Dresden University of Technology
Institute for Theoretical Computer Science
Chair for Automata Theory

LTCS–Report

SAT Encoding of Unification in $\mathcal{EL}$

Franz Baader Barbara Morawska

LTCS-Report 10-04

Lehrstuhl für Automatentheorie
Institut für Theoretische Informatik
TU Dresden
http://lat.inf.tu-dresden.de

Hans-Grundig-Str. 25
01062 Dresden
Germany

SAT Encoding of Unification in $\mathcal{EL}$

Franz Baader
TU Dresden, Germany
baader@tcs.inf.tu-dresden.de

Barbara Morawska*
TU Dresden, Germany
morawska@tcs.inf.tu-dresden.de

May 27, 2010

Abstract

The Description Logic $\mathcal{EL}$ is an inexpressive knowledge representation language, which nevertheless has recently drawn considerable attention in the knowledge representation and the ontology community since, on the one hand, important inference problems such as the subsumption problem are polynomial. On the other hand, $\mathcal{EL}$ is used to define large biomedical ontologies. Unification in Description Logics has been proposed as a novel inference service that can, for example, be used to detect redundancies in ontologies. In a recent paper, we have shown that unification in $\mathcal{EL}$ is NP-complete, and thus of a complexity that is considerably lower than in other Description Logics of comparably restricted expressive power.

In this paper, we introduce a new NP-algorithm for solving unification problem in $\mathcal{EL}$, which is based on a reduction to satisfiability in propositional logic (SAT). The advantage of this new algorithm is, on the one hand, that it allows us to employ highly optimized state of the art SAT solvers when implementing an $\mathcal{EL}$-unification algorithm. On the other hand, this reduction provides us with a proof of the fact that $\mathcal{EL}$-unification is in NP that is much simpler than the one given in our previous paper on $\mathcal{EL}$-unification.

1 Introduction

Description logics (DLs) [3] are a well-investigated family of logic-based knowledge representation formalisms. They can be used to represent the relevant concepts of

*supported by DFG under grant BA 1122/14-1

an application domain using concept terms, which are built from concept names and role names using certain concept constructors. The DL $\mathcal{EL}$ offers the constructors conjunction ($\sqcap$), existential restriction ($\exists r.C$), and the top concept ($\top$). This description logic has recently drawn considerable attention since, on the one hand, important inference problems such as the subsumption problem are polynomial in $\mathcal{EL}$ [1, 2]. On the other hand, though quite inexpressive, $\mathcal{EL}$ can be used to define biomedical ontologies. For example, both the large medical ontology SNOMED CT and the Gene Ontology[1] can be expressed in $\mathcal{EL}$.

Unification in description logics has been proposed in [5] as a novel inference service that can, for example, be used to detect redundancies in ontologies. There, it was shown that, for the DL $\mathcal{FL}_0$, which differs from $\mathcal{EL}$ by offering value restrictions ($\forall r.C$) in place of existential restrictions, deciding unifiability is an ExpTime-complete problem. In [4], we were able to show that unification in $\mathcal{EL}$ is of considerably lower complexity: the decision problem is "only" NP-complete. However, the unification algorithm introduced in [4] to establish the NP upper bound is a brutal "guess and then test" NP-algorithm, and thus it is unlikely that a direct implementation of it will perform well in practice.

In this report, we present a new decision procedure for $\mathcal{EL}$-unification that takes a given $\mathcal{EL}$-unification problem Γ and translates it into a set of propositional clauses $C(\Gamma)$ such that (i) the size of $C(\Gamma)$ is polynomial in the size of Γ, and (ii) Γ is unifiable iff $C(\Gamma)$ is satisfiable. This allows us to use a highly-optimized SAT-solver such as MiniSat[2] to decide solvability of $\mathcal{EL}$-unification problems. Our SAT-translation is inspired by Kapur and Narendran's translation of ACIU-unification problems into satisfiability in propositional Horn logic (Horn-SAT) [8]. The connection between $\mathcal{EL}$-unification and ACIU-unification is due to the fact that (modulo equivalence) the conjunction constructor in $\mathcal{EL}$ is associative, commutative, and idempotent, and has the top concept $\top$ as a unit. However, to treat also existential restrictions correctly, we need to introduce clauses that are not Horn.

It should be noted that the proof of correctness of our translation into SAT does *not* depend on the results in [4]. Consequently, this translation provides us with a new proof of the fact that $\mathcal{EL}$-unification is in NP. This proof is much simpler than the original proof of this fact in [4].

2 Unification in $\mathcal{EL}$

Starting with a set N_{con} of concept names and a set N_{role} of role names, *$\mathcal{EL}$-concept terms* are built using the following concept constructors: the nullary constructor *top-concept* ($\top$), the binary constructor *conjunction* ($C \sqcap D$), and for every role

[1] see http://www.ihtsdo.org/snomed-ct/ and http://www.geneontology.org/

[2] http://minisat.se/

Name	Syntax	Semantics
concept name	A	$A^{\mathcal{I}} \subseteq \mathcal{D}_{\mathcal{I}}$
role name	r	$r^{\mathcal{I}} \subseteq \mathcal{D}_{\mathcal{I}} \times \mathcal{D}_{\mathcal{I}}$
top-concept	$\top$	$\top^{\mathcal{I}} = \mathcal{D}_{\mathcal{I}}$
conjunction	$C \sqcap D$	$(C \sqcap D)^{\mathcal{I}} = C^{\mathcal{I}} \cap D^{\mathcal{I}}$
existential restriction	$\exists r.C$	$(\exists r.C)^{\mathcal{I}} = \{x \mid \exists y : (x,y) \in r^{\mathcal{I}} \wedge y \in C^{\mathcal{I}}\}$
subsumption	$C \sqsubseteq D$	$C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$
equivalence	$C \equiv D$	$C^{\mathcal{I}} = D^{\mathcal{I}}$

Table 1: Syntax and semantics of $\mathcal{EL}$

name $r \in N_{role}$, the unary constructor *existential restriction* ($\exists r.C$). The semantics of $\mathcal{EL}$ is defined in the usual way, using the notion of an interpretation $\mathcal{I} = (\mathcal{D}_{\mathcal{I}}, \cdot^{\mathcal{I}})$, which consists of a nonempty domain $\mathcal{D}_{\mathcal{I}}$ and an interpretation function $\cdot^{\mathcal{I}}$ that assigns binary relations on $\mathcal{D}_{\mathcal{I}}$ to role names and subsets of $\mathcal{D}_{\mathcal{I}}$ to concept terms, as shown in the semantics column of Table 1.

The concept term C *is subsumed by* the concept term D (written $C \sqsubseteq D$) iff $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ holds for all interpretations $\mathcal{I}$. We say that C *is equivalent to* D (written $C \equiv D$) iff $C \sqsubseteq D$ and $D \sqsubseteq C$, i.e., iff $C^{\mathcal{I}} = D^{\mathcal{I}}$ holds for all interpretations $\mathcal{I}$.

The following lemma provides us with a useful *characterization of subsumption in* $\mathcal{EL}$ [4].

Lemma 2.1 *Let C, D be $\mathcal{EL}$-concept terms such that*

$$C = A_1 \sqcap \ldots \sqcap A_k \sqcap \exists r_1.C_1 \sqcap \ldots \sqcap \exists r_m.C_m,$$
$$D = B_1 \sqcap \ldots \sqcap B_\ell \sqcap \exists s_1.D_1 \sqcap \ldots \sqcap \exists s_n.D_n,$$

where $A_1, \ldots, A_k, B_1, \ldots, B_\ell$ are concept names. Then $C \sqsubseteq D$ iff

- *$\{B_1, \ldots, B_\ell\} \subseteq \{A_1, \ldots, A_k\}$ and*
- *for every $j, 1 \leq j \leq n$, there exists an $i, 1 \leq i \leq m$, such that $r_i = s_j$ and $C_i \sqsubseteq D_j$.*

When defining unification in $\mathcal{EL}$, we assume that the set of concepts names is partitioned into a set N_v of concept variables (which may be replaced by substitutions) and a set N_c of concept constants (which must not be replaced by substitutions). A *substitution* σ is a mapping from N_v into the set of all $\mathcal{EL}$-concept terms. This mapping is extended to concept terms in the usual way, i.e., by replacing all occurrences of variables in the term by their σ-images.

A substitution σ induces the following binary relation $>_\sigma$ on variables:

$X >_\sigma Y$ iff there are $n \geq 1$ role names $r_1, \ldots, r_n \in N_{role}$ such that

$$\sigma(X) \sqsubseteq \sigma(\exists r_1. \cdots \exists r_n. Y).$$

The following lemma is an easy consequence of Lemma 2.1.

Lemma 2.2 *The relation $>_\sigma$ is a strict partial order.*

Unification tries to make concept terms equivalent by applying a substitution.

Definition 2.3 *An $\mathcal{EL}$-*unification problem *is of the form $\Gamma = \{C_1 \equiv^? D_1, \ldots, C_n \equiv^? D_n\}$, where $C_1, D_1, \ldots C_n, D_n$ are $\mathcal{EL}$-concept terms. The substitution σ is a* unifier *(or* solution*) of Γ iff $\sigma(C_i) \equiv \sigma(D_i)$ for $i = 1, \ldots, n$. In this case, Γ is called* solvable *or* unifiable*.*

Note that Lemma 2.2 implies that the variable X cannot unify with the concept term $\exists r_1. \cdots \exists r_n. X$ ($n \geq 1$), i.e., the $\mathcal{EL}$-unification problem $\{X \equiv^? \exists r_1. \cdots \exists r_n. X\}$ does not have a solution. This means that an $\mathcal{EL}$-unification algorithm has to realize a kind of *occurs check*.

We will assume without loss of generality that our $\mathcal{EL}$-unification problems are flattened in the sense that they do not contain nested existential restrictions. To define this notion in more detail, we need to introduce the notion of an atom.

An $\mathcal{EL}$-concept term is called an *atom* iff it is a concept name (i.e., concept constant or concept variable) or an existential restriction $\exists r.D$. Obviously, any $\mathcal{EL}$-concept term is (equivalent to) a conjunction of atoms, where the empty conjunction is $\top$. The set $At(C)$ of *atoms of an $\mathcal{EL}$-concept term C* is defined inductively: if $C = \top$, then $At(C) := \emptyset$; if C is a concept name, then $At(C) := \{C\}$; if $C = \exists r.D$ then $At(C) := \{C\} \cup At(D)$; if $C = C_1 \sqcap C_2$, then $At(C) := At(C_1) \cup At(C_2)$.

The following lemma is an easy consequence of Lemma 2.1.

Lemma 2.4 *Let C, D be $\mathcal{EL}$-concept terms such that $C = C_1 \sqcap \ldots \sqcap C_m$ and $D = D_1 \sqcap \ldots \sqcap D_n$, where $D_1, \ldots, D_n$ are atoms. Then $C \sqsubseteq D$ iff for every $j, 1 \leq j \leq n$, there exists an $i, 1 \leq i \leq m$, such that $C_i \sqsubseteq D_j$.*

In our reduction, we will restrict the attention (without loss of generality) to unification problems that are built from atoms without nested existential restrictions. To be more precise, concept names and existential restrictions $\exists r.D$ where D is a concept name are called *flat atoms*. An $\mathcal{EL}$-concept term is *flat* iff it is a conjunction of flat atoms (where the empty conjunction is $\top$). The $\mathcal{EL}$-unification problem Γ is *flat* iff it consists of equations between flat $\mathcal{EL}$-concept terms.

By introducing new concept variables and eliminating $\top$, any $\mathcal{EL}$-unification problem Γ can be transformed in polynomial time into a flat $\mathcal{EL}$-unification problem Γ' such that Γ is solvable iff Γ' is solvable. Thus, we may assume without loss of generality that our input $\mathcal{EL}$-unification problems are flat. Given a flat $\mathcal{EL}$-unification problem $\Gamma = \{C_1 \equiv^? D_1, \ldots, C_n \equiv^? D_n\}$, we call the atoms of $C_1, D_1, \ldots, C_n, D_n$ the *atoms of* Γ.

3 The SAT encoding

In the following, let Γ be a flat $\mathcal{EL}$-unification problem. We show how to translate Γ into a set of propositional clauses $C(\Gamma)$ such that (i) the size of $C(\Gamma)$ is polynomial in the size of Γ, and (ii) Γ is unifiable iff $C(\Gamma)$ is satisfiable. The main idea underlying this translation is that we want to guess, for every pair of atoms A, B of the flat unification problem Γ, whether or not A is subsumed by B after the application of the unifier σ to be computed. In addition, we need to guess a strict partial order $>$ on the variables of Γ, which corresponds to (a subset of) the strict partial order $>_\sigma$ induced by σ.

Thus, we use the following propositional variables:

- $[A \not\sqsubseteq B]$ for every pair A, B of atoms of Γ;
- $[X > Y]$ for every pair of variables occurring in Γ.

Note that we use non-subsumption rather than subsumption for the propositional variables of the first kind since this will allow us to translate the equations of the unification problem into Horn clauses (*à la* Kapur and Narendran [8]). However, we will have to "pay" for this since expressing transitivity of subsumption then requires the use of non-Horn clauses.

Given a flat $\mathcal{EL}$-unification problem Γ, the set $C(\Gamma)$ consists of the following clauses:

(1) Translation of the equations of Γ. For every equation $A_1 \sqcap \cdots \sqcap A_m \equiv^? B_1 \sqcap \cdots \sqcap B_n$ of Γ, we create the following Horn clauses, which express that any atom that occurs as a top-level conjunct on one side of an equivalence must subsume a top-level conjunct on the other side:[3]

1. For every non-variable atom $C \in \{A_1, \ldots, A_m\}$:
   $[B_1 \not\sqsubseteq C] \wedge \ldots \wedge [B_n \not\sqsubseteq C] \rightarrow$
2. For every non-variable atom $C \in \{B_1, \ldots, B_n\}$:
   $[A_1 \not\sqsubseteq C] \wedge \ldots \wedge [A_m \not\sqsubseteq C] \rightarrow$
3. For every non-variable atom C of Γ s.t. $C \notin \{A_1, \ldots, A_m, B_1, \ldots, B_n\}$:
   $[A_1 \not\sqsubseteq C] \wedge \ldots \wedge [A_m \not\sqsubseteq C] \rightarrow [B_j \not\sqsubseteq C]$ for $j = 1, \ldots, n$
   $[B_1 \not\sqsubseteq C] \wedge \ldots \wedge [B_n \not\sqsubseteq C] \rightarrow [A_i \not\sqsubseteq C]$ for $i = 1, \ldots, m$

(2) Translation of the relevant properties of subsumption in $\mathcal{EL}$.

1. For every pair of distinct concept constants A, B occurring in Γ, we say that A cannot be subsumed by B:
   $\rightarrow [A \not\sqsubseteq B]$

[3] see Lemma 2.4.

2. For every pair of distinct role names r, s and atoms $\exists r.A, \exists s.B$ of Γ, we say that $\exists r.A$ cannot be subsumed by $\exists s.B$:
   $\rightarrow [\exists r.A \not\sqsubseteq \exists s.B]$

3. For every pair $\exists r.A, \exists r.B$ of atoms of Γ, we say that $\exists r.A$ can only be subsumed by $\exists r.B$ if A is already subsumed by B:
   $[A \not\sqsubseteq B] \rightarrow [\exists r.A \not\sqsubseteq \exists r.B]$

4. For every concept constant A and every atom $\exists r.B$ of Γ, we say that A and $\exists r.B$ are not in a subsumption relationship
   $\rightarrow [A \not\sqsubseteq \exists r.B]$ and $\rightarrow [\exists r.B \not\sqsubseteq A]$

5. Transitivity of subsumption is expressed using the *non-Horn* clauses:
   $[C_1 \not\sqsubseteq C_3] \rightarrow [C_1 \not\sqsubseteq C_2] \vee [C_2 \not\sqsubseteq C_3]$ where C_1, C_2, C_3 are atoms of Γ.

Note that there are further properties that hold for subsumption in $\mathcal{EL}$ (e.g., the fact that $A \sqsubseteq B$ implies $\exists r.A \sqsubseteq \exists r.B$), but that are not needed to ensure soundness of our translation.

(3) Translation of the relevant properties of $>$.

1. Transitivity and irreflexivity of $>$ can be expressed using the Horn clauses:
   $[X{>}X] \rightarrow$ and $[X{>}Y] \wedge [Y{>}Z] \rightarrow [X{>}Z]$,
   where X, Y, Z are concept variables occurring in Γ.

2. The connection between this order and the order $>_\sigma$ is expressed using the *non-Horn* clauses:
   $\rightarrow [X{>}Y] \vee [X \not\sqsubseteq \exists r.Y]$,
   where X, Y are concept variables occurring in Γ and $\exists r.Y$ is an atom of Γ.

Since the number of atoms of Γ is linear in the size of Γ, it is easy to see that $C(\Gamma)$ is of size polynomial in the size of Γ, and that it can be computed in polynomial time. Note, however, that without additional optimizations, the polynomial can be quite big. If the size of Γ is n, then the number of atoms of Γ is in $O(n)$. The number of possible propositional variables is thus in $O(n^2)$. The size of $C(\Gamma)$ is dominated by the number of clauses expressing the transitivity of subsumption and the transitivity of the order on variables. Thus, the size of $C(\Gamma)$ is in $O((n^2)^3) = O(n^6)$.

Example 3.1 *The following $\mathcal{EL}$-unification problem does not have a solution:*

$$\Gamma := \{X \sqcap \exists r.X \equiv^? X\}.$$

The set of clauses $C(\Gamma)$ has the following elements:

(1) The only clause created in (1) is:

$$[X \not\sqsubseteq \exists r.X] \rightarrow \ .$$

(2) Among the clauses introduced in (2) is the following:

5. $[\exists r.X \not\sqsubseteq \exists r.X] \rightarrow [\exists r.X \not\sqsubseteq X] \vee [X \not\sqsubseteq \exists r.X]$

(3) The following clauses are created in (3):

1. $[X{>}X] \rightarrow$
2. $\rightarrow [X{>}X] \vee [X \not\sqsubseteq \exists r.X]$.

It is easy to see that this set of clauses is unsatisfiable. In fact, $[X \not\sqsubseteq \exists r.X]$ *needs to be assigned the truth value* 0 *because of (1). Consequently, (3)2. implies that* $[X{>}X]$ *needs to be assigned the truth value* 1*, which then falsifies (3)1.*

The next example considers an equation where the right-hand side is the top concept, which is the empty conjunction of flat atoms.

Example 3.2 *The following* $\mathcal{EL}$*-unification problem does not have a solution:*

$$\Gamma := \{A \sqcap B \equiv^? \top\}.$$

In (1)1. we need to construct clauses for the atoms A *and* B *on the left-hand side. Since the right-hand side is the empty conjunction (i.e.,* $n = 0$*), the left-hand sides of the implications generated this way is empty, i.e., both atoms yield the implication* $\rightarrow$ *, in which both the left-hand side and the right-hand side is empty. An empty left-hand side is read as true* (1)*, whereas an empty right-hand side is read as false* (0)*. Thus, this implication is unsatisfiable.*

Theorem 3.3 (Soundness and completeness) *Let* Γ *be a flat* $\mathcal{EL}$*-unification problem. Then,* Γ *is solvable iff* $C(\Gamma)$ *is satisfiable.*

We prove this theorem in the next two subsections, one devoted to the proof of soundness and the other to the proof of completeness. After the formal proof, we will also explain the reduction on a more intuitive level.

Since our translation into SAT is polynomial and SAT is in NP, the above theorem shows that $\mathcal{EL}$-unification is in NP. NP-hardness follows from the fact that $\mathcal{EL}$-matching is known to be NP-hard [9].

Corollary 3.4 *$\mathcal{EL}$-unification is NP-complete.*

Soundness

To prove soundness, we assume that $C(\Gamma)$ is satisfiable. We must show that this implies that Γ is solvable. In order to define a unifier of Γ, we take a propositional valuation τ that satisfies $C(\Gamma)$, and use τ to define an *assignment* of sets S_X of non-variable atoms of Γ to the variables X of Γ:

$$S_X := \{C \mid C \text{ non-variable atom of } \Gamma \text{ s.t. } \tau([X \not\sqsubseteq C]) = 0\}.$$

Given this assignment of sets of non-variable atoms to the variables in Γ, we say that the variable X *directly depends on* the variable Y if Y occurs in an atom of S_X. Let *depends on* be the transitive closure of *directly depends on*.

Lemma 3.5 *Let X, Y be variables occurring in Γ.*

1. *If X* depends on *Y, then $\tau([X{>}Y]) = 1$.*
2. *The* depends on *relation is irreflexive, i.e., X cannot* depend on *itself.*

Proof. (1) If X *directly depends on* the variable Y, then Y appears in a non-variable atom of S_X. This atom must be of the form $\exists r.Y$. By the construction of S_X, $\exists r.Y \in S_X$ can only be the case if $\tau([X \not\sqsubseteq \exists r.Y]) = 0$. Since $C(\Gamma)$ contains the clause $\rightarrow [X{>}Y] \vee [X \not\sqsubseteq \exists r.Y]$, this implies $\tau([X{>}Y]) = 1$.

Since the transitivity clauses introduced in (3)1. are satisfied by τ, we also have that $\tau([X{>}Y]) = 1$ whenever X *depends on* the variable Y.

(2) If X depends on itself, then $\tau([X{>}X]) = 1$ by the first part of this lemma. This is, however, impossible since τ satisfies the clause $[X{>}X] \rightarrow$. ❑

The second part of this lemma shows that the *depends on* relation, which is transitive by definition, defines a strict partial order on variables:

$$X >_d Y \text{ iff } X \text{ depends on } Y.$$

We can now use the sets S_X to define a substitution σ along the strict partial order $>_d$:

- If X is a minimal variable w.r.t. $>_d$, then $\sigma(X)$ is the conjunction of the elements of S_X, where the empty conjunction is $\top$.
- Assume that $\sigma(Y)$ is already defined for all variables Y such that $X >_d Y$, and let $S_X = \{D_1, \ldots, D_n\}$. We define $\sigma(X) := \sigma(D_1) \sqcap \ldots \sqcap \sigma(D_n)$, where again the empty conjunction (in case $n = 0$) is $\top$.

Note that the substitution σ defined this way is actually a *ground* substitution, i.e., for all variables X occurring in Γ we have that $\sigma(X)$ does not contain variables. In the following, we will say that this substitution is *induced by* the assignment Γ.

Before we can show that σ is a unifier of Γ, we must first prove the following lemma.

Lemma 3.6 *Let C_1, C_2 be atoms of Γ. If $\tau([C_1 \not\sqsubseteq C_2]) = 0$, then $\sigma(C_1) \sqsubseteq \sigma(C_2)$.*

Proof. Assume that $\tau([C_1 \not\sqsubseteq C_2]) = 0$.

First, consider the case where C_1 is a variable. By the construction of σ, our assumption $\tau([C_1 \not\sqsubseteq C_2]) = 0$ implies that $\sigma(C_2)$ is a conjunct of $\sigma(C_1)$, and hence $\sigma(C_1) \sqsubseteq \sigma(C_2)$.

Second, consider the case where $\sigma(C_2) = \top$. Then $\sigma(C_1) \sqsubseteq \sigma(C_2)$ is obviously satisfied.

Hence, it remains to prove the lemma for the cases when C_1 is not a variable (i.e., it is a concept constant or an existential restriction) and $\sigma(C_2)$ is not $\top$. We use induction on the role depth of $\sigma(C_1) \sqcap \sigma(C_2)$, where the role depth of an $\mathcal{EL}$-concept term is the maximal nesting of existential restrictions in this term. To be more precise, if D_1, D_2, C_1, C_2 are atoms of Γ, then we define $(D_1, D_2) \succ (C_1, C_2)$ iff the role depth of $\sigma(D_1) \sqcap \sigma(D_2)$ is greater than the role depth of $\sigma(C_1) \sqcap \sigma(C_2)$.

We prove the lemma by induction on $\succ$. The base case for this induction is the case where $\sigma(C_1)$ and $\sigma(C_2)$ have role depth 0, i.e., both are conjunctions of concept constants. Since C_1 is not a variable, this implies that C_1 is a concept constant. The atom C_2 is either a concept constant or a concept variable. We consider these two cases:

- Let C_2 be a concept constant (and thus $C_2 = \sigma(C_2)$). Since $\tau([C_1 \not\sqsubseteq C_2]) = 0$ and the clauses introduced in (2)1. of the translation to SAT are satisfied by τ, we have $C_2 = C_1$, and thus $\sigma(C_1) \sqsubseteq \sigma(C_2)$.

- Assume that C_2 is a variable. Since the role depth of $\sigma(C_2)$ is 0 and $\sigma(C_2)$ is not $\top$, $\sigma(C_2)$ is a non-empty conjunction of concept constants, i.e., $\sigma(C_2) = B_1 \sqcap \cdots \sqcap B_n$ for $n \geq 1$ constants $B_1, \ldots, B_n$ such that $\tau([C_2 \not\sqsubseteq B_i]) = 0$ for $i = \{1, \ldots, n\}$. Then, since τ satisfies the transitivity clauses introduced in (2)5. of the translation to SAT, $\tau([C_1 \not\sqsubseteq B_i]) = 0$ for $i = \{1, \ldots, n\}$. Since τ satisfies the clauses introduced in (2)1. of the translation to SAT, B_i must be identical to C_1 for $i = \{1, \ldots, n\}$. Hence, $\sigma(C_2) = B_1 \sqcap \cdots \sqcap B_n \equiv C_1 = \sigma(C_1)$, which implies $\sigma(C_1) \sqsubseteq \sigma(C_2)$.

Now we assume by induction that the statement of the lemma holds for all pairs of atoms D_1, D_2 such that $(C_1, C_2) \succ (D_1, D_2)$. Notice that, if C_1 is a constant, then $\sigma(C_2)$ cannot contain an atom of the form $\exists r.D$ as a top-level conjunct. In fact, this could only be the case if either C_2 is an existential restriction, or C_2 is a variable and S_{C_2} contains an existential restriction. In the first case, $\tau([C_1 \not\sqsubseteq C_2]) = 0$ would then imply that one of the clauses introduced in (2)4. is not satisfied by τ. In the second case, τ would either need to violate one of the transitivity clauses introduced in (2)5. or one of the clauses introduced in (2)4. Thus, $\sigma(C_2)$ cannot contain an atom of the form $\exists r.D$ as a top-level conjunct. This implies that $\sigma(C_1) \sqcap \sigma(C_2)$ has role depth 0, which actually means that we are in the base case. Therefore, we can assume that C_1 is not a constant.

Since C_1 is not a variable, we have only one case to consider: C_1 is of the form $C_1 = \exists r.C$. Then, because of the clauses in (2)4. and the transitivity clauses in (2)5., $\sigma(C_2)$ cannot contain a constant as a conjunct. If C_2 is an existential restriction $C_2 = \exists s.D$, then $\tau([C_1 \not\sqsubseteq C_2]) = 0$, together with the clauses in (2)2. yields $r = s$. Consequently, $\tau([C_1 \not\sqsubseteq C_2]) = 0$, together with the clauses in (2)3., yields $\tau([C \not\sqsubseteq D] = 0$. By induction, this implies $\sigma(C) \sqsubseteq \sigma(D)$, and thus $\sigma(C_1) = \exists r.\sigma(C) \sqsubseteq \exists r.\sigma(D) = \sigma(C_2)$.

If C_2 is a variable, then $\sigma(C_2)$ must be a conjunction of atoms of the form $\exists r_1.\sigma(D_1), \ldots, \exists r_n.\sigma(D_n)$, where $\tau([C_2 \not\sqsubseteq \exists r_i.D_i]) = 0$ for $i = 1, \ldots, n$. The transitivity clauses in (2)5. yield $\tau([\exists r.C \not\sqsubseteq \exists r_1.D_1]) = \ldots = \tau([\exists r.C \not\sqsubseteq \exists r_n.D_n]) = 0$, and the clauses in (2)2. yield $r_1 = \cdots = r_n = r$. Using the clauses in (2)3., we thus obtain $\tau([C \not\sqsubseteq D_1]) = \ldots = \tau([C \not\sqsubseteq D_n]) = 0$. By induction, this implies $\sigma(C) \sqsubseteq \sigma(D_1), \ldots, \sigma(C) \sqsubseteq \sigma(D_n)$, which in turn yields $\sigma(C_1) = \exists r.\sigma(C) \sqsubseteq \exists r_1.\sigma(D_1) \sqcap \cdots \sqcap \exists r_n.\sigma(D_n) = \sigma(C_2)$. ❑

Now we can easily prove the soundness of the translation.

Proposition 3.7 (Soundness) *The substitution σ induced by a satisfying assignment of $C(\Gamma)$ is a unifier of Γ.*

Proof. We must show, for each equation $A_1 \sqcap \ldots \sqcap A_m \equiv^? B_1 \sqcap \ldots \sqcap B_n$ in Γ, that $\sigma(A_1) \sqcap \ldots \sqcap \sigma(A_m) \equiv \sigma(B_1) \sqcap \ldots \sqcap \sigma(B_n)$. Both sides of this equivalence are conjunctions of ground atoms, i.e., $\sigma(A_1) \sqcap \ldots \sqcap \sigma(A_m) = E_1 \sqcap \ldots \sqcap E_l$ and $\sigma(B_1) \sqcap \ldots \sqcap \sigma(B_n) = F_1 \sqcap \ldots \sqcap F_k$.

To prove that the equivalence holds, it is enough to show that, for each F_i, there is an A_j such that $\sigma(A_j) \sqsubseteq F_i$, and for each E_j, there is a B_i such that $\sigma(B_i) \sqsubseteq E_j$. Here we show only the first part since the other one can be shown in the same way.

First, assume that $F_i = \sigma(B_\nu)$ for a non-variable atom $B_\nu \in \{B_1, \ldots, B_n\}$. Since the clauses introduced in (1)1. of the translation are satisfied by τ, there is an A_j such that $\tau([A_j \not\sqsubseteq B_\nu]) = 0$. By Lemma 3.6, this implies $\sigma(A_j) \sqsubseteq \sigma(B_\nu) = F_i$.

If there is no non-variable atom $B_\nu \in \{B_1, \ldots, B_n\}$ such that $\sigma(B_\nu) = F_i$, then there is a variable B_ν such that the atom F_i is a conjunct of $\sigma(B_\nu)$. By the construction of σ, we know that there is a non-variable atom C of Γ such that $F_i = \sigma(C)$ and $\tau([B_\nu \not\sqsubseteq C]) = 0$. By our assumption, C is not in $\{B_1, \ldots, B_n\}$. Since the clauses created in (1)3. are satisfied by τ, there is an A_j such that $\tau([A_j \not\sqsubseteq C]) = 0$. By Lemma 3.6, this implies $\sigma(A_j) \sqsubseteq \sigma(C) = F_i$. ❑

Completeness

To show *completeness*, assume that Γ is solvable, and let γ be a unifier Γ. We must show that there is an assignment τ satisfying all the clauses in $C(\Gamma)$.

We define the propositional assignment τ as follows:

- for all non-variable atoms C, D of Γ, we define $\tau([C \not\sqsubseteq D]) := 1$ if $\gamma(C) \not\sqsubseteq \gamma(D)$; and $\tau([C \not\sqsubseteq D]) := 0$ if $\gamma(C) \sqsubseteq \gamma(D)$.
- for all variables X, Y occurring in Γ, we define $\tau([X > Y]) := 1$ if $\gamma(X) >_\gamma \gamma(Y)$; and $\tau([X > Y]) := 0$ otherwise.

In the following, we call τ the assignment *induced by* σ.

We show that τ satisfies all the clauses that are created by our translation:

(1) In (1) of the translation we create three types of Horn clauses for each equation $A_1 \sqcap \cdots \sqcap A_m \equiv^? B_1 \sqcap \cdots \sqcap B_n$.

1. If $C \in \{A_1, \ldots, A_m\}$ is a non-variable atom, then $C(\Gamma)$ contains the clause $[B_1 \not\sqsubseteq C] \wedge \cdots \wedge [B_n \not\sqsubseteq C] \rightarrow$.

   The fact that C is a non-variable atom (i.e., a concept constant or an existential restriction) implies that $\gamma(C)$ is also a concept constant or an existential restriction. Since γ is a unifier of the equation, Lemma 2.4 implies there must be an atom B_i such that $\gamma(B_i) \sqsubseteq \gamma(C)$. Therefore $\tau([B_i \not\sqsubseteq C]) = 0$, and the clause is satisfied by τ.
2. The clauses generated in (1)2. of the translation can be treated similarly.
3. If C is a non-variable atom of Γ that does not belong to $\{A_1, \ldots, A_m, B_1, \ldots, B_n\}$, then $C(\Gamma)$ contains the clause $[A_1 \not\sqsubseteq C] \wedge \cdots \wedge [A_m \not\sqsubseteq C] \rightarrow [B_k \not\sqsubseteq C]$ for $k = 1, \ldots, n$. (The symmetric clauses also introduced in (1)3. can be treated similarly.)

   To show that this clause is satisfied by τ, assume that $\tau([B_k \not\sqsubseteq C]) = 0$, i.e., $\gamma(B_k) \sqsubseteq \gamma(C)$. We must show that this implies $\tau([A_j \not\sqsubseteq C]) = 0$ for some j.

   Now, $\gamma(A_1) \sqcap \cdots \sqcap \gamma(A_m) \equiv \gamma(B_1) \sqcap \cdots \sqcap \gamma(B_n) \sqsubseteq \gamma(B_k) \sqsubseteq \gamma(C)$ implies that there is an A_j such that $\gamma(A_j) \sqsubseteq \gamma(C)$, by Lemma 2.4. Thus, or definition of τ yields $\tau([A_j \not\sqsubseteq C]) = 0$.

(2) Now we look at the clauses introduced in (2). Since two constants cannot be in a subsumption relationship, the clauses in (2)1. are satisfied by τ. Similarly, the clauses in (2)2. are satisfied by τ since no existential restriction can subsume another one built using a different role name. The clauses in (2)3. are satisfied because $\gamma(\exists r.A) \sqsubseteq \gamma(\exists r.B)$ implies $\gamma(A) \sqsubseteq \gamma(B)$, by Lemma 2.1. In a similar way we can show that all clauses in (2) are satisfied by our assignment τ. Indeed, these clauses just describe valid properties of the subsumption relation in $\mathcal{EL}$.

(3) The clauses introduced in (3) all describe valid properties of the strict partial order $>_\gamma$; hence they are satisfied by τ.

Proposition 3.8 (Completeness) *The assignment τ induced by a unifier of Γ satisfies $C(\Gamma)$.*

Some comments regarding the reduction

We have shown above that our SAT reduction is sound and complete in the sense that the (flat) $\mathcal{EL}$-unification problem Γ is solvable iff its translation $C(\Gamma)$ into a SAT problem is satisfiable. This proof is, of course, a formal justification of our definition of this translation. Here, we want to explain some aspects of this translation on a more intuitive level.

Soundness

Basically, the clauses generated in (1) enforce that "enough" subsumption relationships hold to have a unifier, i.e., solve each equation. What "enough" means is based on Lemma 2.4: once we have applied the unifier, every atom on one side of the (instantiated) equation must subsume an (instantiated) conjunct on the other side. Such an atom can either be an instance of a non-variable atom (i.e., an existential restriction or a concept constant) occurring on this side of the equation, or it is introduced by the instantiation of a variable. The first case is dealt with by the clauses in (1)1. and (1)2. whereas the second case is dealt with by (1)3. An assignment to the propositional variables of the form $[A \not\sqsubseteq B]$ guesses such subsumptions, and the clauses generated in (1) ensure that enough of them are guessed for solving all equations. However, it is not sufficient to guess enough subsumptions. We also must make sure that these subsumptions can really be made to hold by applying an appropriate substitution. This is the rôle of the clauses introduced in (2). Basically, they say that two existential restrictions can only subsume each other if they are built using the same role name, and their direct subterms subsume each other. Two concept constants subsume each other iff they are equal, and there cannot be a subsumption relation between a concept constant and an existential restriction. To ensure that all such consequences of the guessed subsumptions are really taken into account, transitivity of subsumption is needed. Otherwise, we would, for example, not detect the conflict caused by guessing that $[A \not\sqsubseteq X]$ and $[X \not\sqsubseteq B]$ should be evaluated to 0, i.e., that (for the unifier σ to be constructed) we have $\sigma(A) \sqsubseteq \sigma(X) \sqsubseteq \sigma(B)$ for distinct concept constants A, B. These kinds of conflicts correspond to what is called a *clash failure* in syntactic unification [7].

Example 3.9 *To see the clauses generated in (1) and (2) of the translation at work, let us consider a simple example, where we assume that A, B are distinct concept constants and X, Y are distinct concept variables. Consider the equation*

$$\exists r.X \equiv^{?} \exists r.Y, \tag{1}$$

which in (1)1. and (1)2. yields the clauses

$$[\exists r.Y \not\sqsubseteq \exists r.X] \rightarrow \quad \textit{and} \quad [\exists r.X \not\sqsubseteq \exists r.Y] \rightarrow \tag{2}$$

These clauses state that, for any unifier σ of the equation (1) *we must have $\sigma(\exists r.Y) \sqsubseteq \sigma(\exists r.X)$ and $\sigma(\exists r.X) \sqsubseteq \sigma(\exists r.Y)$. However, stating just these two clauses is not sufficient: we must also ensure that the assignments for the variables X and Y really realize these subsumptions. To see this, assume that we have the additional equation*

$$X \sqcap Y \equiv^? A \sqcap B, \tag{3}$$

which yields the clauses

$$[X \not\sqsubseteq A] \wedge [Y \not\sqsubseteq A] \rightarrow \quad \textit{and} \quad [X \not\sqsubseteq B] \wedge [Y \not\sqsubseteq B] \rightarrow \tag{4}$$

One possible way of satisfying these two clauses is to set

$$\tau([X \not\sqsubseteq A]) = 0 = \tau([Y \not\sqsubseteq B]) \quad \textit{and} \quad \tau([X \not\sqsubseteq B]) = 1 = \tau([Y \not\sqsubseteq A]). \tag{5}$$

The substitution σ induced by this assignment replaces X by A and Y by B, and thus clearly does not *satisfy the subsumptions $\sigma(\exists r.Y) \sqsubseteq \sigma(\exists r.X)$ and $\sigma(\exists r.X) \sqsubseteq \sigma(\exists r.Y)$. Choosing the incorrect assignment* (5) *is prevented by the clauses introduced in (2) of the translation. In fact, in (2)4. we introduce the clauses*

$$[X \not\sqsubseteq Y] \rightarrow [\exists r.X \not\sqsubseteq \exists r.Y] \quad \textit{and} \quad [Y \not\sqsubseteq X] \rightarrow [\exists r.Y \not\sqsubseteq \exists r.X] \tag{6}$$

Together with the clauses (2)*, these clauses can be used to deduce the clauses*

$$[X \not\sqsubseteq Y] \rightarrow \quad \textit{and} \quad [Y \not\sqsubseteq X] \rightarrow \tag{7}$$

Together with the transitivity clauses introduced in (2)5.:

$$[X \not\sqsubseteq B] \rightarrow [X \not\sqsubseteq Y] \vee [Y \not\sqsubseteq B] \quad \textit{and} \quad [Y \not\sqsubseteq A] \rightarrow [Y \not\sqsubseteq X] \vee [X \not\sqsubseteq A] \tag{8}$$

the clauses (7) *prevent the assignment* (5)*.*

This example illustrates, among other things, why the clauses introduced in (2)3. of the translation are needed. In fact, without the clauses (6), the incorrect assignment (5) could not have been prevented.

One may wonder why we only construct the implications in (2)3., but *not* the implications in the other direction:

$$[\exists r.A \not\sqsubseteq \exists r.B] \rightarrow [A \not\sqsubseteq B]$$

The reason is that these implications are not needed to ensure soundness.

Example 3.10 *Consider the unification problem*

$$\{X \equiv^? A, \quad Y \equiv^? \exists r.X, \quad Z \equiv^? \exists r.A\},$$

which produces the clauses

$$[X \not\sqsubseteq A] \rightarrow \ , \ [Y \not\sqsubseteq \exists r.X] \rightarrow \ , \ [Z \not\sqsubseteq \exists r.A] \rightarrow$$

The clause $[X \not\sqsubseteq A] \rightarrow$ *states that, in any unifier* σ *of the first equation, we must have* $\sigma(X) \sqsubseteq \sigma(A)$*. Though this does imply that* $\sigma(\exists r.X) \sqsubseteq \sigma(\exists r.A)$*, there is no need to state this with the clause* $[\exists r.X \not\sqsubseteq \exists r.A] \rightarrow$ *since this subsumption is not needed to solve the equation. Thus, it actually does not hurt if an assignment evaluates* $[\exists r.X \not\sqsubseteq \exists r.A]$ *with* 1*. In fact, this decision does not influence the substitution for* X *that is computed from the assignment.*

Expressed on a more technical level, the crucial tool for proving soundness is Lemma 3.6, which says that $\tau([C_1 \not\sqsubseteq C_2]) = 0$ implies $\sigma(C_1) \sqsubseteq \sigma(C_2)$ for the substitution σ induced by τ. This lemma does not state, and our proof of soundness does not need, the implication in the other direction. As illustrated in the above example, it may well be the case that $\sigma(C_1) \sqsubseteq \sigma(C_2)$ although the satisfying assignment τ evaluates $[C_1 \not\sqsubseteq C_2]$ to 1. The proof of Lemma 3.6 is by induction on the role depth, and thus reduces the problem of showing a subsumption relationship for terms of a higher role depth to the problem of showing subsumption relationships for terms of a lower role depth. This is exactly what the clauses in (2)3. allow us to do. The implications in the other direction are not required for this. They would be needed for proving the other direction of the lemma, but this is not necessary for proving soundness.

Until now, we have not mentioned the clauses generated in (3). Intuitively, they are there to detect what are called *occurs check failures* in the terminology of syntactic unification [7]. To be more precise, the variables of the form $[X{>}Y]$ together with the clauses generated in (3)1. are used to guess a strict partial order on the variables occurring in the unification problem. The clauses generated in (3)2. are used to enforce that only variables Y smaller than X can occur in the set S_X defined by a satisfying assignment. This makes it possible to use the sets S_X to define a substitution σ by induction on the strict partial order. Thus, this order realizes what is called a *constant restriction* in the literature on combining unification algorithms [6]. We have already seen the clauses generated in (3) at work in Example 3.1.

Connection to ACIU-unification *á la* Kapur and Narendran

Our reduction to SAT is an extension of Kapur and Narendran's reduction to HornSAT of unification modulo ACIU. In fact, since $\sqcap$ is associative, commutative, idempotent, and has $\top$ as a unit, $\mathcal{EL}$-unification problems not containing

existential restrictions are exactly ACIU-unification problems with constants. In this case, our reduction is basically identical to the one introduced by Kapur and Narendran in [8]. Kapur and Narendran use propositional variables $P_{A \notin X}$ in place of our variables $[X \not\sqsubseteq A]$, but the way a truth assignment τ for these variables is used to construct a substitution σ is the same as in our proof of soundness: if $\tau(P_{A \notin X}) = 0$, then the constant A is a conjunct in $\sigma(X)$.

Existential restrictions are now treated in a way similar to the treatment of free unary function symbols in the literature on combining unification algorithms [6]: the "clash rules" in (2)1., (2)2., and (2)4. and the decomposition rules in (2)3. are similar to the Martelli-Montanari-style rules for syntactic unification [7], existential restrictions are treated like constants by the ACIU-part of the unification algorithm since they are "alien subterms," and the clauses in (3) enforce a constant restriction, as already mentioned before.

However, existential restrictions are not really *free* function symbols since they are monotonic w.r.t. subsumption. This is taken into account in our translation by guessing subsumption relationships not only between variables and constants/existential restrictions, but also between existential restriction, and by introducing clauses that state relevant properties of subsumption (like the fact that this relation is transitive).

Non-Horn clauses

In our translation, we follow the approach by Kapur and Narendran, and use propositional variables that express non-subsumption rather than subsumption. This ensures that the clauses introduced in (1) are Horn, but it causes the clauses introduced in (2)5. and (3)2. to become non-Horn.

Since HornSAT can be solved in polynomial time and $\mathcal{EL}$-unification is NP-hard, it is clear that there cannot by a polynomial time translation of $\mathcal{EL}$-unification into HornSAT (unless P=NP). Consequently, some non-Horn clauses must show up in such a translation.

Instead of using propositional variables $[C \not\sqsubseteq D]$ that express non-subsumption, we could also use propositional variables $[C \sqsubseteq D]$ expressing subsumption. Then, the clauses in (2)5. would become Horn:

$$[C_1 \sqsubseteq C_2] \wedge [C_2 \sqsubseteq C_3] \rightarrow [C_1 \sqsubseteq C_3]$$

and the same would be true for the clauses in (3)2.:

$$[X \sqsubseteq \exists r.Y] \rightarrow [X > Y]$$

but the clauses in (1) would become non-Horn:

1. For every non-variable atom $C \in \{A_1, \ldots, A_m\}$:
   $\rightarrow [B_1 \sqsubseteq C] \vee \ldots \vee [B_n \sqsubseteq C]$

2. For every non-variable atom $C \in \{B_1, \ldots, B_n\}$:
   $\rightarrow [A_1 \sqsubseteq C] \vee \ldots \vee [A_m \sqsubseteq C]$

3. For every non-variable atom C of Γ s.t. $C \notin \{A_1, \ldots, A_m, B_1, \ldots, B_n\}$:
   $[A_j \sqsubseteq C] \rightarrow [B_1 \sqsubseteq C] \vee \ldots \vee [B_n \sqsubseteq C]$ for $j = 1, \ldots, m$
   $[B_i \sqsubseteq C] \rightarrow [A_1 \sqsubseteq C] \vee \ldots \vee [A_n \sqsubseteq C]$ for $i = 1, \ldots, m$

It is a priori not clear which of these approaches is better in practice; it may well be the case that none is uniformly better than the other, i.e., it may depend on the specific unification problem which one behaves better.

4 Connection to the original "in NP" proof for $\mathcal{EL}$-unification

It should be noted that in the present paper we give a proof of the fact that $\mathcal{EL}$-unification is in NP that is independent of the proof in [4]. The only result from [4] that we have used is the characterization of subsumption (Lemma 2.1), which is an easy consequence of known results for $\mathcal{EL}$ [9].

In [4], the "in NP" result is basically shown as follows:

1. define a well-founded partial order $\succ$ on substitutions and use this to show than any solvable $\mathcal{EL}$-unification problem has a ground unifier that is minimal w.r.t. this order;[4]

2. show that minimal ground unifiers are local in the sense that they are built from atoms of Γ;

3. use the locality of minimal ground unifiers to devise a "guess and then test" NP-algorithm for generating a minimal ground unifier.

The proof of 2., which shows that a non-local unifier cannot be minimal, is quite involved. Compared to that proof, the proof of soundness and completeness given in the present paper is much simpler.

In order to give a closer comparison between the approach used in [4] and the one employed in the present paper, let us recall some of the definitions and results from [4] in more detail:

Definition 4.1 *Let Γ be a flat $\mathcal{EL}$-unification problem, and γ a ground unifier of Γ. Then γ is called* local *if, for each variable X in Γ, there are $n \geq 0$ non-variable atoms $D_1, \ldots, D_n$ of Γ such that $\gamma(X) = \gamma(D_1) \sqcap \cdots \sqcap \gamma(D_n)$, where the empty conjunction is $\top$.*

[4]Recall that a unifier γ of Γ is ground if, for all variables X occurring in Γ, the concept term $\gamma(X)$ does not contain variables.

The "guess and then test" algorithm in [4] crucially depends on the fact that any solvable $\mathcal{EL}$-unification problem has a local unifier. This result can be obtained as an easy consequence of our proof of soundness and completeness.

Corollary 4.2 *Let Γ be a flat $\mathcal{EL}$-unification problem that is solvable. Then Γ has a local unifier.*

Proof. Since Γ is solvable, our completeness result implies that $C(\Gamma)$ is satisfiable. Let τ be an assignment that satisfies $C(\Gamma)$, and let σ be the unifier of Γ induced by τ in our proof of soundness. Locality of σ is an immediate consequence of the definition of σ. ❑

This shows that one does not really need the notion of minimality, and the quite involved proof that minimal unifiers are local given in [4], to justify the completeness of the "guess and then test" algorithm from [4]. However, in [4] minimal unifiers are also used to show a stronger completeness result for the "guess and then test" algorithm: it is shown that (up to equivalence) every minimal ground unifier is computed by the algorithm. In the following, we show that this is also the case for the unification algorithm obtained through our reduction.

Definition 4.3 *Let σ and γ be substitutions, and Γ an $\mathcal{EL}$-unification problem. We define*

- *$\gamma \succeq \sigma$ if, for each variable X in Γ, we have $\gamma(X) \sqsubseteq \sigma(X)$;*
- *$\gamma \equiv \sigma$ if $\gamma \succeq \sigma$ and $\sigma \succeq \gamma$, and $\gamma \succ \sigma$ if $\gamma \succeq \sigma$ and $\sigma \not\equiv \gamma$;*
- *γ is a* minimal *unifier of Γ if there is no unifier σ of Γ such that $\gamma \succ \sigma$.*

As a corollary to our soundness and completeness proof, we can show that any minimal ground unifier σ of Γ is computed by our reduction, in the sense that it is induced by a satisfying assignment of $C(\Gamma)$.

Corollary 4.4 *Let Γ be a flat $\mathcal{EL}$-unification problem. If γ is a minimal ground unifier of Γ, then there is a unifier σ, induced by a satisfying assignment τ of $C(\Gamma)$, such that $\sigma \equiv \gamma$.*

Proof. Let γ be a minimal ground unifier of Γ, and τ the satisfying assignment of $C(\Gamma)$ induced by γ. We show that the unifier σ of Γ induced by τ satisfies $\gamma \succeq \sigma$. Minimality of γ then implies $\gamma \equiv \sigma$.

We must show that, for each variable X occurring in Γ, we have $\gamma(X) \sqsubseteq \sigma(X)$. We prove this by well-founded induction on the strict partial order $>$ defined as[5]

$$X > Y \quad \text{iff} \quad \tau([X{>}Y]) = 1.$$

[5]The clauses in $C(\Gamma)$ make sure that this is indeed a strict partial order. It is trivially well-founded since Γ contains only finitely many variables.

Let X be a minimal variable with respect to this order. Since τ satisfies the clauses in (3)2., the set S_X induced by τ (see the proof of soundness) contains only ground atoms. Let $S_X = \{C_1, \ldots, C_n\}$ for $n \geq 0$ ground atoms. If $n = 0$, then $\sigma(X) = \top$, and thus $\gamma(X) \sqsubseteq \sigma(X)$ is trivially satisfied. Otherwise, we have $\sigma(X) = \sigma(C_1) \sqcap \ldots \sqcap \sigma(C_n) = C_1 \sqcap \ldots \sqcap C_n$, and we know, for each $i \in \{1, \ldots, n\}$, that $\tau([X \not\sqsubseteq C_i]) = 0$ by the definition of S_X. Since τ is the assignment induced by the unifier γ, this implies that $\gamma(X) \sqsubseteq \gamma(C_i) = C_i$. Consequently, we have shown that $\gamma(X) \sqsubseteq C_1 \sqcap \ldots \sqcap C_n = \sigma(X)$.

Now we assume, by induction, that we have $\gamma(Y) \sqsubseteq \sigma(Y)$ for all variables Y such that $X > Y$. Let $S_X = \{C_1, \ldots, C_n\}$ for $n \geq 0$ non-variable atoms of Γ. If $n = 0$, then $\sigma(X) = \top$, and thus $\gamma(X) \sqsubseteq \sigma(X)$ is again trivially satisfied. Otherwise, we have $\sigma(X) = \sigma(C_1) \sqcap \cdots \sqcap \sigma(C_n)$, and we know, for each $i \in \{1, \ldots, n\}$, that $\tau([X \not\sqsubseteq C_i]) = 0$ by the definition of S_X. Since τ is the assignment induced by the unifier γ, this implies that $\gamma(X) \sqsubseteq \gamma(C_i)$. for each $i \in \{1, \ldots, n\}$. Since all variables occurring in $C_1, \ldots, C_n$ are smaller than X and since the concept constructors of $\mathcal{EL}$ are monotonic w.r.t. subsumption, we have by induction that $\gamma(C_i) \sqsubseteq \sigma(C_i)$ for each $i \in \{1, \ldots, n\}$. Consequently, we have $\gamma(X) \sqsubseteq \gamma(C_1) \sqcap \ldots \sqcap \gamma(C_n) \sqsubseteq \sigma(C_1) \sqcap \cdots \sqcap \sigma(C_n) = \sigma(X)$. ❑

5 Conclusion

The results presented in this paper are of interest both from a theoretical and a practical point of view. From the theoretical point of view, this paper gives a new proof of the fact that $\mathcal{EL}$-unification is in NP, which is considerably simpler than the original proof given in [4]. We have also shown that the stronger completeness result for the "guess and then test" NP algorithm of [4] (all minimal ground unifiers are computed) holds as well for the new algorithm presented in this paper.

From the practical point of view, the translation into propositional satisfiability allows us to employ highly optimized state of the art SAT solvers when implementing an $\mathcal{EL}$-unification algorithm. We have actually implemented the SAT translation described in this paper in Java, and have used MiniSat for the satisfiability check. Until now, we have tested the algorithm only on relatively small unification problems, which nevertheless produced thousands of clauses with hundreds of propositional variables. However, MiniSat had no problem testing satisfiability within a few milliseconds in each case, which we find quite promising.

References

[1] Franz Baader. Terminological cycles in a description logic with existential restrictions. In *Proc. of the 18th Int. Joint Conf. on Artificial Intelligence*

(IJCAI 2003), pages 325–330, 2003. Morgan Kaufmann, Los Altos.

[2] Franz Baader, Sebastian Brandt, and Carsten Lutz. Pushing the $\mathcal{EL}$ envelope. In *Proc. of the 19th Int. Joint Conf. on Artificial Intelligence (IJCAI 2005)*, pages 364–369, Edinburgh (UK), 2005. Morgan Kaufmann, Los Altos.

[3] Franz Baader, Diego Calvanese, Deborah McGuinness, Daniele Nardi, and Peter F. Patel-Schneider, editors. *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press, 2003.

[4] Franz Baader and Barbara Morawska. Unification in the description logic $\mathcal{EL}$. In *Proceedings of the 20th International Conference on Rewriting Techniques and Applications (RTA 2009)*, volume 5595 of Lecture Notes in Computer Science, pages 350364. Springer-Verlag, 2009.

[5] Franz Baader and Paliath Narendran. Unification of concepts terms in description logics. *J. of Symbolic Computation*, 31(3):277–305, 2001.

[6] Franz Baader and Klaus Schulz. Unification in the union of disjoint equational theories: Combining decision procedures. *J. of Symbolic Computation*, 21(2):211–243, 1996.

[7] Franz Baader and Wayne Snyder. Unification theory. In *Handbook of Automated Reasoning*, volume I. Elsevier Science Publishers, 2001.

[8] Deepak Kapur and Paliath Narendran. Complexity of unification problems with associative-commutative operators. *J. Automated Reasoning*, 9:261–288, 1992.

[9] Ralf Küsters. *Non-standard Inferences in Description Logics*, volume 2100 of *Lecture Notes in Artificial Intelligence*. Springer-Verlag, 2001.