$ is marked as *solved*.

Fig. 2. The nondeterministic rules *Decomposition 1 and 2* and *Extension*

(3) **Eager application of Decomposition:** If in the previous step one of the rules *Mutation 2 or 3* was applied, do the following for all subsumptions $\mathfrak{s}'$ added to Γ by this rule application: If one of the rules *Decomposition 1 or 2* applies to $\mathfrak{s}'$, nondeterministically choose one of the applicable decomposition rules and apply it to $\mathfrak{s}'$.[5]

Once all subsumptions are solved, return the substitution σ induced by the current assignment.

In step (2), the choice which unsolved subsumption to consider next is don't care nondeterministic. However, choosing which rule to apply to the chosen subsumption is don't know nondeterministic. Additionally, the application of nondeterministic rules requires don't know nondeterministic guessing.

The *eager rules* are mainly there for optimization purposes, i.e., to avoid nondeterministic choices if a deterministic decision can be made. For example, a ground subsumption, as considered in the *Eager Ground Solving* rule, either follows from the ontology, in which case any substitution solves it, or it does not, in which case it does not have a solution. This condition can be checked in polynomial time using the polynomial time subsumption algorithm for $\mathcal{ELH}_{R^+}$ [7]. In the case considered in the *Eager Solving* rule, the substitution induced by the current assignment obviously already solves the subsumption. The *Eager Extension* rule solves a subsumption that contains only a variable X and some elements of S_X on the left-hand side. The rule is motivated by the following observation: for any assignment S' extending the current assignment, the induced

[5] Note that *Decomposition 1* always applies to the new subsumptions. Whether *Decomposition 2* is also applicable depends on the existence of an appropriate transitive role t.

Mutation 1:

Condition: This rule applies to $\mathfrak{s} = C_1 \sqcap \cdots \sqcap C_n \sqsubseteq^? D$ if $n > 1$ and there are atoms $A_1, \ldots, A_k, B$ of $\mathcal{O}$ such that $A_1 \sqcap \cdots \sqcap A_k \sqsubseteq_\mathcal{O} B$ holds.
Action: Its application chooses such atoms, marks $\mathfrak{s}$ as *solved*, and generates the following subsumptions:

- it chooses for each $\eta \in \{1, \ldots, k\}$ an $i \in \{1, \ldots, n\}$ and adds the subsumption $C_i \sqsubseteq^? A_\eta$ to Γ,
- it adds the subsumption $B \sqsubseteq^? D$ to Γ.

Mutation 2:

Condition: This rule applies to $\mathfrak{s} = \exists r.X \sqsubseteq^? D$ if X is a variable, D is ground, and there are atoms $\exists r_1.A_1, \ldots, \exists r_k.A_k$ of $\mathcal{O}$ such that $r \trianglelefteq_\mathcal{O} r_1$, ..., $r \trianglelefteq_\mathcal{O} r_k$, and $\exists r_1.A_1 \sqcap \cdots \sqcap \exists r_k.A_k \sqsubseteq_\mathcal{O} D$ hold.
Action: Its application chooses such atoms, adds the subsumptions $\exists r.X \sqsubseteq^? \exists r_1.A_1$, ..., $\exists r.X \sqsubseteq^? \exists r_k.A_k$ to Γ, and marks $\mathfrak{s}$ as *solved*.

Mutation 3:

Condition: This rule applies to $\mathfrak{s} = \exists r.X \sqsubseteq^? \exists s.Y$ if X and Y are variables, and there are atoms $\exists r_1.A_1, \ldots, \exists r_k.A_k, \exists u.B$ of $\mathcal{O}$ such that $r \trianglelefteq_\mathcal{O} r_1$, ..., $r \trianglelefteq_\mathcal{O} r_k$, $u \trianglelefteq_\mathcal{O} s$, and $\exists r_1.A_1 \sqcap \cdots \sqcap \exists r_k.A_k \sqsubseteq_\mathcal{O} \exists u.B$ hold.
Action: Its application chooses such atoms, adds the subsumptions $\exists r.X \sqsubseteq^? \exists r_1.A_1$, ..., $\exists r.X \sqsubseteq^? \exists r_k.A_k$, $\exists u.B \sqsubseteq^? \exists s.Y$ to Γ, and marks $\mathfrak{s}$ as *solved*.

Mutation 4:

Condition: This rule applies to $\mathfrak{s} = C \sqsubseteq^? \exists s.Y$ if C is a ground atom or $\top$, Y is a variable, and there is an atom $\exists u.B$ of $\mathcal{O}$ such that either

- $C \sqsubseteq_\mathcal{O} \exists u.B$ and $u \trianglelefteq_\mathcal{O} s$, or
- $C \sqsubseteq_\mathcal{O} \exists t.B$ for a transitive role t with $u \trianglelefteq_\mathcal{O} t \trianglelefteq_\mathcal{O} s$.

Action: Its application chooses such an atom, adds the subsumption $B \sqsubseteq^? Y$ to Γ, and marks $\mathfrak{s}$ as *solved*.

Fig. 3. The nondeterministic *Mutation* rules of the unification algorithm

substitution σ' satisfies $\sigma'(X) \equiv \sigma'(C_1) \sqcap \ldots \sqcap \sigma'(C_n)$. Thus, if S'_X contains D, then $\sigma'(X) \sqsubseteq_\mathcal{O} \sigma'(D)$, and σ' solves the subsumption. Conversely, if σ' solves the subsumption, then $\sigma'(X) \sqsubseteq_\mathcal{O} \sigma'(D)$, and thus adding D to S'_X yields an equivalent induced substitution.

The *nondeterministic rules* only come into play if no eager rules can be applied. In order to solve an unsolved subsumption $\mathfrak{s} = C_1 \sqcap \cdots \sqcap C_n \sqsubseteq^? D$, we consider the two conditions of Lemma 1. Regarding the first condition, which is addressed by the rules *Decomposition 1 and 2* and *Extension*, assume that γ is induced by an acyclic assignment S. To satisfy the first condition of the lemma with γ, the atom $\gamma(D)$ must structurally subsume a top-level atom in $\gamma(C_1) \sqcap \cdots \sqcap \gamma(C_n)$. This atom can either be of the form $\gamma(C_i)$ for an atom C_i, or it can be of the form $\gamma(C)$ for an atom $C \in S_{C_i}$ and a variable C_i. In the second case, the atom C can either already be in S_{C_i} or it can be put into S_{C_i} by an application of the Extension rule. The two versions of *Decomposition* correspond to the cases (2) and (3) in the definition of structural subsumption.

The *Mutation rules* cover the second condition in Lemma 1. For example, let us analyze how *Mutation 1* ensures that all the requirements of this condition are satisfied. The rule guesses atoms $A_1, \ldots, A_k, B$ such that $A_1 \sqcap \cdots \sqcap A_k \sqsubseteq_{\mathcal{O}} B$ holds. This can be checked using the polynomial-time subsumption algorithm for $\mathcal{ELH}_{R^+}$. Whenever the second condition of Lemma 1 requires a structural subsumption $\gamma(E) \sqsubseteq^{\mathsf{s}}_{\mathcal{O}} \gamma(F)$ to hold for a (hypothetical) unifier γ of Γ, the rule creates the new subsumption $E \sqsubseteq^{?} F$, which has to be solved later on. This way, the rule ensures that the substitution built by the algorithm actually satisfies the conditions of the lemma. The *other mutation rules* follow the same idea, but they consider cases where only a single atom occurs on the left-hand side of the subsumption to be solved. The reason for considering these cases separately is that in the proof of soundness we need the newly introduced subsumptions to be "smaller" than the subsumption that triggered their introduction. For *Mutation 1* this is the case due to the smaller left-hand side (only one atom), whereas for the other mutation rules this is not so clear. Actually, for *Mutation 2 and 3*, the new subsumptions turn out to be smaller only after *Decomposition* is applied to them. *Mutation 4* implicitly applies a form of decomposition.

Due to the space restrictions, we cannot give more details on how to prove that the algorithm is correct. Complete proofs of soundness, completeness and termination can be found in [3].

Theorem 5. *Algorithm 4 is an* NP*-decision procedure for testing solvability of* $\mathcal{ELH}_{R^+}$*-unification problems w.r.t. cycle-restricted ontologies.*

5 Conclusions

Above, we have presented a goal-oriented NP-algorithm for unification in $\mathcal{ELH}_{R^+}$ w.r.t. cycle-restricted ontologies. In [5], we have developed a reduction of this problem to SAT, which is based on a characterization of subsumption different from the one in Lemma 1. Though clearly better than the brute-force algorithm introduced in [2], both algorithms suffer from a high degree of nondeterminism due to having to guess true subsumptions between concepts built from atoms of the background cycle-restricted ontology. We must find optimizations to tackle this problem before an implementation becomes feasible.

On the theoretical side, the main topic for future research is to consider unification w.r.t. unrestricted $\mathcal{ELH}_{R^+}$-ontologies. In order to generalize the brute-force algorithm in this direction, we need to find a more general notion of locality. Starting with the goal-oriented algorithm, one idea could be not to fail when a cyclic assignment is generated, but rather to add rules that can break such cycles, similar to what is done in procedures for general E-unification [16].

Another idea could be to use just the rules of our goal-oriented algorithm, and not fail when a cyclic assignment S is generated. Our conjecture is that then the background ontology $\mathcal{O}$ together with the cyclic TBox $\mathcal{T}_S := \{X \equiv \bigsqcap_{C \in S_X} C \mid X \in N_v\}$ induced by S satisfies $C \sqsubseteq_{\mathcal{O} \cup \mathcal{T}_S} D$ for all subsumptions $C \sqsubseteq^{?} D$ in Γ_0 if an appropriate hybrid semantics [13] for the combined ontology $\mathcal{O} \cup \mathcal{T}_S$ is used.

All the results on unification in Description Logics mentioned in this paper are restricted to relatively inexpressive logics that do not support all Boolean operators. If we close $\mathcal{EL}$ under negation, then we obtain the DL $\mathcal{ALC}$, which corresponds to the modal logic K [17]. Whether unification in K is decidable is a long-standing open problem. It is only known that relatively minor extensions of K have an undecidable unification problem [18].

References

1. Baader, F., Borgwardt, S., Mendez, J., Morawska, B.: UEL: Unification solver for $\mathcal{EL}$. In: Proc. DL 2012. CEUR Workshop Proceedings, vol. 846 (2012)
2. Baader, F., Borgwardt, S., Morawska, B.: Extending unification in $\mathcal{EL}$ towards general TBoxes. In: Proc. KR 2012, pp. 568–572. AAAI Press (2012) (short paper)
3. Baader, F., Borgwardt, S., Morawska, B.: A goal-oriented algorithm for unification in $\mathcal{EL}\mathcal{H}_{R^+}$ w.r.t. cycle-restricted ontologies. LTCS-Report 12-05, TU Dresden, Germany (2012), `http://lat.inf.tu-dresden.de/research/reports.html`
4. Baader, F., Borgwardt, S., Morawska, B.: A goal-oriented algorithm for unification in $\mathcal{EL}$ w.r.t. cycle-restricted TBoxes. In: Proc. DL 2012. CEUR Workshop Proceedings, vol. 846 (2012)
5. Baader, F., Borgwardt, S., Morawska, B.: SAT Encoding of Unification in $\mathcal{EL}\mathcal{H}_{R^+}$ w.r.t. Cycle-Restricted Ontologies. In: Gramlich, B., Miller, D., Sattler, U. (eds.) IJCAR 2012. LNCS, vol. 7364, pp. 30–44. Springer, Heidelberg (2012)
6. Baader, F., Borgwardt, S., Morawska, B.: SAT encoding of unification in $\mathcal{EL}\mathcal{H}_{R^+}$ w.r.t. cycle-restricted ontologies. LTCS-Report 12-02, TU Dresden, Germany (2012), `http://lat.inf.tu-dresden.de/research/reports.html`
7. Baader, F., Brandt, S., Lutz, C.: Pushing the $\mathcal{EL}$ envelope. In: Proc. IJCAI 2005, pp. 364–369. Morgan Kaufmann (2005)
8. Baader, F., Morawska, B.: Unification in the Description Logic $\mathcal{EL}$. In: Treinen, R. (ed.) RTA 2009. LNCS, vol. 5595, pp. 350–364. Springer, Heidelberg (2009)
9. Baader, F., Morawska, B.: SAT Encoding of Unification in $\mathcal{EL}$. In: Fermüller, C.G., Voronkov, A. (eds.) LPAR-17. LNCS, vol. 6397, pp. 97–111. Springer, Heidelberg (2010)
10. Baader, F., Morawska, B.: Unification in the description logic $\mathcal{EL}$. Log. Meth. Comput. Sci. 6(3) (2010)
11. Baader, F., Narendran, P.: Unification of concept terms in description logics. J. Symb. Comput. 31(3), 277–305 (2001)
12. Brandt, S.: Polynomial time reasoning in a description logic with existential restrictions, GCI axioms, and—what else? In: Proc. ECAI 2004, pp. 298–302 (2004)
13. Brandt, S., Model, J.: Subsumption in $\mathcal{EL}$ w.r.t. hybrid TBoxes. In: Furbach, U. (ed.) KI 2005. LNCS (LNAI), vol. 3698, pp. 34–48. Springer, Heidelberg (2005)
14. Gomes, C.P., Kautz, H., Sabharwal, A., Selman, B.: Satisfiability solvers. In: Handbook of Knowledge Representation, pp. 89–134. Elsevier (2008)
15. Hofmann, M.: Proof-theoretic approach to description-logic. In: Proc. LICS 2005. pp. 229–237. IEEE Press (2005)
16. Morawska, B.: General E-unification with eager variable elimination and a nice cycle rule. J. Autom. Reasoning 39(1), 77–106 (2007)
17. Schild, K.: A correspondence theory for terminological logics: Preliminary report. In: Proc. IJCAI 1991, pp. 466–471 (1991)
18. Wolter, F., Zakharyaschev, M.: Undecidability of the unification and admissibility problems for modal and description logics. ACM Trans. Comput. Log. 9(4) (2008)

SAT Encoding of Unification in $\mathcal{EL}\mathcal{H}_{R^+}$ w.r.t. Cycle-Restricted Ontologies*

Franz Baader, Stefan Borgwardt, and Barbara Morawska

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

Abstract. Unification in Description Logics has been proposed as an inference service that can, for example, be used to detect redundancies in ontologies. For the Description Logic $\mathcal{EL}$, which is used to define several large biomedical ontologies, unification is NP-complete. An NP unification algorithm for $\mathcal{EL}$ based on a translation into propositional satisfiability (SAT) has recently been presented. In this paper, we extend this SAT encoding in two directions: on the one hand, we add general concept inclusion axioms, and on the other hand, we add role hierarchies ($\mathcal{H}$) and transitive roles (R^+). For the translation to be complete, however, the ontology needs to satisfy a certain cycle restriction. The SAT translation depends on a new rewriting-based characterization of subsumption w.r.t. $\mathcal{EL}\mathcal{H}_{R^+}$-ontologies.

1 Introduction

The Description Logic (DL) $\mathcal{EL}$, which offers the constructors conjunction ($\sqcap$), existential restriction ($\exists r.C$), and the top concept ($\top$), has recently drawn considerable attention since, on the one hand, important inference problems such as the subsumption problem are polynomial in $\mathcal{EL}$, even in the presence of general concept inclusion axioms (GCIs) [11,4]. On the other hand, though quite inexpressive, $\mathcal{EL}$ can be used to define biomedical ontologies, such as the large medical ontology SNOMED CT.[1]

Unification in DLs has been proposed in [8] as a novel inference service that can, for instance, be used to detect redundancies in ontologies. For example, assume that one developer of a medical ontology defines the concept of a *patient with severe injury of the frontal lobe* as

$$\exists \mathsf{finding}.(\mathsf{Frontal\_lobe\_injury} \sqcap \exists \mathsf{severity}.\mathsf{Severe}), \tag{1}$$

whereas another one represents it as

$$\exists \mathsf{finding}.(\mathsf{Severe\_injury} \sqcap \exists \mathsf{finding\_site}.\exists \mathsf{part\_of}.\mathsf{Frontal\_lobe}). \tag{2}$$

These two concept descriptions are not equivalent, but they are nevertheless meant to represent the same concept. They can obviously be made equivalent by

* Supported by DFG under grant BA 1122/14-1.

[1] See http://www.ihtsdo.org/snomed-ct/

B. Gramlich, D. Miller, and U. Sattler (Eds.): IJCAR 2012, LNAI 7364, pp. 30–44, 2012.

treating the concept names Frontal_lobe_injury and Severe_injury as variables, and substituting the first one by Injury ⊓ ∃finding_site.∃part_of.Frontal_lobe and the second one by Injury ⊓ ∃severity.Severe. In this case, we say that the descriptions are unifiable, and call the substitution that makes them equivalent a *unifier*.

To motivate our interest in unification w.r.t. GCIs, role hierarchies, and transitive roles, assume that the developers use the descriptions (3) and (4) instead of (1) and (2):

$$\exists \mathsf{finding}.\exists \mathsf{finding\_site}.\exists \mathsf{part\_of}.\mathsf{Brain} \sqcap \exists \mathsf{finding}.(\mathsf{Frontal\_lobe\_injury} \sqcap \exists \mathsf{severity}.\mathsf{Severe}) \tag{3}$$

$$\exists \mathsf{status}.\mathsf{Emergency} \sqcap \exists \mathsf{finding}.(\mathsf{Severe\_injury} \sqcap \exists \mathsf{finding\_site}.\exists \mathsf{part\_of}.\mathsf{Frontal\_lobe}) \tag{4}$$

The descriptions (3) and (4) are not unifiable without additional background knowledge, but they are unifiable, with the same unifier as above, if the GCIs

$$\exists \mathsf{finding}.\exists \mathsf{severity}.\mathsf{Severe} \sqsubseteq \exists \mathsf{status}.\mathsf{Emergency},$$
$$\mathsf{Frontal\_lobe} \sqsubseteq \exists \mathsf{proper\_part\_of}.\mathsf{Brain}$$

are present in a background ontology and this ontology additionally states that part_of is transitive and proper_part_of is a subrole of part_of.

Most of the previous results on unification in DLs did not consider such additional background knowledge. In [8] it was shown that, for the DL $\mathcal{FL}_0$, which differs from $\mathcal{EL}$ by offering value restrictions ($\forall r.C$) in place of existential restrictions, deciding unifiability is an ExpTime-complete problem. In [5], we were able to show that unification in $\mathcal{EL}$ is of considerably lower complexity: the decision problem is NP-complete. The original unification algorithm for $\mathcal{EL}$ introduced in [5] was a brutal "guess and then test" NP-algorithm, but we have since then also developed more practical algorithms. On the one hand, in [7] we describe a goal-oriented unification algorithm for $\mathcal{EL}$, in which nondeterministic decisions are only made if they are triggered by "unsolved parts" of the unification problem. On the other hand, in [6], we present an algorithm that is based on a reduction to satisfiability in propositional logic (SAT). In [7] it was also shown that the approaches for unification of $\mathcal{EL}$-concept descriptions (without any background ontology) can easily be extended to the case of an acyclic TBox as background ontology without really changing the algorithms or increasing their complexity. Basically, by viewing defined concepts as variables, an acyclic TBox can be turned into a unification problem that has as its unique unifier the substitution that replaces the defined concepts by unfolded versions of their definitions.

For GCIs, this simple trick is not possible, and thus handling them requires the development of new algorithms. In [1,2] we describe two such new algorithms: one that extends the brute-force "guess and then test" NP-algorithm from [5] and a more practical one that extends the goal-oriented algorithm from [7]. Both algorithms are based on a new characterization of subsumption w.r.t. GCIs in $\mathcal{EL}$,

which we prove using a Gentzen-style proof calculus for subsumption. Unfortunately, these algorithms are complete only for cycle-restricted TBoxes, i.e., finite sets of GCIs that satisfy a certain restriction on cycles, which, however, does not prevent all cycles. For example, the cyclic GCI $\exists \mathsf{child}.\mathsf{Human} \sqsubseteq \mathsf{Human}$ satisfies this restriction, whereas the cyclic GCI $\mathsf{Human} \sqsubseteq \exists \mathsf{parent}.\mathsf{Human}$ does not.

In the present paper, we still cannot get rid of cycle-restrictedness of the ontology, but extend the results of [2] in two other directions: (i) we add transitive roles (indicated by the subscript R^+ in the name of the DL) and role hierarchies (indicated by adding the letter $\mathcal{H}$ to the name of the DL) to the language, which are important for medical ontologies [17,15]; (ii) we provide an algorithm that is based on a translation into SAT, and thus allows us to employ highly optimized state-of-the-art SAT solvers [10] for implementing the unification algorithm. In order to obtain the SAT translation, using the characterization of subsumption from [2] is not sufficient, however. We had to develop a new rewriting-based characterization of subsumption.

In the next section, we introduce the DLs considered in this paper and the important inference problem subsumption. In Section 3 we define unification for these DLs and recall some of the existing results for unification in $\mathcal{EL}$. In particular, we introduce in this section the notion of cycle-restrictedness, which is required for the results on unification w.r.t. GCIs to hold. In Section 4 we then derive rewriting-based characterizations of subsumption. Section 5 contains the main result of this paper, which is a reduction of unification in $\mathcal{ELH}_{R^+}$ w.r.t. cycle-restricted ontologies to propositional satisfiability. The proof of correctness of this reduction strongly depends on the characterization of subsumption shown in the previous section.

2 The Description Logics $\mathcal{EL}$, $\mathcal{EL}^+$, and $\mathcal{ELH}_{R^+}$

The expressiveness of a DL is determined both by the formalism for describing concepts (the concept description language) and the terminological formalism, which can be used to state additional constraints on the interpretation of concepts and roles in a so-called ontology.

Syntax and Semantics

The concept description language considered in this paper is called $\mathcal{EL}$. Starting with a finite set N_C of *concept names* and a finite set N_R of *role names*, *$\mathcal{EL}$-concept descriptions* are built from concept names using the constructors *conjunction* ($C \sqcap D$), *existential restriction* ($\exists r.C$ for every $r \in N_R$), and *top* ($\top$).

Since in this paper we only consider $\mathcal{EL}$-concept descriptions, we will sometimes dispense with the prefix $\mathcal{EL}$.

On the semantic side, concept descriptions are interpreted as sets. To be more precise, an *interpretation* $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ consists of a non-empty domain $\Delta^{\mathcal{I}}$ and an interpretation function $\cdot^{\mathcal{I}}$ that maps concept names to subsets of $\Delta^{\mathcal{I}}$ and role names to binary relations over $\Delta^{\mathcal{I}}$. This function is extended to concept descriptions as shown in the semantics column of Table 1.

Table 1. Syntax and semantics of $\mathcal{EL}$

Name	Syntax	Semantics
concept name	A	$A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$
role name	r	$r^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$
top	$\top$	$\top^{\mathcal{I}} = \Delta^{\mathcal{I}}$
conjunction	$C \sqcap D$	$(C \sqcap D)^{\mathcal{I}} = C^{\mathcal{I}} \cap D^{\mathcal{I}}$
existential restriction	$\exists r.C$	$(\exists r.C)^{\mathcal{I}} = \{x \mid \exists y : (x,y) \in r^{\mathcal{I}} \wedge y \in C^{\mathcal{I}}\}$
general concept inclusion	$C \sqsubseteq D$	$C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$
role inclusion	$r_1 \circ \cdots \circ r_n \sqsubseteq s$	$r_1^{\mathcal{I}} \circ \cdots \circ r_n^{\mathcal{I}} \subseteq s^{\mathcal{I}}$

Ontologies

A *general concept inclusion (GCI)* is of the form $C \sqsubseteq D$ for concept descriptions C, D, and a *role inclusion* is of the form $r_1 \circ \cdots \circ r_n \sqsubseteq s$ for role names $r_1, \ldots, r_n, s$. Both are called *axioms*. Role inclusions of the form $r \circ r \sqsubseteq r$ are called *transitivity axioms* and of the form $r \sqsubseteq s$ *role hierarchy axioms*. An interpretation $\mathcal{I}$ *satisfies* such an axiom if the corresponding condition in the semantics column of Table 1 holds, where $\circ$ in this column stands for composition of binary relations.

An $\mathcal{EL}^+$*-ontology* is a finite set of axioms. It is an $\mathcal{ELH}_{R^+}$*-ontology* if all its role inclusions are transitivity or role hierarchy axioms, and an $\mathcal{EL}$*-ontology* if it contains only GCIs. An interpretation is a *model* of an ontology if it satisfies all its axioms.

Subsumption, Equivalence, and Role Hierarchy

A concept description C is *subsumed* by a concept description D w.r.t. an ontology $\mathcal{O}$ (written $C \sqsubseteq_{\mathcal{O}} D$) if every model of $\mathcal{O}$ satisfies the GCI $C \sqsubseteq D$. We say that C is *equivalent* to D w.r.t. $\mathcal{O}$ ($C \equiv_{\mathcal{O}} D$) if $C \sqsubseteq_{\mathcal{O}} D$ and $D \sqsubseteq_{\mathcal{O}} C$. If $\mathcal{O}$ is empty, we also write $C \sqsubseteq D$ and $C \equiv D$ instead of $C \sqsubseteq_{\mathcal{O}} D$ and $C \equiv_{\mathcal{O}} D$, respectively. As shown in [11,4], subsumption w.r.t. $\mathcal{EL}^+$-ontologies (and thus also w.r.t. $\mathcal{ELH}_{R^+}$- and $\mathcal{EL}$-ontologies) is decidable in polynomial time.

Since conjunction is interpreted as intersection, the concept descriptions $(C \sqcap D) \sqcap E$ and $C \sqcap (D \sqcap E)$ are always equivalent. Thus, we dispense with parentheses and write nested conjunctions in flat form $C_1 \sqcap \cdots \sqcap C_n$. Nested existential restrictions $\exists r_1.\exists r_2. \ldots \exists r_n.C$ will sometimes also be written as $\exists r_1 r_2 \ldots r_n.C$, where $r_1 r_2 \ldots r_n$ is viewed as a word over the alphabet of role names, i.e., an element of N_R^*.

The *role hierarchy* induced by $\mathcal{O}$ is a binary relation $\trianglelefteq_{\mathcal{O}}$ on N_R, which is defined as the reflexive-transitive closure of the relation $\{(r,s) \mid r \sqsubseteq s \in \mathcal{O}\}$. Using elementary reachability algorithms, the role hierarchy can be computed in polynomial time in the size of $\mathcal{O}$. It is easy to see that $r \trianglelefteq_{\mathcal{O}} s$ implies that $r^{\mathcal{I}} \subseteq s^{\mathcal{I}}$ for all models $\mathcal{I}$ of $\mathcal{O}$.

3 Unification

In order to define unification, we first introduce the notion of a substitution operating on concept descriptions. For this purpose, we partition the set N_C of concepts names into a set N_v of concept variables (which may be replaced by substitutions) and a set N_c of concept constants (which must not be replaced by substitutions). A *substitution* σ maps every variable to an $\mathcal{EL}$-concept description. It can be extended from variables to $\mathcal{EL}$-concept descriptions as follows:

- $\sigma(A) := A$ for all $A \in N_c \cup \{\top\}$,
- $\sigma(C \sqcap D) := \sigma(C) \sqcap \sigma(D)$ and $\sigma(\exists r.C) := \exists r.\sigma(C)$.

A concept description C is *ground* if it does not contain variables, and a substitution is *ground* if all concept descriptions in its range are ground. Obviously, a ground concept description is not modified by applying a substitution, and if we apply a ground substitution to any concept description, then we obtain a ground description. An ontology is *ground* if it does not contain variables.

Definition 1. *Let $\mathcal{O}$ be a ground ontology. A* unification problem *w.r.t. $\mathcal{O}$ is a finite set $\Gamma = \{C_1 \sqsubseteq^? D_1, \ldots, C_n \sqsubseteq^? D_n\}$ of subsumptions between $\mathcal{EL}$-concept descriptions. A substitution σ is a* unifier *of Γ w.r.t. $\mathcal{O}$ if σ* solves *all the subsumptions in Γ w.r.t. $\mathcal{O}$, i.e., if $\sigma(C_1) \sqsubseteq_{\mathcal{O}} \sigma(D_1), \ldots, \sigma(C_n) \sqsubseteq_{\mathcal{O}} \sigma(D_n)$. We say that Γ is* unifiable *w.r.t. $\mathcal{O}$ if it has a unifier w.r.t. $\mathcal{O}$.*

We call Γ w.r.t. $\mathcal{O}$ an $\mathcal{EL}$-, $\mathcal{EL}^+$-, or $\mathcal{ELH}_{R^+}$-unification problem depending on whether and what kind of role inclusions are contained in $\mathcal{O}$.

Three remarks regarding the definition of unification problems are in order. First, note that some of the previous papers on unification in DLs used equivalences $C \equiv^? D$ instead of subsumptions $C \sqsubseteq^? D$. This difference is, however, irrelevant since $C \equiv^? D$ can be seen as a shorthand for the two subsumptions $C \sqsubseteq^? D$ and $D \sqsubseteq^? C$, and $C \sqsubseteq^? D$ has the same unifiers as $C \sqcap D \equiv^? C$.

Second, note that—as in [2]—we have restricted the background ontology $\mathcal{O}$ to be ground. This is not without loss of generality. In fact, if $\mathcal{O}$ contained variables, then we would need to apply the substitution also to its axioms, and instead of requiring $\sigma(C_i) \sqsubseteq_{\mathcal{O}} \sigma(D_i)$ we would thus need to require $\sigma(C_i) \sqsubseteq_{\sigma(\mathcal{O})} \sigma(D_i)$, which would change the nature of the problem considerably. The treatment of unification w.r.t. acyclic TBoxes in [7] actually considers a more general setting, where some of the primitive concepts occurring in the TBox may be variables. The restriction to ground general TBoxes is, however, appropriate for the application scenario sketched in the introduction. In this scenario, there is a fixed background ontology, which is extended with definitions of new concepts by several knowledge engineers. Unification w.r.t. the background ontology is used to check whether some of these new definitions actually are redundant, i.e., define the same intuitive concept. Here, some of the primitive concepts newly introduced by one knowledge engineer may be further defined by another one, but we assume that the knowledge engineers use the vocabulary from the background ontology unchanged, i.e., they define *new* concepts rather than adding definitions

for concepts that already occur in the background ontology. An instance of this scenario can, e.g., be found in [12], where different extensions of SNOMED CT are checked for overlaps, albeit not by using unification, but by simply testing for equivalence.

Third, though arbitrary substitutions σ are used in the definition of a unifier, it is actually sufficient to consider ground substitutions such that all concept descriptions $\sigma(X)$ in the range of σ contain only concept and role names occurring in Γ or $\mathcal{O}$. It is an easy consequence of well-known results from unification theory [9] that Γ has a unifier w.r.t. $\mathcal{O}$ iff it has such a ground unifier.

Relationship to Equational Unification

Unification was originally not introduced for Description Logics, but for equational theories [9]. In [7] it was shown that unification in $\mathcal{EL}$ (w.r.t. the empty ontology) is the same as unification in the equational theory $SLmO$ of semilattices with monotone operators [16]. As argued in [2], unification in $\mathcal{EL}$ w.r.t. a ground $\mathcal{EL}$-ontology corresponds to unification in $SLmO$ extended with a finite set of ground identities. In contrast to GCIs, role inclusions add non-ground identities to $SLmO$ (see [16] and [3] for details).

This unification-theoretic point of view sheds some light on our decision to restrict unification w.r.t. general TBoxes to the case of general TBoxes that are ground. In fact, if we lifted this restriction, then we would end up with a generalization of rigid E-unification [14,13], in which the theory $SLmO$ extended with the identities expressing role inclusions is used as a background theory. To the best of our knowledge, such variants of rigid E-unification have not been considered in the literature, and are probably quite hard to solve.

Flat Ontologies and Unification Problems

To simplify the technical development, it is convenient to normalize the TBox and the unification problem appropriately. To introduce this normal form, we need the notion of an atom.

An *atom* is a concept name or an existential restriction. Obviously, every $\mathcal{EL}$-concept description C is a finite conjunction of atoms, where $\top$ is considered to be the empty conjunction. We call the atoms in this conjunction the *top-level atoms* of C. An atom is called *flat* if it is a concept name or an existential restriction of the form $\exists r.A$ for a concept name A.

The GCI $C \sqsubseteq D$ or subsumption $C \sqsubseteq^? D$ is called *flat* if C is a conjunction of $n \geq 0$ flat atoms and D is a flat atom. The ontology $\mathcal{O}$ (unification problem Γ) is called *flat* if all the GCIs in $\mathcal{O}$ (subsumptions in Γ) are flat. Given a ground ontology $\mathcal{O}$ and a unification problem Γ, we can compute in polynomial time (see [3]) a flat ontology $\mathcal{O}'$ and a flat unification problem Γ' such that

- Γ has a unifier w.r.t. $\mathcal{O}$ iff Γ' has a unifier w.r.t. $\mathcal{O}'$;
- the type of the unification problem ($\mathcal{EL}$, $\mathcal{EL}^+$, or $\mathcal{ELH}_{R^+}$) is preserved.

For this reason, we will assume in the following that all ontologies and unification problems are flat.

Cycle-Restricted Ontologies

The decidability and complexity results for unification w.r.t. $\mathcal{EL}$-ontologies in [2], and also the corresponding ones in the present paper, only hold if the ontologies satisfy a restriction that prohibits certain cyclic subsumptions.

Definition 2. *The $\mathcal{EL}^+$-ontology $\mathcal{O}$ is called* cycle-restricted *iff there is no nonempty word $w \in N_R^+$ and $\mathcal{EL}$-concept description C such that $C \sqsubseteq_{\mathcal{O}} \exists w.C$.*

Note that cycle-restrictedness is not a syntactic condition on the form of the axioms in $\mathcal{O}$, but a semantic one on what follows from $\mathcal{O}$. Nevertheless, for $\mathcal{ELH}_{R^+}$-ontologies, this condition can be decided in polynomial time [3]. Basically, one first shows that the $\mathcal{ELH}_{R^+}$-ontology $\mathcal{O}$ is cycle-restricted iff $A \not\sqsubseteq_{\mathcal{O}} \exists w.A$ holds for all nonempty words $w \in N_R^+$ and all $A \in N_C \cup \{\top\}$. Then, one shows that $A \sqsubseteq_{\mathcal{O}} \exists w.A$ for some $w \in N_R^+$ and $A \in N_C \cup \{\top\}$ implies that there are $n \geq 1$ role names $r_1, \ldots, r_n$ and $A_1, \ldots, A_n \in N_C \cup \{\top\}$ such that

$$(*) \quad A \sqsubseteq_{\mathcal{O}} \exists r_1.A_1, A_1 \sqsubseteq_{\mathcal{O}} \exists r_2.A_2, \ldots, A_{n-1} \sqsubseteq_{\mathcal{O}} \exists r_n.A_n \text{ and } A_n = A.$$

Using the polynomial-time subsumption algorithm for $\mathcal{ELH}_{R^+}$, we can build a graph whose nodes are the elements of $N_C \cup \{\top\}$ and where there is an edge from A to B with label r iff $A \sqsubseteq_{\mathcal{O}} \exists r.B$. Then we can use standard reachability algorithms to check whether this graph contains a cycle of the form $(*)$. The restriction to $\mathcal{ELH}_{R^+}$ stems from the fact that the proof of correctness of this algorithm is based on Lemma 7 below, which we cannot show for $\mathcal{EL}^+$.

The main reason why we need cycle-restrictedness of $\mathcal{O}$ is that it ensures that a substitution always induces a strict partial order on the variables.[2] To be more precise, assume that γ is a substitution. For $X, Y \in N_v$ we define

$$X >_\gamma Y \text{ iff } \gamma(X) \sqsubseteq_{\mathcal{O}} \exists w.\gamma(Y) \text{ for some } w \in N_R^+. \tag{5}$$

Transitivity of $>_\gamma$ is an easy consequence of transitivity of subsumption, and cycle-restrictedness of $\mathcal{O}$ yields irreflexivity of $>_\gamma$.

Lemma 3. *If $\mathcal{O}$ is a cycle-restricted $\mathcal{EL}^+$-ontology, then $>_\gamma$ is a strict partial order on N_v.*

4 Subsumption w.r.t. $\mathcal{EL}^+$- and $\mathcal{ELH}_{R^+}$-Ontologies

Subsumption w.r.t. $\mathcal{EL}^+$-ontologies can be decided in polynomial time [4]. For the purpose of deciding unification, however, we do not simply want a decision procedure for subsumption, but are more interested in a characterization of subsumption that helps us to find unifiers. The characterization of subsumption derived here is based on a rewrite relation that uses axioms as rewrite rules from right to left.

[2] Why we need this order will become clear in Section 5.

Proving Subsumption by Rewriting

Throughout this subsection, we assume that $\mathcal{O}$ is a flat $\mathcal{EL}^+$-ontology. Intuitively, an axiom of the form $A_1 \sqcap \ldots \sqcap A_n \sqsubseteq B \in \mathcal{O}$ is used to replace B by $A_1 \sqcap \ldots \sqcap A_n$ and an axiom of the form $r_1 \circ \ldots \circ r_n \sqsubseteq s \in \mathcal{O}$ to replace $\exists s.C$ by $\exists r_1 \ldots r_n.C$. In order to deal with associativity, commutativity, and idempotency of conjunction, it is convenient to represent concept descriptions as sets of atoms rather than as conjunctions of atoms.

Given an $\mathcal{EL}$-concept description C, the *description set* $\mathsf{s}(C)$ associated with C is defined by induction:

- $\mathsf{s}(A) := \{A\}$ for $A \in N_C$ and $\mathsf{s}(\top) := \emptyset$;
- $\mathsf{s}(C \sqcap D) := \mathsf{s}(C) \cup \mathsf{s}(D)$ and $\mathsf{s}(\exists r.C) := \{\exists r.\mathsf{s}(C)\}$.

For example, if $C = A \sqcap \exists r.(A \sqcap \exists r.\top)$, then $\mathsf{s}(C) = \{A, \exists r.\{A, \exists r.\emptyset\}\}$. We call *set positions* the positions in $\mathsf{s}(C)$ at which there is a set. In our example, we have three set positions, corresponding to the sets $\{A, \exists r.\{A, \exists r.\emptyset\}\}$, $\{A, \exists r.\emptyset\}$, and $\emptyset$. The set position that corresponds to the whole set $\mathsf{s}(C)$ is called the *root position*.

Our *rewrite rules* are of the form $N \leftarrow M$, where N, M are description sets. Such a rule *applies* at a set position p in $\mathsf{s}(C)$ if the corresponding set $\mathsf{s}(C)|_p$ contains M, and its *application* replaces $\mathsf{s}(C)|_p$ by $(\mathsf{s}(C)|_p \setminus M) \cup N$ (see [3] for a more formal definition of set positions and of the application of rewrite rules).

Given a flat $\mathcal{EL}^+$-ontology $\mathcal{O}$, the corresponding rewrite system $R(\mathcal{O})$ consists of the following rules:

- *Concept inclusion* ($\mathbf{R}_c$): For every $C \sqsubseteq D \in \mathcal{O}$, $R(\mathcal{O})$ contains the rule

$$\mathsf{s}(C) \leftarrow \mathsf{s}(D).$$

- *Role inclusion* ($\mathbf{R}_r$): For every $r_1 \circ \cdots \circ r_n \sqsubseteq s \in \mathcal{O}$ and every $\mathcal{EL}$-concept description C, $R(\mathcal{O})$ contains the rule

$$\mathsf{s}(\exists r_1 \ldots r_n.C) \leftarrow \mathsf{s}(\exists s.C).$$

- *Monotonicity* ($\mathbf{R}_m$): For every atom D, $R(\mathcal{O})$ contains the rule

$$\mathsf{s}(D) \leftarrow \emptyset.$$

Definition 4. *Let N, M be description sets. We write $N \leftarrow_{\mathcal{O}} M$ if N can be obtained from M by the application of a rule in $R(\mathcal{O})$. The relation $\overset{*}{\leftarrow}_{\mathcal{O}}$ is defined to be the reflexive, transitive closure of $\leftarrow_{\mathcal{O}}$, i.e., $N \overset{*}{\leftarrow}_{\mathcal{O}} M$ iff there is a chain*

$$N = M_\ell \leftarrow_{\mathcal{O}} M_{\ell-1} \leftarrow_{\mathcal{O}} \ldots \leftarrow_{\mathcal{O}} M_0 = M$$

of $\ell \geq 0$ rule applications. We call such a chain a derivation *of N from M w.r.t. $\mathcal{O}$. A rewriting step in such a derivation is called a* root step *if it applies a rule of the form ($\boldsymbol{R}_c$) at the root position. We write $N \overset{(n)}{\longleftarrow}_{\mathcal{O}} M$ to express that there is a derivation of N from M w.r.t. $\mathcal{O}$ that uses* at most n root steps.

For example, if $\mathcal{O}$ contains the axioms $\top \sqsubseteq \exists r.B$ and $s \sqsubseteq r$, then the following is a derivation w.r.t. $\mathcal{O}$:

$$\{A, \exists s.\{A\}\} \leftarrow_{\mathcal{O}} \{A, \exists r.\{A\}\} \leftarrow_{\mathcal{O}} \{A, \exists r.\{A, \exists r.\{B\}\}\} \leftarrow_{\mathcal{O}} \{A, \exists r.\{A, \exists r.\emptyset\}\}$$

This is a derivation without a root step, which first applies a rule of the form $(\mathbf{R}_m)$, then one of the form $(\mathbf{R}_c)$ (not at the root position), and finally one of the form $(\mathbf{R}_r)$. This shows $\mathsf{s}(A \sqcap \exists s.A) \xleftarrow{(0)}_{\mathcal{O}} \mathsf{s}(A \sqcap \exists r.(A \sqcap \exists r.\top))$.

The following theorem states that subsumption w.r.t. $\mathcal{O}$ corresponds to the existence of a derivation w.r.t. $\mathcal{O}$ whose root steps are bounded by the number of GCIs in $\mathcal{O}$ (see [3] for a proof of this result).

Theorem 5. *Let $\mathcal{O}$ be a flat $\mathcal{EL}^+$-ontology containing n GCIs and C, D be two $\mathcal{EL}$-concept descriptions. Then $C \sqsubseteq_{\mathcal{O}} D$ iff $\mathsf{s}(C) \xleftarrow{(n)}_{\mathcal{O}} \mathsf{s}(D)$.*

A Structural Characterization of Subsumption in $\mathcal{ELH}_{R^+}$

Our translation of unification problems into propositional satisfiability problems depends on a structural characterization of subsumption, which we can unfortunately only show for $\mathcal{ELH}_{R^+}$ ontologies. Throughout this subsection, we assume that $\mathcal{O}$ is a flat $\mathcal{ELH}_{R^+}$-ontology. We say that r is *transitive* if the transitivity axiom $r \circ r \sqsubseteq r$ belongs to $\mathcal{O}$.

Definition 6. *Let C, D be atoms. We say that C is* structurally subsumed *by D w.r.t. $\mathcal{O}$ ($C \sqsubseteq^{\mathsf{s}}_{\mathcal{O}} D$) iff*

- *$C = D$ is a concept name,*
- *$C = \exists r.C'$, $D = \exists s.D'$, $C' \sqsubseteq_{\mathcal{O}} D'$, and $r \trianglelefteq_{\mathcal{O}} s$, or*
- *$C = \exists r.C'$, $D = \exists s.D'$, and $C' \sqsubseteq_{\mathcal{O}} \exists t.D'$ for a transitive role t with $r \trianglelefteq_{\mathcal{O}} t \trianglelefteq_{\mathcal{O}} s$.*

On the one hand, structural subsumption is a stronger property than $C \sqsubseteq_{\mathcal{O}} D$ since it requires the atoms C and D to have "compatible" top-level structures. On the other hand, it is weaker than subsumption w.r.t. the empty ontology, i.e., whenever $C \sqsubseteq D$ holds for two atoms C and D, then $C \sqsubseteq^{\mathsf{s}}_{\mathcal{O}} D$, but not necessarily vice versa. If $\mathcal{O} = \emptyset$, then the three relations $\sqsubseteq$, $\sqsubseteq^{\mathsf{s}}_{\mathcal{O}}$, $\sqsubseteq_{\mathcal{O}}$ coincide on atoms. Like $\sqsubseteq$ and $\sqsubseteq_{\mathcal{O}}$, $\sqsubseteq^{\mathsf{s}}_{\mathcal{O}}$ is reflexive, transitive, and closed under applying existential restrictions (see [3] for proofs of the results mentioned in this paragraph).

Using the connection between subsumption and rewriting stated in Theorem 5, we can now prove a characterization of subsumption in the presence of an $\mathcal{ELH}_{R^+}$-ontology $\mathcal{O}$ that expresses subsumption in terms of structural subsumptions and derivations w.r.t. $\leftarrow_{\mathcal{O}}$. Recall that all $\mathcal{EL}$-concept descriptions are conjunctions of atoms, that $C \sqsubseteq_{\mathcal{O}} D_1 \sqcap \cdots \sqcap D_m$ iff $C \sqsubseteq_{\mathcal{O}} D_j$ for all $j \in \{1, \ldots, m\}$, and $C \sqsubseteq_{\mathcal{O}} D$ iff there is an ℓ such that $\mathsf{s}(C) \xleftarrow{(\ell)}_{\mathcal{O}} \mathsf{s}(D)$.

Lemma 7. *Let $\mathcal{O}$ be a flat $\mathcal{ELH}_{R^+}$-ontology, $C_1, \ldots, C_n$, D be atoms, and $\ell \geq 0$. Then $\mathsf{s}(C_1 \sqcap \cdots \sqcap C_n) \xleftarrow{(\ell)}_{\mathcal{O}} \mathsf{s}(D)$ iff there is*

1. *an index $i \in \{1, \ldots, n\}$ such that $C_i \sqsubseteq^{\mathsf{s}}_{\mathcal{O}} D$; or*
2. *a GCI $A_1 \sqcap \cdots \sqcap A_k \sqsubseteq B$ in $\mathcal{T}$ such that*
   a) *for every $p \in \{1, \ldots, k\}$ we have $\mathsf{s}(C_1 \sqcap \cdots \sqcap C_n) \xleftarrow{(\ell-1)}_{\mathcal{O}} \mathsf{s}(A_p)$,*
   b) *$\mathsf{s}(C_1 \sqcap \cdots \sqcap C_n) \xleftarrow{(\ell)}_{\mathcal{O}} \mathsf{s}(B)$, and*
   c) *$B \sqsubseteq^{\mathsf{s}}_{\mathcal{O}} D$.*

A detailed proof of this lemma is given in [3]. Here, we only want to point out that this proof makes extensive use of the transitivity of $\sqsubseteq^{\mathsf{s}}_{\mathcal{O}}$, and that this is the main reason why we cannot deal with general $\mathcal{EL}^+$-ontologies. In fact, while it is not hard to extend the definition of structural subsumption to more general kinds of ontologies, it is currently not clear to us how to do this such that the resulting relation is transitive; and without transitivity of structural subsumption, we cannot show a characterization analogous to the one in Lemma 7.

5 Reduction of Unification w.r.t. Cycle-Restricted $\mathcal{ELH}_{R^+}$-Ontologies to SAT

The main idea underlying the NP-membership results in [5] and [2] is to show that any $\mathcal{EL}$-unification problem that is unifiable w.r.t. the empty ontology and w.r.t. a cycle-restricted $\mathcal{EL}$-ontology, respectively, has a so-called local unifier. Here, we generalize the notion of a local unifier to the case of unification w.r.t. cycle-restricted $\mathcal{ELH}_{R^+}$-ontologies, but then go a significant step further. Instead of using an algorithm that "blindly" generates all local substitutions and then checks whether they are unifiers, we reduce the search for a local unifier to a propositional satisfiability problem.

Local Unifiers

Let Γ be a flat unification problem and $\mathcal{O}$ be a flat, cycle-restricted $\mathcal{ELH}_{R^+}$-ontology. We denote by At the set of atoms occurring as subdescriptions in subsumptions in Γ or axioms in $\mathcal{O}$ and define

$$\mathsf{At}_{\mathsf{tr}} := \mathsf{At} \cup \{\exists t.D' \mid \exists s.D' \in \mathsf{At},\ t \trianglelefteq_{\mathcal{O}} s,\ t \text{ transitive}\}.$$

Furthermore, we define the set of *non-variable atoms* by $\mathsf{At}_{\mathsf{nv}} := \mathsf{At}_{\mathsf{tr}} \backslash N_v$. Though the elements of $\mathsf{At}_{\mathsf{nv}}$ cannot be variables, they may contain variables if they are of the form $\exists r.X$ for some role r and a variable X. We call a function S that associates every variable $X \in N_v$ with a set $S_X \subseteq \mathsf{At}_{\mathsf{nv}}$ an *assignment*. Such an assignment induces the following relation $>_S$ on N_v: $>_S$ is the transitive closure of

$$\{(X, Y) \in N_v \times N_v \mid Y \text{ occurs in an element of } S_X\}.$$

We call the assignment S *acyclic* if $>_S$ is irreflexive (and thus a strict partial order). Any acyclic assignment S induces a unique substitution σ_S, which can be defined by induction along $>_S$:

- If X is a minimal element of N_v w.r.t. $>_S$, then we set $\sigma_S(X) := \bigsqcap_{D \in S_X} D$.
- Assume that $\sigma(Y)$ is already defined for all Y such that $X >_S Y$. Then we define $\sigma_S(X) := \bigsqcap_{D \in S_X} \sigma_S(D)$.

We call a substitution σ *local* if it is of this form, i.e., if there is an acyclic assignment S such that $\sigma = \sigma_S$. Since N_v and $\mathsf{At_{nv}}$ are finite, there are only finitely many local substitutions. Thus, if we know that any solvable unification problem has a local unifier, then we can enumerate (or guess, in a nondeterministic machine) all local substitutions and then check whether any of them is a unifier. Thus, in general many substitutions will be generated that only in the subsequent check turn out not to be unifiers. In contrast, our SAT reduction will ensure that only unifiers are generated.

The Reduction

Here, we reduce unification w.r.t. cycle-restricted $\mathcal{ELH}_{R^+}$-ontologies to the satisfiability problem for propositional logic, which is NP-complete. This shows that this unification problem is in NP. But more importantly, it immediately allows us to apply highly optimized SAT solvers for solving such unification problems.

As before, we assume that Γ is a flat unification problem and $\mathcal{O}$ is a flat, cycle-restricted $\mathcal{ELH}_{R^+}$-ontology. Let $\mathcal{T}$ be the subset of $\mathcal{O}$ that consists of the GCIs in $\mathcal{O}$. We define the set

$$\text{Left} := \mathsf{At} \cup \{C_1 \sqcap \cdots \sqcap C_n \mid C_1 \sqcap \cdots \sqcap C_n \sqsubseteq^? D \in \Gamma \text{ for some } D \in \mathsf{At}\}$$

that contains all atoms of Γ and $\mathcal{O}$ and all left-hand sides of subsumptions from Γ. For $L \in \text{Left}$ and $C \in \mathsf{At}$, we write "$C \in L$" if C is a top-level atom of L.

The propositional variables we use for the reduction are of the form $[L \sqsubseteq D]^i$ for $L \in \text{Left}$, $D \in \mathsf{At_{tr}}$, and $i \in \{0, \ldots, |\mathcal{T}|\}$. The intuition underlying these variables is that every satisfying propositional valuation induces an acyclic assignment S such that the following holds for the corresponding substitution σ_S: $[L \sqsubseteq D]^i$ is evaluated to true by the assignment iff $\mathsf{s}(\sigma_S(L))$ can be derived from $\mathsf{s}(\sigma_S(D))$ using at most i root steps, i.e., $\mathsf{s}(\sigma_S(L)) \xleftarrow{(i)}_{\mathcal{O}} \mathsf{s}(\sigma_S(D))$.

Additionally, we use the propositional variables $[X > Y]$ for $X, Y \in N_v$ to express the strict partial order $>_S$ induced by the acyclic assignment S.

The auxiliary function Dec is defined as follows for $C \in \mathsf{At}$, $D \in \mathsf{At_{tr}}$:

$$\text{Dec}(C \sqsubseteq D) = \begin{cases} \mathbf{1} & \text{if } C = D \\ [C \sqsubseteq D]^{|\mathcal{T}|} & \text{if } C \text{ and } D \text{ are ground} \\ \text{Trans}(C \sqsubseteq D) & \text{if } C = \exists r.C',\ D = \exists s.D', \text{ and } r \trianglelefteq_{\mathcal{O}} s\ , \\ [C \sqsubseteq D]^{|\mathcal{T}|} & \text{if } C \text{ is a variable} \\ \mathbf{0} & \text{otherwise} \end{cases}$$

$$\text{Trans}(C \sqsubseteq D) = [C' \sqsubseteq D']^{|\mathcal{T}|} \vee \bigvee_{\substack{t \text{ transitive} \\ r \trianglelefteq_{\mathcal{O}} t \trianglelefteq_{\mathcal{O}} s}} [C' \sqsubseteq \exists t.D']^{|\mathcal{T}|}.$$

Note that $C' \in \mathsf{At}$ and $D', \exists t.D' \in \mathsf{At}_{\mathsf{tr}}$ by the definition of $\mathsf{At}_{\mathsf{tr}}$ and since Γ and $\mathcal{O}$ are flat. Here, **0** and **1** are Boolean constants representing the truth values 0 (*false*) and 1 (*true*), respectively.

The unification problem will be reduced to satisfiability of the following set of propositional formulae. For simplicity, we do not use only clauses here. However, our formulae can be transformed into clausal form by introducing polynomially many auxiliary propositional variables and clauses.

Definition 8. *Let Γ be a flat unification problem and $\mathcal{O}$ a flat, cycle-restricted $\mathcal{ELH}_{R^+}$-ontology. The set $C(\Gamma, \mathcal{O})$ contains the following propositional formulae:*

(I) Translation of the subsumptions of Γ. *For every $L \sqsubseteq^? D$ in Γ, we introduce a clause asserting that this subsumption must hold:*

$$\rightarrow [L \sqsubseteq D]^{|\mathcal{T}|}.$$

(II) Translation of the relevant properties of subsumption.

1) For all ground atoms $C \in \mathsf{At}$, $D \in \mathsf{At}_{\mathsf{tr}}$ and $i \in \{0, \ldots, |\mathcal{T}|\}$ such that $C \not\sqsubseteq_{\mathcal{O}} D$, we introduce a clause preventing this subsumption:

$$[C \sqsubseteq D]^i \rightarrow \quad .$$

2) For every variable Y, $B \in \mathsf{At}_{\mathsf{nv}}$, $i, j \in \{0, \ldots, |\mathcal{T}|\}$, and $L \in \mathrm{Left}$, we introduce the clause

$$[L \sqsubseteq Y]^i \wedge [Y \sqsubseteq B]^j \rightarrow [L \sqsubseteq B]^{\min\{|\mathcal{T}|, i+j\}}.$$

3) For every $L \in \mathrm{Left} \setminus N_v$ and $D \in \mathsf{At}_{\mathsf{tr}}$, we introduce the following formulae, depending on L and D:

a) If D is a ground atom and L is not a ground atom, we introduce

$$[L \sqsubseteq D]^i \rightarrow \bigvee_{C \in L} \mathrm{Dec}(C \sqsubseteq D) \vee \bigvee_{\substack{A_1 \sqcap \cdots \sqcap A_k \sqsubseteq B \in \mathcal{O} \\ B \sqsubseteq_{\mathcal{O}} D}} ([L \sqsubseteq A_1]^{i-1} \wedge \cdots \wedge [L \sqsubseteq A_k]^{i-1})$$

for all $i \in \{1, \ldots, |\mathcal{T}|\}$ and

$$[L \sqsubseteq D]^0 \rightarrow \bigvee_{C \in L} \mathrm{Dec}(C \sqsubseteq D).$$

b) If D is a non-variable, non-ground atom, we introduce

$$[L \sqsubseteq D]^i \rightarrow \bigvee_{C \in L} \mathrm{Dec}(C \sqsubseteq D) \vee \bigvee_{A \text{ atom of } \mathcal{O}} ([L \sqsubseteq A]^i \wedge \mathrm{Dec}(A \sqsubseteq D))$$

for all $i \in \{1, \ldots, |\mathcal{T}|\}$ and

$$[L \sqsubseteq D]^0 \rightarrow \bigvee_{C \in L} \mathrm{Dec}(C \sqsubseteq D).$$

(III) Translation of the relevant properties of $>$.

1) Transitivity and irreflexivity of $>$ is expressed by the clauses

$$[X > X] \rightarrow \quad \text{and } [X > Y] \wedge [Y > Z] \rightarrow [X > Z]$$

for all $X, Y, Z \in N_v$.

2) The connection between $>$ and $\sqsubseteq$ is expressed using the clause

$$[X \sqsubseteq \exists r.Y]^i \rightarrow [X > Y]$$

for every $X, Y \in N_v$, $\exists r.Y \in \mathsf{At_{tr}}$, and $i \in \{0, \ldots, |\mathcal{T}|\}$.

It is easy to see that the set $C(\Gamma, \mathcal{O})$ can be constructed in time polynomial in the size of Γ and $\mathcal{O}$. In particular, subsumptions $B \sqsubseteq_{\mathcal{O}} D$ between ground atoms B, D can be checked in polynomial time in the size of $\mathcal{O}$ [4].

There are several differences between $C(\Gamma, \mathcal{O})$ and the clauses constructed in [6] to solve unification in $\mathcal{EL}$ w.r.t. the empty ontology. The propositional variables employed in [6] are of the form $[C \not\sqsubseteq D]$ for atoms C, D of Γ, i.e., they stand for non-subsumption rather than subsumption. The use of single atoms C instead of whole left-hand sides L also leads to a different encoding of the subsumptions from Γ in part (I). The clauses in (III) are identical up to negation of the variables $[X \sqsubseteq \exists r.Y]^i$. But most importantly, in [6] the properties of subsumption expressed in (II) need only deal with subsumption w.r.t. the empty ontology, whereas here we have to take a cycle-restricted $\mathcal{ELH}_{R^+}$-ontology into account. We do this by expressing the characterization of subsumption given in Lemma 7. This is also the reason why the propositional variables $[L \sqsubseteq D]^i$ have an additional index i: in fact, in Lemma 7 we refer to the number of root steps in the derivation that shows the subsumption, and this needs to be modeled in our SAT reduction.

Theorem 9. *The unification problem Γ is solvable w.r.t. $\mathcal{O}$ iff $C(\Gamma, \mathcal{O})$ is satisfiable.*

Since $C(\Gamma, \mathcal{O})$ can be constructed in polynomial time and SAT is in NP, this shows that unification w.r.t. cycle-restricted $\mathcal{ELH}_{R^+}$-ontologies is in NP. NP-hardness follows from the known NP-hardness of $\mathcal{EL}$-unification w.r.t. the empty ontology [5].

Corollary 10. *Unification w.r.t. cycle-restricted $\mathcal{ELH}_{R^+}$-ontologies is an NP-complete problem.*

To prove Theorem 9, we must show soundness and completeness of the reduction.

Soundness of the Reduction. Let τ be a valuation of the propositional variables that satisfies $C(\Gamma, \mathcal{O})$. We must show that then Γ has a unifier w.r.t. $\mathcal{O}$. To this purpose, we use τ to define an assignment S by

$$S_X := \{D \in \mathsf{At_{nv}} \mid \exists i \in \{0, \ldots, |\mathcal{T}|\} : \tau([X \sqsubseteq D]^i) = 1\}.$$

Using the clauses in (III), it is not hard to show [3] that $X >_S Y$ implies $\tau([X > Y]) = 1$. Due to the irreflexivity clause in (III), this yields that the assignment S is acyclic. Thus, it induces a substitution σ_S. A proof of the following lemma can be found in [3].

Lemma 11. *If $\tau([L \sqsubseteq D]^i) = 1$ for $L \in$ Left, $D \in \mathsf{At_{tr}}$, and $i \in \{0, \ldots, |\mathcal{T}|\}$, then $\sigma_S(L) \sqsubseteq_{\mathcal{O}} \sigma_S(D)$.*

Because of the clauses in (I), this lemma immediately implies that σ_S is a unifier of Γ w.r.t. $\mathcal{O}$.

Completeness of the Reduction. Given a unifier γ of Γ w.r.t. $\mathcal{O}$, we can define a valuation τ that satisfies $C(\Gamma, \mathcal{O})$ as follows.

Let $L \in$ Left and $D \in \mathsf{At_{tr}}$ and $i \in \{0, \ldots, |\mathcal{T}|\}$. We set $\tau([L \sqsubseteq D]^i) := 1$ iff $\mathsf{s}(\gamma(L)) \xleftarrow{(i)}_{\mathcal{O}} \mathsf{s}(\gamma(D))$. According to Theorem 5, we thus have $\tau([L \sqsubseteq D]^i) = 0$ for all $i \in \{0, \ldots, |\mathcal{T}|\}$ iff $\gamma(L) \not\sqsubseteq_{\mathcal{O}} \gamma(D)$. Otherwise, there is an $i \in \{0, \ldots, |\mathcal{T}|\}$ such that $\tau([L \sqsubseteq D]^j) = 1$ for all $j \geq i$, and $\tau([L \sqsubseteq D]^j) = 0$ for all $j < i$.

To define the valuation of the remaining propositional variables $[X > Y]$ with $X, Y \in N_v$, we set $\tau([X > Y]) = 1$ iff $X >_\gamma Y$, where $>_\gamma$ is defined as in (5), i.e., $X >_\gamma Y$ iff $\gamma(X) \sqsubseteq_{\mathcal{O}} \exists w.\gamma(Y)$ for some $w \in N_R^+$.

The following lemma, whose proof can be found in [3], shows completeness of our reduction using Lemma 7.

Lemma 12. *The valuation τ satisfies $C(\Gamma, \mathcal{O})$.*

Note that cycle-restrictedness of $\mathcal{O}$ is needed in order to satisfy the irreflexivity clause $[X > X] \rightarrow$ (see Lemma 3). We cannot dispense with this clause since it is needed in the proof of soundness to obtain acyclicity of the assignment S constructed there. In fact, only because S is acyclic can we define the substitution σ_S, which is then shown to be a unifier.

6 Conclusions

We have shown that unification w.r.t. cycle-restricted $\mathcal{ELH}_{R^+}$-ontologies can be reduced to propositional satisfiability. This improves on the results in [1,2] in two respects. First, it allows us to deal also with ontologies that contain transitivity and role hierarchy axioms, which are important for medical ontologies. Second, the SAT reduction can easily be implemented and enables us to make use of highly optimized SAT solvers, whereas the goal-oriented algorithm in [1], while having the potential of becoming quite efficient, requires a high amount of additional optimization work. The main topic for future research is to investigate whether we can get rid of cycle-restrictedness.

References

1. Baader, F., Borgwardt, S., Morawska, B.: Unification in the description logic $\mathcal{EL}$ w.r.t. cycle-restricted TBoxes. LTCS-Report 11-05, Theoretical Computer Science, TU Dresden (2011), `http://lat.inf.tu-dresden.de/research/reports.html`

2. Baader, F., Borgwardt, S., Morawska, B.: Extending unification in $\mathcal{EL}$ towards general TBoxes. In: Proc. of the 13th Int. Conf. on Principles of Knowledge Representation and Reasoning. AAAI Press (2012) (short paper)
3. Baader, F., Borgwardt, S., Morawska, B.: SAT encoding of unification in $\mathcal{ELH}_{\mathcal{R}+}$ w.r.t. cycle-restricted ontologies. LTCS-Report 12-02, Theoretical Computer Science, TU Dresden (2012), `http://lat.inf.tu-dresden.de/research/reports.html`
4. Baader, F., Brandt, S., Lutz, C.: Pushing the $\mathcal{EL}$ envelope. In: Kaelbling, L.P., Saffiotti, A. (eds.) Proc. of the 19th Int. Joint Conf. on Artificial Intelligence, pp. 364–369. Morgan Kaufmann, Los Altos (2005)
5. Baader, F., Morawska, B.: Unification in the Description Logic $\mathcal{EL}$. In: Treinen, R. (ed.) RTA 2009. LNCS, vol. 5595, pp. 350–364. Springer, Heidelberg (2009)
6. Baader, F., Morawska, B.: SAT Encoding of Unification in $\mathcal{EL}$. In: Fermüller, C.G., Voronkov, A. (eds.) LPAR-17. LNCS, vol. 6397, pp. 97–111. Springer, Heidelberg (2010)
7. Baader, F., Morawska, B.: Unification in the description logic $\mathcal{EL}$. Logical Methods in Computer Science 6(3) (2010)
8. Baader, F., Narendran, P.: Unification of concept terms in description logics. J. of Symbolic Computation 31(3), 277–305 (2001)
9. Baader, F., Snyder, W.: Unification theory. In: Robinson, J.A., Voronkov, A. (eds.) Handbook of Automated Reasoning, pp. 445–532. The MIT Press (2001)
10. Biere, A., Heule, M., van Maaren, H., Walsh, T. (eds.): Handbook of Satisfiability. IOS Press (2009)
11. Brandt, S.: Polynomial time reasoning in a description logic with existential restrictions, GCI axioms, and—what else? In: de Mántaras, R.L., Saitta, L. (eds.) Proc. of the 16th Eur. Conf. on Artificial Intelligence. pp. 298–302 (2004)
12. Campbell, J.R., Lopez Osornio, A., de Quiros, F., Luna, D., Reynoso, G.: Semantic interoperability and SNOMED CT: A case study in clinical problem lists. In: Kuhn, K., Warren, J., Leong, T.Y. (eds.) Proc. of the 12th World Congress on Health (Medical) Informatics, pp. 2401–2402. IOS Press (2007)
13. Degtyarev, A., Voronkov, A.: The undecidability of simultaneous rigid E-unification. Theor. Comput. Sci. 166(1&2), 291–300 (1996)
14. Gallier, J.H., Narendran, P., Plaisted, D.A., Snyder, W.: Rigid E-unification: NP-completeness and applications to equational matings. Inf. Comput. 87(1/2), 129–195 (1990)
15. Seidenberg, J., Rector, A.L.: Representing Transitive Propagation in OWL. In: Embley, D.W., Olivé, A., Ram, S. (eds.) ER 2006. LNCS, vol. 4215, pp. 255–266. Springer, Heidelberg (2006)
16. Sofronie-Stokkermans, V.: Locality and subsumption testing in $\mathcal{EL}$ and some of its extensions. In: Proc. Advances in Modal Logic (2008)
17. Suntisrivaraporn, B., Baader, F., Schulz, S., Spackman, K.: Replacing SEP-Triplets in SNOMED CT Using Tractable Description Logic Operators. In: Bellazzi, R., Abu-Hanna, A., Hunter, J. (eds.) AIME 2007. LNCS (LNAI), vol. 4594, pp. 287–291. Springer, Heidelberg (2007)

Dismatching and Local Disunification in $\mathcal{EL}$

Franz Baader, Stefan Borgwardt, and Barbara Morawska*

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

Abstract

Unification in Description Logics has been introduced as a means to detect redundancies in ontologies. We try to extend the known decidability results for unification in the Description Logic $\mathcal{EL}$ to disunification since negative constraints on unifiers can be used to avoid unwanted unifiers. While decidability of the solvability of general $\mathcal{EL}$-disunification problems remains an open problem, we obtain NP-completeness results for two interesting special cases: *dismatching problems*, where one side of each negative constraint must be ground, and *local* solvability of disunification problems, where we restrict the attention to solutions that are built from so-called atoms occurring in the input problem. More precisely, we first show that dismatching can be reduced to local disunification, and then provide two complementary NP-algorithms for finding local solutions of (general) disunification problems.

1998 ACM Subject Classification I.2.3 Deduction and Theorem Proving, I.2.4 Knowledge Representation Formalisms and Methods

Keywords and phrases Unification, Description Logics, SAT

Digital Object Identifier 10.4230/LIPIcs.RTA.2015.40

1 Introduction

Description logics (DLs) [6] are a family of logic-based knowledge representation formalisms, which can be used to represent the conceptual knowledge of an application domain in a structured and formally well-understood way. They are employed in various application areas, but their most notable success so far is the adoption of the DL-based language OWL [21] as standard ontology language for the semantic web. DLs allow their users to define the important notions (classes, relations) of the domain using concepts and roles; to state constraints on the way these notions can be interpreted using terminological axioms; and to deduce consequences such as subsumption (subclass) relationships from the definitions and constraints. The expressivity of a particular DL is determined by the constructors available for building concepts.

The DL $\mathcal{EL}$, which offers the concept constructors conjunction ($\sqcap$), existential restriction ($\exists r.C$), and the top concept ($\top$), has drawn considerable attention in the last decade since, on the one hand, important inference problems such as the subsumption problem are polynomial in $\mathcal{EL}$, even with respect to expressive terminological axioms [16]. On the other hand, though quite inexpressive, $\mathcal{EL}$ is used to define biomedical ontologies, such as the large medical ontology SNOMED CT.[1] For these reasons, the most recent OWL version, OWL 2, contains the profile OWL 2 EL,[2] which is based on a maximally tractable extension of $\mathcal{EL}$ [5].

* Supported by DFG under grant BA 1122/14-2.

[1] http://www.ihtsdo.org/snomed-ct/

[2] http://www.w3.org/TR/owl2-profiles/

Unification in Description Logics was introduced in [12] as a novel inference service that can be used to detect redundancies in ontologies. It is shown there that unification in the DL $\mathcal{FL}_0$, which differs from $\mathcal{EL}$ in that existential restriction is replaced by value restriction ($\forall r.C$), is ExpTime-complete. The applicability of this result was not only hampered by this high complexity, but also by the fact that $\mathcal{FL}_0$ is not used in practice to formulate ontologies.

In contrast, as mentioned above, $\mathcal{EL}$ is employed to build large biomedical ontologies for which detecting redundancies is a useful inference service. For example, assume that one developer of a medical ontology defines the concept of a *patient with severe head injury* as

$$\mathsf{Patient} \sqcap \exists \mathsf{finding}.(\mathsf{Head\_injury} \sqcap \exists \mathsf{severity}.\mathsf{Severe}), \tag{1}$$

whereas another one represents it as

$$\mathsf{Patient} \sqcap \exists \mathsf{finding}.(\mathsf{Severe\_finding} \sqcap \mathsf{Injury} \sqcap \exists \mathsf{finding\_site}.\mathsf{Head}). \tag{2}$$

Formally, these two concepts are not equivalent, but they are nevertheless meant to represent the same concept. They can obviously be made equivalent by treating the concept names Head_injury and Severe_finding as variables, and substituting the first one by Injury $\sqcap$ $\exists$finding_site.Head and the second one by $\exists$severity.Severe. In this case, we say that the concepts are unifiable, and call the substitution that makes them equivalent a *unifier*. In [10], we were able to show that unification in $\mathcal{EL}$ is of considerably lower complexity than unification in $\mathcal{FL}_0$: the decision problem for $\mathcal{EL}$ is NP-complete. The main idea underlying the proof of this result is to show that any solvable $\mathcal{EL}$-unification problem has a local unifier, i.e., a unifier built from a polynomial number of so-called atoms determined by the unification problem. However, the brute-force "guess and then test" NP-algorithm obtained from this result, which guesses a local substitution and then checks (in polynomial time) whether it is a unifier, is not useful in practice. We thus developed a goal-oriented unification algorithm for $\mathcal{EL}$, which is more efficient since nondeterministic decisions are only made if they are triggered by "unsolved parts" of the unification problem. Another option for obtaining a more efficient unification algorithm is a translation to satisfiability in propositional logic (SAT): in [9] it is shown how a given $\mathcal{EL}$-unification problem Γ can be translated in polynomial time into a propositional formula whose satisfying valuations correspond to the local unifiers of Γ.

Intuitively, a unifier of two $\mathcal{EL}$ concepts proposes definitions for the concept names that are used as variables: in our example, we know that, if we define Head_injury as Injury $\sqcap$ $\exists$finding_site.Head and Severe_finding as $\exists$severity.Severe, then the two concepts (1) and (2) are equivalent w.r.t. these definitions. Of course, this example was constructed such that the unifier (which is actually local) provides sensible definitions for the concept names used as variables. In general, the existence of a unifier only says that there is a structural similarity between the two concepts. The developer who uses unification as a tool for finding redundancies in an ontology or between two different ontologies needs to inspect the unifier(s) to see whether the definitions it suggests really make sense. For example, the substitution that replaces Head_injury by Patient $\sqcap$ Injury $\sqcap$ $\exists$finding_site.Head and Severe_finding by Patient $\sqcap$ $\exists$severity.Severe is also a local unifier, which however does not make sense. Unfortunately, even small unification problems like the one in our example can have too many local unifiers for manual inspection. In [2] we propose to restrict the attention to so-called minimal unifiers, which form a subset of all local unifiers. In our example, the nonsensical unifier is indeed not minimal. In general, however, the restriction to minimal unifiers may preclude interesting local unifiers. In addition, as shown in [2], computing minimal unifiers is actually harder than computing local unifiers (unless the polynomial hierarchy collapses). In the present paper, we propose disunification as a more direct approach

for avoiding local unifiers that do not make sense. In addition to positive constraints (requiring equivalence or subsumption between concepts), a disunification problem may also contain negative constraints (preventing equivalence or subsumption between concepts). In our example, the nonsensical unifier can be avoided by adding the dissubsumption constraint

$$\mathsf{Head\_injury} \not\sqsubseteq^? \mathsf{Patient} \tag{3}$$

to the equivalence constraint (1) $\equiv^?$ (2).

Unification and disunification in DLs is actually a special case of unification and disunification modulo equational theories (see [12] and [10] for the equational theories respectively corresponding to $\mathcal{FL}_0$ and $\mathcal{EL}$). Disunification modulo equational theories has, e.g., been investigated in [17, 18]. It is well-known in unification theory that for effectively finitary equational theories, i.e., theories for which finite complete sets of unifiers can effectively be computed, disunification can be reduced to unification: to decide whether a disunification problem has a solution, one computes a finite complete set of unifiers of the equations and then checks whether any of the unifiers in this set also solves the disequations. Unfortunately, for $\mathcal{FL}_0$ and $\mathcal{EL}$, this approach is not feasible since the corresponding equational theories have unification type zero [10, 12], and thus finite complete sets of unifiers need not even exist. Nevertheless, it was shown in [14] that the approach used in [12] to decide unification (reduction to language equations, which are then solved using tree automata) can be adapted such that it can also deal with disunification. This yields the result that disunification in $\mathcal{FL}_0$ has the same complexity (ExpTime-complete) as unification.

For $\mathcal{EL}$, going from unification to disunification appears to be more problematic. In fact, the main reason for unification to be decidable and in NP is locality: if the problem has a unifier then it has a local unifier. We will show that disunification in $\mathcal{EL}$ is not local in this sense by providing an example of a disunification problem that has a solution, but no local solution. Decidability and complexity of disunification in $\mathcal{EL}$ remains an open problem, but we provide partial solutions that are of interest in practice. On the one hand, we investigate *dismatching problems*, i.e., disunification problems where the negative constraints are dissubsumptions $C \not\sqsubseteq^? D$ for which C or D is ground (i.e., does not contain a variable). Note that the dissubsumption (3) from above actually satisfies this restriction since $\mathsf{Patient}$ is not a variable. We prove that (general) solvability of dismatching problems can be reduced to *local disunification*, i.e., the question whether a given $\mathcal{EL}$-disunification problem has a *local* solution, which shows that dismatching in $\mathcal{EL}$ is NP-complete. On the other hand, we develop two specialized algorithms to solve local disunification problems that extend the ones for unification [9, 10]: a goal-oriented algorithm that reduces the amount of nondeterministic guesses necessary to find a local solution, as well as a translation to SAT. The reason we present two kinds of algorithms is that, in the case of unification, they have proved to complement each other well in first evaluations [1]: the goal-oriented algorithm needs less memory and finds minimal solutions faster, while the SAT reduction generates larger data structures (of cubic size), but outperforms the goal-oriented algorithm on unsolvable problems.

Full proofs of the results presented below can be found in [4].

2 Subsumption and dissubsumption in $\mathcal{EL}$

The syntax of $\mathcal{EL}$ is defined based on two sets $\mathsf{N_C}$ and $\mathsf{N_R}$ of *concept names* and *role names*, respectively. *Concept terms* are built from concept names using the constructors *conjunction* ($C \sqcap D$), *existential restriction* ($\exists r.C$ for $r \in \mathsf{N_R}$), and *top* ($\top$). An *interpretation* $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$

Table 1 Syntax and semantics of $\mathcal{EL}$.

Name	Syntax	Semantics
top	$\top$	$\top^{\mathcal{I}} := \Delta^{\mathcal{I}}$
conjunction	$C \sqcap D$	$(C \sqcap D)^{\mathcal{I}} := C^{\mathcal{I}} \cap D^{\mathcal{I}}$
existential restriction	$\exists r.C$	$(\exists r.C)^{\mathcal{I}} := \{x \mid \exists y.(x,y) \in r^{\mathcal{I}} \wedge y \in C^{\mathcal{I}}\}$

consists of a non-empty domain $\Delta^{\mathcal{I}}$ and an interpretation function that maps concept names to subsets of $\Delta^{\mathcal{I}}$ and role names to binary relations over $\Delta^{\mathcal{I}}$. This function is extended to concept terms as shown in the semantics column of Table 1.

A concept term C is *subsumed* by a concept term D (written $C \sqsubseteq D$) if for every interpretation $\mathcal{I}$ it holds that $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$. We write a *dissubsumption* $C \not\sqsubseteq D$ to abbreviate the fact that $C \sqsubseteq D$ does not hold. The two concept terms C and D are *equivalent* (written $C \equiv D$) if $C \sqsubseteq D$ and $D \sqsubseteq C$. Note that we use "=" to denote *syntactic* equality between concept terms, whereas "≡" denotes semantic equivalence.

Since conjunction is interpreted as intersection, we can treat $\sqcap$ as a commutative and associative operator, and thus dispense with parentheses in nested conjunctions. An *atom* is a concept name or an existential restriction. Hence, every concept term C is a conjunction of atoms or $\top$. We call the atoms in this conjunction the *top-level atoms* of C. Obviously, C is equivalent to the conjunction of its top-level atoms, where the empty conjunction corresponds to $\top$. An atom is *flat* if it is a concept name or an existential restriction of the form $\exists r.A$ with $A \in \mathsf{N_C}$.

Subsumption in $\mathcal{EL}$ is decidable in polynomial time [8] and can be checked by recursively comparing the top-level atoms of the two concept terms.

▶ **Lemma 1** ([10]). *For two atoms C, D, we have $C \sqsubseteq D$ iff $C = D$ is a concept name or $C = \exists r.C'$, $D = \exists r.D'$, and $C' \sqsubseteq D'$. If C, D are concept terms, then $C \sqsubseteq D$ iff for every top-level atom D' of D there is a top-level atom C' of C such that $C' \sqsubseteq D'$.*

We obtain the following contrapositive formulation characterizing dissubsumption.

▶ **Lemma 2.** *For two concept terms C, D, we have $C \not\sqsubseteq D$ iff there is a top-level atom D' of D such that for all top-level atoms C' of C it holds that $C' \not\sqsubseteq D'$.*

In particular, $C \not\sqsubseteq D$ is characterized by the existence of a top-level atom D' of D for which $C \not\sqsubseteq D'$ holds. By further analyzing the structure of atoms, we obtain the following.

▶ **Lemma 3.** *Let C, D be two atoms. Then we have $C \not\sqsubseteq D$ iff either*
1. *C or D is a concept name and $C \neq D$; or*
2. *$D = \exists r.D'$, $C = \exists s.C'$, and $r \neq s$; or*
3. *$D = \exists r.D'$, $C = \exists r.C'$, and $C' \not\sqsubseteq D'$.*

3 Disunification

As described in the introduction, we now partition the set $\mathsf{N_C}$ into a set of *(concept) variables* ($\mathsf{N_v}$) and a set of *(concept) constants* ($\mathsf{N_c}$). A concept term is *ground* if it does not contain any variables. We define a quite general notion of disunification problems that is similar to the equational formulae used in [18].

▶ **Definition 4.** A *disunification problem* Γ is a formula built from subsumptions of the form $C \sqsubseteq^? D$, where C and D are concept terms, using the logical connectives $\wedge$, $\vee$, and $\neg$. We use equations $C \equiv^? D$ to abbreviate $(C \sqsubseteq^? D) \wedge (D \sqsubseteq^? C)$, disequations $C \not\equiv^? D$ for $\neg(C \sqsubseteq^? D) \vee \neg(D \sqsubseteq^? C)$, and dissubsumptions $C \not\sqsubseteq^? D$ instead of $\neg(C \sqsubseteq^? D)$. A *basic disunification problem* is a conjunction of subsumptions and dissubsumptions. A *dismatching problem* is a basic disunification problem in which all dissubsumptions $C \not\sqsubseteq^? D$ are such that C or D is ground. Finally, a *unification problem* is a conjunction of subsumptions.

The definition of dismatching problems is partially motivated by the definition of *matching* in description logics, where similar restrictions are imposed on unification problems [7, 11, 23]. Another motivation comes from our experience that dismatching problems already suffice to formulate most of the negative constraints one may want to put on unification problems, as described in the introduction.

To define the semantics of disunification problems, we now fix a *finite signature* $\Sigma \subseteq \mathsf{N_C} \cup \mathsf{N_R}$ and assume that all disunification problems contain only concept terms constructed over the symbols in Σ. A *substitution* σ maps every variable in Σ to a ground concept term constructed over the symbols of Σ. This mapping can be extended to all concept terms (over Σ) in the usual way. A substitution σ *solves* a subsumption $C \sqsubseteq^? D$ if $\sigma(C) \sqsubseteq \sigma(D)$; it *solves* $\Gamma_1 \wedge \Gamma_2$ if it solves both Γ_1 and Γ_2; it solves $\Gamma_1 \vee \Gamma_2$ if it solves Γ_1 or Γ_2; and it solves $\neg\Gamma$ if it does not solve Γ. A substitution that solves a given disunification problem is called a *solution* of this problem. A disunification problem is *solvable* if it has a solution.

In contrast to unification, in disunification it does make a difference whether or not solutions may contain variables from $\mathsf{N_v} \cap \Sigma$ or additional symbols from $(\mathsf{N_C} \cup \mathsf{N_R}) \setminus \Sigma$ [17]. In the context of the application sketched in the introduction, restricting solutions to ground terms over Σ is appropriate: the finite signature Σ contains exactly the symbols that occur in the ontology to be checked for redundancy, and since a solution σ is supposed to provide definitions for the variables in Σ, it should not use the variables themselves to define them; moreover, definitions that contain symbols that are not in Σ would be meaningless to the user.

Reduction to basic disunification problems

We will consider only basic disunification problems in the following. The reason is that there is a straightforward NP-reduction from solvability of arbitrary disunification problems to solvability of basic disunification problems. In this reduction, we view all subsumptions occurring in the disunification problem as propositional variables and guess a satisfying valuation of the resulting propositional formula. It then suffices to check solvability of the basic disunification problem obtained as the conjunction of all subsumptions evaluated to true and the negations of all subsumptions evaluated to false. Since the problems considered in the following sections are all NP-complete, the restriction to basic disunification problems does not affect our complexity results. In the following, we thus restrict the attention to basic disunification problems, which we simply call *disunification problems* and consider them to be sets of subsumptions and dissubsumptions.

Reduction to flat disunification problems

We further simplify our analysis by considering *flat* disunification problems, which means that they may only contain *flat* dissubsumptions of the form $C_1 \sqcap \cdots \sqcap C_n \not\sqsubseteq^? D_1 \sqcap \cdots \sqcap D_m$

for flat atoms $C_1, \ldots, C_n, D_1, \ldots, D_m$ with $m, n \geq 0$,[3] and *flat* subsumptions of the form $C_1 \sqcap \cdots \sqcap C_n \sqsubseteq^? D_1$ for flat atoms $C_1, \ldots, C_n, D_1$ with $n \geq 0$.

The restriction to flat disunification problems is without loss of generality: to flatten concept terms, one can simply introduce new variables and equations to abbreviate subterms [10]. Moreover, a subsumption of the form $C \sqsubseteq^? D_1 \sqcap \cdots \sqcap D_m$ is equivalent to $C \sqsubseteq^? D_1$, ..., $C \sqsubseteq^? D_m$. Any solution of a disunification problem Γ can be extended to a solution of the resulting flat disunification problem Γ', and conversely every solution of Γ' also solves Γ.

This flattening procedure also works for unification problems. However, dismatching problems cannot without loss of generality be restricted to being flat since the introduction of new variables to abbreviate subterms may destroy the property that one side of each dissubsumption is ground (see also Section 4).

For solving flat unification problems, it has been shown that it suffices to consider so-called local solutions [10], which are restricted to use only the atoms occurring in the input problem. We extend this notion to disunification as follows. Let Γ be a flat disunification problem. We denote by At the set of all (flat) atoms occurring as subterms in Γ, by Var the set of variables occurring in Γ, and by $\mathsf{At_{nv}} := \mathsf{At} \setminus \mathsf{Var}$ the set of *non-variable atoms* of Γ. Let $S \colon \mathsf{Var} \to 2^{\mathsf{At_{nv}}}$ be an *assignment (for Γ)*, i.e. a function that assigns to each variable $X \in \mathsf{Var}$ a set $S_X \subseteq \mathsf{At_{nv}}$ of non-variable atoms. The relation $>_S$ on Var is defined as the transitive closure of $\{(X, Y) \in \mathsf{Var}^2 \mid Y \text{ occurs in an atom of } S_X\}$. If this defines a strict partial order, i.e. $>_S$ is irreflexive, then S is called *acyclic*. In this case, we can define the substitution σ_S inductively along $>_S$ as follows: if X is minimal, then $\sigma_S(X) := \bigsqcap_{D \in S_X} D$; otherwise, assume that $\sigma_S(Y)$ is defined for all $Y \in \mathsf{Var}$ with $X > Y$, and define

$$\sigma_S(X) := \bigsqcap_{D \in S_X} \sigma_S(D).$$

It is easy to see that the concept terms $\sigma_S(D)$ are ground and constructed from the symbols of Σ, and hence σ_S is a valid candidate for a solution of Γ according to Definition 4.

▶ **Definition 5.** Let Γ be a flat disunification problem. A substitution σ is called *local* if there exists an acyclic assignment S for Γ such that $\sigma = \sigma_S$. The disunification problem Γ is *locally solvable* if it has a local solution, i.e. a solution that is a local substitution. *Local disunification* is the problem of checking flat disunification problems for local solvability.

Note that assignments and local solutions are defined only for *flat* disunification problems.

Obviously, local disunification is decidable in NP: We can guess an assignment S, and check it for acyclicity and whether the induced substitution solves the disunification problem in polynomial time. It has been shown [10] that unification in $\mathcal{EL}$ is *local* in the sense that the equivalent flattened problem has a local solution iff the original problem is solvable. Hence not only local, but also general solvability of unification problems in $\mathcal{EL}$ can be decided in NP. In addition, this shows that NP-hardness already holds for local unification, and thus also for local disunification.

▶ **Fact 6.** *Deciding local solvability of flat disunification problems in $\mathcal{EL}$ is* NP*-complete.*

The next example shows that disunification in $\mathcal{EL}$ is *not local* in this sense.

▶ **Example 7.** Consider the flat disunification problem

$$\Gamma := \{X \sqsubseteq^? B,\ A \sqcap B \sqcap C \sqsubseteq^? X,\ \exists r.X \sqsubseteq^? Y,\ \top \not\sqsubseteq^? Y,\ Y \not\sqsubseteq^? \exists r.B\}$$

[3] Recall that the empty conjunction is $\top$.

with variables X, Y and constants A, B, C. The substitution σ with $\sigma(X) := A \sqcap B \sqcap C$ and $\sigma(Y) := \exists r.(A \sqcap C)$ is a solution of Γ. For σ to be local, the atom $\exists r.(A \sqcap C)$ would have to be of the form $\sigma(D)$ for a non-variable atom D occurring in Γ. But the only candidates for D are $\exists r.X$ and $\exists r.B$, none of which satisfy $\exists r.(A \sqcap C) = \sigma(D)$.

We show that Γ cannot have another solution that is local. Assume to the contrary that Γ has a local solution γ. We know that $\gamma(Y)$ cannot be $\top$ since γ must solve the first dissubsumption. Furthermore, none of the constants A, B, C can be a top-level atom of $\gamma(Y)$ since this would contradict the third subsumption. That leaves only the non-variable atoms $\exists r.\gamma(X)$ and $\exists r.B$, which are ruled out by the last dissubsumption since both $\gamma(X)$ and B are subsumed by B.

The decidability and complexity of general solvability of disunification problems is still open. In the following, we first consider the special case of solving dismatching problems, for which we show a similar result as for unification: every dismatching problem can be polynomially reduced to a flat problem that has a local solution iff the original problem is solvable. The main difference is that this reduction is nondeterministic. In this way, we reduce dismatching to local disunification. We then provide two different NP-algorithms for the latter problem by extending the rule-based unification algorithm from [10] and adapting the SAT encoding of unification problems from [9]. These algorithms are more efficient than the brute-force "guess and then test" procedure on which our argument for Fact 6 was based.

4 Reducing dismatching to local disunification

As mentioned in Section 3, we cannot restrict our attention to flat dismatching problems without loss of generality. Instead, the nondeterministic algorithm we present in the following reduces any dismatching problem Γ to a flat *disunification* problem Γ' with the property that local solvability of Γ' is equivalent to the solvability of Γ. Since the algorithm takes at most polynomial time in the size of Γ, this shows that dismatching in $\mathcal{EL}$ is NP-complete. For simplicity, we assume that the subsumptions and the non-ground sides of the dissubsumptions have already been flattened using the approach mentioned in the previous section. This retains the property that all dissubsumptions have one ground side and does not affect the solvability of the problem.

Our procedure exhaustively applies a set of rules to the (dis)subsumptions in a dismatching problem (see Figures 1 and 2). In these rules, $C_1, \ldots, C_n$ and $D_1, \ldots, D_m$ are atoms. The rule Left Decomposition includes the special case where the left-hand side of $\mathfrak{s}$ is $\top$, in which case $\mathfrak{s}$ is simply removed from the problem. Note that at most one rule is applicable to any given (dis)subsumption. The choice which (dis)subsumption to consider next is don't care nondeterministic, but the choices in the rules Right Decomposition and Solving Left-Ground Dissubsumptions are don't know nondeterministic.

▶ **Algorithm 8.** Let Γ_0 be a dismatching problem. We initialize $\Gamma := \Gamma_0$. While any of the rules of Figures 1 and 2 is applicable to any element of Γ, choose one such element and apply the corresponding rule. If any rule application fails, then return "failure".

To see that every run of the nondeterministic algorithm terminates in polynomial time, note that each rule application takes only polynomial time in the size of the chosen (dis)subsumption. In particular, subsumptions between ground atoms can be checked in polynomial time [8]. Additionally, we can show that the algorithm needs at most polynomially many rule applications since each rule application decreases the following measure on Γ: we sum up all sizes of (dis)subsumptions in Γ to which a rule is still applicable, where the size

Right Decomposition:

Condition: This rule applies to $\mathfrak{s} = C_1 \sqcap \cdots \sqcap C_n \not\sqsubseteq^? D_1 \sqcap \cdots \sqcap D_m$ if $m = 0$ or $m > 1$, and $C_1, \ldots, C_n$, $D_1, \ldots, D_m$ are atoms.
Action: If $m = 0$, then *fail*. Otherwise, choose an index $i \in \{1, \ldots, m\}$ and replace $\mathfrak{s}$ by $C_1 \sqcap \cdots \sqcap C_n \not\sqsubseteq^? D_i$.

Left Decomposition:

Condition: This rule applies to $\mathfrak{s} = C_1 \sqcap \cdots \sqcap C_n \not\sqsubseteq^? D$ if $n = 0$ or $n > 1$, $C_1, \ldots, C_n$ are atoms, and D is a non-variable atom.
Action: Replace $\mathfrak{s}$ by $C_1 \not\sqsubseteq^? D, \ldots, C_n \not\sqsubseteq^? D$.

Atomic Decomposition:

Condition: This rule applies to $\mathfrak{s} = C \not\sqsubseteq^? D$ if C and D are non-variable atoms.
Action: Apply the first case that matches $\mathfrak{s}$:
a) if C and D are ground and $C \sqsubseteq D$, then *fail*;
b) if C and D are ground and $C \not\sqsubseteq D$, then remove $\mathfrak{s}$ from Γ;
c) if C or D is a constant, then remove $\mathfrak{s}$ from Γ;
d) if $C = \exists r.C'$ and $D = \exists s.D'$ with $r \neq s$, then remove $\mathfrak{s}$ from Γ;
e) if $C = \exists r.C'$ and $D = \exists r.D'$, then replace $\mathfrak{s}$ by $C' \not\sqsubseteq^? D'$.

Figure 1 Decomposition rules.

of $C \sqsubseteq^? D$ or $C \not\sqsubseteq^? D$ is defined as $|C| \cdot |D|$, and $|C|$ is the number of symbols needed to write down C (for details, see [4]).

Note that the Solving rule for left-ground dissubsumptions is not limited to non-flat dissubsumptions, and thus the algorithm completely eliminates all left-ground dissubsumptions from Γ. It is also easy to see that, if the algorithm is successful, then the resulting disunification problem Γ is flat. We now prove that this nondeterministic procedure is correct in the following sense.

▶ **Lemma 9.** *The dismatching problem Γ_0 is solvable iff there is a successful run of Algorithm 8 such that the resulting flat disunification problem Γ has a local solution.*

Proof Sketch. Soundness (i.e., the if direction) is easy to show, using Lemmas 1–3. Showing completeness (i.e., the only-if direction) is more involved. Basically, given a solution γ of Γ_0, we can use γ to guide the rule applications and extend γ to the newly introduced variables such that each rule application is successful and the invariant "γ *solves all (dis)subsumptions of* Γ" is maintained. Once no more rules can be applied, we have a flat disunification problem Γ of which the extended substitution γ is a (possibly non-local) solution. To obtain a local solution, we denote by At, Var, and $\mathsf{At}_{\mathsf{nv}}$ the sets as defined in Section 3 and define the assignment S induced by γ as:

$$S_X := \{D \in \mathsf{At}_{\mathsf{nv}} \mid \gamma(X) \sqsubseteq \gamma(D)\},$$

for all (old and new) variables $X \in \mathsf{Var}$. It can be shown that this assignment is acyclic and that the induced local substitution σ_S solves Γ, and thus also Γ_0 (see [4] for details). ◀

The disunification problem of Example 7 is in fact a dismatching problem. The rule Solving Left-Ground Dissubsumptions can be used to replace $\top \not\sqsubseteq^? Y$ with $Y \sqsubseteq^? \exists r.Z$. The presence of the new atom $\exists r.Z$ makes the solution σ introduced in Example 7 local.

Flattening Right-Ground Dissubsumptions:

Condition: This rule applies to $\mathfrak{s} = X \not\sqsubseteq^? \exists r.D$ if X is a variable and D is ground and is not a concept name.
Action: Introduce a new variable X_D and replace $\mathfrak{s}$ by $X \not\sqsubseteq^? \exists r.X_D$ and $D \sqsubseteq^? X_D$.

Flattening Left-Ground Subsumptions:

Condition: This rule applies to $\mathfrak{s} = C_1 \sqcap \cdots \sqcap C_n \sqcap \exists r_1.D_1 \sqcap \cdots \sqcap \exists r_m.D_m \sqsubseteq^? X$ if $m > 0$, X is a variable, $C_1, \ldots, C_n$ are flat ground atoms, and $\exists r_1.D_1, \ldots, \exists r_m.D_m$ are non-flat ground atoms.
Action: Introduce new variables $X_{D_1}, \ldots, X_{D_m}$ and replace $\mathfrak{s}$ by $D_1 \sqsubseteq^? X_{D_1}, \ldots, D_m \sqsubseteq^? X_{D_m}$ and $C_1 \sqcap \cdots \sqcap C_n \sqcap \exists r_1.X_{D_1} \sqcap \cdots \sqcap \exists r_m.X_{D_m} \sqsubseteq^? X$.

Solving Left-Ground Dissubsumptions:

Condition: This rule applies to $\mathfrak{s} = C_1 \sqcap \cdots \sqcap C_n \not\sqsubseteq^? X$ if X is a variable and $C_1, \ldots, C_n$ are ground atoms.
Action: Choose one of the following options:

- Choose a constant $A \in \Sigma$ and replace $\mathfrak{s}$ by $X \sqsubseteq^? A$. If $C_1 \sqcap \cdots \sqcap C_n \sqsubseteq A$, then *fail*.
- Choose a role $r \in \Sigma$, introduce a new variable Z, replace $\mathfrak{s}$ by $X \sqsubseteq^? \exists r.Z$, $C_1 \not\sqsubseteq^? \exists r.Z$, ..., $C_n \not\sqsubseteq^? \exists r.Z$, and immediately apply Atomic Decomposition to each of these dissubsumptions.

Figure 2 Flattening and solving rules.

Together with Fact 6 and the NP-hardness of unification in $\mathcal{EL}$ [10], Lemma 9 yields the following complexity result.

▶ **Theorem 10.** *Deciding solvability of dismatching problems in $\mathcal{EL}$ is* NP*-complete.*

5 A goal-oriented algorithm for local disunification

In this section, we present an algorithm for local disunification that is based on transformation rules. Basically, to solve the subsumptions, this algorithm uses the rules of the goal-oriented algorithm for unification in $\mathcal{EL}$ [10, 3], which produces only local unifiers. Since any local solution of the disunification problem is a local unifier of the subsumptions in the problem, one might think that it is then sufficient to check whether any of the produced unifiers also solves the dissubsumptions. This would not be complete, however, since the goal-oriented algorithm for unification does *not* produce *all* local unifiers. For this reason, we have additional rules for solving the dissubsumptions. Both rule sets contain (deterministic) *eager* rules that are applied with the highest priority, and *nondeterministic* rules that are only applied if no eager rule is applicable. The goal of the eager rules is to enable the algorithm to detect obvious contradictions as early as possible in order to reduce the number of nondeterministic choices it has to make.

Let now Γ_0 be the flat disunification problem for which we want to decide local solvability, and let the sets At, Var, and $\mathsf{At}_{\mathsf{nv}}$ be defined as in Section 3. We assume without loss of generality that the dissubsumptions in Γ_0 have only a single atom on the right-hand side. If this is not the case, it can easily be achieved by exhaustive application of the nondeterministic rule Right Decomposition (see Figure 1) without affecting the complexity of the overall procedure.

Starting with Γ_0, the algorithm maintains a current disunification problem Γ and a current acyclic assignment S, which initially assigns the empty set to all variables. In addition, for each subsumption or dissubsumption in Γ, it maintains the information on whether it is *solved* or not. Initially, all subsumptions of Γ_0 are unsolved, except those with a variable on the

right-hand side, and all dissubsumptions in Γ_0 are unsolved, except those with a variable on the left-hand side and a non-variable atom on the right-hand side. Subsumptions of the form $C_1 \sqcap \cdots \sqcap C_n \sqsubseteq^? X$ and dissubsumptions of the form $X \not\sqsubseteq^? D$, for a non-variable atom D, are called *initially solved*. Intuitively, they only specify constraints on the assignment S_X. More formally, this intuition is captured by the process of *expanding* Γ w.r.t. the variable X, which performs the following actions:

- every initially solved subsumption $\mathfrak{s} \in \Gamma$ of the form $C_1 \sqcap \cdots \sqcap C_n \sqsubseteq^? X$ is expanded by adding the subsumption $C_1 \sqcap \cdots \sqcap C_n \sqsubseteq^? E$ to Γ for every $E \in S_X$, and
- every initially solved dissubsumption $X \not\sqsubseteq^? D \in \Gamma$ is expanded by adding $E \not\sqsubseteq^? D$ to Γ for every $E \in S_X$.

A (non-failing) application of a rule of our algorithm does the following:

- it solves exactly one unsolved subsumption or dissubsumption,
- it may extend the current assignment S by adding elements of $\mathsf{At}_{\mathsf{nv}}$ to some set S_X,
- it may introduce new flat subsumptions or dissubsumptions built from elements of At,
- it keeps Γ expanded w.r.t. all variables X.

Subsumptions and dissubsumptions are only added by a rule application or by expansion if they are not already present in Γ. If a new subsumption or dissubsumption is added to Γ, it is marked as unsolved, unless it is initially solved (because of its form). Solving subsumptions and dissubsumptions is mostly independent, except for expanding Γ, which can add new unsolved subsumptions and dissubsumptions at the same time, and may be triggered by solving a subsumption or a dissubsumption.

The rules dealing with subsumptions are depicted in Figure 3; these three eager and two nondeterministic rules are essentially the same as the ones in [3], with the only difference that the background ontology $\mathcal{T}$ used there is empty for our purposes. Note that several rules may be applicable to the same subsumption, and there is no preference between them. Using Eager Ground Solving, the algorithm can immediately evaluate ground subsumptions via the polynomial-time algorithm of [8]. If the required subsumption holds, it is marked as solved, and otherwise Γ cannot be solvable and hence the algorithm fails. Eager Solving detects when a subsumption trivially holds because the atom D from the right-hand side is already present on the left-hand side, either directly or via the assignment of a variable. Eager Extension is applicable in case the left-hand side of a subsumption is essentially equivalent to a single variable X due to all its atoms being "subsumed by" S_X. In this case, there is no other option but to add the right-hand side atom to S_X to solve the subsumption, and to expand Γ w.r.t. this new assignment. In case none of the eager rules apply to a subsumption, it can be solved nondeterministically by either extending the assignment of a variable that occurs on the left-hand side (Extension), or decomposing the subsumption by looking for matching existential restrictions on both sides (cf. Lemma 1).

The new rules for solving dissubsumptions are listed in Figure 4. These include variants of the Left Decomposition and Atomic Decomposition rules from the previous section (see Figure 1). In these two rules, which are eager, instead of removing dissubsumptions we mark them as solved. Additionally, Γ may have to be expanded if such a rule adds a new dissubsumption that is initially solved. The new nondeterministic rule Local Extension follows the same idea as the Solving rule for left-ground dissubsumptions (see Figure 2), but does not have to introduce new variables and atoms since we are looking only for local solutions. Note that the left-hand side of $\mathfrak{s}$ may be a variable, and then $\mathfrak{s}$ is of the form $Y \not\sqsubseteq^? X$. This dissubsumption is not initially solved, because X is not a non-variable atom.

▶ **Algorithm 11.** Let Γ_0 be a flat disunification problem. We initialize $\Gamma := \Gamma_0$ and $S_X := \emptyset$ for all variables $X \in \mathsf{Var}$. While Γ contains an unsolved subsumption or dissubsumption, do

Eager Ground Solving:

Condition: This rule applies to $\mathfrak{s} = C_1 \sqcap \cdots \sqcap C_n \sqsubseteq^? D \in \Gamma$, if $\mathfrak{s}$ is ground.
Action: The rule application fails if $\mathfrak{s}$ does not hold. Otherwise, $\mathfrak{s}$ is marked as *solved*.

Eager Solving:

Condition: This rule applies to $\mathfrak{s} = C_1 \sqcap \cdots \sqcap C_n \sqsubseteq^? D \in \Gamma$, if there is an index $i \in \{1, \ldots, n\}$, such that $C_i = D$ or $C_i = X \in \mathsf{Var}$ and $D \in S_X$.
Action: The application of the rule marks $\mathfrak{s}$ as *solved*.

Eager Extension:

Condition: This rule applies to $\mathfrak{s} = C_1 \sqcap \cdots \sqcap C_n \sqsubseteq^? D \in \Gamma$, if there is an index $i \in \{1, \ldots, n\}$, such that $C_i = X \in \mathsf{Var}$ and $\{C_1, \ldots, C_n\} \setminus \{X\} \subseteq S_X$.
Action: The application of the rule adds D to S_X. If this makes S cyclic, the rule application fails. Otherwise, Γ is expanded w.r.t. X and $\mathfrak{s}$ is marked as *solved*.

Decomposition:

Condition: This rule applies to $\mathfrak{s} = C_1 \sqcap \cdots \sqcap C_n \sqsubseteq^? \exists s.D \in \Gamma$, if there is at least one index $i \in \{1, \ldots, n\}$ with $C_i = \exists s.C$.
Action: The application of the rule chooses such an index i, adds $C \sqsubseteq^? D$ to Γ, expands Γ w.r.t. D if D is a variable, and marks $\mathfrak{s}$ as *solved*.

Extension:

Condition: This rule applies to $\mathfrak{s} = C_1 \sqcap \cdots \sqcap C_n \sqsubseteq^? D \in \Gamma$, if there is at least one index $i \in \{1, \ldots, n\}$ with $C_i \in \mathsf{Var}$.
Action: The application of the rule chooses such an index i and adds D to S_{C_i}. If this makes S cyclic, the rule application fails. Otherwise, Γ is expanded w.r.t. C_i and $\mathfrak{s}$ is marked as *solved*.

Figure 3 Rules for subsumptions.

the following:

1. **Eager rule application:** If eager rules are applicable to some unsolved subsumption or dissubsumption $\mathfrak{s}$ in Γ, apply an arbitrarily chosen one to $\mathfrak{s}$. If the rule application fails, return "failure".
2. **Nondeterministic rule application:** If no eager rule is applicable, let $\mathfrak{s}$ be an unsolved subsumption or dissubsumption in Γ. If one of the nondeterministic rules applies to $\mathfrak{s}$, choose one and apply it. If none of these rules apply to $\mathfrak{s}$ or the rule application fails, then return "failure".

Once all (dis)subsumptions in Γ are solved, return the substitution σ_S that is induced by the current assignment.

As with Algorithm 8, the choice which (dis)subsumption to consider next and which eager rule to apply is don't care nondeterministic, while the choice of which nondeterministic rule to apply and the choices inside the rules are don't know nondeterministic. Each of these latter choices may result in a different solution σ_S. All proof details for the following results can be found in [4].

▶ **Lemma 12.** *Every run of Algorithm 11 terminates in time polynomial in the size of* Γ_0.

Proof Sketch. We can show that each (dis)subsumption that is added by a rule or by expansion is either of the form $C \sqsubseteq^? D$ or $C \not\sqsubseteq^? D$, where $C, D \in \mathsf{At}$, or of the form $C_1 \sqcap \cdots \sqcap C_n \sqsubseteq^? E$, where $C_1 \sqcap \cdots \sqcap C_n$ is the left-hand side of a subsumption from the original problem Γ_0 and $E \in \mathsf{At}$. Obviously, there are only polynomially many such

Eager Top Solving:

Condition: This rule applies to $\mathfrak{s} = C \not\sqsubseteq^? \top \in \Gamma$.
Action: The rule application fails.

Eager Left Decomposition:

Condition: This rule applies to $\mathfrak{s} = C_1 \sqcap \cdots \sqcap C_n \not\sqsubseteq^? D \in \Gamma$ if $n = 0$ or $n > 1$, and $D \in \mathsf{At}_{\mathsf{nv}}$.
Action: The application of the rule marks $\mathfrak{s}$ as *solved* and, for each $i \in \{1, \ldots, n\}$, adds $C_i \not\sqsubseteq^? D$ to Γ and expands Γ w.r.t. C_i if C_i is a variable.

Eager Atomic Decomposition:

Condition: This rule applies to $\mathfrak{s} = C \not\sqsubseteq^? D \in \Gamma$ if $C, D \in \mathsf{At}_{\mathsf{nv}}$.
Action: The application of the rule applies the first case that matches $\mathfrak{s}$:

a) if C and D are ground and $C \sqsubseteq D$, then the rule application fails;

b) if C and D are ground and $C \not\sqsubseteq D$, then $\mathfrak{s}$ is marked as *solved*;

c) if C or D is a concept name, then $\mathfrak{s}$ is marked as *solved*;

d) if $C = \exists r.C'$ and $D = \exists s.D'$ with $r \neq s$, then $\mathfrak{s}$ is marked as *solved*;

e) if $C = \exists r.C'$ and $D = \exists r.D'$, then $C' \not\sqsubseteq^? D'$ is added to Γ, Γ is expanded w.r.t. C' if C' is a variable and D' is not a variable, and $\mathfrak{s}$ is marked as *solved*.

Local Extension:

Condition: This rule applies to $\mathfrak{s} = C \not\sqsubseteq^? X \in \Gamma$ if $X \in \mathsf{Var}$.
Action: The application of the rule chooses $D \in \mathsf{At}_{\mathsf{nv}}$ and adds D to S_X. If this makes S cyclic, the rule application fails. Otherwise, the new dissubsumption $C \not\sqsubseteq^? D$ is added to Γ, Γ is expanded w.r.t. X, Γ is expanded w.r.t. C if C is a variable, and $\mathfrak{s}$ is marked as *solved*.

Figure 4 New rules for dissubsumptions.

(dis)subsumptions. Additionally, each rule application solves at least one (dis)subsumption and takes at most polynomial time. ◀

To show *soundness* of the procedure, assume that a run of the algorithm terminates with success, i.e. all subsumptions and dissubsumptions are solved. Let $\hat{\Gamma}$ be the set of all subsumptions and dissubsumptions produced by this run, S be the final assignment, and σ_S the induced substitution (see Section 3). To show that σ_S solves $\hat{\Gamma}$, and hence also Γ_0, we use induction on the following order on (dis)subsumptions.

▶ **Definition 13.** Consider any (dis)subsumption $\mathfrak{s}$ of the form $C_1 \sqcap \cdots \sqcap C_n \sqsubseteq^? C_{n+1}$ or $C_1 \sqcap \cdots \sqcap C_n \not\sqsubseteq^? C_{n+1}$ in $\hat{\Gamma}$.

- We define $m(\mathfrak{s}) := (m_1(\mathfrak{s}), m_2(\mathfrak{s}))$, where
  - $m_1(\mathfrak{s}) := \emptyset$ if $\mathfrak{s}$ is ground; otherwise, $m_1(\mathfrak{s}) := \{X_1, \ldots, X_m\}$, where $\{X_1, \ldots, X_m\}$ is the multiset of all variables occurring in $C_1, \ldots, C_n, C_{n+1}$.
  - $m_2(\mathfrak{s}) := |\mathfrak{s}|$, where $|\mathfrak{s}|$ is the size of $\mathfrak{s}$, i.e. the number of symbols in $\mathfrak{s}$.
- The strict partial order $\succ$ on such pairs is the lexicographic order, where the second components are compared w.r.t. the usual order on natural numbers, and the first components are compared w.r.t. the multiset extension of $>_S$ [13].
- We extend $\succ$ to $\hat{\Gamma}$ by setting $\mathfrak{s}_1 \succ \mathfrak{s}_2$ iff $m(\mathfrak{s}_1) \succ m(\mathfrak{s}_2)$.

Since multiset extensions and lexicographic products of well-founded strict partial orders are again well-founded [13], $\succ$ is a well-founded strict partial order on $\hat{\Gamma}$. We can then use the fact that the (dis)subsumptions produced by Algorithm 11 are always smaller w.r.t. this

order than the (dis)subsumptions they were created from to prove the following lemma by well-founded induction over $\succ$.

▶ **Lemma 14.** *σ_S is a local solution of $\hat{\Gamma}$, and thus also of its subset Γ_0.*

To prove *completeness*, assume that σ is a local solution of Γ_0. We can show that σ can guide the choices of Algorithm 11 to obtain a local solution σ' of Γ_0 such that, for every variable X, we have $\sigma(X) \sqsubseteq \sigma'(X)$. The following invariants will be maintained throughout the run of the algorithm for the current set of (dis)subsumptions Γ and the current assignment S:

I. σ is a solution of Γ. II. For each $D \in S_X$, we have that $\sigma(X) \sqsubseteq \sigma(D)$.

By Lemma 1, chains of the form $\sigma(X_1) \sqsubseteq \sigma(\exists r_1.X_2)$, $\dots \sigma(X_{n-1}) \sqsubseteq \sigma(\exists r_{n-1}.X_n)$ with $X_1 = X_n$ are impossible, and thus invariant II implies that S is acyclic. Hence, if extending S during a rule application preserves this invariant, this extension will not cause the algorithm to fail. In [4] it is shown that

- the invariants are maintained by the operation of expanding Γ;
- the application of an eager rule never fails and maintains the invariants; and
- if $\mathfrak{s}$ is an unsolved (dis)subsumption of Γ to which no eager rule applies, then there is a nondeterministic rule that can be successfully applied to $\mathfrak{s}$ while maintaining the invariants.

This concludes the proof of correctness of Algorithm 11, which provides a more goal-directed way to solve local disunification problems than blindly guessing an assignment as described in Section 4.

▶ **Theorem 15.** *The flat disunification problem Γ_0 has a local solution iff there is a successful run of Algorithm 11 on Γ_0.*

6 Encoding local disunification into SAT

The following reduction to SAT is a generalization of the one for unification problems in [9]. We again consider a flat disunification problem Γ and the sets At, Var, and $\mathsf{At}_{\mathsf{nv}}$ as in Section 3. Since we are restricting our considerations to *local* solutions, we can without loss of generality assume that the sets N_{v}, N_{c}, and N_{R} contain exactly the variables, constants, and role names occurring in Γ. To further simplify the reduction, we assume in the following that all flat dissubsumptions in Γ are of the form $X \not\sqsubseteq^? Y$ for variables X, Y. This is without loss of generality, which can be shown using a transformation similar to the flattening rules from Section 4.

The translation into SAT uses the propositional variables $[C \sqsubseteq D]$ for all $C, D \in \mathsf{At}$. The SAT problem consists of a set of clauses $\mathsf{Cl}(\Gamma)$ over these variables that express properties of (dis)subsumption in $\mathcal{EL}$ and encode the elements of Γ. The intuition is that a satisfying valuation of $\mathsf{Cl}(\Gamma)$ induces a local solution σ of Γ such that $\sigma(C) \sqsubseteq \sigma(D)$ holds whenever $[C \sqsubseteq D]$ is true under the valuation. The solution σ is constructed by first extracting an acyclic assignment S out of the satisfying valuation and then computing $\sigma := \sigma_S$. We additionally introduce the variables $[X > Y]$ for all $X, Y \in \mathsf{N}_{\mathsf{v}}$ to ensure that the generated assignment S is indeed acyclic. This is achieved by adding clauses to $\mathsf{Cl}(\Gamma)$ that express that $>_S$ is a strict partial order, i.e. irreflexive and transitive.

Finally, we use the auxiliary variables $p_{C,X,D}$ for all $X \in \mathsf{N}_{\mathsf{v}}$, $C \in \mathsf{At}$, and $D \in \mathsf{At}_{\mathsf{nv}}$ to express the restrictions imposed by dissubsumptions of the form $C \not\sqsubseteq^? X$ in clausal form. More precisely, whenever $[C \sqsubseteq X]$ is false for some $X \in \mathsf{N}_{\mathsf{v}}$ and $C \in \mathsf{At}$, then the

dissubsumption $\sigma(C) \not\sqsubseteq \sigma(X)$ should hold. By Lemma 2, this means that we need to find an atom $D \in \mathsf{At_{nv}}$ that is a top-level atom of $\sigma(X)$ and satisfies $\sigma(C) \not\sqsubseteq \sigma(D)$. This is enforced by making the auxiliary variable $p_{C,X,D}$ true, which makes $[X \sqsubseteq D]$ true and $[C \sqsubseteq D]$ false (see Definition 167).

▶ **Definition 16.** The set $\mathsf{CI}(\Gamma)$ contains the following propositional clauses:

(I) *Translation of* Γ.

- **a.** For every subsumption $C_1 \sqcap \cdots \sqcap C_n \sqsubseteq^? D$ in Γ with $D \in \mathsf{At_{nv}}$:
  $\to [C_1 \sqsubseteq D] \vee \cdots \vee [C_n \sqsubseteq D]$
- **b.** For every subsumption $C_1 \sqcap \cdots \sqcap C_n \sqsubseteq^? X$ in Γ with $X \in \mathsf{N_v}$, and every $E \in \mathsf{At_{nv}}$:
  $[X \sqsubseteq E] \to [C_1 \sqsubseteq E] \vee \cdots \vee [C_n \sqsubseteq E]$
- **c.** For every dissubsumption $X \not\sqsubseteq^? Y$ in Γ: $[X \sqsubseteq Y] \to$

(IV) *Properties of subsumptions between non-variable atoms.*

- **a.** For every $A \in \mathsf{N_c}$: $\to [A \sqsubseteq A]$
- **b.** For every $A, B \in \mathsf{N_c}$ with $A \neq B$: $[A \sqsubseteq B] \to$
- **c.** For every $\exists r.A, \exists s.B \in \mathsf{At_{nv}}$ with $r \neq s$: $[\exists r.A \sqsubseteq \exists s.B] \to$
- **d.** For every $A \in \mathsf{N_c}$ and $\exists r.B \in \mathsf{At_{nv}}$:
  $[A \sqsubseteq \exists r.B] \to$ and $[\exists r.B \sqsubseteq A] \to$
- **e.** For every $\exists r.A, \exists r.B \in \mathsf{At_{nv}}$:
  $[\exists r.A \sqsubseteq \exists r.B] \to [A \sqsubseteq B]$ and $[A \sqsubseteq B] \to [\exists r.A \sqsubseteq \exists r.B]$

(VI) *Transitivity of subsumption.*
For every $C_1, C_2, C_3 \in \mathsf{At}$: $[C_1 \sqsubseteq C_2] \wedge [C_2 \sqsubseteq C_3] \to [C_1 \sqsubseteq C_3]$

(VII) *Dissubsumptions of the form* $C \not\sqsubseteq^? X$ *with a variable* X.
For every $C \in \mathsf{At}$, $X \in \mathsf{N_v}$:
$\to [C \sqsubseteq X] \vee \bigvee_{D \in \mathsf{At_{nv}}} p_{C,X,D}$,
and additionally for every $D \in \mathsf{At_{nv}}$:
$p_{C,X,D} \to [X \sqsubseteq D]$ and $p_{C,X,D} \wedge [C \sqsubseteq D] \to$

(VIII) *Properties of* $>$.

- **a.** For every $X \in \mathsf{N_v}$: $[X > X] \to$
- **b.** For every $X, Y, Z \in \mathsf{N_v}$: $[X > Y] \wedge [Y > Z] \to [X > Z]$
- **c.** For every $X, Y \in \mathsf{N_v}$ and $\exists r.Y \in \mathsf{At}$: $[X \sqsubseteq \exists r.Y] \to [X > Y]$

The main difference to the encoding in [9] (apart from the fact that we consider (dis)subsumptions here instead of equivalences) lies in the clauses 7 that ensure the presence of a non-variable atom D that solves the dissubsumption $C \not\sqsubseteq^? X$ (cf. Lemma 2). We also need some additional clauses in 4 to deal with dissubsumptions. It is easy to see that $\mathsf{CI}(\Gamma)$ can be constructed in time cubic in the size of Γ (due to the clauses in 6 and 2).

To show *soundness* of the reduction, let τ be a valuation of the propositional variables that satisfies $\mathsf{CI}(\Gamma)$. We define the assignment S^τ as follows:

$$S^\tau_X := \{D \in \mathsf{At_{nv}} \mid \tau([X \sqsubseteq D]) = 1\}.$$

In [4] it is shown that $X >_{S^\tau} Y$ implies $\tau([X > Y]) = 1$ and that this implies irreflexivity of $>_{S^\tau}$. This in particular shows that S^τ is acyclic. In the following, let σ_τ denote the substitution σ_{S^τ} induced by S^τ. In [4] it is shown that σ_τ is a solution of Γ by proving that for all atoms $C, D \in \mathsf{At}$ it holds that $\tau([C \sqsubseteq D]) = 1$ iff $\sigma_\tau(C) \sqsubseteq \sigma_\tau(D)$.

Since σ_τ is obviously local, this suffices to show soundness of the reduction.

▶ **Lemma 17.** *If* $\mathsf{CI}(\Gamma)$ *is solvable, then* Γ *has a local solution.*

To show *completeness*, let σ be a local solution of Γ and $>_\sigma$ the resulting partial order on $\mathsf{N_v}$, defined as follows for all $X, Y \in \mathsf{N_v}$:

$$X >_\sigma Y \text{ iff } \sigma(X) \sqsubseteq \exists r_1. \ldots \exists r_n. \sigma(Y) \text{ for some } r_1, \ldots, r_n \in \mathsf{N_R} \text{ with } n \geq 1.$$

Note that $>_\sigma$ is irreflexive since $X >_\sigma X$ is impossible by Lemma 1, and it is transitive since $\sqsubseteq$ is transitive and closed under applying existential restrictions on both sides. Thus, $>_\sigma$ is a strict partial order. We define a valuation τ_σ as follows for all $C, D \in \mathsf{At}$, $E \in \mathsf{At_{nv}}$, and $X, Y \in \mathsf{N_v}$:

$$\tau_\sigma([C \sqsubseteq D]) := \begin{cases} 1 & \text{if } \sigma(C) \sqsubseteq \sigma(D) \\ 0 & \text{otherwise} \end{cases} \qquad \tau_\sigma([X > Y]) := \begin{cases} 1 & \text{if } X >_\sigma Y \\ 0 & \text{otherwise} \end{cases}$$

$$\tau_\sigma(p_{C,X,E}) := \begin{cases} 1 & \text{if } \sigma(X) \sqsubseteq \sigma(E) \text{ and } \sigma(C) \not\sqsubseteq \sigma(E) \\ 0 & \text{otherwise} \end{cases}$$

In [4] it is proved that τ_σ satisfies $\mathsf{CI}(\Gamma)$, which shows completeness of the reduction.

▶ **Lemma 18.** *If Γ has a local solution, then $\mathsf{CI}(\Gamma)$ is solvable.*

This completes the proof of the correctness of the translation presented in Definition 16, which provides us with a reduction of local disunification (and thus also of dismatching) to SAT. This SAT reduction has been implemented in our prototype system UEL,[4] which uses SAT4J[5] as external SAT solver. First experiments show that dismatching is indeed helpful for reducing the number and the size of unifiers. The runtime performance of the solver for dismatching problems is comparable to the one for pure unification problems.

7 Related and future work

Since Description Logics and Modal Logics are closely related [26], results on unification in one of these two areas carry over to the other one. In Modal Logics, unification has mostly been considered for expressive logics with all Boolean operators [19, 20, 25]. An important open problem in the area is the question whether unification in the basic modal logic K, which corresponds to the DL $\mathcal{ALC}$, is decidable. It is only known that relatively minor extensions of K have an undecidable unification problem [27]. Disunification also plays an important role in Modal Logics since it is basically the same as the admissibility problem for inference rules [15, 22, 24] (see [4] for details).

Regarding future work, we want to investigate the decidability and complexity of general disunification in $\mathcal{EL}$, and consider also the case where non-ground solutions are allowed. From a more practical point of view, we plan to implement also the goal-oriented algorithm for local disunification, and to evaluate the performance of both presented algorithms on real-world problems.

References

1 Franz Baader, Stefan Borgwardt, Julian Alfredo Mendez, and Barbara Morawska. UEL: Unification solver for $\mathcal{EL}$. In *Proc. DL'12*, volume 846 of *CEUR-WS*, pages 26–36, 2012.

[4] version 1.3.0, available at `http://uel.sourceforge.net/`
[5] `http://www.sat4j.org/`

2 Franz Baader, Stefan Borgwardt, and Barbara Morawska. Computing minimal $\mathcal{EL}$-unifiers is hard. In *Proc. AiML'12*, 2012.

3 Franz Baader, Stefan Borgwardt, and Barbara Morawska. A goal-oriented algorithm for unification in $\mathcal{EL}$ w.r.t. cycle-restricted TBoxes. In *Proc. DL'12*, volume 846 of *CEUR-WS*, pages 37–47, 2012.

4 Franz Baader, Stefan Borgwardt, and Barbara Morawska. Dismatching and local disunification in $\mathcal{EL}$. LTCS-Report 15-03, Chair for Automata Theory, TU Dresden, Germany, 2015. See `http://lat.inf.tu-dresden.de/research/reports.html`.

5 Franz Baader, Sebastian Brandt, and Carsten Lutz. Pushing the $\mathcal{EL}$ envelope further. In *Proc. OWLED'08*, 2008.

6 Franz Baader, Diego Calvanese, Deborah McGuinness, Daniele Nardi, and Peter F. Patel-Schneider, editors. *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press, 2003.

7 Franz Baader, Ralf Küsters, Alex Borgida, and Deborah L. McGuinness. Matching in description logics. *J. Logic Comput.*, 9(3):411–447, 1999.

8 Franz Baader, Ralf Küsters, and Ralf Molitor. Computing least common subsumers in description logics with existential restrictions. In *Proc. IJCAI'99*, pages 96–101. Morgan Kaufmann, 1999.

9 Franz Baader and Barbara Morawska. SAT encoding of unification in $\mathcal{EL}$. In *Proc. LPAR'10*, volume 6397 of *LNCS*, pages 97–111. Springer, 2010.

10 Franz Baader and Barbara Morawska. Unification in the description logic $\mathcal{EL}$. *Log. Meth. Comput. Sci.*, 6(3), 2010.

11 Franz Baader and Barbara Morawska. Matching with respect to general concept inclusions in the description logic $\mathcal{EL}$. In *Proc. KI'14*, volume 8736 of *LNCS*, pages 135–146. Springer, 2014.

12 Franz Baader and Paliath Narendran. Unification of concept terms in description logics. *J. Symb. Comput.*, 31(3):277–305, 2001.

13 Franz Baader and Tobias Nipkow. *Term Rewriting and All That*. Cambridge University Press, 1999.

14 Franz Baader and Alexander Okhotin. Solving language equations and disequations with applications to disunification in description logics and monadic set constraints. In *Proc. LPAR'12*, volume 7180 of *LNCS*, pages 107–121. Springer, 2012.

15 Sergey Babenyshev, Vladimir V. Rybakov, Renate Schmidt, and Dmitry Tishkovsky. A tableau method for checking rule admissibility in $S4$. In *Proc. M4M-6*, 2009.

16 Sebastian Brandt. Polynomial time reasoning in a description logic with existential restrictions, GCI axioms, and—what else? In *Proc. ECAI'04*, pages 298–302, 2004.

17 Wray L. Buntine and Hans-Jürgen Bürckert. On solving equations and disequations. *J. of the ACM*, 41(4):591–629, 1994.

18 Hubert Comon. Disunification: A survey. In J.-L. Lassez and G. Plotkin, editors, *Computational Logic: Essays in Honor of Alan Robinson*, pages 322–359. MIT Press, 1991.

19 Silvio Ghilardi. Unification through projectivity. *J. Logic and Computation*, 7(6):733–752, 1997.

20 Silvio Ghilardi. Unification in intuitionistic logic. *J. Logic and Computation*, 64(2):859–880, 1999.

21 Ian Horrocks, Peter F. Patel-Schneider, and Frank van Harmelen. From SHIQ and RDF to OWL: The making of a web ontology language. *J. Web Sem.*, 1(1):7–26, 2003.

22 Rosalie Iemhoff and George Metcalfe. Proof theory for admissible rules. *Ann. Pure Appl. Logic*, 159(1-2):171–186, 2009.

23 Ralf Küsters. Chapter 6: Matching. In *Non-Standard Inferences in Description Logics*, volume 2100 of *LNCS*, pages 153–227. Springer, 2001.

24 Vladimir V. Rybakov. *Admissibility of logical inference rules*, volume 136 of *Studies in Logic and the Foundations of Mathematics*. North-Holland Publishing Co., Amsterdam, 1997.

25 Vladimir V. Rybakov. Multi-modal and temporal logics with universal formula - reduction of admissibility to validity and unification. *J. Logic and Computation*, 18(4):509–519, 2008.

26 Klaus Schild. A correspondence theory for terminological logics: Preliminary report. In *Proc. IJCAI'91*, pages 466–471, 1991.

27 Frank Wolter and Michael Zakharyaschev. Undecidability of the unification and admissibility problems for modal and description logics. *ACM Trans. Comput. Log.*, 9(4), 2008.