# D W Q

Foundations of **D**ata **W**arehouse **Q**uality

National Technical University of Athens (NTUA)
Informatik V & Lehr- und Forschungsgebiet Theoretische Informatik (RWTH)
Institute National de Recherche en Informatique et en Automatique (INRIA)
Deutsche Forschungszentrum für künstliche Intelligenz (DFKI)
University of Rome «La Sapienza» (Uniroma)
Istituto per la Ricerca Scientifica e Tecnologica (IRST)

F. Baader

## Unification of Knowledge Bases

Proc. of the International Workshop on Description Logics 97

Gif sur Yvette, France

**DWQ : ESPRIT Long Term Research Project, No 22469**
**Contact Person : Prof. Yannis Vassiliou, National Technical University of Athens,
15773 Zographou, GREECE Tel +30-1-772-2526 FAX: +30-1-772-2527, e-mail: yv@cs.ntua.gr**

# 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
and

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 substitute 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 Motivation

The first motivation for considering unification of concept terms comes from an application in chemical process engineering [4]. 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

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

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

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

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 substituting 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 $\sigma$, where a substitution replaces (some of the) atomic concepts by concept terms. A substitution is a unifier of $C, D$ iff $\sigma(C) \equiv \sigma(D)$ (where $\equiv$ denotes equivalence of 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.

Another motivation for considering unification of concept terms comes from the work of Borgida and McGuinness [6], 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 $\sigma$ such that $C \sqsubseteq \sigma(D)$. More precisely, they are interested in finding "minimal" substitutions 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.

## 2 Unification of $\mathcal{FL}_0$-concept terms

As a first test case, we have investigated the unification problem for the rather small DL language $\mathcal{FL}_0$, which allows for concept conjunction ($C \sqcap D$), value restriction ($\forall R.C$), and the top concept ($\top$). The semantics of these operators is defined in the usual way. Two $\mathcal{FL}_0$-concept terms are equivalent ($C \equiv D$) iff they denote the same concept in every interpretation (i.e., $C^I = D^I$ 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).

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

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 1** 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*.

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 $Y \sqcap \forall R.Y \sqcap \forall R.\forall S.A$.

### Reduction to an equational unification problem

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

$$(x \wedge y) \wedge z = x \wedge (y \wedge z), \quad x \wedge y = y \wedge x,$$
$$x \wedge x = x, \qquad x \wedge \mathsf{T} = x,$$

and for all $R \in \mathcal{R}$

$$h_R(x \wedge y) = h_R(x) \wedge h_R(y), \quad h_R(\mathsf{T}) = \mathsf{T}.$$

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

**Lemma 2** *Let $C$ and $D$ by $\mathcal{FL}_0$-concept terms. Then*

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

*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 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 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 $\tau(C) =_{\mathrm{ACUIh}} \tau(C')$.

Now, assume that $\tau(C) \neq_{\mathrm{ACUIh}} \tau(D)$. Consequently, the corresponding normal forms $C', D'$ also satisfy $\tau(C') \neq_{\mathrm{ACUIh}} \tau(D')$. 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, $d_0 \in D'^I = D^I$. □

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., $\sigma(t_1) =_{\mathrm{ACUIh}} \sigma(t_2)$.[2]

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

[2]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.

**Lemma 3** *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 $\forall R_1.C_1 \sqcap \cdots \sqcap \forall R_n.C_n \equiv \forall R_1.D_1 \sqcap \cdots \sqcap \forall R_n.D_n$ iff $C_1 \equiv D_1, \ldots, C_n \equiv D_n$.*

**Reduction to solving linear equations**

The theory ACUIh belongs to the class of so-called commutative theories [1], for which solving unification problems can be reduced to solving systems of linear equations over a corresponding semiring [9, 3]. Conversely, every system of linear equations over this semiring corresponds to a unification problem. (A semiring is similar to a ring, but the addition need not have an inverse; e.g., the nonnegative integers with addition and multiplication are a semiring, but not a ring.) Since we have seen above that in the case of ACUIh-unification, solving systems of equations is equivalent to solving a single equation, we can restrict our attention to the problem of how to solve a single linear equation over the semiring corresponding to ACUIh.

This semiring can be described as follows:

- its elements are finite sets of words (over the alphabet 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.

For example, $\{RS, S\}$ and $\{\varepsilon, R\}$ are elements of this semiring, and multiplying them yields $\{RS, S\}\{\varepsilon, R\} = \{RS, S, RSR, SR\}$.

ACUIh-unification problems (consisting w.l.o.g. of a single equation) are translated into linear equations of the form

$$S_0 \cup S_1X_1 \cup \cdots \cup S_nX_n = T_0 \cup T_1X_1 \cup \cdots \cup T_nX_nT_{m,n}X_n \qquad (*)$$

The coefficients $S_i, T_i$ of this equation are semiring elements (i.e., finite sets of words). A solution of this equation assigns finite sets of words to the variables $X_i$ such that the equation holds. For example, the ACUIh-unification problem $h_R(h_R(a)) \wedge h_R(x) =^?_{\text{ACUIh}} y \wedge h_R(y) \wedge h_R(h_S(a))$ from above is translated into the (inhomogeneous) linear equation

$$\{RR\} \cup \{R\}X \cup \emptyset Y = \{RS\} \cup \emptyset X \cup \{\varepsilon, R\}Y,$$

and $X = \{\varepsilon, S\}$, $Y = \{R\}$ is a solution.

**Theorem 4** *Solvability of ACUIh-unification problems (with free constants) can be decided in deterministic exponential time, and the problem is at least PSPACE hard.*

*The decidability result* can be obtained by reducing solvability of linear equations in the above semiring to the emptiness problem for (root-to-frontier) tree automata working on finite trees [8]. 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.

Our approach for solving linear equations with the help of tree automata cannot treat the equation $(*)$ from above directly: 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 of the involved languages. For a word $w = R_1 \dots R_m$, its mirror image is defined as $w^R := R_m \dots R_1$, and for a finite set of words $L = \{w_1, \dots, w_\ell\}$, its mirror image is $L^R := \{w_1^R, \dots, w_\ell^R\}$. Obviously, $X_1 = L_1, \dots, X_n = L_n$ is a solution of $(*)$ iff $Y_1 = L_1^R, \dots, Y_n = L_n^R$ is a solution of the corresponding mirrored equation $(**)$:

$$S_0^R \cup Y_1 S_1^R \cup \dots \cup Y_n S_n^R = T_0^R \cup Y_1 T_1^R \cup \dots \cup Y_n T_n^R \qquad (**)$$

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. To achieve this goal, the automaton guesses at each node whether it (more precisely, the path leading to it) belongs to one of the $Y_i$ (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 $S_i^R$ and $T_i^R$ is realized: if $S_i^R$ contains a word $w$, and the automaton has decided that a given node $\kappa$ belongs to $Y_i$, 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 $T_i^R$ in place of $S_i^R$. 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.

*The hardness result* can be shown by reduction from the Finite State Automata Intersection problem, which has been shown to be PSPACE-complete by Kozen (see [7]). This problem can be described as follows: given a sequence $\mathcal{A}_1, \dots, \mathcal{A}_n$ 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 $\mathcal{A}$ 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 $\mathcal{A}_1, \dots, \mathcal{A}_n$ 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 $\Sigma$ with final state $q_f$ and initial state $q_0$, we consider the alphabet $\Delta$ that consists of $\Sigma$ 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 $\Delta$:

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

It can be shown (see [2]) that for any solution of this equation, $X$ is a nonempty subset of the language accepted by $\mathcal{A}$. In addition, for any finite subset $T$ of this language, there exists a solution of the equation such that $X = T$.

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 system of equations can be translated into a corresponding ACUIh-unification problem, which proves the PSPACE-hardness result.

# 3 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 section, 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 $\Delta$. 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 $\Delta$. The term $\forall\emptyset.A$ is considered to be equivalent to $\top$. Using these abbreviations, the terms $C, D$ can be rewritten as

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

for finite sets of words $S_{0,i}, S_j, T_{0,i}, T_j$ $(i = 1, \ldots, k, j = 1, \ldots, n)$. The following lemma is not hard to show.

**Lemma 5** *$C, D$ are unifiable iff for all $i = 1, \ldots, k$, the linear equation*

$$S_{0,i} \cup S_1 X_1 \cup \cdots \cup S_n X_n = T_{0,i} \cup T_1 X_1 \cup \cdots \cup T_n X_n$$

*has a solution.*

Note that this is not a system of $k$ equations that must be solved simultaneously: each of these equations can be solved separately.

For example, consider the concept terms $C = \forall R.(A_1 \sqcap \forall R.A_2) \sqcap \forall R.\forall S.X_1$ and $D = \forall R.\forall S.(\forall S.A_1 \sqcap \forall R.A_2) \sqcap \forall R.X_1 \sqcap \forall R.\forall R.A_2$. The corresponding concept terms in normal form are $C' = \forall\{R\}.A_1 \sqcap \forall\{RR\}.A_2 \sqcap \forall\{RS\}.X_1$ and $D' = \forall\{RSS\}.A_1 \sqcap \forall\{RSR, RR\}.A_2 \sqcap \forall\{R\}.X_1$, which lead to the two linear equations

$$\begin{aligned} \{R\} \cup \{RS\}X_1 &= \{RSS\} \cup \{R\}X_1 \\ \{RR\} \cup \{RS\}X_1 &= \{RSR, RR\} \cup \{R\}X_1 \end{aligned}$$

The first equation (the one for $A_1$) has $X_1 = \{\varepsilon, S\}$ as a solution, and second (the one for $A_2$) has $X_1 = \{R\}$ as a solution. These two solutions yield the following unifier of $C, D$:

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

# 4 Future work

Beside 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. 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 [6], 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 and P. Narendran. Unification of concept terms. In *Proceedings of the 11th International Workshop on Unification, UNIF-97, LIFO Technical Report 97-8*. LIFO, Universitè de Orlèans, 1997.

[3] 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.

[4] 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.

[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] 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.

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

[9] 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.