Unification of Concept Terms in Description Logics

Franz Baader Paliath Narendran

LTCS-Report 97-02

Abridged versions have appeared in *Proc. UNIF'97* and *DL'97.*

Unification of Concept Terms in Description Logics

Franz Baader[*]
LuFg Theoretical Computer Science,
RWTH Aachen
Ahornstraße 55, 52074 Aachen, Germany
e-mail: baader@informatik.rwth-aachen.de

Paliath Narendran[†]
Department of Computer Science
State University of New York at Albany
Albany, NY 12222, USA
e-mail: dran@cs.albany.edu

Abstract

Unification of concept terms is a new kind of inference problem for Description Logics, which extends the equivalence problem by allowing to replace certain concept names by concept terms before testing for equivalence. We show that this inference problem is of interest for applications, and present first decidability and complexity results for a small concept description language.

1 Introduction

Knowledge representation languages based on Description Logics (DL languages) can be used to represent the terminological knowledge of an application domain in a structured and formally well-understood way [8, 3]. With the help of these languages, the important notions of the domain can be described by *concept terms*, i.e., expressions that are built from atomic concepts (unary predicates) and atomic roles (binary predicates) using the concept constructors provided

[*]Partially supported by the EC Working Group CCL II.
[†]Partially supported by the NSF grants CCR-9404930 and INT-9401087.

by the DL language. The atomic concepts and concept terms represent sets of individuals, whereas roles represent binary relations between individuals. For example, using the atomic concept Woman and the atomic role child, the concept of all *women having only daughters* (i.e., women such that all their children are again women) can be represented by the concept term

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

Knowledge representation systems based on Description Logics provide their users with various inference capabilities that allow them to deduce implicit knowledge from the explicitly represented knowledge. For instance, the *subsumption* algorithm allows one to determine subconcept-superconcept relationships: C is subsumed by D ($C \sqsubseteq D$) iff all instances of C are also instances of D, i.e., the first term is always interpreted as a subset of the second term. For example, the concept term Woman obviously subsumes the concept term $\mathsf{Woman} \sqcap \forall \mathsf{child}.\mathsf{Woman}$. 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* ($C \equiv D$) iff they subsume each other, i.e., iff they always represent the same set of individuals. For example, the terms $\mathsf{Woman} \sqcap \forall \mathsf{child}.\mathsf{Woman}$ and $(\forall \mathsf{child}.\mathsf{Woman}) \sqcap \mathsf{Woman}$ are equivalent since $\sqcap$ is interpreted as set intersection, which is obviously commutative. 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. Unification of concept terms extends this inference capability by allowing to replace certain concept names by concept terms before testing for equivalence.

The first motivation for considering unification of concept terms comes from an application in chemical process engineering [5]. In this application, the DL system is used to support the design of a large terminology of concepts describing parts of chemical plants as well as processes that take place in these plants. Since several knowledge engineers are involved in defining new concepts, and since this knowledge acquisition process takes rather long (several years), it happens that the same (intuitive) concept is introduced several times, often with slightly differing descriptions. Our goal was to use the reasoning capabilities of the DL system (in particular, testing for equivalence of concept terms) to support avoiding this kind of redundancy. 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 only daughters*[1] by the concept term

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

[1]We use an example from the family domain since examples from process engineering would require too much explanation.

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

$$\mathsf{Female} \sqcap \mathsf{Human} \sqcap \forall \mathsf{child}.(\mathsf{Female} \sqcap \mathsf{Human})$$

are not equivalent, but they are meant to represent the same concept. The two terms can obviously be made equivalent by replacing the atomic concept Woman in the first term by the concept term $\mathsf{Female} \sqcap \mathsf{Human}$. This leads us to *unification of concept terms*, i.e., the question whether two concept terms C, D can be made equivalent by applying an appropriate substitution σ, where a substitution replaces (some of the) atomic concepts by concept terms. A substitution is a unifier of C, D iff $\sigma(C) \equiv \sigma(D)$. 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.

Another motivation for considering unification of concept terms comes from the work of Borgida and McGuinness [7], who introduce matching of concept terms (of the DL language used by the CLASSIC system) modulo subsumption: for given concept terms C and D they ask for a substitution σ such that $C \sqsubseteq \sigma(D)$. More precisely, they are interested in finding "minimal" substitutions for which this is the case, i.e., σ should satisfy the property that there does not exist another substitution δ such that $C \sqsubseteq \delta(D) \sqsubset \sigma(D)$. Since $C \sqsubseteq D$ iff $C \sqcap D \equiv C$, this matching problem can be reduced to a unification problem.

In the following, we consider the unification problem for a rather small DL language, called $\mathcal{FL}_0$ in the literature [2]. We shall see that this problem can be viewed as a unification problem modulo an appropriate equational theory: the theory ACUIh of a binary associative, commutative, and idempotent function symbol with a unit and several homomorphisms. This theory turns out to be a so-called commutative (or monoidal) theory [1, 13, 4], in which unification can be reduced to solving equations in a corresponding semiring, which in the case of ACUIh is the polynomial semiring (in non-commuting indeterminates) over the Boolean semiring.[2] The problem of solving linear equations over this semiring can in turn be reduced to a certain formal language problem, which can be solved using automata on finite trees. This provides us with an exponential time algorithm for deciding solvability of ACUIh-unification problems, and thus also for unification of concept terms of the DL language $\mathcal{FL}_0$. It can also be shown that the problem is at least PSPACE-hard. Finally, we consider the matching problem for $\mathcal{FL}_0$-concept terms, and show that it is decidable in polynomial time.

[2]Note that this is not the Boolean ring (with operations *conjunction* and *ex-or*), but the Boolean semiring (with operations *conjunction* and *disjunction*).

2 The DL language $\mathcal{FL}_0$

In this section, we introduce syntax and semantics of the knowledge representation language $\mathcal{FL}_0$, and give a formal definition of subsumption, equivalence, and unification of concept terms.

Definition 1 Let $\mathcal{C}$ and $\mathcal{R}$ be disjoint finite sets, the set of *atomic concepts* and the set of *atomic roles*. The set of all $\mathcal{FL}_0$*-concept terms* over $\mathcal{C}$ and $\mathcal{R}$ is inductively defined as follows:

- Every element of $\mathcal{C}$ is a concept term (atomic concept).
- The symbol $\top$ is a concept term (top concept).
- If C and D are concept terms, then $C \sqcap D$ are concept terms (concept conjunction).
- If C is a concept term and R is an atomic role (i.e., $R \in \mathcal{R}$), then $\forall R.C$ is a concept term (value restriction).

The following definition provides a model-theoretic semantics for $\mathcal{FL}_0$:

Definition 2 An *interpretation* I consists of a nonempty set Δ^I, the domain of the interpretation, and an interpretation function that assigns to every atomic concept $A \in \mathcal{C}$ a set $A^I \subseteq \Delta^I$, and to every atomic role $R \in \mathcal{R}$ a binary relation $R^I \subseteq \Delta^I \times \Delta^I$. The interpretation function is extended to complex concept terms as follows:

$$\begin{array}{rcl}
\top^I & := & \Delta^I, \\
(C \sqcap D)^I & := & C^I \cap D^I, \\
(\forall R.C)^I & := & \{d \in \Delta^I \mid \forall e \in \Delta^I\colon (d,e) \in R^I \rightarrow e \in C^I\}.
\end{array}$$

Based on this semantics, subsumption and equivalence of concept terms is defined as follows: Let C and D be $\mathcal{FL}_0$-concept terms.

- C is *subsumed* by D ($C \sqsubseteq D$) iff $C^I \subseteq D^I$ for all interpretations I.
- C is *equivalent* to D ($C \equiv D$) iff $C^I = D^I$ for all interpretations I.

In order to define unification of concept terms, we must first introduce the notion of a substitution operating on concept terms. To this purposes, we partition the set of atomic concepts into a set $\mathcal{C}_v$ of concept variables (which may be replaced by substitutions) and a set $\mathcal{C}_c$ of concept constants (which must not be replaced by substitutions). Intuitively, $\mathcal{C}_v$ are the atomic concepts that have possibly been given another name or been specified in more detail in another concept term describing the same notion. The elements of $\mathcal{C}_c$ 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 $\mathcal{C}_v$ into the set of all $\mathcal{FL}_0$-concept terms. This mapping is extended to concept terms in the obvious way, i.e.,

- $\sigma(A) := A$ for all $A \in \mathcal{C}_c$,
- $\sigma(\top) := \top$,
- $\sigma(C \sqcap D) := \sigma(C) \sqcap \sigma(D)$, and
- $\sigma(\forall R.C) := \forall R.\sigma(C)$.

Definition 3 Let C and D be $\mathcal{FL}_0$-concept terms. The substitution σ is a *unifier* of C and D iff $\sigma(C) \equiv \sigma(D)$. In this case, the concept terms C and D are called *unifiable*.

For example, if $A \in \mathcal{C}_c$ and $X, Y \in \mathcal{C}_v$, then $\sigma = \{X \mapsto A \sqcap \forall S.A, Y \mapsto \forall R.A\}$ is a unifier of the concept terms $\forall R.\forall R.A \sqcap \forall R.X$ and $Y \sqcap \forall R.Y \sqcap \forall R.\forall S.A$.

3 The equational theory ACUIh

Unification of $\mathcal{FL}_0$-concept terms can be reduced to the well-known notion of *unification modulo an equational theory*, which allows us to employ methods and results developed in unification theory [6].

First, we show how concept terms can be translated into terms over an appropriate signature $\Sigma_{\mathcal{R}}$, which consists of a binary function symbol $\wedge$, a constant symbol T, and for each $R \in \mathcal{R}$ a unary function symbol h_R. In addition, every element of $\mathcal{C}_v$ is considered as variable symbol, and every element of $\mathcal{C}_c$ as a (free) constant. The translation function τ is defined by induction on the structure of concept terms:

- $\tau(A) := A$ for all $A \in \mathcal{C}$,
- $\tau(\top) := \mathsf{T}$,
- $\tau(C \sqcap D) := \tau(C) \wedge \tau(D)$, and
- $\tau(\forall R.C) := h_R(\tau(C))$.

Obviously, τ is a bijective mapping between the set of all $\mathcal{FL}_0$-concept terms (with atomic concepts from $\mathcal{C} = \mathcal{C}_v \cup \mathcal{C}_c$ and atomic roles from $\mathcal{R}$) and the set of all terms over the signature $\Sigma_\mathcal{R}$ built using variables from $\mathcal{C}_v$ and free constants from $\mathcal{C}_c$.

The equational theory ACUIh that axiomatizes equivalence of $\mathcal{FL}_0$-concept terms consists of the following identities:

$$\begin{array}{rcl} \text{ACUIh} & := & \{\ (x \wedge y) \wedge z = x \wedge (y \wedge z),\ x \wedge y = y \wedge x,\ x \wedge x = x,\ x \wedge \mathsf{T} = x\ \} \\ & \cup & \{\ h_R(x \wedge y) = h_R(x) \wedge h_R(y),\ h_R(\mathsf{T}) = \mathsf{T} \mid R \in \mathcal{R}\ \}. \end{array}$$

Let $=_{\text{ACUIh}}$ denote the congruence relation on terms induced by ACUIh, i.e., $s =_{\text{ACUIh}} t$ holds iff s can be transformed into t using identities from ACUIh.

Lemma 4 *Let C and D be $\mathcal{FL}_0$-concept terms. Then*

$$C \equiv D \ \ \textit{iff}\ \ \tau(C) =_{\text{ACUIh}} \tau(D).$$

Proof. The if-direction is an easy consequence of the semantics of $\mathcal{FL}_0$-concept terms. In fact, since concept conjunction is interpreted as set intersection, it inherits associativity, commutativity, and idempotency (modulo equivalence) from set intersection. In addition, it is easy to see that $C \sqcap \top \equiv C$, $\forall R.\top \equiv \top$, and $\forall R.(C \sqcap D) \equiv (\forall R.C) \sqcap (\forall R.D)$ hold for arbitrary concept terms C and D.

To show the only-if-direction, we first represent $\mathcal{FL}_0$-concept terms in a certain normal form. Using the equivalences noted in the proof of the if-direction, any $\mathcal{FL}_0$-concept term can be transformed into an equivalent $\mathcal{FL}_0$-concept term C' that is either $\top$ or a (nonempty) conjunction of terms of the form $\forall R_1.\cdots\forall R_n.A$ for $n \geq 0$ (not necessarily distinct) role names $R_1, \ldots, R_n$ and a concept name $A \neq \top$. Since the transformation into this normal form uses only identities from ACUIh, we have $\tau(C) =_{\text{ACUIh}} \tau(C')$.

Now, assume that $\tau(C) \neq_{\text{ACUIh}} \tau(D)$. Consequently, the corresponding normal forms C', D' also satisfy $\tau(C') \neq_{\text{ACUIh}} \tau(D')$. This implies that one of these two normal forms contains a conjunct $\forall R_1.\cdots\forall R_n.A$ (for $n \geq 0$ and $A \neq \top$) that does not occur in the other normal form. We assume without loss of generality that this conjunct occurs in C', but not in D'.

We use this conjunct to construct an interpretation I such that $C'^I \neq D'^I$, which implies $C' \not\equiv D'$ and thus $C \not\equiv D$. The domain Δ^I of this interpretation consists of $n+1$ distinct individuals $d_0, \ldots, d_n$. The interpretation of the concept names is given by $B^I := \Delta^I$ for all names $B \neq A$, and $A^I := \Delta^I \setminus \{d_n\}$. Finally, the role names are interpreted as $S^I := \{(d_{i-1}, d_i) \mid S = R_i\}$. As an obvious consequence of this definition, we obtain $d_0 \notin (\forall R_1.\cdots\forall R_n.A)^I$, and thus $d_0 \notin C'^I = C^I$. On the other hand, $d_0 \in \top^I$ and $d_0 \in (\forall S_1.\cdots\forall S_m.B)^I$ for all concept terms of the form $\forall S_1.\cdots\forall S_m.B$ that are different to $\forall R_1.\cdots\forall R_n.A$. Consequently, $d_0 \in D'^I = D^I$. □

As a consequence of this lemma, the concept terms C and D are unifiable iff the corresponding terms $\tau(C)$ and $\tau(D)$ are unifiable modulo ACUIh. For example, the concept terms $\forall R.\forall R.A \sqcap \forall R.X$ and $Y \sqcap \forall R.Y \sqcap \forall R.\forall S.A$ are translated into the terms $t_1 := h_R(h_R(a)) \wedge h_R(x)$ and $t_2 := y \wedge h_R(y) \wedge h_R(h_S(a))$, and the substitution $\sigma' := \{x \mapsto a \wedge h_S(a), y \mapsto h_R(a)\}$ is an ACUIh-unifier of these terms, i.e., $\sigma(t_1) =_{\text{ACUIh}} \sigma(t_2)$.[3]

In unification theory, one usually considers unification problems that consist of a finite set of term equations $\Gamma = \{s_1 =^? t_1, ..., s_n =^? t_n\}$ rather than a single equation $s =^? t$. For ACUIh, we can show that the system Γ has an ACUIh-unifier iff the single equation

$$h_{R_1}(s_1) \wedge \cdots \wedge h_{R_n}(s_n) =^? h_{R_1}(t_1) \wedge \cdots \wedge h_{R_n}(t_n)$$

has an ACUIh-unifier, provided that $h_{R_1}, \ldots, h_{R_n}$ are n distinct unary function symbols in $\Sigma_{\mathcal{R}}$. Thus, solving systems of equations is equivalent to solving a single equation in this case. The correctness of this reduction is an easy consequence of the following lemma.

Lemma 5 *Let $C_1, \ldots, C_n, D_1, \ldots, D_n$ be $\mathcal{FL}_0$-concept terms, and $R_1, \ldots, R_n$ be n pairwise distinct role names. Then*

$$\forall R_1.C_1 \sqcap \cdots \sqcap \forall R_n.C_n \equiv \forall R_1.D_1 \sqcap \cdots \sqcap \forall R_n.D_n \quad \textit{iff} \quad C_1 \equiv D_1, \ldots, C_n \equiv D_n.$$

Proof. The if-direction of the lemma is trivially satisfied. In order to show the only-if-direction, assume that $C_i \not\equiv D_i$ for some $i, 1 \leq i \leq n$. Thus, there exists an interpretation I such that $C_i^I \neq D_i^I$. We assume (without loss of generality) that there exists $d \in \Delta^I$ such that $d \in C_i^I \setminus D_i^I$. We extend the interpretation I to an interpretation I' by defining $\Delta^{I'} := \Delta^I \cup \{e\}$, where $e \notin \Delta^I$. The interpretation in I' of all concept names and of all role names different from R_i coincides with their interpretation in I. Finally, $R_i^{I'} := R_i^I \cup \{(e, d)\}$. By construction of I', we have $e \notin (\forall R_i.D_i)^{I'}$. In addition, $e \in (\forall R_j.C_j)^{I'}$ for all $j, 1 \leq j \leq n$. Thus, $e \in (\forall R_1.C_1 \sqcap \cdots \sqcap \forall R_n.C_n)^{I'}$, but $e \notin (\forall R_1.D_1 \sqcap \cdots \sqcap \forall R_n.D_n)^{I'}$, which shows that the two terms are not equivalent. □

For readers that are familiar with unification theory, we want to point out that the *unification type* of ACUIh has already been determined in [1]: ACUIh is of type zero, which means that ACUIh-unification problems need not have a minimal complete set of ACUIh-unifiers. In particular, this implies that there exist ACUIh-unification problems for which the set of all unifiers cannot be represented as the set of all instances of a *finite* set of unifiers. For our application in knowledge representation, this result seems not to be very relevant since we

[3]To distinguish between concept names in concept terms and variable and constant symbols in terms over $\Sigma_{\mathcal{R}}$, we use upper-case letters for concept names and the corresponding lower-case letters for constants and variables.

are mainly interested in ground solutions of the unification problems, i.e., in unifiers that do not introduce concept variables. In the present paper, we restrict our attention to the decision problem, i.e., the problem of deciding solvability of ACUIh-unification problems. This problem has not been considered in [1]. In the following, we will show that this problem is decidable. Note that unification in the closely related theory ACUh, which is obtained from ACUIh by removing the axiom $x \wedge x = x$, has been shown to be undecidable [12].

4 Reducing ACUIh-unification to solving linear equations

The purpose of this section is to show that ACUIh-unification can be reduced to solving the following formal language problem: Let $S_0, S_1, \ldots, S_n, T_0, T_1, \ldots, T_n$ be finite sets of words over the alphabet of all role names. We consider the equation

$$S_0 \cup S_1X_1 \cup \cdots \cup S_nX_n = T_0 \cup T_1X_1 \cup \cdots \cup T_nX_n. \quad (*)$$

A solution of this equation assigns finite sets of words to the variables X_i such that the equation holds. The operation $\cup$ stands for set union and expressions like S_iX_i for element-wise concatenation of sets of words; e.g., $\{SR, S\}\{R, RR\} = \{SRR, SR, SRRR\}$.

This reduction can either be obtained directly, or as a consequence of results from unification theory. In the following, we consider both approaches.

4.1 Commutative theories and semirings

The theory ACUIh is a so-called commutative theory [1], for which solving unification problems can be reduced to solving systems of linear equations over a corresponding semiring [13, 4]. Conversely, every system of linear equations over this semiring corresponds to a unification problem.

Let us first consider the theory ACUI, which consists of the axioms specifying that $\wedge$ is associative, commutative and idempotent, and that $\top$ is a unit element with respect to $\wedge$. The corresponding semiring is obtained by considering the ACUI-free algebra in one generator (say x), and then taking the set of all endomorphisms of this algebra. Since the ACUI-free algebra generated by x consists of two congruence classes, with representatives x and $\top$, respectively, there are two possible endomorphisms: 0, which is defined by $x \mapsto \top$, and 1, which is defined by $x \mapsto x$. The multiplication $\cdot$ of this semiring is just composition of endomorphisms, and the addition $+$ is obtained by applying $\wedge$ argument-wise, e.g., $(1 + 0)(x) := 1(x) \wedge 0(x) = x \wedge \top =_{\text{ACUI}} x = 1(x)$. It is easy to see that $+$

behaves like disjunction and $\cdot$ like conjunction on the truth values 0 and 1. Thus, the semiring corresponding to ACUI is the Boolean semiring.

As shown in [4], adding homomorphisms to a commutative theory corresponds to going to a polynomial semiring (in non-commuting indeterminates) on the semiring side, where every indeterminate corresponds to one of the homomorphisms. Thus, the semiring $\mathcal{S}_{\mathrm{ACUIh}}$ corresponding to ACUIh is the polynomial semiring (in $|\mathcal{R}|$ non-commuting indeterminates) over the Boolean semiring. Let Δ be the set of these indeterminates (which are w.l.o.g. just the role names). Monomials in $\mathcal{S}_{\mathrm{ACUIh}}$ are simply words over the alphabet Δ, and since the addition operation in the semiring is idempotent, the elements of the semiring can be seen as finite sets of words over this alphabet. Thus, the semiring $\mathcal{S}_{\mathrm{ACUIh}}$ can be described as follows:

- its elements are finite sets of words (over the alphabet Δ of all role names),
- its addition operation is union of sets with the empty set $\emptyset$ as unit,
- its multiplication operation is element-wise concatenation with the set $\{\varepsilon\}$ consisting of the empty word as unit.

As described in [13, 4], ACUIh-unification problems (consisting w.l.o.g. of a single equation) are now translated into (inhomogeneous) linear equations over this semiring. According to the above description of $\mathcal{S}_{\mathrm{ACUIh}}$, these are just equations of the form $(*)$. In the next section we explain in more detail how these equations can be obtained from a given unification problem.

4.2 A direct reduction to linear equations

The fact that equivalence of $\mathcal{FL}_0$-concept terms can be axiomatized by a *commutative* equational theory has allowed us to employ known results from unification theory about the connection between unification modulo commutative theories and solving linear equations in semirings. In this subsection, we show how the linear equations corresponding to a unification problem between $\mathcal{FL}_0$-concept terms can be obtained directly, without the detour through equational unification. On the one hand, this may be helpful for readers not familiar with the relevant literature in unification theory. On the other hand, it opens the possibility to use a similar approach for concept languages for which equivalence cannot be axiomatized by a commutative theory.

Let C, D be the two $\mathcal{FL}_0$-concept terms to be unified, and assume that $\emptyset \neq \{A_1, \ldots, A_k\} \subseteq C_c$ contains all the concept names of C_c that occur in C, D. In addition, let $X_1, \ldots, X_n$ be the concept names of C_v that occur in C, D.

First, we show that C, D can be transformed into a certain normal form. We know that any $\mathcal{FL}_0$-concept term can be transformed into an equivalent $\mathcal{FL}_0$-concept term that is either $\top$ or a (nonempty) conjunction of terms of the form $\forall R_1.\cdots\forall R_m.A$ for $m \geq 0$ (not necessarily distinct) role names $R_1, \ldots, R_m$ and a concept name $A \neq \top$. We abbreviate $\forall R_1.\cdots\forall R_m.A$ by $\forall R_1 \ldots R_m.A$, where $R_1 \ldots R_m$ is considered as a word over the alphabet of all role names Δ. In addition, instead of $\forall w_1.A \sqcap \ldots \sqcap \forall w_\ell.A$ we write $\forall L.A$ where $L := \{w_1, \ldots, w_\ell\}$ is a finite set of words over Δ. The term $\forall\emptyset.A$ is considered to be equivalent to $\top$. Using these abbreviations, the terms C, D can be rewritten as

$$\begin{aligned} C &\equiv \forall S_{0,1}.A_1 \sqcap \ldots \sqcap \forall S_{0,k}.A_k \sqcap \forall S_1.X_1 \sqcap \ldots \sqcap \forall S_n.X_n, \\ D &\equiv \forall T_{0,1}.A_1 \sqcap \ldots \sqcap \forall T_{0,k}.A_k \sqcap \forall T_1.X_1 \sqcap \ldots \sqcap \forall T_n.X_n, \end{aligned}$$

for finite sets of words $S_{0,i}, S_j, T_{0,i}, T_j$ $(i = 1, \ldots, k, j = 1, \ldots, n)$. If C, D are ground terms, i.e., $\mathcal{FL}_0$-concept terms that do not contain concept variables, then we have $S_1 = \ldots = S_n = \emptyset = T_1 = \ldots = T_n$. In fact, the terms $\forall\emptyset.X_i$ are equivalent to $\top$, and can thus be removed from the conjunction.

The next lemma characterizes equivalence of ground terms in $\mathcal{FL}_0$.

Lemma 6 *Let C, D be ground terms such that*

$$\begin{aligned} C &\equiv \forall U_1.A_1 \sqcap \ldots \sqcap \forall U_k.A_k, \\ D &\equiv \forall V_1.A_1 \sqcap \ldots \sqcap \forall V_k.A_k. \end{aligned}$$

Then $C \equiv D$ iff $U_i = V_i$ for all $i = 1, \ldots, k$.

Proof. The if-direction is trivial. To show the only-if-direction, assume that $U_i \neq V_i$. Without loss of generality, let $w = R_1 \ldots R_r$ be such that $w \in U_i \setminus V_i$. Thus, the conjunct $\forall R_1.\cdots\forall R_r.A_i$ occurs in C, but not in D. As in the proof of the only-if-direction of Lemma 4, this fact can be used to construct an interpretation I such that $D^I \setminus C^I \neq \emptyset$, which shows that the two terms cannot be equivalent. $\square$

As an easy consequence of this lemma, we can now characterize unifiability of $\mathcal{FL}_0$-concept terms:

Theorem 7 *Let C, D be $\mathcal{FL}_0$-concept terms such that*

$$\begin{aligned} C &\equiv \forall S_{0,1}.A_1 \sqcap \ldots \sqcap \forall S_{0,k}.A_k \sqcap \forall S_1.X_1 \sqcap \ldots \sqcap \forall S_n.X_n, \\ D &\equiv \forall T_{0,1}.A_1 \sqcap \ldots \sqcap \forall T_{0,k}.A_k \sqcap \forall T_1.X_1 \sqcap \ldots \sqcap \forall T_n.X_n. \end{aligned}$$

Then C, D are unifiable iff for all $i = 1, \ldots, k$, the linear equation $E_{C,D}(A_i)$:

$$S_{0,i} \cup S_1 X_{1,i} \cup \cdots \cup S_n X_{n,i} \quad = \quad T_{0,i} \cup T_1 X_{1,i} \cup \cdots \cup T_n X_{n,i}$$

has a solution.

Note that this is not a system of k equations that must be solved simultaneously: since they do not share variables, each of these equations can be solved separately.

Before proving the theorem, let us consider a simple example: The concept terms in normal form corresponding to

$$\begin{aligned} C &= \forall R.(A_1 \sqcap \forall R.A_2) \sqcap \forall R.\forall S.X_1, \\ D &= \forall R.\forall S.(\forall S.A_1 \sqcap \forall R.A_2) \sqcap \forall R.X_1 \sqcap \forall R.\forall R.A_2 \end{aligned}$$

are

$$\begin{aligned} C' &= \forall\{R\}.A_1 \sqcap \forall\{RR\}.A_2 \sqcap \forall\{RS\}.X_1, \\ D' &= \forall\{RSS\}.A_1 \sqcap \forall\{RSR, RR\}.A_2 \sqcap \forall\{R\}.X_1. \end{aligned}$$

Thus, unification of C, D leads to the two linear equations

$$\begin{aligned} \{R\} \cup \{RS\}X_{1,1} &= \{RSS\} \cup \{R\}X_{1,1}, \\ \{RR\} \cup \{RS\}X_{1,2} &= \{RSR, RR\} \cup \{R\}X_{1,2}. \end{aligned}$$

The first equation (the one for A_1) has $X_{1,1} = \{\varepsilon, S\}$ as a solution, and the second (the one for A_2) has $X_{1,2} = \{R\}$ as a solution. These two solutions yield the following unifier of C, D:

$$\{X_1 \mapsto A_1 \sqcap \forall S.A_1 \sqcap \forall R.A_2\}.$$

Proof of the theorem. It is easy to see that the unification problem for C, D has a solution iff it has a ground solution, i.e., a unifier that replaces the variables X_i by terms containing no other concept names than $A_1, \ldots, A_k$. In fact, in a given unifier, concept constants not occurring in C, D and concept variables can simply be instantiated by (arbitrary) ground terms. The obtained substitution is ground and still a unifier.

Now, let $\sigma := \{X_1 \mapsto \bigsqcap_{i=1}^{k} \forall U_{1,i}.A_i,\ \ldots,\ X_n \mapsto \bigsqcap_{i=1}^{k} \forall U_{n,i}.A_i\}$ be a ground substitution. Using the identities in ACUIh, it is easy to see that

$$\begin{aligned} \sigma(C) &\equiv \bigsqcap_{i=1}^{k} \forall\left(S_{0,i} \cup S_1 U_{1,i} \cup \cdots \cup S_n U_{n,i}\right).A_i, \\ \sigma(D) &\equiv \bigsqcap_{i=1}^{k} \forall\left(T_{0,i} \cup T_1 U_{1,i} \cup \cdots \cup T_n U_{n,i}\right).A_i. \end{aligned}$$

Lemma 6 implies that $\sigma(C) \equiv \sigma(D)$ iff, for all $i = 1, \ldots, k$,

$$S_{0,i} \cup S_1 U_{1,i} \cup \cdots \cup S_n U_{n,i} = T_{0,i} \cup T_1 U_{1,i} \cup \cdots \cup T_n U_{n,i}.$$

Thus, if σ is a unifier of C, D, then $X_{1,i} := U_{1,i}, \ldots, X_{n,i} := U_{n,i}$ is a solution of $E_{C,D}(A_i)$ $(i = 1, \ldots, k)$. Conversely, solutions of $E_{C,D}(A_i)$ for $i = 1, \ldots, k$ can be used to build a unifier of C, D. $\square$

5 Solving linear equations in $\mathcal{S}_{\text{ACUIh}}$

In this section, we show that solvability of ACUIh-unification problems, and thus also unification of $\mathcal{FL}_0$-concept terms, is decidable:

Theorem 8 *Solvability of ACUIh-unification problems can be decided in deterministic exponential time.*

This decidability result can be obtained by reducing solvability of linear equations in the semiring $\mathcal{S}_{\text{ACUIh}}$ to the emptiness problem for (root-to-frontier) tree automata working on finite trees [11]. The main idea underlying the proof is as follows. A finite set of words over an alphabet Δ of cardinality k can be represented by a finite tree, where each node has at most k sons. In such a tree, every path from the root to a node can be represented by a unique word over Δ. If the nodes of the tree are labelled with 0 or 1, then we can take the set of all words representing paths from the root to nodes with label 1 as the finite set of words represented by the tree. In the following, we assume w.l.o.g. that $\Delta = \{1, \ldots, k\}$.

Definition 9 A k-ary tree with labels in $\{0, 1\}$ is a mapping $t : dom(t) \to \{0, 1\}$ such that $dom(t)$ is a finite subset of $\{1, \ldots, k\}^*$ such that

- $dom(t)$ is prefix-closed, i.e., $uv \in dom(t)$ implies $u \in dom(t)$.
- $ui \in dom(t)$ for some $i, 1 \leq i \leq k$, implies $uj \in dom(t)$ for all $j = 1, \ldots, k$.

The elements of $dom(t)$ are the nodes of the tree t, and $t(u)$ is called the label of node u. The empty word ε is the root of t, and the nodes u such that $ui \notin dom(t)$ for all $i = 1, \ldots, k$ are the leaves of t. The set of all leaves of t is called the frontier of t. Nodes of t that are not in the frontier are called inner nodes. If $ui \in dom(t)$ then it is called the ith son of u in t. By our definition of k-ary trees, any node of t is either a leaf, or it has exactly k sons.

For a k-ary tree t with labels in $\{0, 1\}$ we define

$$L(t) := \{u \in dom(t) \mid t(u) = 1\}.$$

Obviously, $L(t)$ is a finite set of words over $\Delta = \{1, \ldots, k\}$, and any finite set of words over Δ can be represented in this way.

Definition 10 A (nondeterministic) root-to-frontier (or top-down) tree automaton that works on k-ary trees with labels in $\{0, 1\}$ is a 4-tuple $\mathcal{A} = (Q, I, T, F)$ where

- Q is a finite set of states,

- $I \subseteq Q$ is the set of initial states,
- $T \subseteq Q \times \{0,1\} \times Q^k$ is the transition relation, and
- $F : \{0,1\} \to 2^Q$ assigns to each label l in $\{0,1\}$ a set of final states $F(l) \subseteq Q$.

A run of $\mathcal{A}$ on the tree t is a mapping $r : dom(t) \to Q$ such that

- $(r(u), t(u), r(u1), \ldots, r(uk)) \in T$ for all inner nodes u.

The run r is called successful iff

- $r(\varepsilon) \in I$ (root condition),
- $r(u) \in F(t(u))$ for all leaves u (leaf condition).

The tree language accepted by $\mathcal{A}$ is defined as

$$\mathcal{L}(\mathcal{A}) := \{t \mid \text{there exists a successful run of } \mathcal{A} \text{ on } t\}.$$

The emptiness problem for $\mathcal{A}$ is the question whether $\mathcal{L}(\mathcal{A}) \neq \emptyset$.

The following theorem is well-known (see, e.g., [14]):

Theorem 11 *The emptiness problem for root-to-frontier tree automata is decidable in polynomial time.*

Our approach for solving linear equations in $\mathcal{S}_{\mathrm{ACUIh}}$ with the help of tree automata cannot treat equations of the form

$$S_0 \cup S_1X_1 \cup \cdots \cup S_nX_n = T_0 \cup T_1X_1 \cup \cdots \cup T_nX_n \quad (*)$$

directly:[4] it needs an equation where the variables X_i are in front of the coefficients S_i. However, such an equation can easily be obtained from $(*)$ by considering the mirror images (or reverse) of the involved languages. For a word $w = i_1 \ldots i_m$, its mirror image is defined as $w^{mi} := i_m \ldots i_1$, and for a finite set of words $L = \{w_1, \ldots, w_\ell\}$, its mirror image is $L^{mi} := \{w_1^{mi}, \ldots, w_\ell^{mi}\}$. Obviously, $X_1 = L_1, \ldots, X_n = L_n$ is a solution of $(*)$ iff $Y_1 = L_1^{mi}, \ldots, Y_n = L_n^{mi}$ is a solution of the corresponding mirrored equation $(**)$:

$$S_0^{mi} \cup Y_1S_1^{mi} \cup \cdots \cup Y_nS_n^{mi} \;=\; T_0^{mi} \cup Y_1T_1^{mi} \cup \cdots \cup Y_nT_n^{mi}. \quad (**)$$

In principle, we build a tree automaton that accepts the trees representing the finite sets of words obtained by instantiating this equation with its solutions. To

[4] Basically, this is due to Theorem 11.6, (b) in [14].

achieve this goal, the automaton guesses at each node whether it (more precisely, the path leading to it) belongs to one of the Y_is (more precisely, to the set of words instantiated for Y_i), and then does the necessary book-keeping to make sure that the concatenation with the elements of S_i^{mi} and T_i^{mi} is realized: if S_i^{mi} contains a word w, and the automaton has decided that a given node κ belongs to Y_i, then if one starts at κ and follows the path corresponding to w, one must find a node with label 1. Vice versa, every label 1 in the tree must be justified this way. The same must hold for T_i^{mi} in place of S_i^{mi}. The size of the set of states of this automaton will turn out to be exponential in the the size of the equation (due to the necessary book-keeping). Since the emptiness problem for tree automata working on finite trees can be solved in polynomial time (in the size of the automaton), this will yield the exponential time algorithm claimed in Theorem 8.

Before we can define the automaton corresponding to the (solutions of) equation $(**)$, we need some more notation. For a finite set of words S and a word u, we define $u^{-1}S := \{v \mid uv \in S\}$. The suffix closure of S is the set $\mathit{suf}(S) := \{u \mid \text{there exists } v \text{ such that } vu \in S\}$. Obviously, the cardinality of $\mathit{suf}(S)$ is linear in the size of S (which is the sum of the length of the words in S), and $u^{-1}S \subseteq \mathit{suf}(S)$.

The root-to-frontier tree automaton $\mathcal{A}_{**} = (Q, I, T, F)$ corresponding to equation $(**)$ is defined as follows:

- Let $M_L := \mathit{suf}(\bigcup_{i=0}^{n} S_i^{mi})$, $M_R := \mathit{suf}(\bigcup_{i=0}^{n} T_i^{mi})$ and $N := \{1, \ldots, n\}$. Then $Q := 2^N \times 2^{M_L} \times 2^{M_R}$, i.e., the states of $\mathcal{A}_{**}$ are triples whose first component is a subset of the set of indices of the variables in $(**)$, the second component is a finite set of words that are suffixes of words occurring on the left-hand side of $(**)$, and the third component is a finite set of words that are suffixes of words occurring on the right-hand side of $(**)$. Obviously, the size of Q is exponential in the size of equation $(**)$.

  Intuitively, the first component of a state "guesses" to which of the Y_is the word represented by the current node of the tree belongs. The second component does the book-keeping for the left-hand side of the equation: if u is the word represented by the current node of the tree and v belongs to the second component of the state, then uv must belong to the (evaluated) left-hand side. The third component does the same for the right-hand side.

- The set of initial states is defined as

  $$I := \{(G, L, R) \mid \ G \subseteq N,\ L = S_0^{mi} \cup \bigcup_{i \in G} S_i^{mi},\ R = T_0^{mi} \cup \bigcup_{i \in G} T_i^{mi}\}.$$

  Intuitively, G is our initial guess which of the Y_is contain the empty word. Every word in S_0^{mi} belongs to the (evaluated) left-hand side, and if $\varepsilon \in Y_i$, then every word in S_i^{mi} also belongs to the left-hand side.

- The transition relation T consists of all tuples

  $$((G_0, L_0, R_0), l, (G_1, L_1, R_1), \ldots, (G_k, L_k, R_k)) \in Q \times \{0,1\} \times Q^k$$

  such that

  - $\varepsilon \in L_0$ iff $\varepsilon \in R_0$ iff $l = 1$.
    This makes sure that the left-hand side is evaluated to the same set of words as the right-hand side, and that this is the set of words represented by the accepted tree.
  - For $i = 1, \ldots, k$,

    $$\begin{aligned} L_i &= i^{-1}L_0 \cup \bigcup_{j \in G_i} S_j^{mi}, \\ R_i &= i^{-1}R_0 \cup \bigcup_{j \in G_i} T_j^{mi}. \end{aligned}$$

    This updates the book-keeping information: if $iu \in L_0$ then u must belong to L_i, the corresponding book-keeping component of the ith son of the current node. If G_i contains j, i.e., we have guessed that the word represented by the ith son belongs to Y_j, then the book-keeping component L_i of this son must also contain all elements of S_j^{mi}. The equation for R_i can be explained similarly.

- The assignment of sets of final states to labels is defined as follows:

  $$\begin{aligned} F(0) &:= \{(G, L, R) \mid L = R = \emptyset\}, \\ F(1) &:= \{(G, L, R) \mid L = R = \{\varepsilon\}\}. \end{aligned}$$

  Again, this makes sure that the left-hand side is evaluated to the same set of words as the right-hand side, and that this is the set of words represented by the accepted tree.

Lemma 12 *Let t be a k-ary tree with labels in $\{0,1\}$. Then the following are equivalent:*

1. *$t \in \mathcal{L}(\mathcal{A}_{**})$.*
2. *There are finite sets of words $\theta(Y_1), \ldots, \theta(Y_n)$ such that*

   $$S_0^{mi} \cup \theta(Y_1)S_1^{mi} \cup \cdots \cup \theta(Y_n)S_n^{mi} = L(t) = T_0^{mi} \cup \theta(Y_1)T_1^{mi} \cup \cdots \cup \theta(Y_n)T_n^{mi}.$$

Proof. If $t \in \mathcal{L}(\mathcal{A}_{**})$, then there exists a successful run of $\mathcal{A}_{**}$ on t. From the first components of the states assigned to the nodes of t we can read off appropriate sets $\theta(Y_1), \ldots, \theta(Y_n)$: if the first component of the state assigned to the node κ

contains i, then the word represented by κ belongs to $\theta(Y_i)$. The definition of $\mathcal{A}_{**}$ makes sure that this assignment of finite sets of words to the variables in $(**)$ satisfies the equations in statement 2 of the lemma.

Conversely, if $\theta(Y_1), \ldots, \theta(Y_n)$ is an assignment of finite sets of words to the variables Y_i, then this assignment can be used to determine appropriate first components of states for a run of $\mathcal{A}_{**}$. Once these first components are fixed, an appropriate tree t and the full run of $\mathcal{A}_{**}$ on t can be reconstructed. The fact that the equations in statement 2 are satisfied guarantees that this run exists and that it is successful. $\square$

As an immediate consequence of this lemma we obtain that equation $(**)$ has a solution iff $\mathcal{L}(\mathcal{A}_{**}) \neq \emptyset$. Since the emptiness problem can be decided in time polynomial in the size of $\mathcal{A}_{**}$, and since $\mathcal{A}_{**}$ is exponential in the size of $(**)$, this completes the proof of Theorem 8.

6 ACUIh-unification is PSPACE-hard

We show in this section that the ACUIh-unification problem is PSPACE-hard. The reduction is from the Finite State Automata Intersection problem, which has been shown to be PSPACE-complete by Kozen (see [10]). This problem can be described as follows: given a sequence $\mathcal{A}_1, \ldots, \mathcal{A}_n$ of deterministic finite state automata (dfa) over the same input alphabet Σ, decide whether there exists a word w accepted by each of these automata. (Note that the problem is polynomial for any fixed number n of automata.)

For simplicity assume that the transition relation of a dfa $\mathcal{A}$ is represented using a finite word rewriting system $R = \{l_i \to r_i \mid 1 \leq i \leq k\}$. Each left-hand side l_i is of the form $p_i a_i$ for a state p_i of the automaton and an input symbol $a_i \in \Sigma$, and the right-hand side is a state q_i of the automaton. Thus, $l_i \to r_i$ represents the transition that says: if the automaton is in state p_i and reads the symbol a_i, then it goes into state q_i. Since the automata are assumed to be deterministic, there exists at most one rule with left-hand side $p_i a_i$ for each pair (p_i, a_i). The automaton represented by R accepts the word w iff $q_0 w$ can be reduced to q_f (where q_0 is the initial state of the automaton, and q_f is one of the final states).

We may also assume that the dfa has exactly one final state. In fact, if we modify the automata $\mathcal{A}_1, \ldots, \mathcal{A}_n$ by adding a new symbol $\sharp$ and a new final state, and a transition with $\sharp$ from each original final state to this new one, then the resulting automata accept a common word iff the original ones did.

Given such a dfa $\mathcal{A}$ over Σ with final state q_f and initial state q_0, we consider the alphabet Δ that consists of Σ and the states of $\mathcal{A}$. We construct the following linear equation, where the variables X, X_i range over finite sets of words over Δ:

$$\{q_0\}X \cup \{r_1\}X_1 \cup \ldots \cup \{r_k\}X_k = \{q_f\} \cup \{l_1\}X_1 \cup \ldots \cup \{l_k\}X_k$$

We want to show that for any solution θ of this equation, all elements of X belong to the language accepted by $\mathcal{A}$.

To this purpose, we will consider a more general situation. Let T be a finite set of words over Δ. We want to show that the problem

$$T \cup \{r_1\}X_1 \cup \ldots \cup \{r_k\}X_k = \{q_f\} \cup \{l_1\}X_1 \cup \ldots \cup \{l_k\}X_k$$

has a solution iff all the words in T can be reduced to q_f.

The if-direction is not hard to see. Thus, let us consider the only-if direction, i.e., assume that θ is a solution of the equation.

We prove the statement by induction on the sum of the lengths of the elements of T, i.e., $\sum_{w \in T} |w|$.

Note that, for $i \neq j$, the sets of words $\{l_i\}\theta(X_i)$ and $\{l_j\}\theta(X_j)$ are disjoint, no matter what solution θ we consider (since the dfa is deterministic). But this need not be the case for the sets on the left-hand side.

Let w be an element of T. Since θ was assumed to be a solution of the equation, there are two cases: either $w = q_f$, or w belongs to (exactly) one of the sets $\{l_i\}\theta(X_i)$ on the right-hand side. In the first case, there is nothing to show. Thus, assume that w is contained in the set $\{l_i\}\theta(X_i)$. Thus, $w = l_i u$ for a word $u \in \theta(X_i)$, and hence $r_i u$ is contained in $\{r_i\}\theta(X_i)$. Note that w reduces to $r_i u$ in one step using the transition rules. Consider

$$T' := (T \setminus \{w\}) \cup \{r_i u\},$$

and define a new substitution β, which coincides with θ on all variables different from X_i. For the definition of $\beta(X_i)$ we distinguish two cases:

1. if w occurs in one of the sets $\{r_j\}\theta(X_j)$, then $\beta(X_i) := \theta(X_i)$;
2. otherwise, $\beta(X_i) := \theta(X_i) \setminus \{u\}$.

This substitution is a solution for

$$T' \cup \{r_1\}X_1 \cup \ldots \cup \{r_k\}X_k = \{q_f\} \cup \{l_1\}X_1 \cup \ldots \cup \{l_k\}X_k.$$

In fact, in the second case, $w = l_i u$ is removed from the left-hand side of the equation as well as from the right-hand side. The word $r_i u$ is removed from $\{r_i\}\theta(X_i)$ on the left-hand side, but it is added to T'. Since it has occurred on the right-hand side for solution θ (and is different from $w = l_i u$ because it is shorter), it still occurs on the right-hand side for solution β. In the first case, $w = l_i u$ remains on the right-hand side (since u is still in $\beta(X_i)$). It is also contained in the left-hand side (by the assumption that it is contained in some $\{r_j\}\theta(X_j)$).

Obviously, T' is lower in our measure than T since $|l_i u| = |r_i u| + 1$. Thus, induction yields that $r_i u$ reduces to the final state q_f, which shows that $w = l_i u$ does the same.

It is also easy to show that for $T = \emptyset$ there does not exist a solution. In fact, for a solution θ, the left-hand side of the equation cannot be empty (since the right-hand side isn't). Thus, take a word of maximal length in $\{r_1\}\theta(X_1) \cup \ldots \cup \{r_k\}\theta(X_k)$, and assume that it belongs to $\{r_i\}\theta(X_i)$. Then, $\{l_i\}\theta(X_i)$ contains a longer word, which yields the necessary contradiction.

For n deterministic finite automata, we can thus construct a system consisting of n such equations. We assume that the only variable shared by these equations is the variable X. Since in a solution of this system the variable X cannot be replaced by the empty set, and since the words in the set substituted for X always belong to the languages accepted by the automata, we have thus reduced the Finite State Automata Intersection problem to the problem of solving a system of linear equations over sets of finite words. This proves

Theorem 13 *Solvability of ACUIh-unification problems is PSPACE-hard.*

Consequently, unifiability of $\mathcal{FL}_0$-concept terms is also a PSPACE-hard problem.

7 ACUIh-matching is polynomial

For the purpose of this article, where we are only interested in the existence of matchers, matching can be seen as the special case of unification where the term t on the right-hand side of the equation $s =^? t$ does not contain variables [9]. As for unification, we can restrict our attention to the case of a single such equation.

As an easy consequence of Theorem 7 we obtain that matching of $\mathcal{FL}_0$-concept terms (equivalently, ACUIh-matching) can be reduced to solving linear equations of the form

$$S_0 \cup S_1 X_1 \cup \cdots \cup S_n X_n \;=\; T_0, \qquad (***)$$

where $S_0, \ldots, S_n, T_0$ are finite sets of words over the alphabet of all role names. A solution of this equation assigns finite sets of words to the variables X_i such that the equation holds.

Lemma 14 *Equation* $(***)$ *has a solution iff the following is a solution of* $(***)$*:*

$$\theta(X_i) := \bigcap_{u \in S_i} u^{-1}T_0 \quad (i = 1, \ldots, n).$$

Proof. The if-direction is trivial. To show the only-if-direction, we assume that $\tau(X_1), \ldots, \tau(X_n)$ are finite sets of words that solve $(***)$.

First, we prove that $\tau(X_i) \subseteq \theta(X_i)$ holds for all $i = 1, \ldots, n$. Thus, let $v \in \tau(X_i)$ and $u \in S_i$. Since $S_i\tau(X_i) \subseteq T_0$, we know that $uv \in T_0$, and thus $v \in u^{-1}T_0$. This shows that $\tau(X_i) \subseteq u^{-1}T_0$ for all $u \in S_i$, which yields $\tau(X_i) \subseteq \theta(X_i)$.

As an immediate consequence, we obtain

$$T_0 = S_0 \cup S_1\tau(X_1) \cup \cdots \cup S_n\tau(X_n) \subseteq S_0 \cup S_1\theta(X_1) \cup \cdots \cup S_n\theta(X_n).$$

It remains to be shown that the other inclusion holds as well. Obviously, we have $S_0 \subseteq T_0$ since there exists a solution. To conclude the proof, let $u \in S_i$ and $v \in \theta(X_i)$. We must show that $uv \in T_0$. By definition of $\theta(X_i)$, we know that $v \in u^{-1}T_0$, and thus $uv \in T_0$. □

Obviously, computing the sets $\theta(X_i)$ and checking whether they yield a solution of $(***)$ can be done in time polynomial in the size of $(***)$. Thus, we have proved the following theorem:

Theorem 15 *Solvability of ACUIh-matching problems can be decided in polynomial time.*

Consequently, matching of $\mathcal{FL}_0$-concept terms is also polynomial.

The connection to the work of Borgida and McGuinness

In [7], Borgida and McGuinness consider a slightly different matching problem: matching modulo subsumption. For given concept terms C and D, where C does not contain variables, they ask for a substitution σ such that $C \sqsubseteq \sigma(D)$. Moreover, they are interested in a substitution σ such that $\sigma(D)$ is as small as possible w.r.t. the subsumption hierarchy.

Obviously, since C does not contain variables, $C \sqsubseteq \sigma(D)$ iff $\sigma(C \sqcap D) = C \sqcap \sigma(D) \equiv C$, which shows that matching modulo subsumption can be reduced to matching as considered above. In particular, this shows that for $\mathcal{FL}_0$-concept terms matching modulo subsumption is polynomial:

Corollary 16 *The following problem is decidable in polynomial time:*

Instance: *$\mathcal{FL}_0$-concept terms C and D, where C does not contain variables.*

Question: *Does there exist a substitution σ such that $C \sqsubseteq \sigma(D)$?*

As an easy consequence of the proof of Lemma 14, we can also compute a substitution σ such that $\sigma(D)$ is as small as possible w.r.t. the subsumption hierarchy, if the matching problem is solvable. In fact, we have shown that the solution θ of $({*}{*}{*})$ constructed in the proof is larger (w.r.t. set inclusion) than all other solutions of $({*}{*}{*})$. Since each word in a solution of $({*}{*}{*})$ gives rise to an additional value restriction, it is clear that the largest solution of $({*}{*}{*})$ gives rise to a solution σ of the matching problem such that $\sigma(D)$ is as small as possible w.r.t. subsumption.

Borgida and McGuinness consider a language that is more expressive than $\mathcal{FL}_0$. In addition, they allow for role variables (which may be replaced by role constants). They present a polynomial matching algorithm, which is, however, not complete. In addition they state (without proof) that matching for $\mathcal{FL}_0$-concept terms containing role variables is NP-complete. This result can easily be proved as follows:

Theorem 17 *Solvability of matching problems for $\mathcal{FL}_0$-concept terms containing role variables is NP-complete.*

Proof. Since role variables may only be replaced by role constants (and not by complex role terms), we can nondeterministically guess the right assignment of role names to role variables, and then apply our polynomial decision procedure for matching of $\mathcal{FL}_0$-concept terms without role variables. This shows that the problem is in NP.

To show the hardness result, we reduce monotone 1-in-3-SAT (see [10]) to the matching problem for $\mathcal{FL}_0$-concept terms containing role variables. For every propositional variable p in an instance of monotone 1-in-3-SAT we introduce a role variable R_p. In addition, we use role constants R_0 and R_1 to represent the truth values. A clause $p \vee q \vee r$ is translated into the matching problem

$$\forall R_p.\forall R_q.\forall R_r.A \sqcap X =^{?} \forall R_0.\forall R_0.\forall R_1.A \sqcap \forall R_0.\forall R_1.\forall R_0.A \sqcap \forall R_1.\forall R_0.\forall R_0.A,$$

where X is a concept variable used only in this equation. It is easy to see that a solution of this problem assigns R_1 to exactly one of the three role variables R_p, R_q, R_r, and R_0 to the other two. Vice versa, any such assignment can be extended to a solution of the matching problem by assigning an appropriate value to X. Thus, the system of all matching problems obtained from the clauses of the instance of monotone 1-in-3-SAT is solvable iff the monotone 1-in-3-SAT problem has a solution. Since solving systems of matching problems can be reduced to solving a single matching problem, this reduction also shows NP-hardness for single matching problems. □

8 Future work

Beside the technical problem of obtaining a tight complexity bound for unification of $\mathcal{FL}_0$-concept terms, the main topic for future work is to extend the decidability results to more expressive DL languages. Using a direct reduction of the unification problem to a corresponding formal language problem (as described in the previous section), our approach may also be applicable to languages for which equivalence of concept terms is not axiomatizable by a commutative equational theory.

Another interesting problem is how to define an appropriate ordering on unifiers. For the instantiation preorder usually employed in unification theory, ACUIh is not well-behaved [1]: it is not possible to represent all unifiers by finitely many most general ones. However, note that a more expressive language might lead to a theory with a better behaviour (since in a richer signature there are more substitutions available). Second, it might well be the case that the instantiation ordering on substitutions (which is appropriate for the applications of equational unification in theorem proving, term rewriting, and logic programming) is not the right ordering to use when dealing with substitutions operating on concept terms. As indicated by the work of Borgida and McGuinness [7], another ordering, induced by the subsumption hierarchy, might be more appropriate.

References

[1] F. Baader. Unification in commutative theories. *J. Symbolic Computation*, 8:479–497, 1989.

[2] F. Baader. Terminological cycles in KL-ONE-based knowledge representation languages. In *Proceedings of the Eighth National Conference on Artificial Intelligence, AAAI-90*, pages 621–626, Boston (USA), 1990.

[3] F. Baader and B. Hollunder. A terminological knowledge representation system with complete inference algorithms. In *Proceedings of the First International Workshop on Processing Declarative Knowledge*, volume 572 of *Lecture Notes in Computer Science*, pages 67–85, Kaiserslautern (Germany), 1991. Springer–Verlag.

[4] F. Baader and W. Nutt. Combination problems for commutative/monoidal theories: How algebra can help in equational reasoning. *J. Applicable Algebra in Engineering, Communication and Computing*, 7(4):309–337, 1996.

[5] F. Baader and U. Sattler. Knowledge representation in process engineering. In *Proceedings of the International Workshop on Description Logics*, Cambridge (Boston), MA, U.S.A., 1996. AAAI Press/The MIT Press.

[6] F. Baader and J.H. Siekmann. Unification theory. In D.M. Gabbay, C.J. Hogger, and J.A. Robinson, editors, *Handbook of Logic in Artificial Intelligence and Logic Programming*. Oxford University Press, Oxford, UK, 1994.

[7] A. Borgida and D.L. McGuinness. Asking queries about frames. In *Proceedings of the Fifth International Conference on Principles of Knowledge Representation and Reasoning, KR'96*, pages 340–349, Cambridge, MA (USA), 1996.

[8] R. J. Brachman and J. G. Schmolze. An overview of the KL-ONE knowledge representation system. *Cognitive Science*, 9(2):171–216, 1985.

[9] H.-J. Bürckert. Matching—a special case of unification? *J. Symbolic Computation*, 8(5):532–536, 1989.

[10] M.R. Garey and D.S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W.H. Freeman and Company, New York, 1979.

[11] F. Gécseg and M. Steinby. *Tree Automata*. Akadémiai Kiaddá, Budapest, Hungary, 1984.

[12] P. Narendran. Solving linear equations over polynomial semirings. In *11th Annual Symposium on Logic in Computer Science, LICS'96*, pages 466–472, Rutger University (NJ), 1996. IEEE Computer Society Press.

[13] W. Nutt. Unification in monoidal theories. In M.E. Stickel, editor, *Proceedings of the 10th International Conference on Automated Deduction*, volume 449 of *Lecture Notes in Artificial Intelligence*, pages 618–632, Kaiserslautern, Germany, 1990. Springer-Verlag.

[14] W. Thomas. Automata on infinite objects. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*, volume B, pages 133–191, Amsterdam, 1990. Elsevier Science Publishers.