Unification in a Description Logic with Inconsistency and Transitive Closure of Roles

Franz Baader
LuFG Theoretische Informatik
RWTH Aachen, Germany
`baader@informatik.rwth-aachen.de`

Ralf Küsters
Institut für Informatik und Praktische Mathematik
CAU Kiel, Germany
`kuesters@ti.informatik.uni-kiel.de`

Abstract

Unification considers concept patterns, i.e., concept descriptions with variables, and tries to make these descriptions equivalent by replacing the variables by appropriate concept descriptions. Baader and Küsters have shown that unification in $\mathcal{FL}_{reg}$, a description logic that allows for the concept constructors top concept, concept conjunction, and value restrictions as well as the role constructors union, composition, and transitive closure, is an ExpTime-complete problem and that solvable $\mathcal{FL}_{reg}$-unification problems always have least unifiers. In the present paper, we generalize these results to a DL which extends $\mathcal{FL}_{reg}$ by the bottom concept. The proof strongly depends on the existence of least unifiers in $\mathcal{FL}_{reg}$.

1 Introduction

Unification of concept descriptions was introduced by Baader and Narendran [5] as a new inference service for detecting and avoiding redundancies in DL knowledge bases. Unification considers concept patterns, i.e., concept descriptions with variables, and tries to make these descriptions equivalent by replacing the variables by appropriate concept descriptions. The technical results in [5] were concerned with unification in the small DL $\mathcal{FL}_0$, which allows for conjunction of concepts ($C \sqcap D$), value restriction ($\forall R.C$), and the top concept ($\top$): unification of $\mathcal{FL}_0$-concept patterns is an ExpTime-complete problem.

Since then, extending this result to more expressive DLs has turned out to be quite hard. Even if one just adds the bottom concept $\bot$ to $\mathcal{FL}_0$, the methods employed for unification in $\mathcal{FL}_0$ no longer apply. Consequently, it is also not clear how to handle unification in descriptions logics that can express inconsistency (like DLs with (atomic) negation, number restrictions, etc.). Instead of adding concept constructors to $\mathcal{FL}_0$, [4] considers unification in $\mathcal{FL}_{reg}$, which extends $\mathcal{FL}_0$ by the role constructors union, composition, and transitive closure. For this DL, unification is still an ExpTime-complete problem. In contrast to the case of $\mathcal{FL}_0$, solvable $\mathcal{FL}_{reg}$-unification problems always have a least solution (w.r.t. subsumption), which can be computed in exponential time.

In the present paper we extend the results for $\mathcal{FL}_{reg}$ to $\mathcal{FL}\bot_{reg}$, the extension of $\mathcal{FL}_{reg}$ by the bottom concept. The proof strongly depends on the existence of least unifiers in $\mathcal{FL}_{reg}$.

The results for unification in $\mathcal{FL}_{reg}$ have been obtained by reduction from/to the problem of solving (systems of) linear equations over regular languages, and solvability of such equations has been shown to be ExpTime-complete, by reductions from/to decision problems for tree-automata on infinite trees. Closely related to the problem of solving linear language equations is the problem of solving set constraints [1], i.e., relations between sets of terms. Set constraints are usually more general than linear language equations, which most closely correspond to positive set constraints [1] for terms over unary and nullary function symbols where only union of sets is allowed. In fact, it is easy to see that linear language equations can be expressed using positive set constraints. However, whereas we are interested in greatest solutions (corresponding to least solutions on the unification side), for set constraints one usually considers least solutions.

To solve unification in $\mathcal{FL}\bot_{reg}$, we have extended linear language equations to so-called linear Σ^*-equations: Linear language equations only allow for left concatenation of regular languages and variables, i.e., concatenation of the form $L \cdot X$, where L is a regular language and X is a variable. In linear Σ^*-equations one can in addition concatenate Σ^* from right, where Σ^* is the set of all words over the alphabet Σ, yielding terms of the form $L \cdot X \cdot \Sigma^*$, which cannot be expressed by set constraints.

2 Preliminaries

First, we introduce the DLs $\mathcal{FL}_0$-, $\mathcal{FL}\bot$-, $\mathcal{FL}_{reg}$-, and $\mathcal{FL}\bot_{reg}$. Starting from the finite and disjoint sets N_C of concept names and N_R of role names, $\mathcal{FL}_0$-concept descriptions are built using the concept constructors conjunction ($C \sqcap D$), value restriction ($\forall r.C$), and the top concept ($\top$). $\mathcal{FL}\bot$ additionally allows for the bottom concept ($\bot$). $\mathcal{FL}_{reg}$ extends $\mathcal{FL}_0$ by the role constructors identity role (ε), empty role ($\emptyset$), union ($R \cup S$), composition ($R \circ S$),

Syntax	Semantics	$\mathcal{FL}_0$	$\mathcal{FL}\bot$	$\mathcal{FL}_{reg}$	$\mathcal{FL}\bot_{reg}$
$\top$	Δ^I	x	x	x	x
$\bot$	$\emptyset$		x		x
$C \sqcap D$	$C^I \cap D^I$	x	x	x	x
$\forall R.C$	$\{x \in \Delta^I \mid \forall y : (x,y) \in R^I \to y \in C^I\}$	x	x	x	x
ε	$\{(x,x) \mid x \in \Delta^I\}$			x	x
$\emptyset$	$\emptyset$			x	x
$R \circ S$	$\{(x,z) \mid \exists y : (x,y) \in R^I \wedge (y,z) \in S^I\}$			x	x
R^*	$\bigcup_{n\geq 0}(R^I)^n$			x	x

Table 1: Syntax and semantics of concept descriptions.

and reflexive-transitive closure (R^*). Finally, $\mathcal{FL}\bot_{reg}$ adds the bottom concept ($\bot$) to $\mathcal{FL}_{reg}$. As an example, consider the $\mathcal{FL}\bot_{reg}$-concept description Woman $\sqcap$ $\forall$child*.Woman $\sqcap$ $\forall$pet.$\bot$, which represents the set of all women with only female offspring and no pets.

Role names will be denoted by lower case letters ($r, s, \ldots \in N_R$), and complex roles by upper case letters ($R, S, T \ldots$). Note that a complex role can be viewed as a regular expression over N_R where ε is taken as the empty word, role names as elements of the alphabet, the empty role as the empty language, union as union of languages, composition as concatenation, and reflexive-transitive closure as Kleene star. Therefore, we sometimes view a complex role R as a regular expression. In the following, we will abuse notation by identifying regular expressions with the languages they describe. In particular, if R and R' are regular expressions, then $R = R'$ will mean that the corresponding languages are equal.

The semantics of concept and role descriptions is defined as usual in terms of an *interpretation* $\mathcal{I} = (\Delta^I, \cdot^I)$. The domain Δ^I of $\mathcal{I}$ is a non-empty set and the interpretation function $\cdot^I$ maps each concept name $A \in N_C$ to a set $A^I \subseteq \Delta^I$ and each role name $r \in N_R$ to a binary relation $r^I \subseteq \Delta^I \times \Delta^I$. The extension of $\cdot^I$ to arbitrary concept and role descriptions is defined inductively, as shown in the second column of Table 1. The interested reader may note that $\mathcal{FL}\bot_{reg}$-concept descriptions can also be viewed as concepts defined by cyclic $\mathcal{FL}\bot$-TBoxes interpreted with the greatest fixed-point semantics [2].

The concept description D *subsumes* the description C ($C \sqsubseteq D$) iff $C^I \subseteq D^I$ for all interpretations $\mathcal{I}$. Two concept descriptions C, D are *equivalent* ($C \equiv D$) iff they subsume each other.

In order to define unification of concept descriptions, we first have to introduce the notions concept pattern and substitution operating on concept patterns. To this purpose, we consider a set of *concept variables* N_X (disjoint from $N_C \cup N_R$). Now, *$\mathcal{FL}\bot_{reg}$-concept patterns* are $\mathcal{FL}\bot_{reg}$-concept descriptions defined over the set $N_C \cup N_X$ of concept names and the set N_R of role names. For example, given $A \in N_C$, $X \in N_X$, and $r \in N_R$, $\forall r.A \sqcap \forall r^*.X$ is an $\mathcal{FL}\bot_{reg}$-concept pattern.

A *substitution* σ is a mapping from N_X into the set of all $\mathcal{FL}\bot_{reg}$-concept descriptions. This mapping is extended from variables to concept patterns in the obvious way, i.e.,

- $\sigma(\top) := \top$, $\sigma(\bot) := \bot$, and $\sigma(A) := A$ for all $A \in N_C$,
- $\sigma(C \sqcap D) := \sigma(C) \sqcap \sigma(D)$ and $\sigma(\forall R.C) := \forall R.\sigma(C)$.

Definition 1 *An $\mathcal{FL}\bot_{reg}$-*unification problem *is of the form $C \equiv^? D$, where C, D are $\mathcal{FL}\bot_{reg}$-concept patterns. The substitution σ is a* unifier *of this problem iff $\sigma(C) \equiv \sigma(D)$. In this case, the unification problem is* solvable*, and C and D are called* unifiable*.*

For sublanguages of $\mathcal{FL}\bot_{reg}$, concept patterns, substitutions, and unification problems are defined analogously.

For example, the substitution $\sigma = \{X \mapsto \forall r \circ r^*.A,\ Y \mapsto \forall r.\bot\}$ is a unifier of the unification problem

$$\forall s.\forall r.\bot \sqcap \forall r.A \sqcap \forall r.X \quad \equiv^? \quad \forall s.\forall r.\forall s.A \sqcap X \sqcap \forall s.Y.$$

Note that this problem can also be viewed as an $\mathcal{FL}\bot$-unification problem. However, in this case it does not have a solution since there are no $\mathcal{FL}\bot$-concept descriptions that, when substituted for X and Y, make the two concept patterns equivalent.

In case a unification problem is solvable, one is usually interested in obtaining an actual solution. Since a given unification problem may have infinitely many unifiers, one must decide which ones to prefer. As mentioned in the introduction, in many applications least unifiers are of interest. To define least unifiers, we extend the subsumption quasi-ordering from concept descriptions to substitutions: if σ and σ' are substitutions, then $\sigma \sqsubseteq \sigma'$ iff $\sigma(X) \sqsubseteq \sigma'(X)$ for all variables X. The unifier σ is a *least unifier* of an unification problem if $\sigma \sqsubseteq \sigma'$ for all unifiers σ'. Note that least unifiers (if they exist) are uniquely determined up to equivalence. By abuse of language, we therefore refer to *the* least unifier.

For $\mathcal{FL}_0$ and $\mathcal{FL}\bot$, least unifiers need not exist. For example, assume that $N_C = \{A\}$, $N_R = \{r\}$, and $N_X = \{X\}$, and consider the unification problem $\forall r.A \sqcap \forall r.X \equiv^? X \sqcap \forall r.A$. Substituting $\top$ for X solves the problem in $\mathcal{FL}_0$ and $\mathcal{FL}\bot$. However, it is not hard to show that there does not exist a least unifier in $\mathcal{FL}\bot$ or $\mathcal{FL}_0$. On the other hand, σ with $\sigma(X) = \forall r^*.A$ is the least unifier of this problem in $\mathcal{FL}\bot_{reg}$. More generally, we will show that every solvable $\mathcal{FL}\bot_{reg}$-unification problem has a least unifier.

The example $X \equiv^? X$ shows that there are problems that have least unifiers in $\mathcal{FL}\bot$ but not in $\mathcal{FL}_0$. In $\mathcal{FL}\bot$, mapping X to $\bot$ yields the least unifier of the problem, but there is no least unifier of this problem in $\mathcal{FL}_0$.

3 Unification in $\mathcal{FL}_{reg}$

In this section, we briefly recall the results from [4] for unification in $\mathcal{FL}_{reg}$, since our results for $\mathcal{FL}\bot_{reg}$ depend on them.

The following is shown in [4] for unification in $\mathcal{FL}_{reg}$:

1. Deciding the solvability of $\mathcal{FL}_{reg}$-unification problems is an ExpTime-complete problem.
2. Every solvable $\mathcal{FL}_{reg}$-unification problem has a least unifier. The size of this unifier may grow exponentially in the size of the problem, and it can be computed in exponential time.

These results have been obtained by reduction to/from the problem of solving (systems of) linear equations over regular languages built using the alphabet N_R of role names. Any $\mathcal{FL}_{reg}$-unification problem can be transformed into a system of linear equations over regular languages such that the unifiers of the problem correspond to the solutions of the system (see the next section for the corresponding reduction for $\mathcal{FL}\bot_{reg}$).

This type of equations is defined as follows. For languages $L, M \subseteq N_R^*$, their concatenation is defined as $LM := \{vw \mid v \in L, w \in M\}$. Let $X_1, \dots, X_n$ be variables. Given regular languages $S_0, S_1, \dots, S_n, T_0, T_1, \dots, T_n$[1] over N_R, a *linear equation over regular languages* is of the form

$$S_0 \cup S_1X_1 \cup \cdots \cup S_nX_n = T_0 \cup T_1X_1 \cup \cdots \cup T_nX_n. \tag{1}$$

A *(regular) solution* θ of this equation is a substitution assigning to each variable a (regular) language over N_R such that the equation holds. A *system of regular language equations* is a finite collection of linear equations over regular languages. A substitution θ solves such a system if it solves every equation in it simultaneously.

We are particularly interested in *regular* solutions since they correspond to unifiers in $\mathcal{FL}_{reg}$. We are also interested in greatest solutions since they correspond to least unifiers. To define greatest solutions we extend the inclusion relation on languages to solutions of linear equations: if θ and θ' are solutions of a given linear equation, then $\theta \subseteq \theta'$ iff $\theta(X) \subseteq \theta(X')$ for every variable X occurring in the equation. A solution θ is called *greatest solution* iff $\theta' \subseteq \theta$ for every solution θ'.

For systems of regular language equations the following results have been shown in [4]:

1. Deciding the solvability of systems of regular language equations is an ExpTime-complete problem.

[1]We assume that these languages are given by regular expressions or nondeterministic finite automata.

2. Every solvable system of regular language equations has a greatest solution θ, and this solution is regular, i.e., the languages $\theta(X)$ for all variables X occurring in the system are regular. There exists an exponential time algorithm that computes deterministic finite automata accepting the regular languages $\theta(X)$, and the size of these automata may grow exponentially in the size of the system of equations.

The complexity results for the decision problem are obtained by reductions to/from decision problems for a certain class of Büchi tree-automata. Deterministic finite automata accepting the greatest solution of a (solvable) system of regular language equations can be constructed from a Büchi tree-automaton that "accepts" all the solutions of the system (see [4] for details).

4 Unification in $\mathcal{FL}\bot_{reg}$

In this section, we extend the results above from $\mathcal{FL}_{reg}$ to $\mathcal{FL}\bot_{reg}$. To be more precise, we show the following theorem.

Theorem 2 *Deciding the solvability of an $\mathcal{FL}\bot_{reg}$-unification problem is an ExpTime-complete problem. A solvable $\mathcal{FL}\bot_{reg}$-unification problem always has a least solution, which can be computed in exponential time. The size of such a solution may grow exponentially in the size of the unification problem.*

While for $\mathcal{FL}_{reg}$ we have proved these results by reduction to the solvability of systems of linear equations over regular languages of the form (1), we now need *linear Σ^*-equations over regular languages*, which have the form

$$S_0 \cup \bigcup_{i=1}^{n} S_i X_i \cup \bigcup_{i=1}^{n'} S'_i X'_i \Sigma^* = T_0 \cup \bigcup_{i=1}^{n} T_i X_i \cup \bigcup_{i=1}^{n'} T'_i X'_i \Sigma^*, \tag{2}$$

where the sets $\{X_i\}$ and $\{X'_i\}$ of variables are disjoint. In the following, we will first show how unification in $\mathcal{FL}\bot_{reg}$ can be reduced to deciding the solvability of certain systems of such linear Σ^*-equations. We then show how to solve such systems.

The Reduction

It is easy to see that $\mathcal{FL}\bot_{reg}$-concept patterns can be written in the following normal form:

$$\forall R_\bot.\bot \sqcap \bigsqcap_{A \in N_C} \forall R_A.A \sqcap \bigsqcap_{X \in N_X} \forall R_X.X,$$

where $R_\bot$, R_A, and R_X are regular expressions over N_R. These normal forms are obtained by exhaustively applying the equivalence preserving normalization rule $\forall R.C \sqcap \forall R'.C \longrightarrow \forall(R \cup R').C$, where R, R' are regular languages over N_R and C is some $\mathcal{FL}\bot_{reg}$-concept pattern.

To establish the correctness of our reduction, we need the following characterization of equivalence between $\mathcal{FL}\bot_{reg}$-concept descriptions.

Lemma 3 *Let C, D be $\mathcal{FL}\bot_{reg}$-concept descriptions such that*

$$C \equiv \forall S_\bot.\bot \sqcap \bigsqcap_{A \in N_C} \forall S_A.A \quad \text{and} \quad D \equiv \forall T_\bot.\bot \sqcap \bigsqcap_{A \in N_C} \forall T_A.A.$$

Then $C \equiv D$ iff (i) *$S_\bot\Sigma^* = T_\bot\Sigma^*$, and* (ii) *$S_A \cup S_\bot\Sigma^* = T_A \cup T_\bot\Sigma^*$ for all $A \in N_C$.*

As an easy consequence of this lemma, unification in $\mathcal{FL}\bot_{reg}$ can be reduced to solving the Σ^*-equations $E(\bot)$ and $E(A)$ introduced below. In these equations, the variables $X_\bot$ and X_A are new copies of $X \in N_X$. If E denotes an equation, then E^l shall denote its left hand-side and E^r its right hand-side.

Theorem 4 *Let C, D be $\mathcal{FL}\bot_{reg}$-concept patterns such that*

$$\begin{array}{rcl} C & \equiv & \forall S_\bot.\bot \sqcap \bigsqcap_{A \in N_C} \forall S_A.A \sqcap \bigsqcap_{X \in N_X} \forall S_X.X, \text{ and} \\ D & \equiv & \forall T_\bot.\bot \sqcap \bigsqcap_{A \in N_C} \forall T_A.A \sqcap \bigsqcap_{X \in N_X} \forall T_X.X. \end{array}$$

Then C, D are unifiable iff the system $\{E(\bot)\} \cup \{E(A) \mid A \in N_C\}$ of linear Σ^*-equations has a solution, where

$$\begin{array}{rcl} E(\bot) & := & S_\bot\Sigma^* \cup \bigcup_{X \in N_X} S_X X_\bot \Sigma^* = T_\bot\Sigma^* \cup \bigcup_{X \in N_X} T_X X_\bot \Sigma^*, \\ E(A) & := & E(\bot)^l \cup S_A \cup \bigcup_{X \in N_X} S_X X_A = E(\bot)^r \cup T_A \cup \bigcup_{X \in N_X} T_X X_A. \end{array}$$

Solutions of the system $\{E(\bot)\} \cup \{E(A) \mid A \in N_C\}$ can easily be translated into unifiers of C, D and vice versa. In particular, it is easy to show that least unifiers of C, D correspond to greatest solutions of $\{E(\bot)\} \cup \{E(A) \mid A \in N_C\}$.

Unlike the system considered for $\mathcal{FL}_{reg}$, the equations in the system here cannot be solved separately since the variables in $E(\bot)$ occur also in $E(A)$.

Note that $E(\bot)$ is a special kind of a linear Σ^*-equation. An equation of this form will be called *restricted linear Σ^*-equation* in the sequel.

Solving Restricted Linear Σ^*-Equations

The existence of greatest solutions of solvable restricted linear Σ^*-equation follows from the observation that the set of solutions of systems of (general) linear Σ^*-equations is closed under (arbitrary) union, i.e., if θ_i, $i \in I$, for some index set I, are solutions, then so is θ with $\theta(X) := \bigcup_{i \in I} \theta_i(X)$ for every variable X

occurring in the system. Note, however, that this does not imply that greatest solutions of such systems are regular.

As for the solvability of restricted linear Σ^*-equations, let $E'(\bot)$ denote the equation obtained from $E(\bot)$ by omitting the right-concatenation of the variables $X_\bot$ with Σ^*. We can prove the following lemma (see Appendix A).

Lemma 5 $E'(\bot)$ *is solvable iff* $E(\bot)$ *is solvable. If the equations are solvable, then they have greatest solutions, and their greatest solutions coincide.*

Since $E'(\bot)$ is a linear equation over regular languages, this lemma, together with the results from [4] cited above, yields the following proposition.

Proposition 6 *The solvability of restricted linear* Σ^**-equations can be decided in exponential time. Any solvable restricted linear* Σ^**-equation has a greatest solution, which is regular and can be computed in exponential time.*

Proof of Theorem 2

Because of Theorem 4, it is enough to consider the solvability and the existence of greatest solutions of the system $\{E(\bot)\} \cup \{E(A) \mid A \in N_C\}$. We already know that if such a system is solvable, it has a greatest solution.

As for the solvability, let $E'(A)$ be the equation obtained from $E(A)$ by replacing $E^l(\bot)$ by $S_\bot\Sigma^*$ and $E^r(\bot)$ by $T_\bot\Sigma^*$. See Appendix B for the proof of the following lemma.

Lemma 7 *The system* $\{E(\bot)\} \cup \{E'(A) \mid A \in N_C\}$ *is solvable iff* $\{E(\bot)\} \cup \{E(A) \mid A \in N_C\}$ *is solvable. If these systems are solvable, then they have greatest solutions, and their greatest solutions coincide.*

By Proposition 6, the solvability of $E(\bot)$ can be decided in exponential time. Since each $E'(A)$ is a linear equation over regular languages, its solvability can also be decided in exponential time. Because $E(\bot)$ and the equations $E'(A)$ do not share variables, solvability of $\{E(\bot)\} \cup \{E'(A) \mid A \in N_C\}$ can be decided by testing these equations separately for solvability. We know that each of these equations (if solvable) has a greatest solution, which is regular and can be computed in exponential time. The greatest solution of $\{E(\bot)\} \cup \{E'(A) \mid A \in N_C\}$ (which is also the greatest solution of $\{E(\bot)\} \cup \{E(A) \mid A \in N_C\}$ by Lemma 7) can simply be obtained by combining the greatest solutions of the single equations. This proves the following theorem:

Theorem 8 *The solvability of systems of equations of the form*

$$\{E(\bot)\} \cup \{E(A) \mid A \in N_C\}$$

can be decided in exponential time. Solvable systems always have a greatest solution, which is regular and can be computed in exponential time.

From this theorem it immediately follows that deciding the solvability of $\mathcal{FL}\bot_{reg}$-unification problems is in ExpTime, that least solutions always exist, and that they can be computed in exponential time. Given this, to complete the proof of Theorem 2 it remains to show ExpTime-hardness and the fact that the size of greatest solutions may grow exponentially.

ExpTime-hardness follows from the following lemma (see Appendix C for the proof) and the fact that unification in $\mathcal{FL}_{reg}$ is an ExpTime-hard problem (see Section 3).

Lemma 9 *Given an $\mathcal{FL}_{reg}$-unification problem, this problem is solvable in $\mathcal{FL}_{reg}$ iff it is solvable when considered as $\mathcal{FL}\bot_{reg}$-unification problem.*

The lower bound on the size of a least solution claimed in Theorem 2 can be proved as follows. Let $L_1, \dots, L_k$ be regular languages and $r_1, \dots, r_k$ be pairwise distinct role names which do not occur in the languages L_i. Then the least unifier of the $\mathcal{FL}\bot_{reg}$-unification problem

$$\forall(\{r_1\}L_1 \cup \ldots \cup \{r_k\}L_k).A \equiv \forall(\{r_1\}L_1 \cup \ldots \{r_k\}L_k).A \sqcap \forall\{r_1, \dots, r_k\}.X$$

is $\sigma(X) \equiv \forall(L_1 \cap \cdots \cap L_k).A$. From results shown in [7] it follows that the size of automata accepting the intersection of the L_is may grow exponentially in the size of automata accepting $L_1, \dots, L_k$.

5 Conclusion

Unification in DLs that can express inconsistency has turned out to be a very hard problem. In the present paper, we have shown that unification in the extension of $\mathcal{FL}_{reg}$ by the bottom concept is not harder than unification in $\mathcal{FL}_{reg}$ itself. However, this result strongly depends on the fact that solvable $\mathcal{FL}_{reg}$-unification problems always have a least unifier in $\mathcal{FL}_{reg}$. Consequently, and unfortunately, the method employed in this paper cannot be adapted to the extension of $\mathcal{FL}_0$ by the bottom concept since in $\mathcal{FL}_0$ least unifiers need not exist. Thus, the unification in this extension still remains a wide open problem. More accessible appears to be unification in extensions of $\mathcal{FL}_{reg}$ by constructors that can express inconsistency (like atomic negation and number restrictions).

References

[1] A. Aiken, D. Kozen, M. Vardi, and E. Wimmers. The Complexity of Set Constraints. In E. Börger, Y. Gurevich, and K. Meinke, editors, *Proceedings 1993 Conf. Computer Science Logic (CSL'93)*, volume 832 of *Lecture Notes in Computer Science*, pages 1–17. European Association Computer Science Logic, Springer, September 1993.

[2] F. Baader. Augmenting Concept Languages by Transitive Closure of Rules: An Alternativ to Terminological Cycles. In J. Mylopoulos and R. Reiter, editors, *Proceedings of the 12th International Joint Conference on Artificial Intelligence (IJCAI'91)*, pages 446–451, Sydney, 1991. Morgan Kaufmann Publishers.

[3] F. Baader and B. Hollunder. A Terminological Knowledge Representation System with Complete Inference Algorithms. In *Proceedings of the First International Workshop on Processing Declarative Knowledge*, volume 572 of *Lecture Notes in Computer Science*, pages 67–85, Kaiserslautern (Germany), 1991. Springer–Verlag.

[4] F. Baader and R. Küsters. Unification in a Description Logic with Transitive Closure of Roles. In R. Nieuwenhuis and A. Voronkov, editors, *Proceedings of the 8th International Conference on Logic for Programming, Artificial Intelligence and Reasoning (LPAR 2001)*, volume 2250 of *Lecture Notes in Artificial Intelligence*, Vienna, Austria, 2001. Springer–Verlag.

[5] F. Baader and P. Narendran. Unification of Concept Terms in Description Logics. In H. Prade, editor, *Proceedings of the 13th European Conference on Artificial Intelligence (ECAI-98)*, pages 331–335, Brighton, UK, 1998. John Wiley & Sons Ltd. An extended version has appeared in J. Symbolic Computation 31:277–305, 2001.

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

[7] S. Yu and Q. Zhuang. On the State Complexity of Intersection of Regular Languages. *ACM SIGACT News*, 22(3):52–54, 1991.

A Proof of Lemma 5

Assume that θ is a solution of $E'(\bot)$, i.e.,

$$S_\bot\Sigma^* \cup \bigcup_{X\in N_X} S_X\theta(X_\bot) = T_\bot\Sigma^* \cup \bigcup_{X\in N_X} T_X\theta(X_\bot).$$

Now, consider the identity obtained by concatenating both sides from the right with Σ^*. Using the fact that $\Sigma^*\Sigma^* = \Sigma^*$, it is easy to see that the identity obtained this way is just $E(\bot)$, where the variables $X_\bot$ are substituted by $\theta(X_\bot)$. Thus, θ solves $E(\bot)$. Conversely, let θ be a solution of $E(\bot)$. Then clearly θ' with $\theta'(X_\bot) := \theta(X_\bot)\Sigma^*$ solves $E'(\bot)$.

Assume that the equations are solvable. We know that $E'(\bot)$ has a greatest solution, say θ. As argued above, every solution of $E'(\bot)$ is also a solution of

$E(\bot)$, and thus θ is a solution of $E(\bot)$. It remains to show that θ is the greatest solution of $E(\bot)$. Assume that τ is a solution of $E(\bot)$. As shown above, τ' with $\tau'(X_\bot) := \tau(X_\bot)\Sigma^*$ is a solution of $E'(\bot)$, and thus we have for all variables $X_\bot$ occurring in $E(\bot)$ and $E'(\bot)$:

$$\tau(X_\bot) \subseteq \tau(X_\bot)\Sigma^* = \tau'(X_\bot) \subseteq \theta(X_\bot).$$

The first inclusion holds since the empty word belongs to Σ^* and the second inclusion holds since θ is the greatest solution of $E'(\bot)$. Summing up, we have shown that the (arbitrarily chosen) solution τ of $E(\bot)$ is contained in θ, which proves that θ is the greatest solution of $E(\bot)$.

B Proof of Lemma 7

Let $\mathcal{S}$ denote the system $\{E(\bot)\}\cup\{E(A) \mid A \in N_C\}$ and $\mathcal{S}'$ the system $\{E(\bot)\}\cup \{E'(A) \mid A \in N_C\}$.

Let θ be a solution of $\mathcal{S}$. We show that θ' with $\theta'(X_\bot) := \theta(X_\bot)$ and $\theta'(X_A) := \theta(X_A)\cup\theta(X_\bot)\Sigma^*$ solves $\mathcal{S}'$. Obviously θ' solves $E(\bot)$. When replacing the variables in $E'(A)$ by their θ'-images, we obtain on the left-hand side

$$S_\bot\Sigma^* \cup S_A \cup \bigcup_{X\in N_X} S_X(\theta(X_A) \cup (\theta(X_\bot)\Sigma^*))$$

and on the right-hand side

$$T_\bot\Sigma^* \cup T_A \cup \bigcup_{X\in N_X} T_X(\theta(X_A) \cup (\theta(X_\bot)\Sigma^*)).$$

By applying known rules for concatenation and union of regular languages, the left-hand side can be transformed into

$$S_\bot\Sigma^* \cup \bigcup_{X\in N_X} S_X\theta(X_\bot)\Sigma^* \cup S_A \cup \bigcup_{X\in N_X} S_X\theta(X_A)$$

and the right-hand side into

$$T_\bot\Sigma^* \cup \bigcup_{X\in N_X} T_X\theta(X_\bot)\Sigma^* \cup T_A \cup \bigcup_{X\in N_X} T_X\theta(X_A).$$

By our assumption, θ solves both $E(\bot)$ and $E(A)$, and thus both sides are equal. Thus, we have shown that if $\mathcal{S}$ is solvable, then so is $\mathcal{S}'$.

Conversely, if θ solves $\mathcal{S}'$, then also $\mathcal{S}$: Obviously, θ solves $E(\bot)$. Now, consider $E'(A)$ where the variables are instantiated by θ. Since θ solves $E(\bot)$, we can add $E^l(\bot)$ and $E^r(\bot)$ (again with the variables instantiated by θ) to the left and the right hand-side of $E'(A)$, respectively, without destroying the fact that both sides are equal. The identity obtained this way is just $E(A)$, with the variables instantiated by θ. Hence, θ also solves $E(A)$.

Assume that the systems are solvable. We know that $\mathcal{S}'$ has a greatest solution, say θ. As shown above, each solution of $\mathcal{S}'$ is also a solution of $\mathcal{S}$. Thus, θ is a solution of $\mathcal{S}$. It remains to show that it is the greatest solution of $\mathcal{S}$. Assume that τ is an arbitrary solution of $\mathcal{S}$. We must show that $\tau \subseteq \theta$. Let τ' be constructed from τ as at the beginning of this proof, i.e., $\tau'(X_\bot) := \tau(X_\bot)$ and $\tau'(X_A) := \tau(X_A) \cup \tau(X_\bot)\Sigma^*$. Then, $\tau \subseteq \tau'$ and τ' is a solution of $\mathcal{S}'$. Since θ is the greatest solution of $\mathcal{S}'$, this implies $\tau \subseteq \tau' \subseteq \theta$, which shows the required inclusion $\tau \subseteq \theta$.

C Proof of Lemma 9

The lemma is an easy consequence of Lemma 7 and the reduction of unification in $\mathcal{FL}_{reg}$ to solving systems of regular language equation: Analogously to Theorem 4 and as stated in [4], the solvability of an $\mathcal{FL}_{reg}$-unification problem can be reduced to the solvability of a system of regular language equations, which coincides with the one in Theorem 4, except that the equation $E(\bot)$ is omitted and $E(A)$ does not contain $E(\bot)^l$ and $E(\bot)^r$. Let us call this version of $E(A)$, $E_{\mathcal{FL}_{reg}}(A)$. Given an $\mathcal{FL}_{reg}$-unification problem, the sets $S_\bot$ and $T_\bot$ are empty, thus $E'(A)$ (cf. Lemma 7) coincides with $E_{\mathcal{FL}_{reg}}(A)$. As a consequence, if $\{E(\bot)\} \cup \{E'(A) \mid A \in N_C\}$ has a solution, then also $\{E_{\mathcal{FL}_{reg}}(A) \mid A \in N_C\}$. Conversely, if $\{E_{\mathcal{FL}_{reg}}(A) \mid A \in N_C\}$ is solvable, then also $\{E(\bot)\} \cup \{E'(A) \mid A \in N_C\}$ (the variables $X_\bot$ can simply be substituted by the empty set). By Lemma 7, the system $\{E(\bot)\} \cup \{E(A) \mid A \in N_C\}$ has a solution iff $\{E(\bot)\} \cup \{E'(A) \mid A \in N_C\}$ has a solution, and thus, $\{E(\bot)\} \cup \{E(A) \mid A \in N_C\}$ is solvable iff $\{E_{\mathcal{FL}_{reg}}(A) \mid A \in N_C\}$ is solvable. Together with Theorem 4 and the reduction of unification in $\mathcal{FL}_{reg}$ to solving the system $\{E_{\mathcal{FL}_{reg}}(A) \mid A \in N_C\}$, Lemma 9 follows.