*J. Symbolic Computation*

# Unification of Concept Terms in Description Logics

FRANZ BAADER[†] AND PALIATH NARENDRAN[‡]

[†] *Theoretical Computer Science, RWTH Aachen, Germany*
[‡] *Department of Computer Science, State University of New York at Albany, USA*

*(Received )*

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 systems based on Description Logics (DL systems) can be used to represent the terminological knowledge of an application domain in a structured and formally well-understood way (Brachman and Schmolze 1985, Baader and Hollunder 1991, Brachman *et al.* 1991, Woods and Schmolze 1992). With the help of the underlying description language (DL language), 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 $\mathsf{Woman}$ and the atomic role $\mathsf{child}$, 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

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

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

0747–7171/90/000000 + 00 $03.00/0 

and $(\forall \mathsf{child}.\mathsf{Woman}) \sqcap \mathsf{Woman}$ are equivalent since $\sqcap$ is interpreted as set intersection, which is obviously commutative.

The traditional inference problems for DL systems (like subsumption) are now well-investigated, which means that there are algorithms available for solving the subsumption problem and related inference problems in a great variety of DL languages of differing expressive power (see, e.g., (Levesque and Brachman 1987, Schmidt-Schauß and Smolka 1991, Nebel 1990a, Hollunder *et al.* 1990, Baader 1991, Baader and Hanschke 1991, Hollunder and Baader 1991, Buchheit *et al.* 1993, Borgida and Patel-Schneider 1994, Baader and Sattler 1996a, Baader *et al.* 1996, Baader and Sattler 1996a, Baader and Sattler 1996c)). In addition, the computational complexity of these inference problems has been investigated in detail (Levesque and Brachman 1987, Nebel 1988, Nebel 1990b, Donini *et al.* 1991b, Donini *et al.* 1991a, Donini *et al.* 1992, Schaerf 1993, Donini *et al.* 1994). DL systems are employed in various application domains, such as natural language processing (Quantz and Schmitz 1994), configuration of technical systems (Wright *et al.* 1993, Buchheit *et al.* 1994b, Rychtyckyj 1996, McGuinness *et al.* 1995), software information systems (Devanbu *et al.* 1991), optimizing queries to databases (Buchheit *et al.* 1994a, Bergamaschi *et al.* 1997, Bergamaschi and Beneventano 1997), and planning (Koehler 1994).

In some of these applications it has turned out, however, that building and maintaining large DL knowledge bases requires support by additional inference capabilities, which have not been considered in the DL literature until very recently. The present article, which is an extended version of a paper first presented at ECAI'98 (Baader and Narendran 1998), is concerned with such a new inference service, namely, *unification* of concept terms.

Our motivation for considering unification of concept terms comes from an application in chemical process engineering (Baader and Sattler 1996b). 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*[†] 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 replacing the atomic concept $\mathsf{Woman}$ in the first term

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

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 $\sigma(C) \equiv \sigma(D)$. 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 (1996), 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.

In the following, we consider the unification problem for a rather small DL language, called $\mathcal{FL}_0$ in the literature (Baader 1990). This language is not expressive enough to represent all the relevant knowledge in the chemical process engineering application mentioned above. However, since unification is a new and apparently difficult inference problem in Description Logics, we decided to study the problem first for the small language $\mathcal{FL}_0$, and then try to extend the results to larger languages (see Section 9 for a discussion of the extensibility of the results obtained in this article).

We shall see that the unification problem for $\mathcal{FL}_0$ 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 monoidal (or commutative) theory (Baader 1989, Nutt 1990, Baader and Nutt 1996), in which unification can be reduced to solving linear equations in a corresponding semiring. In the case of ACUIh, the corresponding semiring $\mathcal{S}_{\mathrm{ACUIh}}$ is the polynomial semiring (in non-commuting indeterminates) over the Boolean semiring.† The reduction from unification of $\mathcal{FL}_0$-concept terms to solving linear equations in $\mathcal{S}_{\mathrm{ACUIh}}$ can, however, also be obtained without the detour through unification modulo ACUIh.

From an algebraic point of view, $\mathcal{S}_{\mathrm{ACUIh}}$ is a rather unusual semiring. Its elements are finite sets of words over a certain finite alphabet, its addition operation is set union ($\cup$), and its multiplication operation is element-wise concatenation of sets of words ($\cdot$). Consequently, the problem of solving linear equations over this semiring turns out to be the same as the following formal language problem: given finite sets of words $S_0, S_1, \ldots, S_n, T_0, T_1, \ldots, T_n$, we want to know whether there exist finite sets of words $X_1, \ldots, X_n$ such that

$$S_0 \cup S_1 \cdot X_1 \cup \cdots \cup S_n \cdot X_n = T_0 \cup T_1 \cdot X_1 \cup \cdots \cup T_n \cdot X_n.$$

The main contribution of this article is to prove that this problem (and thus also the unification problem for $\mathcal{FL}_0$) is Exptime-complete. Both, the proof that the problem can be solved in (deterministic) exponential time and that the problem is Exptime-hard, is shown with the help of automata working on finite trees.

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

Finally, we consider the matching problem for $\mathcal{FL}_0$-concept terms, and show that it is decidable in polynomial time.

## 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 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 2.2. An *interpretation* $I$ consists of a nonempty 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 $R^I \subseteq \Delta^I \times \Delta^I$. 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 $I$.

It is well-known that subsumption (and thus also equivalence) of $\mathcal{FL}_0$-concept terms can be decided in polynomial time (Levesque and Brachman 1987).

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*.

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

The above unifier $\sigma$ is a *ground* substitution since the concept terms substituted for the variables do not contain variables. If we are only interested in testing unifiability, rather than computing unifiers, then it is sufficient to restrict the attention to ground substitutions: a given unification problem has a unifier iff it has a ground unifier. In fact, given a non-ground unifier $\sigma$, any instance of $\sigma$ obtained by replacing the variables occurring in $\sigma$ by ground terms is also a unifier. But even if we are interested in computing a unifier (or at least the unified term), in the application sketched in the introduction it appears to be more appropriate to compute ground unifiers. As mentioned above, the concept constants (i.e., the elements of $\mathcal{C}_c$) are assumed to be used in the same way by all knowledge engineers, whereas for the concept variables (i.e., the elements of $\mathcal{C}_v$) the intuitive meaning is less clear. Thus, if we want to offer the computed unifier or the unified term to a knowledge engineer for inspection, a ground unifier or a ground term will make more sense to this person.

## 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 (Baader and Siekmann 1994).

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

LEMMA 3.1. *Let $C$ and $D$ be $\mathcal{FL}_0$-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 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 $\tau(C) =_{\mathrm{ACUIh}} \tau(C')$.

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, $d_0 \in D'^I = D^I$. $\square$

As an immediate consequence of this lemma, unification in $\mathcal{FL}_0$ is just a syntactic variant of unification modulo ACUIh. In fact, if $\sigma$ is a unifier of the $\mathcal{FL}_0$-concept terms $C$ and $D$, then the lemma implies that the substitution $\sigma^\tau$ defined as $\sigma^\tau(X) := \tau(\sigma(X)$ $(X \in \mathcal{C}_v)$ is an ACUIh-unifier of $\tau(C)$ and $\tau(D)$.

PROPOSITION 3.2. *The $\mathcal{FL}_0$-concept terms $C$ and $D$ are unifiable iff the corresponding terms $\tau(C)$ and $\tau(D)$ are unifiable modulo ACUIh.*

In our 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 $\sigma^\tau := \{x \mapsto a \wedge h_S(a), y \mapsto h_R(a)\}$ is an ACUIh-unifier of these terms, i.e., $\sigma(t_1) =_{\mathrm{ACUIh}} \sigma(t_2)$.†

In unification theory, one usually considers unification problems that consist of a finite

† 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.

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 3.3. *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*

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

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 $i, 1 \le i \le n$. 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 $j, 1 \le j \le n$. 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. $\square$

For readers that are familiar with unification theory, we want to point out that the *unification type* of ACUIh has already been determined in (Baader 1989): 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 are mainly interested in ground solutions of the unification problems, i.e., in unifiers that do not introduce concept variables (see the remark at the end of Section 2). Anyway, 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 (Baader 1989). In the following, we will show that this problem is decidable, but of a rather high complexity. 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 (Narendran 1996). This theory is also of unification type zero (Baader 1993).

## 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

$$S_0 \cup S_1 \cdot X_1 \cup \cdots \cup S_n \cdot X_n \quad = \quad T_0 \cup T_1 \cdot X_1 \cup \cdots \cup T_n \cdot X_n. \tag{4.1}$$

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 "$S_i \cdot X_i$" for element-wise concatenation of sets of words; for example, $\{SR, S\}\cdot\{R, RR\} = \{SRR, SR, SRRR\}$. The reason for calling such an equation a *linear equation* is that we can view "$\cup$" as addition and "$\cdot$" as multiplication in an appropriate semiring.

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

### 4.1. Monoidal theories and semirings

The theory ACUIh is a so-called monoidal theory (Nutt 1990), for which solving unification problems can be reduced to solving systems of linear equations over a corresponding semiring (Nutt 1990, Baader and Nutt 1996). Conversely, every system of linear equations over this semiring corresponds to a unification problem.

Before considering the semiring corresponding to ACUIh, let us recall the definition of a monoidal theory. Given an equational theory $E$, its signature is the set of function symbols occurring in the identities of $E$.

Definition 4.1. An equational theory $E$ is called *monoidal* iff it satisfies the following properties:

1. its signature contains a binary function symbol $f$ and a constant symbol $e$, and all other function symbols in the signature of $E$ are unary.
2. The symbol $f$ is associative-commutative with unit $e$, i.e., $f(f(x,y),z) =_E f(x, f(y,z))$, $f(x,y) =_E f(y,x)$, and $f(x,e) =_E x$.
3. Every unary function symbol $h$ in the signature of $E$ is a homomorphism for $f$ and $e$, i.e., $h(f(x,y)) =_E f(h(x), h(y))$ and $h(e) =_E e$.

It is an immediate consequence of this definition that ACUIh is monoidal. In fact, we have one binary associative-commutative function symbol, $\wedge$, with unit $\top$, and all additional symbols $h_R$ $(R \in \mathcal{R})$ are homomorphisms for $\wedge$ and $\top$.

In order to determine the semiring corresponding to ACUIh, 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 $\wedge$. Obviously, this theory is also monoidal.† As shown in (Nutt 1990), 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., $(1+0)(x) := 1(x) \wedge 0(x) = x \wedge \top =_{\mathrm{ACUI}} x = 1(x)$. 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 (Baader and Nutt 1996), adding homomorphisms to a monoidal 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 $\Delta$ be the set of these

† Note that the symbols $h_R$ $(R \in \mathcal{R})$ do not belong to the signature of ACUI.

indeterminates (which are w.l.o.g. just the role names). Monomials in $\mathcal{S}_{\mathrm{ACUIh}}$ are simply words over the alphabet $\Delta$, and since the addition operation in the semiring is associative, commutative, and 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 $\Delta$ 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 (Nutt 1990, Baader and Nutt 1996), 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 (4.1). 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 *monoidal* equational theory has allowed us to employ known results from unification theory about the connection between unification modulo monoidal 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 monoidal theory.

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

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

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 4.2. *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 3.1, 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. □

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

THEOREM 4.3. *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 $E_{C,D}(A_i)$:*

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

*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

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

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

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

Lemma 4.2 implies that $\sigma(C) \equiv \sigma(D)$ iff, for all $i = 1, \ldots, k$,

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

Thus, if $\sigma$ 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. Automata on finite trees

Our method for solving linear equations in $\mathcal{S}_{\mathrm{ACUIh}}$ employs automata that work on finite trees. In this section we define tree automata and recall the results that will be used in subsequent sections (see (Gécseg and Steinby 1984, Comon *et al.* 1997) for more information on tree automata).

We consider trees with labels in the ranked alphabet $\Sigma$, where the number of successors of a node is determined by the rank of its label. Obviously, such trees are simply representations of terms over the signature $\Sigma$.

Definition 5.1. Let $\Sigma$ be a finite alphabet, where each $f \in \Sigma$ is associated with a rank $rank(f) \geq 0$, and let $k$ be the maximal rank of the elements of $\Sigma$. A (finite) $\Sigma$-tree is a mapping $t : dom(t) \to \Sigma$ such that $dom(t)$ is a finite subset of $\{1, \ldots, k\}^*$ such that

- the empty word $\varepsilon$ belongs to $dom(t)$;
- for all $u \in \{1, \ldots, k\}^*$ and $i \in \{1, \ldots, k\}$, we have $ui \in dom(t)$ iff $u \in dom(t)$ and $i \leq rank(t(u))$.

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 $\varepsilon$ 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 $t$. By the above definition, the leaves are the nodes labeled with a symbol of rank zero, i.e., $rank(t(u)) = 0$ iff $u$ is a leaf of $t$. We denote the set of symbols of rank 0 by

$$\Sigma_0 := \{f \in \Sigma \mid rank(f) = 0\}.$$

We always assume $\Sigma_0 \neq \emptyset$ since otherwise there is no finite $\Sigma$-tree. The set of all leaves of the tree $t$ 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 $i$th son of $u$ in $t$.

Definition 5.2. A (nondeterministic) *root-to-frontier tree automaton (rfa)* that works on $\Sigma$-trees is a 5-tuple $\mathcal{A} = (\Sigma, Q, I, T, F)$ where

- $\Sigma$ is a finite, ranked alphabet,
- $Q$ is a finite set of states,
- $I \subseteq Q$ is the set of initial states,
- $T$ assigns to each $f \in \Sigma \setminus \Sigma_0$ of rank $n$ a transition relation $T(f) \subseteq Q \times Q^n$, and
- $F : \Sigma_0 \to 2^Q$ assigns to each label $c$ of rank zero a set of final states $F(c) \subseteq Q$.

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

- $(r(u), r(u1), \ldots, r(un)) \in T(t(u))$ for all inner nodes $u$ with label $t(u)$ of rank $n$.

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., (Thomas 1990)):

THEOREM 5.3. *The emptiness problem for root-to-frontier tree automata is decidable in polynomial time.*

We will use this result in the next section to show that solvability of linear equations of the form (4.1) is decidable in deterministic exponential time. The idea is that a given equation is translated into an rfa of size exponential in the size of the equation such that the equation has a solution iff the corresponding automaton accepts a nonempty set of trees. In Section 7 we will show that this is the best we can do since solvability of linear equations of the form (4.1) is also Exptime-hard. This will be proved by a reduction from the intersection emptiness problem of *deterministic* root-to-frontier automata.

DEFINITION 5.4. The rfa $\mathcal{A} = (\Sigma, Q, I, T, F)$ is a *deterministic root-to-frontier automaton (drfa)* iff

- the set $I$ of initial states consists of a single initial state $q_0$,
- for all states $q \in Q$ and all symbols $f$ of rank $n > 0$ there exists exactly one $n$-tuple $(q_1, \ldots, q_n)$ such that $(q, q_1, \ldots, q_n) \in T(f)$.
  For deterministic automata it is often more convenient to use a transition function $\delta$ in place of the (functional) transition relations. This function is defined as $\delta(q, f) := (q_1, \ldots, q_n)$, where $(q_1, \ldots, q_n)$ is the unique tuple satisfying $(q, q_1, \ldots, q_n) \in T(f)$.

It should be noted that deterministic root-to-frontier automata are weaker than nondeterministic ones. For example, the language consisting of the trees (written in term notation) $f(a, a)$ and $f(b, b)$ cannot be accepted by a deterministic root-to-frontier automaton since a drfa accepting these two trees would also accept $f(a, b)$ and $f(b, a)$. It is easy to see that $\{f(a, a), f(b, b)\}$ can be accepted by a nondeterministic rfa (see (Gécseg and Steinby 1984), Example 2.11). More generally, the values assigned by a run $r$ of a drfa to the nodes on a path from the root to a leaf in a given tree are uniquely determined by the labels of the nodes on the path, i.e., they do not depend on labels occurring on other paths. This fact will be important for our reduction.

The *intersection emptiness problem* for deterministic root-to-frontier automata is defined as follows: given a sequence $\mathcal{A}_1, \ldots, \mathcal{A}_n$ of deterministic root-to-frontier automata over the same ranked alphabet $\Sigma$, decide whether there exists a common tree $t$ accepted by each of these automata. The following result is due to Seidl (1994).

THEOREM 5.5. *The intersection emptiness problem for deterministic root-to-frontier automata is Exptime-hard.*

Note that, as a consequence of Theorem 5.3 and the fact that intersection can as usual be handled via product of automata, the problem is polynomial for any fixed number $n$ of automata.

## 6. Solving linear equations in $\mathcal{S}_{\mathrm{ACUIh}}$

In this section, we show that linear equations of the form (4.1) (and thus also ACUIh- and $\mathcal{FL}_0$-unification problems) can be solved in exponential time.

THEOREM 6.1. *Solvability of linear equations in $\mathcal{S}_{\mathrm{ACUIh}}$ can be decided in deterministic exponential time.*

COROLLARY 6.2. *Solvability of unification problems in $\mathcal{FL}_0$ and of ACUIh-unification problems can be decided in deterministic exponential time.*

The main idea underlying our proof of Theorem 6.1 is that (i) finite sets of words can be represented by finite trees; and (ii) the trees that describe solution sets of a given linear equation of the form (4.1) (i.e., the sets of words obtained by inserting the solutions of the equation into one side of the equation) form a tree language that can be accepted by an rfa of size exponential in the size of the equation. It should be noted that the idea of representing sets of words by trees is not new. It goes back at least to Büchi and Rabin, and has also been employed in the area of set constraints.

There are actually different ways of filling in the details of the proof sketched above. One possible way of proving the result would be to translate linear equations of the form (4.1) into formulae of the logic WSkS[†] (where $k$ is the cardinality of the alphabet) such that a given equation is solvable iff the corresponding formula is valid. It is actually not hard to come up with a polynomial translation that has this property. However, since the complexity of deciding validity in WSkS is a lot higher than Exptime, this would show only decidability of the problem without directly yielding the desired complexity result. Decidability of WSkS can be shown by a translation into automata working on finite trees (see, e.g., (Comon *et al.* 1997), Chapter 3.3). By analyzing what this translation does on formulae that encode linear equations of the form (4.1), one could then show that the automaton obtained from such a formula is only exponential in the size of the formula.

† See, e.g., (Thomas 1990, Comon *et al.* 1997) for information on WSkS.

Another way of proving the result could be to adapt results about solvability of set constraints. In fact, linear equations of the form (4.1) are very similar to so-called monadic set constraints, for which an Exptime-completeness result was shown in (Aiken *et al.* 1993). However, in the context of set constraints one usually allows for arbitrary (possibly infinite) sets as solutions, whereas we are only interested in finite solutions. Thus, one would need to check whether the hypergraph technique employed in (Aiken *et al.* 1993) can also be used to decide *finite* solvability of monadic set constraints. Gilleron *et al.* (1994) treat finite solvability of set constraints using automata on infinite trees. Due to the fact that they consider no-monadic set constraints, their automata are quite complex, and the complexity of the decision procedure obtained this way is higher than ours. Thus, one would need to check whether the restriction of their approach to the monadic case really yields an Exptime decision procedure.

We have decided to give a direct reduction to the emptiness problem for tree automata since we think that this reduction is quite simple and instructive, and since we believe that working out the details of one of the above approaches would need (at least) as much space as presenting the reduction from scratch.

In order to represent finite sets of words over an alphabet $\Delta$ of cardinality $k$ by finite trees, we consider the ranked alphabet $\Sigma := \{c_0, c_1, f_0, f_1\}$, where $c_0, c_1$ have rank 0 and $f_0, f_1$ have rank $k$. We can take the set of all words representing paths from the root to nodes having a label with index 1 as the finite set of words represented by the tree. In the following, we assume w.l.o.g. that $\Delta = \{1, \ldots, k\}$.

For a $\Sigma$-tree $t$ we define

$$L(t) := \{u \in dom(t) \mid t(u) = c_1 \text{ or } t(u) = f_1\}.$$

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

Our approach for solving linear equations in $\mathcal{S}_{\mathrm{ACUIh}}$ with the help of tree automata cannot directly treat equations of the form (4.1):

$$S_0 \cup S_1 \cdot X_1 \cup \cdots \cup S_n \cdot X_n = T_0 \cup T_1 \cdot X_1 \cup \cdots \cup T_n \cdot X_n.$$

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 (4.1) 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 (4.1) iff $Y_1 = L_1^{mi}, \ldots, Y_n = L_n^{mi}$ is a solution of the corresponding mirrored equation:

$$S_0^{mi} \cup Y_1 \cdot S_1^{mi} \cup \cdots \cup Y_n \cdot S_n^{mi} \quad = \quad T_0^{mi} \cup Y_1 \cdot T_1^{mi} \cup \cdots \cup Y_n \cdot T_n^{mi}. \tag{6.1}$$

As mentioned above, we want to build a tree automaton that accepts the $\Sigma$-trees representing the finite sets of words obtained by instantiating this equation with its solutions (see Lemma 6.3 below for a more formal formulation of which tree language the automaton is supposed to accept). To achieve this goal, the automaton guesses at each node whether it (more precisely, the path leading to it) belongs to one of the $Y_i$s (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 $\kappa$ belongs to $Y_i$, then if one starts at $\kappa$ and follows the path corresponding to $w$, one must find a node whose label has index 1. Vice versa, every label with index 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 6.1.

Before we can define the automaton corresponding to the (solutions of) equation (6.1), 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 thus the size of $\mathit{suf}(S)$ is quadratic in the size of $S$. For all words $u$ we have $u^{-1}S \subseteq \mathit{suf}(S)$.

The root-to-frontier tree automaton $\mathcal{A}_{6.1} = (\Sigma, Q, I, T, F)$ corresponding to equation (6.1) 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 $\mathcal{A}_{6.1}$ are triples whose first component is a subset of the set of indices of the variables in (6.1), the second component is a finite set of words that are suffixes of words occurring on the left-hand side of (6.1), and the third component is a finite set of words that are suffixes of words occurring on the right-hand side of (6.1). Obviously, the size of $Q$ is exponential in the size of equation (6.1).
  Intuitively, the first component of a state "guesses" to which of the $Y_i$s 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 $Y_i$s 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 (and analogously for the right-hand side).
- For $l \in \{0, 1\}$, the transition relation $T(f_l)$ consists of all tuples
  $$((G_0, L_0, R_0), (G_1, L_1, R_1), \ldots, (G_k, L_k, R_k)) \in Q \times Q^k$$
  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 $i$th son of the current node. If $G_i$ contains $j$, i.e., we have guessed that the word represented by the $i$th 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:

$$\begin{array}{rcl} F(c_0) & := & \{(G, L, R) \mid L = R = \emptyset\}, \\ F(c_1) & := & \{(G, L, R) \mid L = R = \{\varepsilon\}\}. \end{array}$$

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 6.3. *For any $\Sigma$-tree $t$, the following statements are equivalent:*

1. $t \in \mathcal{L}(\mathcal{A}_{6.1})$.
2. *There are finite sets of words $\theta(Y_1), \ldots, \theta(Y_n)$ such that*

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

PROOF. $(1 \rightarrow 2)$: If $t \in \mathcal{L}(\mathcal{A}_{6.1})$, then there exists a successful run $r$ of $\mathcal{A}_{6.1}$ on $t$. From the first components of the states assigned to the nodes of $t$ by $r$ we can read off appropriate sets $\theta(Y_1), \ldots, \theta(Y_n)$: if the first component of the state assigned to the node $\kappa$ contains $i$, then the word represented by $\kappa$ belongs to $\theta(Y_i)$. To be more precise, for each $u \in dom(t)$, let $(G_u, L_u, R_u) \in Q$ be such that $r(u) = (G_u, L_u, R_u)$. Then we define

$$\theta(Y_i) := \{u \in dom(t) \mid i \in G_u\}.$$

The definition of $\mathcal{A}_{6.1}$ makes sure that this assignment of finite sets of words to the variables in (6.1) satisfies the identities in statement 2 of the lemma. In order to show this, we concentrate on the first identity. (The second can be shown analogously.)

First, assume that $w \in \theta(Y_i)\cdot S_i^{mi}$ for some $i, 1 \leq i \leq n$. (The case $w \in S_0^{mi}$ can be treated similarly.) Thus, $w = uv$ for words $u \in \theta(Y_i)$ and $v \in S_i^{mi}$. By definition of $\theta(Y_i)$, this implies $i \in G_u$, and thus the definition of the initial states (if $u = \varepsilon$) or of the transition relation (if $u \neq \varepsilon$) imply that $v \in L_u$. Again by definition of the transition relation, this yields $\varepsilon \in L_{uv}$. Thus, the index of $t(uv)$ is 1 (by definition of the transition relation or of the final states), which shows $w = uv \in L(t)$.

Conversely, assume that $w \in L(t)$, i.e., the index of $t(w)$ is 1. By definition of the transition relation and of the final states, this implies that $\varepsilon \in L_w$. By definition of the initial states and of the transition relation this can only be the case if (i) $w \in S_0^{mi}$, or (ii) $w = uv$, $i \in G_u$, and $v \in S_i^{mi}$ for some $i, 1 \leq i \leq n$. In the first case, we are done, and in the second case we know that $u \in \theta(Y_i)$, which shows that $w = uv \in \theta(Y_i)\cdot S_i^{mi}$.

$(2 \rightarrow 1)$: Let $\theta(Y_1), \ldots, \theta(Y_n)$ be an assignment of finite sets of words to the variables $Y_i$ such that the identities in statement 2 of the lemma hold. This assignment can be used to determine appropriate first components of states for a run of $\mathcal{A}_{6.1}$ on $t$: for all $u \in dom(t)$ we define

$$G_u := \{i \mid u \in \theta(Y_i)\}.$$

Once these first components are fixed, the full run of $\mathcal{A}_{6.1}$ on $t$ can be reconstructed. In fact, the nondeterminism of $\mathcal{A}_{6.1}$ is only due to the choice of the first component of the states. Thus, the other two components are uniquely determined. To be more precise, we define $r : dom(t) \to Q$ by $r(u) := (G_u, L_u, R_u)$, where

$$L_\varepsilon = S_0^{mi} \cup \bigcup_{i \in G_\varepsilon} S_i^{mi} \quad \text{and} \quad R_\varepsilon = T_0^{mi} \cup \bigcup_{i \in G_\varepsilon} T_i^{mi},$$

and for $ui \in dom(t)$

$$L_{ui} = i^{-1} L_u \cup \bigcup_{j \in G_{ui}} S_j^{mi} \quad \text{and} \quad R_{ui} = i^{-1} R_u \cup \bigcup_{j \in G_{ui}} T_j^{mi}.$$

The fact that the equations in statement 2 are satisfied guarantees that $r$ is indeed a successful run of $\mathcal{A}_{6.1}$ on $t$.

For example, assume that $t(u) = f_l$ for $l \in \{0, 1\}$ and that $\varepsilon \in L_u$. We must show that $l = 1$ and $\varepsilon \in R_u$. (All the other cases can be treated similarly.) From $\varepsilon \in L_u$ we can deduce that $u \in S_0^{mi} \cup \theta(Y_1) \cdot S_1^{mi} \cup \cdots \cup \theta(Y_n) \cdot S_n^{mi}$ by definition of $r$. Since the identities in statement 2 of the lemma are assumed to be satisfied, this implies $u \in L(t)$ and $u \in T_0^{mi} \cup \theta(Y_1) \cdot T_1^{mi} \cup \cdots \cup \theta(Y_n) \cdot T_n^{mi}$. Now, $u \in L(t)$ immediately yields $l = 1$, and it is easy to see that $u \in T_0^{mi} \cup \theta(Y_1) \cdot T_1^{mi} \cup \cdots \cup \theta(Y_n) \cdot T_n^{mi}$ implies $\varepsilon \in R_u$ (by definition of $r$). $\square$

As an immediate consequence of this lemma we obtain that equation (6.1) has a solution iff $\mathcal{L}(\mathcal{A}_{6.1}) \neq \emptyset$. Since the emptiness problem can be decided in time polynomial in the size of $\mathcal{A}_{6.1}$, and since $\mathcal{A}_{6.1}$ is exponential in the size of (6.1), this completes the proof of Theorem 6.1.

## 7. Solving linear equations in $\mathcal{S}_{\mathrm{ACUIh}}$ is Exptime-hard

We show in this section that the problem of testing linear equations of the form (4.1) for solvability is Exptime-hard. Consequently, this problem as well as solvability of ACUIh- and $\mathcal{FL}_0$-unification problems are Exptime-complete problems. As already mentioned above, Exptime-hardness will be shown by a reduction from the intersection emptiness problem of *deterministic* root-to-frontier automata (drfa).

In principle, the reduction works as follows:

- Each finite $\Sigma$-tree $t$ is encoded by a finite set of words $S(t)$. Let us call such a set of words a tree set.
- A given drfa $\mathcal{A}$ can be translated into an equation of the form (4.1). This equation contains a special variable $X$ and is such that, for each tree $t$ accepted by $\mathcal{A}$, there is a solution assigning $S(t)$ to $X$. The converse is not true, however, i.e., not every solution $\theta$ of the equation satisfies $\theta(X) = S(t)$ for some tree $t$ accepted by $\mathcal{A}$. Intuitively, it should be clear that this cannot be the case since the solutions of equations of the form (4.1) are closed under union, i.e., if $\theta_1$ and $\theta_2$ are solutions of such an equation, then so is $\theta$, where $\theta(X) := \theta_1(X) \cup \theta_2(X)$ for all variables $X$. We will show that the equation obtained from a given drfa $\mathcal{A}$ is such that any solution $\theta$ satisfies $\theta(X) = S(t_1) \cup \ldots \cup S(t_n)$ for some trees $t_1, \ldots, t_n$ accepted by $\mathcal{A}$. Conversely, for any such union $S(t_1) \cup \ldots \cup S(t_n)$ there is a solution $\theta$ satisfying $\theta(X) = S(t_1) \cup \ldots \cup S(t_n)$.
- Given a sequence of drfa $\mathcal{A}_1, \ldots, \mathcal{A}_n$, we construct the corresponding equations (called $(1), \ldots, (n)$ in the following), where the special variable $X$ is the only one

shared by the different equations. As shown in Section 3 (see Lemma 3.3), the system of equations $(1), \ldots, (n)$ can be represented by a single equation (called (0) in the following). It the remains to be shown that (0) has a solution iff the automata $\mathcal{A}_1, \ldots, \mathcal{A}_n$ accept a common tree.
If the automata $\mathcal{A}_1, \ldots, \mathcal{A}_n$ accept a common tree $t$, then each equation $(i), 1 \leq i \leq n$, has a solution $\theta_i$ such that $\theta_i(X) = S(t)$. Since $X$ is the only variable shared by the equations $(1), \ldots, (n)$, the solutions $\theta_1, \ldots, \theta_n$ can be combined into a solution of the system $(1), \ldots, (n)$, and thus of (0).
Conversely, any solution $\theta$ of (0) is also a solution of the equations $(1), \ldots, (n)$. Consequently, for each $i, 1 \leq i \leq n$, there exist trees $t_{i,1}, \ldots, t_{i,m_i}$ accepted by $\mathcal{A}_i$ such that $\theta(X) = S(t_{i,1}) \cup \ldots \cup S(t_{i,m_i})$. The remaining obstacle is that the set $\theta(X)$ can be represented as the union of many different tree sets, and thus it is not clear that the same tree is accepted by all the automata.

- This obstacle is finally removed by showing the following: If $t_1, \ldots, t_m$ are trees accepted by the drfa $\mathcal{A}$, and $t$ is an arbitrary tree such that $S(t) \subseteq S(t_1) \cup \ldots \cup S(t_m)$, then $t$ is also accepted by $\mathcal{A}$. (For this to hold it is crucial that the automata are deterministic.) As an immediate consequence, $t_{1,1}$ is a tree accepted by all the automata $\mathcal{A}_1, \ldots, \mathcal{A}_n$.

For technical reasons, the details of the reduction are actually a bit more complex. In particular, it is not enough to look at tree sets only. We also need to consider sets of words obtained from trees where the leaves are labeled with states of the respective automaton (see the definition of run trees and run tree sets below).

Before starting to describe the reduction in detail, we show that we can restrict our attention to ranked alphabets containing only symbols of rank $\leq 2$, and containing a single symbol of rank 0. Indeed, the Exptime-hardness proof of Seidl (1994) shows that it is sufficient to restrict the attention to alphabets containing only symbols of rank $\leq 2$. In addition, it will be convenient to assume that the alphabet contains exactly one symbol $\sharp$ of rank 0 (i.e., all the leaves are labeled with $\sharp$), and that the drfa under consideration has exactly one final state $q_f$, i.e., its final assignment is $F(\sharp) = \{q_f\}$. In fact, we can simply turn the original symbols of rank 0 into symbols of rank 1, and add the new symbol $\sharp$ of rank 0 to $\Sigma$. For a symbol $a$ of original rank 0, the final assignment $I(a)$ is replaced by a transition satisfying $\delta(q, a) = q_f$ iff $q \in F(a)$.[†] Obviously, this transformation can be done such that the original automaton $\mathcal{A}$ accepts the tree $t$ iff the new automaton $\mathcal{A}_\sharp$ accepts the modified tree $\widehat{t}$ that is obtained from $t$ by adding a son labeled with $\sharp$ to every leaf of $t$. If we apply this transformation to automata $\mathcal{A}_1, \ldots, \mathcal{A}_n$, then the resulting automata accept a common tree iff the original ones did.

In the following, $\Sigma_\sharp$ denotes a ranked alphabet such that $\sharp \in \Sigma_\sharp$ is the only symbol of rank 0, and $\Sigma := \Sigma_\sharp \setminus \{\sharp\}$ contains only symbols of rank 1 and 2.

### Tree sets and run tree sets

Let $t$ be a $\Sigma_\sharp$-tree. We represent such a tree by a set $S(t)$ of words over the alphabet $\Sigma_\sharp \cup \{1, 2\}$, where each word describes a path from a leaf to the root of the tree.[‡] The

[†] To be more precise, if $q \notin F(a)$, then $\delta(q, a)$ is a new non-accepting state which reproduces itself.

[‡] There may be other ways of representing $\Sigma_\sharp$-trees by sets of words; we have chosen this one since it has turned out to be convenient for our reduction.

symbols 1 and 2 are used to represent the left and the right son of a node, respectively. For example, the tree $t := f(g(\sharp,\sharp),h(\sharp))$ yields the set $S(t) := \{\sharp 1g1f, \sharp 2g1f, \sharp 1h2f\}$. More generally, we define the *tree set* induced by a $\Sigma_\sharp$-tree $t$ by induction on the structure of $t$:

- $S(\sharp) := \{\sharp\}$;
- $S(h(t)) := \{u1h \mid u \in S(t)\}$ for all $h \in \Sigma$ of rank 1;
- $S(g(t_1,t_2)) := \{u1g \mid u \in S(t_1)\} \cup \{u2g \mid u \in S(t_2)\}$ for all $g \in \Sigma$ of rank 2.

In order to simulate a given drfa by a linear equation of the form (4.1), it is not sufficient to look at such tree sets only. We must also consider sets induced by trees that represent intermediate configurations of a run of the drfa on a given tree.

Assume that $\mathcal{A}$ is a drfa over $\Sigma_\sharp$ with set of states $Q$, final state $q_f$, and initial state $q_0$. We consider the ranked alphabet $\Sigma \cup Q$, where the states in $Q$ are assumed to be symbols of rank 0. Given a $(\Sigma \cup Q)$-tree, we denote by $t^\sharp$ the $\Sigma_\sharp$-tree that is obtained from $t$ by replacing each symbol in $Q$ by $\sharp$.

DEFINITION 7.1. The $(\Sigma \cup Q)$-tree $t$ is a *run tree* for the drfa $\mathcal{A}$ iff the unique run $r$ on $t^\sharp$ with $r(\varepsilon) = q_0$ is such that $r(u) = t(u)$ for all leaves $u \in dom(t) = dom(t^\sharp)$.

For example, assume that $f, g$ are binary symbols, and let $\delta(q_0, f) := (q_1, q_f)$ and $\delta(q_1, g) := (q_f, q_f)$. Then the unique run starting with $q_0$ at the root of the tree $f(g(\sharp,\sharp),\sharp)$ labels the root with $q_0$, its left son with $q_1$, and all leaves with $q_f$. Consequently, $f(g(q_f,q_f),q_f)$ is a run tree for $\mathcal{A}$. Note that $q_0$ and $f(q_1,q_f)$ are also run trees. Obviously, if all the leaves of a given run tree $t$ are labeled with $q_f$, then the tree $t^\sharp$ obtained from $t$ by replacing the label $q_f$ by $\sharp$ is accepted by the drfa $\mathcal{A}$.

The function $S$ that assigns tree sets to trees can be extended to $(\Sigma \cup Q)$-trees in the obvious way. If $t$ is a run tree for $\mathcal{A}$, the $S(t)$ is called a *run tree set* for $\mathcal{A}$. For example, the run tree $f(g(q_f,q_f),q_f)$ yields the run tree set $\{q_f1g1f, q_f2g1f, q_f2f\}$.

As mentioned in the outline of the reduction given above, the following lemma will become relevant in the proof of correctness of the reduction.

LEMMA 7.2. *Let $t_1,\ldots,t_m$ be run trees for the drfa $\mathcal{A}$, and let $t$ be a $(\Sigma \cup Q)$-tree. If $S(t) \subseteq S(t_1) \cup \ldots \cup S(t_m)$, then $t$ is also a run tree for $\mathcal{A}$.*

PROOF. Let $r$ be the unique run of $\mathcal{A}$ on $t^\sharp$ with $r(\varepsilon) = q_0$, and let $u \in dom(t) = dom(t^\sharp)$ be a leaf. We must show that $t(u)$ coincides with $r(u)$.

There exists a word $w \in \Sigma \cup \{1,2\}$ such that the path from $u$ to the root of $t$ is represented by $t(u)w \in S(t)$. Since $S(t) \subseteq S(t_1) \cup \ldots \cup S(t_m)$, there exists $j, 1 \leq j \leq m$, such that $t(u)w \in S(t_j)$. Obviously, $t(u)w \in S(t_j)$ implies that $u$ is also a leaf in $t_j$ and that $t_j(u) = t(u)$.

Let $r'$ be the unique run of $\mathcal{A}$ on ${t_j}^\sharp$ with $r'(\varepsilon) = q_0$. On the one hand, since $t_j$ is a run tree, we know that $t_j(u) = r'(u)$. On the other hand, since $\mathcal{A}$ is deterministic, the values assigned by a run of $\mathcal{A}$ to the nodes on a path are uniquely determined by the labels of the nodes on the path; they do not depend on any other nodes of the tree. Since $t(u)w$ describes both a path in $t$ and in $t_j$, we thus know that $r$ and $r'$ coincide on the nodes on this path. In particular, this yields $r(u) = r'(u)$, and thus $t(u) = t_j(u) = r'(u) = r(u)$. $\square$

Simulating a drfa by a linear equation

Let $\mathcal{A}$ be a drfa over $\Sigma_\sharp$ with set of states $Q$, transition function $\delta$, initial state $q_0$, and exactly one final state $q_f$, i.e., with final assignment $F(\sharp) = \{q_f\}$. The linear equation†

$$\{q_f\}{\cdot}X \cup \bigcup_{(q,g)\in Q\times\Sigma} \{q\}{\cdot}X_{(q,g)} = \{q_0\} \cup \bigcup_{\delta(q,g)=(q_1,\ldots,q_k)} \{q_1 1g, \ldots, q_k kg\}{\cdot}X_{(q,g)} \tag{7.1}$$

corresponding to $\mathcal{A}$ uses the extended alphabet

$$\Delta := \Sigma \cup Q \cup \{1,2\},$$

that is, the variables $X, X_{(q,g)}$ range over finite sets of words over $\Delta$. Obviously, the size of this equation is polynomial in the size of the drfa $\mathcal{A}$.

We *claim* that, for any solution $\theta$ of equation (7.1), the set $\{q_f\}{\cdot}\theta(X)$ is a finite union of run tree sets (see Lemma 7.5 below). Since the leaves of the run trees $t_i$ generating this set are all labeled with $q_f$, this means that the corresponding trees ${t_i}^\sharp$ are all accepted by the drfa $\mathcal{A}$. To be more precise, $\{q_f\}{\cdot}\theta(X) = \bigcup_{i=1}^m S(t_i)$ for run trees $t_1, \ldots, t_m$ whose leaves are labeled with $q_f$, and thus the trees ${t_1}^\sharp, \ldots, {t_m}^\sharp$ are accepted by $\mathcal{A}$. Before giving a formal proof of this claim, we illustrate it by an example.

Example 7.3. Let $\Sigma = \{f\}$ for a binary symbol $f$. We consider the drfa $\mathcal{A}$ with states $Q := \{q_0, q_1, q_f\}$, initial state $q_0$, final state $q_f$ (i.e., $F(\sharp) = q_f$), and the transition function

$$\delta(q_0, f) := (q_1, q_f), \quad \delta(q_1, f) := (q_f, q_f), \quad \text{and} \quad \delta(q_f, f) := (q_f, q_f).$$

It is easy to see that $q_0$, $f(q_1, q_f)$, and $f(f(q_f, q_f), q_f)$ are run trees. The last run tree shows that $f(f(\sharp,\sharp),\sharp)$ is accepted by $\mathcal{A}$.

The linear equation corresponding to $\mathcal{A}$ is of the form

$$\begin{aligned}\{q_f\}{\cdot}X \cup \{q_0\}{\cdot}X_{(q_0,f)} \cup \{q_1\}{\cdot}X_{(q_1,f)} \cup \{q_f\}{\cdot}X_{(q_f,f)} \quad = \quad & \{q_0\} \cup \\ & \{q_1 1f, q_f 2f\}{\cdot}X_{(q_0,f)} \cup \\ & \{q_f 1f, q_f 2f\}{\cdot}X_{(q_1,f)} \cup \\ & \{q_f 1f, q_f 2f\}{\cdot}X_{(q_f,f)}.\end{aligned}$$

Assume that $\theta$ is a solution of this equation. Since $q_0$ occurs in the right-hand side, it must also occur in the left-hand side. Obviously, this is only possible if $\varepsilon \in \theta(X_{(q_0,f)})$. This implies that $q_1 1f$ and $q_f 2f$ occur in the right-hand side, and thus they must also occur in the left-hand side. Note that this simulates the effect of the transition function applied to $(q_0, f)$ at the root.

Let us first continue with $q_f 2f$. There are two possibilities for this word to occur in the left-hand side. On the one hand, we could have $2f \in \theta(X)$. In this case, no additional words are forced to occur on the right-hand side (i.e., the recursion stops). On the other hand, we could have $2f \in \theta(X_{(q_f,f)})$. In this case, $q_f 1f2f$ and $q_f 2f2f$ must occur in the right-hand side, and the recursion continues. To make things more simple, let us assume that $2f \in \theta(X)$ and $2f \notin \theta(X_{(q_f,f)})$ (in principle, it could, of course, also occur in both).

Now, let us consider $q_1 1f$. Since this word occurs in the left-hand side, we can deduce

† By our assumption on $\Sigma$, we know that $k$ is always in $\{1,2\}$. We use the notation $(q_1, \ldots, q_k)$ etc. to avoid having to distinguish explicitly between the cases $k = 1$ and $k = 2$.

that $1f \in \theta(X_{(q_1,f)})$, and thus $q_f1f1f$ and $q_f2f1f$ occur in the right-hand side. Consequently, both words also occur in the left-hand side. For simplicity, let us assume that this is the case because $1f1f$ and $2f1f$ belong to $\theta(X)$.

If no other words except the ones explicitly mentioned above occur in the images of the variables, then

$$\theta(X) = \{2f, 1f1f, 2f1f\}, \;\; \theta(X_{(q_0,f)}) = \{\varepsilon\}, \;\; \theta(X_{(q_1,f)}) = \{1f\}, \;\; \text{and} \;\; \theta(X_{(q_f,f)}) = \emptyset.$$

It is easy to see that this really is a solution of the above linear equation. The set $\{q_f\}\cdot\theta(X)$ is the run tree set corresponding to the run tree $f(f(q_f, q_f), q_f)$.

Intuitively, the occurrence of $q_0$ in the right-hand side starts the run with $q_0$ at the root. The fact that the variables of the form $X_{(q,g)}$ occur both in the left- and the right-hand side causes a recursion that simulates iterated application of the transition function. Finally, the occurrence of $\{q_f\}\cdot X$ in the left-hand side guarantees that one can terminate the recursion once a leaf labeled by the final state is reached.

To give a formal proof of the claim (and its converse), we consider a more general situation. Let $T$ be a finite set of words over $\Delta$. We consider the equation 7.1($T$) that is obtained from (7.1) by replacing $\{q_f\}\cdot X$ by $T$:

$$T \cup \bigcup_{(q,g)\in Q\times\Sigma} \{q\}\cdot X_{(q,g)} = \{q_0\} \cup \bigcup_{\delta(q,g)=(q_1,\ldots,q_k)} \{q_1 1g, \ldots, q_k kg\}\cdot X_{(q,g)}. \qquad 7.1(T)$$

Lemma 7.4. *Let $\theta$ be a solution of* 7.1($T$), *i.e.,*

$$T \cup \bigcup_{(q,g)\in Q\times\Sigma} \{q\}\cdot\theta(X_{(q,g)}) = \{q_0\} \cup \bigcup_{\delta(q,g)=(q_1,\ldots,q_k)} \{q_1 1g, \ldots, q_k kg\}\cdot\theta(X_{(q,g)}).$$

*If $w$ is a word of maximal length in this set, then $w \in T$. In particular, this implies $T \neq \emptyset$.*

Proof. If $w \notin T$, then $w \in \bigcup_{(q,g)\in Q\times\Sigma}\{q\}\cdot\theta(X_{(q,g)})$. Consequently, $w = qu$ for a word $u \in \theta(X_{(q,g)})$. This implies that, for some state $q_i$, the longer word $q_i igu$ occurs on the right-hand side of the equation, which contradicts the maximality of $w$.

Since at least $q_0$ occurs in $\{q_0\} \cup \bigcup_{\delta(q,g)=(q_1,\ldots,q_k)}\{q_1 1g, \ldots, q_k kg\}\cdot\theta(X_{(q,g)})$, there always exists such a maximal word $w$, which shows that $T$ is nonempty. □

Lemma 7.5. *Equation* 7.1($T$) *has a solution iff $T$ is a finite union of run tree sets.*

Proof. (1) Let us first consider the *if-direction*. Since the set of solutions of linear equations of the form (4.1) is closed under union, it is sufficient to consider the case where $T$ is a run tree set, i.e., there exists a run tree $t$ such that $T = S(t)$. Let $r$ be the unique run of $\mathcal{A}$ on $t^\sharp$ with $r(\varepsilon) = q_0$. The tree $t$ together with the run $r$ can be used to define a solution $\theta$ of 7.1($T$). To this purpose, we consider the inner nodes of $t$, i.e., nodes $u \in dom(t)$ such that $u$ is not a leaf. Such a node $u$ has a corresponding word $w \in \Delta^+$ that describes the path in $t$ (and in $t^\sharp$) from $u$ to the root. For example, the inner node 12 in $f(f(q_f, g(q_f, q_f)), q_f)$ has the corresponding word $g2f1f$.† For $q \in Q$ and $g \in \Sigma$ we

† Note that the words $u$ (12 in the example) describe the path "from the root to the node", whereas the words $w$ ($g2f1f$ in the example) describe the path "from the node to the root."

define

$$\theta(X_{(q,g)}) := \{v \in \Delta^* \mid gv \text{ corresponds to an inner node } u \text{ of } t \text{ such that } r(u) = q \text{ and } t(u) = g\}.$$

In order to show that $\theta$ is in fact a solution of 7.1($T$), we first assume that $w \in \Delta^*$ belongs to the left-hand side, i.e., $w \in T$ or $w \in \{q\}{\cdot}\theta(X_{(q,g)})$ for some $q \in Q$ and $g \in \Sigma$. We consider the case $w \in \{q\}{\cdot}\theta(X_{(q,g)})$, i.e., $w = qv$ for some $v \in \theta(X_{(q,g)})$. (The other case can be treated similarly.) By definition of $\theta$, the word $gv$ corresponds to an inner node $u$ of $t$ such that $r(u) = q$ and $t(u) = g$. If $v = \varepsilon$, then $u$ is the root of $t$, and thus $q = r(u) = q_0$, which shows that $w = qv = q = q_0$ also belongs to the right-hand side. Otherwise, there exists $i \in \{1, 2\}$ and $f \in \Sigma$ such that $v = ifv'$ for some $v' \in \Delta^*$. Let $u'$ be such that $u = u'i$ (i.e., $u'$ is the node to which the word $fv'$ corresponds), let $q'$ be such that $r(u') = q'$, and let $\delta(q', f) = (q_1, \ldots, q_k)$. Since $r$ is a run, we know that $q = q_i$. In addition, $v' \in \theta(X_{(q',f)}$ by definition of $\theta$. Consequently, $w = q_i ifv' \in \{q_i if\}{\cdot}\theta(X_{(q',f)})$ also belongs to the right-hand side.

Conversely, assume that $w \in \Delta^*$ belongs to the right-hand side, i.e., $w = q_0$ or $w = q_i igv$ for some $q_i \in Q$, $i \in \{1, 2\}$, and $g \in \Sigma$ such that $v \in \theta(X_{(q,g)})$ for some $q \in Q$ with $\delta(q, g) = (q_1, \ldots, q_k)$. Again, we restrict our attention to the second case. Let $u'$ be the node of $t$ to which the word $gv$ corresponds. There are again two cases to be distinguished: $u'i$ is either an inner node or a leaf. If $u'i$ is an inner node, then it is easy to see that $igv$ belongs to $\theta(X_{(q_i,f)})$, where $f = t(u'i)$, and thus $w = q_i igv$ belongs to the left-hand side. If $u'i$ is a leaf, then it is easy to see that $w = q_i igv$ belongs to $T = S(t)$ since $r(u'i) = q_i$.

(2) Now, let us consider the *only-if direction*, i.e., assume that $\theta$ is a solution of 7.1($T$). We proof the statement simultaneously for all finite sets $T$ by induction on the size of the solution $\theta$, where the size of $\theta$ is the sum of the cardinalities of the sets $\theta(X_{(q,g)})$. Note that this sum is a well-defined natural number since the sets $\theta(X_{(q,g)})$ are finite.

Let $w$ be a word of maximal length in

$$T \cup \bigcup_{(q,g)\in Q\times\Sigma} \{q\}{\cdot}(X_{(q,g)}) = \{q_0\} \cup \bigcup_{\delta(q,g)=(q_1,\ldots,q_k)} \{q_1 1g, \ldots, q_k kg\}{\cdot}\theta(X_{(q,g)}).$$

By Lemma 7.4 we know that $w \in T$. Again, note that such a maximal word exists since the sets $\theta(X_{(q,g)})$ are finite and the above set is nonempty (since it contains at least $q_0$).

*Case 1:* If $w = q_0$, then the maximality of $w$ implies that $\theta(X_{(q,g)}) = \emptyset$ for all pairs $(q, g) \in Q \times \Sigma$. Consequently, $T = \{q_0\}$, and this is obviously a finite union of run tree sets.

*Case 2:* Assume that $w = p_i ifu$ for a word $u \in \theta(X_{(p,f)})$, let $\ell$ be the rank of $f$, and let $\delta(p, f) := (p_1, \ldots, p_\ell)$. Since $w$ is of maximal length, all the words $p_j jfu$ for $j \in \{1, \ldots, \ell\}$ are of maximal length as well, and thus they are all contained in $T$.

We define a new substitution $\theta'$ and a new set $T'$ as follows:

- $\theta'(X_{(p,f)}) := \theta(X_{(p,f)}) \setminus \{u\}$, and
- on all other variables $\theta'$ coincides with $\theta$.

The set $T'$ is obtained from $T$ by

- adding $pu$,
- for each $j \in \{1, \ldots, \ell\}$, removing $p_j jfu$ unless it is contained in $\bigcup_{\delta(q,g)=(q_1,\ldots,q_k)} \{q_1 1g, \ldots, q_k kg\}{\cdot}\theta'(X_{(q,g)})$.

Obviously, the substitution $\theta'$ is smaller than the substitution $\theta$. Thus, we can apply the induction hypothesis to $T'$ provided that we can show that $\theta'$ solves 7.1($T'$).

The only difference between the old and the new right-hand side of the equation is that some of the words $p_j j f u$ may have been removed from the new right-hand side. However, in this case we have also removed these words from $T'$. In addition, they cannot occur in one of the sets $\{p_j\}\cdot\theta'(X_{(p_j,g)})$ since this would contradict the maximality of $w$.

On the left-hand side, the fact that $pu$ does not occur in $\{p\}\cdot\theta'(X_{(p,f)}$ is compensated by the fact that $pu \in T'$. Thus, the only difference between the old and the new left-hand side is that some of the words $p_j j f u$ do not belong to $T'$. However, these are exactly the words that do not belong to the new right-hand side.

To sum up, we have shown that $\theta'$ solves 7.1($T'$), and thus we know by induction that $T'$ is a finite union of run tree sets, i.e., $T' = S(t_1) \cup \ldots \cup S(t_m)$ for run trees $t_1, \ldots, t_m$. We want to show that $T$ is also a finite union of run tree sets.

First, assume that $pu \notin T$. Thus, we have $T = (T' \setminus \{pu\}) \cup \{p_1 1 f u, \ldots p_\ell \ell f u\}$. Let $t_i$ be such that $pu \in S(t_i)$. Since $\delta(p, f) = (p_1, \ldots, p_\ell)$, the tree $t_i'$ that is obtained from $t_i$ by replacing the leaf corresponding to the word $pu$ by the tree $f(p_1, \ldots, p_\ell)$ is also a run tree. In addition, $S(t_i') = (S(t_i) \setminus \{pu\}) \cup \{p_1 1 f u, \ldots p_\ell \ell f u\}$. Thus, if we replace every tree $t_i$ with $pu \in S(t_i)$ by the corresponding tree $t_i'$, we can represent $T$ as a finite union of run tree sets.

If $pu \in T$, then we simply add one of the trees $t_i'$ without removing the corresponding tree $t_i$. $\square$

### The main theorem

We are now ready to prove the main theorem of this section.

THEOREM 7.6. *Solvability of linear equations in $\mathcal{S}_{\mathrm{ACUIh}}$ is Exptime-hard.*

PROOF. We prove that the intersection problem for deterministic root-to-frontier automata, which is known to be Exptime-hard (Seidl 1994), can be reduced to the problem of solving linear equations in $\mathcal{S}_{\mathrm{ACUIh}}$.

Thus, let $\mathcal{A}_1, \ldots, \mathcal{A}_n$ be a sequence of deterministic root-to-frontier automata. For each automaton, we construct the corresponding linear equation, where the variable $X$ is the only one shared by the different equations. Let $(i)$ denote the equation corresponding to $\mathcal{A}_i$, and let (0) be the single linear equation representing the system of equations $(1), \ldots, (n)$ (see Lemma 3.3). The size of each equation $(i)$ is polynomial in the size of $\mathcal{A}_i$, an thus the size of (0) is polynomial in the size of the sequence $\mathcal{A}_1, \ldots, \mathcal{A}_n$. We show that (0) is solvable iff $\mathcal{A}_1, \ldots, \mathcal{A}_n$ accept a common tree.

If there is a tree $\widehat{t}$ that is accepted by each of the automata $\mathcal{A}_1, \ldots, \mathcal{A}_n$, then the tree $t$ that is obtained from $\widehat{t}$ by replacing $\sharp$ by $q_f$ is a run tree for each of the automata, and $\widehat{t} = t^\sharp$. Consequently, the if-direction of Lemma 7.5 implies that equation $(i)$ corresponding to the automaton $\mathcal{A}_i$ has a solution $\theta_i$ such that $\{q_f\}\cdot\theta_i(X) = S(t)$. Since $X$ is the only variable shared by the equations, and since the solutions $\theta_i$ coincide on $X$, there is a substitution $\theta$ that solves the equations $(i)$ simultaneously. Consequently, $\theta$ is also a solution of (0).

Conversely, assume that $\theta$ solves the equation (0), and thus all the equations $(i)$. The only-if direction of Lemma 7.5 implies that, for each automaton $\mathcal{A}_i$, the set $\{q_f\}\cdot\theta(X)$ is a finite union of run tree sets. Consequently, for each $i, 1 \leq i \leq n$, there exist run trees

$t_{i,1}, \ldots, t_{i,m_i}$ for $\mathcal{A}_i$ such that $\{q_f\}\cdot\theta(X) = S(t_{i,1}) \cup \ldots \cup S(t_{i,m_i})$. By Lemma 7.4 we know that $\{q_f\}\cdot\theta(X) \neq \emptyset$, and thus $m_i \geq 1$.

Now, consider $t_{1,1}$. We know that $S(t_{1,1}) \subseteq S(t_{i,1}) \cup \ldots \cup S(t_{i,m_i})$, and thus Lemma 7.2 implies that $t_{1,1}$ is a run tree for each of the automata $\mathcal{A}_i$. Since $S(t_{1,1}) \subseteq \{q_f\}\cdot\theta(X)$, the leaves of $t_{1,1}$ are all labeled with $q_f$, which implies that $t^\sharp_{1,1}$ is accepted by $\mathcal{A}_i$. Since this is true for all $i, 1 \leq i \leq n$, we know that the automata $\mathcal{A}_1, \ldots, \mathcal{A}_n$ accept a common tree. □

Together with the results proved in previous sections, this theorem yields

COROLLARY 7.7. *The following problems are Exptime-complete:*

- *solvability of linear equations in* $\mathcal{S}_{\mathrm{ACUIh}}$*;*
- *solvability of ACUIh-unification problems;*
- *unifiability of* $\mathcal{FL}_0$*-concept terms.*

## 8. ACUIh-matching is polynomial

Matching is the special case of unification where the term $t$ on the right-hand side of the equation $s =^? t$ does not contain variables (Bürckert 1989). As in the case of unification, we can restrict our attention to the case of a single such equation.

As an easy consequence of Theorem 4.3 we obtain that matching of $\mathcal{FL}_0$-concept terms (equivalently, ACUIh-matching) can be reduced to solving linear equations of the form

$$S_0 \cup S_1\cdot X_1 \cup \cdots \cup S_n\cdot X_n \quad = \quad T_0, \tag{8.1}$$

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 8.1. *Equation* (8.1) *has a solution iff the following is a solution of* (8.1)*:*

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

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 (8.1).

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 $S_i\cdot\tau(X_i) \subseteq T_0$, 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

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

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

(8.1) can be done in time polynomial in the size of (8.1). Thus, we have proved the following theorem:

THEOREM 8.2. *Solvability of linear equations of the form* (8.1) *in* $\mathcal{S}_{\text{ACUIh}}$ *can be decided in polynomial time.*

COROLLARY 8.3. *Matching of* $\mathcal{FL}_0$*-concept terms and ACUIh-matching is polynomial.*

THE CONNECTION TO THE WORK OF BORGIDA AND MCGUINNESS

Borgida and McGuinness (1996) 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 $\sigma$ such that $C \sqsubseteq \sigma(D)$. Moreover, they are interested in a substitution $\sigma$ 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 8.4. *The following problem, called matching modulo subsumption, 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* $\sigma$ *such that* $C \sqsubseteq \sigma(D)$*?*

As an easy consequence of the proof of Lemma 8.1, we can also compute a substitution $\sigma$ 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 $\theta$ of (8.1) constructed in the proof is larger (w.r.t. set inclusion) than all other solutions of (8.1). Since each word in a solution of (8.1) gives rise to an additional value restriction, it is clear that the largest solution of (8.1) gives rise to a solution $\sigma$ of the matching problem such that $\sigma(D)$ is as small as possible w.r.t. subsumption.

COROLLARY 8.5. *Let* $C, D$ *be* $\mathcal{FL}_0$*-concept terms (where* $C$ *does not contain variables) such that the matching problem modulo subsumption induced by* $C$ *and* $D$ *is solvable. Then, we can compute in polynomial time a substitution* $\theta$ *such that*

- $\theta$ *solves the matching problem modulo subsumption, i.e.,* $C \sqsubseteq \theta(D)$*;*
- $\theta(D)$ *is the least* $\mathcal{FL}_0$*-concept term with this property, i.e.,* $\theta(D) \sqsubseteq \sigma(D)$ *for all substitutions* $\sigma$ *satisfying* $C \sqsubseteq \sigma(D)$*.*

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 8.6. *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 (Garey and Johnson 1979)) to the matching problem for $\mathcal{FL}_0$-concept terms containing role variables. An instance of the monotone 1-in-3-SAT problem is given by a finite set of clauses, where each clause is a disjunction of three distinct propositional variables. A solution is an assignment of truth values to the propositional variables such that, in each clause of the problem, *exactly one* variable becomes true.

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

$$\forall R_p.\forall R_q.\forall R_r.A \sqcap X \equiv^? \forall R_0.\forall R_0.\forall R_1.A \sqcap \forall R_0.\forall R_1.\forall R_0.A \sqcap \forall R_1.\forall R_0.\forall R_0.A,$$

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. □

## 9. Future work

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 Subsection 4.2), our approach is also applicable to languages for which equivalence of concept terms is not axiomatizable by a monoidal equational theory. In (Baader *et al.* 1999) it is shown that the results for *matching* carry over to the language $\mathcal{FL}_\neg$, which extends $\mathcal{FL}_0$ by the bottom concept $\bot$ and atomic negation, and to $\mathcal{ALN}$, which extends $\mathcal{FL}_\neg$ by so-called number restrictions. The language $\mathcal{ALN}$ is expressive enough to be used in our chemical process engineering application. Unfortunately, *unification* of $\mathcal{FL}_\neg$- and $\mathcal{ALN}$-concept terms is more problematic: although it can also be reduced to a corresponding formal language problem, it appears that the tree automata approach employed in Section 6 cannot directly be used to solve this problem.

Another interesting problem is how to define an appropriate ordering on unifiers and matchers. For the instantiation preorder usually employed in unification theory, ACUIh is not well-behaved (Baader 1989): the theory ACUIh is of unification type zero, which implies that 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 (1996), another ordering, induced by the subsumption hierarchy, might be more appropriate.

**Acknowledgments.** The authors should like to thank the anonymous referees for their comments, which greatly helped to improve the presentation of this article. The first author was partially supported by the EC Working Group CCL II, and the second author was partially supported by the NSF grants CCR-9404930 and INT-9401087.

## References

Aiken, A., Kozen, D., Vardi, M., Wimmers, E. (1993). The complexity of set constraints. In Börger, E., Gurevich, Y., Meinke, K., editors, *Proceedings of Computer Science Logic, CSL'93*, volume 832 of *Lecture Notes in Computer Science*, pages 1–17. Springer–Verlag.

Baader, F. (1989). Unification in commutative theories. *J. Symbolic Computation*, 8:479–497.

Baader, F. (1990). 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).

Baader, F. (1991). Augmenting concept languages by transitive closure of roles: An alternative to terminological cycles. In Mylopoulos, J., Reiter, R., editors, *Proceedings of the 12th International Joint Conference on Artificial Intelligence*, pages 446–451, Sydney, Australia. Morgan Kaufmann, San Francisco.

Baader, F. (1993). Unification in commutative theories, Hilbert's basis theorem and Gröbner bases. *J. ACM*, 40(3):477–503.

Baader, F., Buchheit, M., Hollunder, B. (1996). Cardinality restrictions on concepts. *Artificial Intelligence*, 88(1–2):195–213.

Baader, F., Hanschke, P. (1991). A scheme for integrating concrete domains into concept languages. In Mylopoulos, J., Reiter, R., editors, *Proceedings of the 12th International Joint Conference on Artificial Intelligence*, pages 452–457, Sydney, Australia. Morgan Kaufmann, San Francisco.

Baader, F., Hollunder, B. (1991). A terminological knowledge representation system with complete inference algorithms. In Richter, M., Boley, H., editors, *Proceedings of the First International Workshop on Processing Declarative Knowledge (PDK-91)*, volume 567 of *Lecture Notes in Computer Science*, pages 67–85, Kaiserslautern (Germany). Springer–Verlag.

Baader, F., Küsters, R., Borgida, A., McGuinness, D. (1999). Matching in description logics. *Journal of Logic and Computation*, 9(3):411–447.

Baader, F., Narendran, P. (1998). Unification of concept terms in description logics. In Prade, H., editor, *Proceedings of the 13th European Conference on Artificial Intelligence (ECAI-98)*, pages 331–335. John Wiley & Sons Ltd.

Baader, F., Nutt, W. (1996). 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.

Baader, F., Sattler, U. (1996a). Description logics with symbolic number restrictions. In Wahlster, W., editor, *Proceedings of the Twelfth European Conference on Artificial Intelligence (ECAI-96)*, pages 283–287. John Wiley & Sons Ltd.

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

Baader, F., Sattler, U. (1996c). Number restrictions on complex roles in description logics. In Aiello, L. C., Doyle, J., Shapiro, S., editors, *Proceedings of the Fifth International Conference on the Principles of Knowledge Representation and Reasoning (KR-96)*, pages 328–339, Cambridge, Mass. Morgan Kaufmann, San Francisco.

Baader, F., Siekmann, J. (1994). Unification theory. In Gabbay, D., Hogger, C., Robinson, J., editors, *Handbook of Logic in Artificial Intelligence and Logic Programming*. Oxford University Press, Oxford, UK.

Bergamaschi, S., Beneventano, D. (1997). Incoherence and subsumption for recursive views and queries in object-oriented data models. *Data and Knowledge Engineering*, 21(3):217–252.

Bergamaschi, S., Sartori, C., Beneventano, D., Vincini, M. (1997). ODB-tools: a description logics based tool for schema validation and semantic query optimization in object oriented databases. In

Lenzerini, M., editor, *Proceedings of the 5th Congress of the Italian Association for Artificial Intelligence: Advances in Artificial Intelligence (AI*IA-97)*, volume 1321 of *Lecture Notes in Artificial Intelligence*, pages 435–438. Springer–Verlag.

Borgida, A., McGuinness, D. (1996). 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).

Borgida, A., Patel-Schneider, P. (1994). A semantics and complete algorithm for subsumption in the CLASSIC description logic. *Journal of Artificial Intelligence Research*, 1:277–308.

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

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

Buchheit, M., Donini, F. M., Schaerf, A. (1993). Decidable reasoning in terminological knowledge representation systems. *Journal of Artificial Intelligence Research*, 1:109–138.

Buchheit, M., Jeusfeld, M., Nutt, W., Staudt, M. (1994a). Subsumption of queries to object-oriented databases. *Information Systems*, 19(1):33–54.

Buchheit, M., Klein, R., Nutt, W. (1994b). Configuration as model construction: The constructive problem solving approach. In *Proceedings of the Third International Conference on Artificial Intelligence in Design, AID'94*, Lausanne, Switzerland.

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

Comon, H., Dauchet, M., Gilleron, R., Jacquemard, F., Lugiez, D., Tison, S., Tommasi, M. (1997). Tree automata techniques and applications. Available on: `http://www.grappa.univ-lille3.fr/tata`.

Devanbu, P., Brachman, R. J., Selfridge, P. G., Ballard, B. W. (1991). LaSSIE: A knowledge-based software information system. *Communications of the ACM*, 34(5):34–49.

Donini, F. M., Hollunder, B., Lenzerini, M., Spaccamela, A. M., Nardi, D., Nutt, W. (1992). The complexity of existential quantification in concept languages. *Journal of Artificial Intelligence*, 53:309–327.

Donini, F. M., Lenzerini, M., Nardi, D., Nutt, W. (1991a). The complexity of concept languages. In Allen, J., Fikes, R., Sandewall, E., editors, *Proceedings of the 2nd International Conference on Principles of Knowledge Representation and Reasoning (KR-91)*, pages 151–162, Cambridge, Mass. Morgan Kaufmann, San Francisco.

Donini, F. M., Lenzerini, M., Nardi, D., Nutt, W. (1991b). Tractable concept languages. In Mylopoulos, J., Reiter, R., editors, *Proceedings of the 12th International Joint Conference on Artificial Intelligence*, pages 458–463, Sydney, Australia. Morgan Kaufmann, San Francisco.

Donini, F. M., Lenzerini, M., Nardi, D., Schaerf, A. (1994). Deduction in concept languages: From subsumption to instance checking. *Journal of Logic and Computation*, 4(4):423–452.

Garey, M., Johnson, D. (1979). *Computers and Intractability: A Guide to the Theory of NP-Completeness.* W.H. Freeman and Company, New York.

Gécseg, F., Steinby, M. (1984). *Tree Automata.* Akadémiai Kiaddá, Budapest, Hungary.

Gilleron, R., Tison, S., Tommassi, M. (1994). Some new decidability results on positive and negative set constraints. In Jouannaid, J.-P., editor, *Proceedings of Constraints in Computational Logics, CCL'94*, volume 845 of *Lecture Notes in Computer Science*, pages 336–351. Springer–Verlag.

Hollunder, B., Baader, F. (1991). Qualifying number restrictions in concept languages. In Allen, J., Fikes, R., Sandewall, E., editors, *Proceedings of the 2nd International Conference on Principles of Knowledge Representation and Reasoning*, pages 335–346, Cambridge, Mass. Morgan Kaufmann, San Francisco.

Hollunder, B., Nutt, W., Schmidt-Schauß, M. (1990). Subsumption algorithms for concept description languages. In Aiello, L. C., editor, *Proceedings of the 9th European Conference on Artificial Intelligence (ECAI-90)*, pages 348–353, Stockholm, Sweden. Pitman Publishing, London.

Koehler, J. (1994). An application of terminological logics to case-based reasoning. In Doyle, J., Sandewall, E., Torasso, P., editors, *Proceedings of the Fourth International Conference on Principles of Knowledge Representation and Reasoning (KR-94)*, pages 351–362, Bonn, Germany. Morgan Kaufmann, San Francisco.

Levesque, H. J., Brachman, R. J. (1987). Expressiveness and tractability in knowledge representation and reasoning. *Computational Intelligence*, 3:78–93.

McGuinness, D. L., Resnick, L. A., Isbell, C. (1995). Description Logic in practice: A CLASSIC application. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence, IJCAI'95*, pages 2045–2046, Montréal, Canada. Morgan Kaufmann, San Francisco. Video Presentation.

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

Nebel, B. (1988). Computational complexity of terminological reasoning in BACK. *Journal of Artificial Intelligence*, 34(3):371–383.

Nebel, B. (1990a). *Reasoning and Revision in Hybrid Representation Systems*, volume 422 of *Lecture Notes in Computer Science*. Springer–Verlag.

Nebel, B. (1990b). Terminological reasoning is inherently intractable. *Journal of Artificial Intelligence*, 43(2):235–249.

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

Quantz, J., Schmitz, B. (1994). Knowledge-based disambiguation for machine translation. *Minds and Machines*, 4:39–57.

Rychtyckyj, N. (1996). DLMS: An evaluation of KL-ONE in the automobile industry. In Aiello, L. C., Doyle, J., Shapiro, S., editors, *Proceedings of the Fifth International Conference on Principles of Knowledge Representation and Reasoning (KR-96)*, pages 588–596, Cambridge, Mass. Morgan Kaufmann, San Francisco.

Schaerf, A. (1993). On the complexity of the instance checking problem in concept languages with existential quantification. *Journal of Intelligent Information Systems*, 2:265–278.

Schmidt-Schauß, M., Smolka, G. (1991). Attributive concept descriptions with complements. *Journal of Artificial Intelligence*, 47:1–26.

Seidl, H. (1994). Haskell overloading is DEXPTIME-complete. *Information Processing Letters*, 52(2):57–60.

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

Woods, W. A., Schmolze, J. G. (1992). The KL-ONE family. *Computers and Mathematics with Applications*, 23(2–5):133–177.

Wright, J. R., Weixelbaum, E. S., Brown, K., Vesonder, G. T., Palmer, S. R., Berman, J. I., Moore, H. H. (1993). A knowledge-based configurator that supports sales, engineering, and manufacturing at AT&T network systems. *AI Magazine*, 14(3):69–80.