# Unification of Concept Terms
## – Extended Abstract –

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

## 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 [7, 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 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

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

from the explicitly represented knowledge. For instance, the subsumption algorithm allows one to determine subconcept-superconcept relationships. For example, the concept term $\mathsf{Woman}$ subsumes the concept term $\mathsf{Woman} \sqcap \forall \mathsf{child}.\mathsf{Woman}$ 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 are *equivalent* if they subsume each other, i.e., if 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. 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.[1] 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* by the concept term

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

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 $\mathsf{Woman}$. 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 substituting the atomic concept $\mathsf{Woman}$ 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 can be made equivalent by applying an appropriate substitution, where a substitution replaces (some of the) atomic concepts 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.

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

[1] This work was motivated by an application in chemical process engineering, in which this situation occurs.

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, 12, 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 languages 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.

# 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 2.1** Let $\mathcal{C}$ and $\mathcal{R}$ be disjoint sets, the set of *atomic concepts* and the set of *atomic roles*. The set of all $\mathcal{FL}_0$-*concept terms* 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 2.2** An *interpretation* $I$ consists of a non-empty set $\Delta^I$, 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

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

$R^I \subseteq \Delta^I \times \Delta^I$. 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* $\sigma$ 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 2.3** Let $C$ and $D$ be $\mathcal{FL}_0$-concept terms. The substitution $\sigma$ 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*.

# 3 The equational theory ACUIh

In this section we show that 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 [5].

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 $\mathsf{T}$, and for each $R \in \mathcal{R}$ a unary function symbol $h_R$. In addition, every element of $\mathcal{C}_v$ is considered as a variable symbol, and every element of $\mathcal{C}_c$ as a (free) constant. The translation function $\tau$ 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, $\tau$ is a bijective mapping between the set of all $\mathcal{FL}_0$-concept terms (with atomic concept 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 that axiomatizes equivalence of $\mathcal{FL}_0$-concept terms is defined by the following identities:

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

**Lemma 3.1** *Let $C$ and $D$ be $\mathcal{FL}_0$-concept terms. Then*

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

*Proof.* The if-direction is an easy consequence of the definition of $\mathcal{FL}_0$-concept terms. In fact, since concept conjunction is interpreted as set union, it inherits associativity, commutativity, and idempotency (modulo equivalence) from set union. 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 respresent $\mathcal{FL}_0$-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 $\Delta^I$ 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, $d_0 \in D'^I = D^I$. □

The lemma shows that the concept terms $C$ and $D$ are unifiable iff the corresponding terms $\tau(C)$ and $\tau(D)$ are unifiable modulo ACUIh. 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 $\Gamma$ 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 3.2** *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 $d_0 \in \Delta^I$ 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. □

The unification type of ACUIh has 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 by a *finite* complete set of unifiers. The problem of deciding solvability of ACUIh-unification problems 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 [11].

# 4 Reducing ACUIh-unification to solving linear equations

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 [12, 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 $\mathsf{T}$ 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 $\mathsf{T}$, respectively, there are two possible endomorphisms: 0, which is defined by $x \mapsto \mathsf{T}$, 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., $(1+0)(x) := 1(x) \wedge 0(x) = x \wedge \mathsf{T} =_{\text{ACUI}} x = 1(x)$. 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 $\mathcal{S}_{\text{ACUIh}}$ corresponding to ACUIh is the polynomial semiring (in $|\mathcal{R}|$ non-commuting indeterminates) over the Boolean semiring.

Let $\Delta$ be the set of these indeterminates. Monomials in $\mathcal{S}_{\text{ACUIh}}$ are simply words over the alphabet $\Delta$, 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. Consequently, the problem of solving systems of linear equations over $\mathcal{S}_{\text{ACUIh}}$ can be reduced to solving the following formal language problem:

For $i = 1, \ldots, m$, let $S_{i,0}, S_{i,1}, \ldots, S_{i,n}, T_{i,0}, T_{i,1}, \ldots, T_{i,n}$ be finite sets of words over the alphabet $\Delta$. We consider the system of equations

$$S_{1,0} \cup X_1 S_{1,1} \cup \cdots \cup X_n S_{1,n} \quad = \quad T_{1,0} \cup X_1 T_{1,1} \cup \cdots \cup X_n T_{1,n}$$

$$\begin{array}{rcl} \vdots & & \vdots \\ S_{m,0} \cup X_1 S_{m,1} \cup \cdots \cup X_n S_{m,n} & = & T_{m,0} \cup X_1 T_{m,1} \cup \cdots \cup X_n T_{m,n} \end{array}$$

A solution of this system assigns finite sets of words over $\Delta$ to the variables $X_i$ such that the equations hold. For example, the equation

$$\{aaa\} \cup X_1\{aa\} \cup X_2\emptyset = \{baa\} \cup X_1\emptyset \cup X_2\{a, aa\}$$

has as a solution $X_1 = \{\varepsilon, b\}$ and $X_2 = \{a\}$ (where $\varepsilon$ denotes the empty word).

**Theorem 4.1** *Solvability of ACUIh-unification problems (with free constants) can be decided in deterministic exponential time.*

This theorem can be proved by reducing solvability of the above formal language problem to the emptiness problem for (root-to-frontier) tree automata [10]. The main idea underlying the proof is as follows. A finite set of words over an alphabet $\Delta$ 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 $\Delta$. 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.

For the sake of simplicity assume that we have only one equation of the form

$$S_0 \cup X_1 S_1 \cup \cdots \cup X_n S_n = T_0 \cup X_1 T_1 \cup \cdots \cup X_n T_n \qquad (*)$$

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. In the above example, the given solution yields the language $\{aa, aaa, baa\}$.

In order to accept the trees corresponding to the finite sets of words obtained by instantiating the equation $(*)$ by one of its solutions, the automaton guesses at each node whether it (more precisely, the path leading to it) belongs to one of the $X_i$ (more precisely, to the set of words instantiated for $X_i$), and then does the necessary book-keeping to make sure that the concatenation with the elements of $S_i$ and $T_i$ is realized: if $S_i$ contains a word $w$, and the automaton has decided that a given node $\kappa$ belongs to $X_i$, then if one starts at $\kappa$ 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 $T_i$ in place of $S_i$.

The size of the set of states of this automaton turns 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 yields the exponential time algorithm claimed in the theorem.

# 5 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 [9]). This problem can be described as follows: given a sequence $A_1, \ldots, A_n$ of deterministic finite state automata (dfa) over the same input alphabet $\Sigma$, 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 $A$ 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 $A_1, \ldots, A_n$ 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 $A$ over $\Sigma$ with final state $q_f$ and initial state $q_0$, we consider the alphabet $\Delta$ that consists of $\Sigma$ and the states of $A$. We construct the following linear equation, where the variables $X, X_i$ range over finite sets of words over $\Delta$:

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

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

To this purpose, we will consider a more general situation. Let $T$ be a finite set of words over $\Delta$. 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 $\theta$ 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 $\theta$ we consider (since the dfa is deterministic). But this need not be the case for the sets on the lhs.

Let $w$ be an element of $T$. Since $\theta$ 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 rhs. 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 $\beta$, which coincides with $\theta$ 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 $\theta$ (and is different from $w = l_i u$ because it is shorter), it still occurs on the right-hand side for solution $\beta$. 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 $\theta$, the left-hand side of the equation cannot be empty (since the rhs 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.

**Theorem 5.1** *Solvability of ACUIh-unification problems (with free constants) is PSPACE-hard.*

The linear equations corresponding to ACUIh-unification problems introduced in the previous section actually differed from the linear equations considered in this section in that the variables occurred in front of the sets of words rather than behind. However, by going to the mirror languages, one can easily reduce solvability of one type of linear equations to solvability of the other.

# 6 Future and related work

Apart from the technical problem of obtaining a tight complexity bound, the main topic for future work is to extend the decidability result to more expressive DL languages. Another interesting problem is how to cope with the fact that ACUIh-unification (and thus unification of $\mathcal{FL}_0$-concept terms) is of type zero. First, note that a more expressive language might lead to a theory with a better unification type (since in a richer signature there are more substitutions available). Second, it might well be the case that the instantiation ordering on substitutions is not the right ordering to use when dealing with substitutions operating on concept terms. It is possible that another ordering, induced by the subsumption hierarchy, is more appropriate.

Borgida and McGuinness [6] consider matching of concept terms (of the DL languages used by the CLASSIC system [8]) modulo subsumption: for given concept terms $C$ and $D$ they ask for a substitution $\sigma$ such that $C \sqsubseteq \sigma(D)$. More precisely, they are interested in finding "minimal" substitution for which this is the case, i.e., $\sigma$ should satisfy the property that there does not exist another substitution $\delta$ 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. This yields an additional motivation for investigating orderings on unifiers that are induced by subsumption.

# 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 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.

[6] A. Borgida and D.L. McGuiness. 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.

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

[8] R.J. Brachman, D.L. McGuinness, P.F. Patel-Schneider, L.A. Resnick, and A. Borgida. Living with CLASSIC: When and how to use a KL-ONE-like language. In J. Sowa, editor, *Principles of Semantic Networks*, pages 401–456. Morgan Kaufmann, San Mateo, Calif., 1991.

[9] 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.

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

[11] 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.

[12] 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.