# A Goal-Oriented Algorithm for Unification in $\mathcal{ELH}_{R^+}$ w.r.t. Cycle-Restricted Ontologies*

Franz Baader, Stefan Borgwardt, and Barbara Morawska
{baader,stefborg,morawska}@tcs.inf.tu-dresden.de

Theoretical Computer Science, TU Dresden, Germany

**Abstract** Unification in Description Logics (DLs) has been proposed as an inference service that can, for example, be used to detect redundancies in ontologies. For the DL $\mathcal{EL}$, which is used to define several large biomedical ontologies, unification is NP-complete. A goal-oriented NP unification algorithm for $\mathcal{EL}$ that uses nondeterministic rules to transform a given unification problem into solved form has recently been presented. In this paper, we extend this goal-oriented algorithm in two directions: on the one hand, we add general concept inclusion axioms (GCIs), and on the other hand, we add role hierarchies ($\mathcal{H}$) and transitive roles ($R^+$). For the algorithm to be complete, however, the ontology consisting of the GCIs and role axioms needs to satisfy a certain cycle restriction.

## 1 Introduction

The DL $\mathcal{EL}$, which offers the constructors conjunction ($\sqcap$), existential restriction ($\exists r.C$), and the top concept ($\top$), has recently drawn considerable attention since, on the one hand, important inference problems such as the subsumption problem are polynomial in $\mathcal{EL}$, even in the presence of general concept inclusions (GCIs) [12]. On the other hand, though quite inexpressive, $\mathcal{EL}$ can be used to define biomedical ontologies, such as the large medical ontology SNOMED CT.[1] A tractable extension of $\mathcal{EL}$ [7], which includes role hierarchy and transitivity axioms, is the basis of the OWL 2 EL profile of the new Web Ontology Language OWL 2.[2]

Unification in DLs has been proposed in [11] as a novel inference service that can, for instance, be used to detect redundancies in ontologies. For example, assume that one developer of a medical ontology defines the concept of a *patient with severe injury of the frontal lobe* as

$$\exists \mathsf{finding}.(\mathsf{Frontal\_lobe\_injury} \sqcap \exists \mathsf{severity}.\mathsf{Severe}), \tag{1}$$

whereas another one represents it as

$$\exists \mathsf{finding}.(\mathsf{Severe\_injury} \sqcap \exists \mathsf{finding\_site}.\exists \mathsf{part\_of}.\mathsf{Frontal\_lobe}). \tag{2}$$

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

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

[2] See http://www.w3.org/TR/owl2-profiles/

These two concept descriptions are not equivalent, but they are nevertheless meant to represent the same concept. They can obviously be made equivalent by treating the concept names Frontal_lobe_injury and Severe_injury as variables, and substituting the first one by Injury ⊓ ∃finding_site.∃part_of.Frontal_lobe and the second one by Injury ⊓ ∃severity.Severe. In this case, we say that the descriptions are unifiable, and call the substitution that makes them equivalent a *unifier*.

Our interest in unification w.r.t. GCIs, role hierarchies, and transitive roles stems from the fact that these features are important for expressing medical knowledge. For example, assume that the developers use the descriptions (3) and (4) instead of (1) and (2):

$$\exists \mathsf{finding}.\exists \mathsf{finding\_site}.\exists \mathsf{part\_of}.\mathsf{Brain} \sqcap \exists \mathsf{finding}.(\mathsf{Frontal\_lobe\_injury} \sqcap \exists \mathsf{severity}.\mathsf{Severe}) \tag{3}$$

$$\exists \mathsf{status}.\mathsf{Emergency} \sqcap \exists \mathsf{finding}.(\mathsf{Severe\_injury} \sqcap \exists \mathsf{finding\_site}.\exists \mathsf{part\_of}.\mathsf{Frontal\_lobe}) \tag{4}$$

The descriptions (3) and (4) are not unifiable without additional background knowledge, but they are unifiable, with the same unifier as above, if the GCIs

$$\exists \mathsf{finding}.\exists \mathsf{severity}.\mathsf{Severe} \sqsubseteq \exists \mathsf{status}.\mathsf{Emergency},$$
$$\mathsf{Frontal\_lobe} \sqsubseteq \exists \mathsf{proper\_part\_of}.\mathsf{Brain}$$

are present in a background ontology and this ontology additionally states that part_of is transitive and proper_part_of is a subrole of part_of.

In [8], we were able to show that unification in the DL $\mathcal{EL}$ (without GCIs and role axioms) is NP-complete. In addition to a brute-force "guess and then test" NP-algorithm [8], we have developed a goal-oriented unification algorithm for $\mathcal{EL}$, in which nondeterministic decisions are only made if they are triggered by "unsolved parts" of the unification problem [10], and an algorithm that is based on a reduction to satisfiability in propositional logic (SAT) [9], which enables the use of highly-optimized SAT solvers [14]. Whereas both approaches are clearly better than the brute-force algorithm, none of them is uniformly better than the other. First experiments with our system UEL [1] show that the SAT translation is usually faster in deciding unifiability, but it needs more space than the goal-oriented algorithm and it produces more uninteresting and large unifiers. In fact, the SAT translation generates all so-called local unifiers, whereas the goal-oriented algorithm produces all so-called minimal unifiers, though it may also produce some non-minimal ones. The set of minimal unifiers is a subset of the set of local unifiers, and in our experiments the minimal unifiers usually made more sense in the application.

In [10] it was shown that the approaches for unification of $\mathcal{EL}$-concept descriptions (without any background ontology) mentioned above can easily be extended to the case of a so-called acyclic TBox (a simple form of GCIs, which basically introduce abbreviations for concept descriptions) as background ontology without really changing the algorithms or increasing their complexity.

For more general GCIs, such a simple solution is no longer possible. In [2], we extended the brute-force "guess and then test" NP-algorithm from [8] to the case of GCIs, which required the development of a new characterization of subsumption w.r.t. GCIs in $\mathcal{EL}$. Unfortunately, the algorithm is complete only for general TBoxes (i.e., finite sets of GCIs) that satisfy a certain restriction on cycles, which, however, does not prevent all cycles. For example, the cyclic GCI $\exists \mathsf{child}.\mathsf{Human} \sqsubseteq \mathsf{Human}$ satisfies this restriction, whereas the cyclic GCI $\mathsf{Human} \sqsubseteq \exists \mathsf{parent}.\mathsf{Human}$ does not. In [5] we provide a more practical unification algorithm that is based on a translation into SAT, and can also deal with role hierarchies and transitive roles, but still needs the ontology (now consisting of GCIs and role axioms) to be cycle-restricted. In the presence of role hierarchies ($\mathcal{H}$) and transitive roles ($R^+$), we use the name $\mathcal{EL}\mathcal{H}_{R^+}$ rather than $\mathcal{EL}$ for the logic.

Motivated by our experience that, for the case of $\mathcal{EL}$ without background ontology, the goal-oriented algorithm sometimes behaves better than the one based on a translation into SAT, we introduce in this paper a goal-oriented algorithm for unification in $\mathcal{EL}\mathcal{H}_{R^+}$ w.r.t. cycle-restricted ontologies.[3] Full proofs of the presented results can be found in [3].

## 2 The Description Logics $\mathcal{EL}$ and $\mathcal{EL}\mathcal{H}_{R^+}$

The expressiveness of a DL is determined both by the formalism for describing concepts (the concept description language) and the terminological formalism, which can be used to state additional constraints on the interpretation of concepts and roles in a so-called ontology.

The *concept description language* considered in this paper is called $\mathcal{EL}$. Starting with a finite set $N_C$ of *concept names* and a finite set $N_R$ of *role names*, *$\mathcal{EL}$-concept descriptions* are built from concept names using the constructors *conjunction* ($C \sqcap D$), *existential restriction* ($\exists r.C$ for every $r \in N_R$), and *top* ($\top$). Since in this paper we only consider $\mathcal{EL}$-concept descriptions, we will sometimes dispense with the prefix $\mathcal{EL}$.

On the *semantic side*, concept descriptions are interpreted as sets. To be more precise, an *interpretation* $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ consists of a non-empty domain $\Delta^{\mathcal{I}}$ and an interpretation function $\cdot^{\mathcal{I}}$ that maps concept names to subsets of $\Delta^{\mathcal{I}}$ and role names to binary relations over $\Delta^{\mathcal{I}}$. This function is inductively extended to concept descriptions as follows:

$$\top^{\mathcal{I}} := \Delta^{\mathcal{I}}, \quad (C \sqcap D)^{\mathcal{I}} := C^{\mathcal{I}} \cap D^{\mathcal{I}}, \quad (\exists r.C)^{\mathcal{I}} := \{x \mid \exists y : (x, y) \in r^{\mathcal{I}} \wedge y \in C^{\mathcal{I}}\}$$

A *general concept inclusion axiom (GCI)* is of the form $C \sqsubseteq D$ for concept descriptions $C, D$, a *role hierarchy axiom* is of the form $r \sqsubseteq s$ for role names

[3] A previous version of this paper, which considers unification in $\mathcal{EL}$ w.r.t. cycle-restricted ontologies, but without role hierarchies and transitive roles, has been presented in 2012 at the Description Logic workshop (see [4]).

$r, s$, and a *transitivity axiom* is of the form $r \circ r \sqsubseteq r$ for a role name $r$. An interpretation $\mathcal{I}$ *satisfies* such an axiom $C \sqsubseteq D$, $r \sqsubseteq s$, $r \circ r \sqsubseteq r$, respectively, iff

$$C^{\mathcal{I}} \subseteq D^{\mathcal{I}}, \quad r^{\mathcal{I}} \subseteq s^{\mathcal{I}}, \quad \text{and} \quad r^{\mathcal{I}} \circ r^{\mathcal{I}} \subseteq r^{\mathcal{I}},$$

where $\circ$ stands for composition of binary relations. An $\mathcal{EL}\mathcal{H}_{R^+}$*-ontology* is a finite set of such axioms. It is an $\mathcal{EL}$*-ontology* if it contains only GCIs. An interpretation is a *model* of an ontology if it satisfies all its axioms.

A concept description $C$ is *subsumed* by a concept description $D$ w.r.t. an ontology $\mathcal{O}$ (written $C \sqsubseteq_{\mathcal{O}} D$) if every model of $\mathcal{O}$ satisfies the GCI $C \sqsubseteq D$. We say that $C$ is *equivalent* to $D$ w.r.t. $\mathcal{O}$ ($C \equiv_{\mathcal{O}} D$) if $C \sqsubseteq_{\mathcal{O}} D$ and $D \sqsubseteq_{\mathcal{O}} C$. If $\mathcal{O}$ is empty, we also write $C \sqsubseteq D$ and $C \equiv D$ instead of $C \sqsubseteq_{\mathcal{O}} D$ and $C \equiv_{\mathcal{O}} D$, respectively. As shown in [12,7], subsumption w.r.t. $\mathcal{EL}\mathcal{H}_{R^+}$-ontologies (and thus also w.r.t. $\mathcal{EL}$-ontologies) is decidable in polynomial time.

Since conjunction is interpreted as intersection, the concept descriptions $(C \sqcap D) \sqcap E$ and $C \sqcap (D \sqcap E)$ are always equivalent. Thus, we dispense with parentheses and write nested conjunctions in flat form $C_1 \sqcap \cdots \sqcap C_n$. Nested existential restrictions $\exists r_1.\exists r_2. \ldots \exists r_n.C$ will sometimes also be written as $\exists r_1 r_2 \ldots r_n.C$, where $r_1 r_2 \ldots r_n$ is viewed as a word over the alphabet of role names, i.e. an element of $N_R^*$.

The *role hierarchy* induced by $\mathcal{O}$ is a binary relation $\trianglelefteq_{\mathcal{O}}$ on $N_R$, which is defined as the reflexive-transitive closure of the relation $\{(r, s) \mid r \sqsubseteq s \in \mathcal{O}\}$. Using elementary reachability algorithms, the role hierarchy can be computed in polynomial time in the size of $\mathcal{O}$. It is easy to see that $r \trianglelefteq_{\mathcal{O}} s$ implies that $r^{\mathcal{I}} \subseteq s^{\mathcal{I}}$ for all models $\mathcal{I}$ of $\mathcal{O}$. Given an $\mathcal{EL}\mathcal{H}_{R^+}$-ontology $\mathcal{O}$, we call the role $t$ *transitive w.r.t.* $\mathcal{O}$ if $\mathcal{O}$ contains the axiom $t \circ t \sqsubseteq t$. If $\mathcal{O}$ is clear from the context, we often omit the suffix "w.r.t. $\mathcal{O}$" and call $t$ a *transitive* role.

An $\mathcal{EL}$-concept description is an *atom* if it is an existential restriction or a concept name. The atoms of an $\mathcal{EL}$-concept description $C$ are the subdescriptions of $C$ that are atoms, and the top-level atoms of $C$ are the atoms occurring in the top-level conjunction of $C$. Obviously, any $\mathcal{EL}$-concept description is the conjunction of its top-level atoms, where the empty conjunction corresponds to $\top$. The atoms of an $\mathcal{EL}\mathcal{H}_{R^+}$-ontology $\mathcal{O}$ are the atoms of all the concept descriptions occurring in GCIs of $\mathcal{O}$.

We say that a subsumption between two atoms is *structural* if their top-level structure is compatible. To be more precise, following [5] we define structural subsumption between atoms as follows: the atom $C$ is *structurally subsumed* by the atom $D$ w.r.t. $\mathcal{O}$ ($C \sqsubseteq^{\mathsf{s}}_{\mathcal{O}} D$) iff one of the following holds:

1. $C = D$ is a concept name,
2. $C = \exists r.C'$, $D = \exists s.D'$, $r \trianglelefteq_{\mathcal{O}} s$, and $C' \sqsubseteq_{\mathcal{O}} D'$.
3. $C = \exists r.C'$, $D = \exists s.D'$, and $C' \sqsubseteq_{\mathcal{O}} \exists t.D'$ for a transitive role $t$ such that $r \trianglelefteq_{\mathcal{O}} t \trianglelefteq_{\mathcal{O}} s$.

It is easy to see that subsumption w.r.t. $\emptyset$ between two atoms implies structural subsumption w.r.t. $\mathcal{O}$, which in turn implies subsumption w.r.t. $\mathcal{O}$. The unification algorithm presented below crucially depends on the following characterization of subsumption:

**Lemma 1.** *Let $\mathcal{O}$ be an $\mathcal{ELH}_{R^+}$-ontology and $C_1, \ldots, C_n$, $D_1, \ldots, D_m$ be atoms. Then $C_1 \sqcap \cdots \sqcap C_n \sqsubseteq_{\mathcal{O}} D_1 \sqcap \cdots \sqcap D_m$ iff for every $j \in \{1, \ldots, m\}$*

1. *there is an index $i \in \{1, \ldots, n\}$ such that $C_i \sqsubseteq^{\mathsf{s}}_{\mathcal{O}} D_j$ or*
2. *there are atoms $A_1, \ldots, A_k, B$ of $\mathcal{O}$ ($k \geq 0$) such that*
   *(a) $A_1 \sqcap \cdots \sqcap A_k \sqsubseteq_{\mathcal{O}} B$,*
   *(b) for every $\eta \in \{1, \ldots, k\}$ there is $i \in \{1, \ldots, n\}$ with $C_i \sqsubseteq^{\mathsf{s}}_{\mathcal{O}} A_\eta$, and*
   *(c) $B \sqsubseteq^{\mathsf{s}}_{\mathcal{O}} D_j$.*

Our proof of this lemma in [3] is based on a Gentzen-style proof calculus for subsumption w.r.t. $\mathcal{ELH}_{R^+}$-ontologies, which is similar to the one developed in [15] for subsumption w.r.t. $\mathcal{EL}$-ontologies. Although this characterization looks identical to the one given in [2] for the case of $\mathcal{EL}$-ontologies it differs from that characterization in that it uses a more general notion of structural subsumption. Also note that the characterization of subsumption w.r.t. $\mathcal{ELH}_{R^+}$-ontologies employed in [5] to show correctness of the the SAT translation is different from the one given above, and it is proved using a rewriting approach rather than a Gentzen-style proof calculus.

As mentioned in the introduction, our unification algorithm is complete only for $\mathcal{ELH}_{R^+}$-ontologies that satisfy a certain restriction on cycles.

**Definition 2.** *The $\mathcal{ELH}_{R^+}$-ontology $\mathcal{O}$ is called* cycle-restricted *iff there is no nonempty word $w \in N_R^+$ and $\mathcal{EL}$-concept description $C$ such that $C \sqsubseteq_{\mathcal{O}} \exists w.C$.*

In [5] we show that a given $\mathcal{ELH}_{R^+}$-ontology can be tested for cycle-restrictedness in polynomial time. The main idea is that it is sufficient to consider the cases where $C$ is a concept name or $\top$.

## 3 Unification in $\mathcal{ELH}_{R^+}$

We partition the set $N_C$ 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* $\sigma$ maps every concept variable to an $\mathcal{EL}$-concept description. It is extended to concept descriptions in the usual way:

- $\sigma(A) := A$ for all $A \in N_c \cup \{\top\}$,
- $\sigma(C \sqcap D) := \sigma(C) \sqcap \sigma(D)$ and $\sigma(\exists r.C) := \exists r.\sigma(C)$.

An $\mathcal{EL}$-concept description $C$ is *ground* if it does not contain variables. Obviously, a ground concept description is not modified by applying a substitution. An $\mathcal{ELH}_{R^+}$-ontology is *ground* if it does not contain variables.

**Definition 3.** *Let $\mathcal{O}$ be an $\mathcal{ELH}_{R^+}$-ontology that is ground. An $\mathcal{ELH}_{R^+}$-unification problem w.r.t. $\mathcal{O}$ is a finite set $\Gamma = \{C_1 \sqsubseteq^? D_1, \ldots, C_n \sqsubseteq^? D_n\}$ of subsumptions between $\mathcal{EL}$-concept descriptions. A substitution $\sigma$ is a* unifier *of $\Gamma$ w.r.t. $\mathcal{O}$ if $\sigma$* solves *all the subsumptions in $\Gamma$, i.e. if $\sigma(C_1) \sqsubseteq_{\mathcal{O}} \sigma(D_1), \ldots, \sigma(C_n) \sqsubseteq_{\mathcal{O}} \sigma(D_n)$. We say that $\Gamma$ is* unifiable *w.r.t. $\mathcal{O}$ if it has a unifier.*

Note that some of the previous papers on unification in DLs use equivalences $C \equiv^? D$ instead of subsumptions $C \sqsubseteq^? D$. This difference is, however, irrelevant since $C \equiv^? D$ can be seen as a shorthand for the two subsumptions $C \sqsubseteq^? D$ and $D \sqsubseteq^? C$, and $C \sqsubseteq^? D$ has the same unifiers as $C \sqcap D \equiv^? C$. Also note that we have restricted the background ontology $\mathcal{O}$ to be ground. This is not without loss of generality. If $\mathcal{O}$ contained variables, then we would need to apply the substitution also to its GCIs, and instead of requiring $\sigma(C_i) \sqsubseteq_{\mathcal{O}} \sigma(D_i)$ we would thus need to require $\sigma(C_i) \sqsubseteq_{\sigma(\mathcal{O})} \sigma(D_i)$, which would change the nature of the problem considerably (see [6] for a more detailed discussion).

**Preprocessing** To simplify the description of the algorithm, it is convenient to first normalize the ontology and the unification problem appropriately. An atom is called *flat* if it is a concept name or an existential restriction of the form $\exists r.A$ for a concept name $A$. The $\mathcal{ELH}_{R^+}$-ontology $\mathcal{O}$ is called *flat* if it contains only GCIs of the form $A \sqcap B \sqsubseteq C$, where $A, B$ are flat atoms or $\top$ and $C$ is a flat atom. The unification problem $\Gamma$ is called *flat* if it contains only flat subsumptions of the form $C_1 \sqcap \cdots \sqcap C_n \sqsubseteq^? D$, where $n \geq 0$ and $C_1, \ldots, C_n, D$ are flat atoms.[4] Let $\Gamma$ be a unification problem and $\mathcal{O}$ an $\mathcal{ELH}_{R^+}$-ontology. By introducing auxiliary variables and concept names, respectively, $\Gamma$ and $\mathcal{O}$ can be transformed in polynomial time into a flat unification problem $\Gamma'$ and a flat $\mathcal{ELH}_{R^+}$-ontology $\mathcal{O}'$ such that the unifiability status remains unchanged, i.e., $\Gamma$ has a unifier w.r.t. $\mathcal{O}$ iff $\Gamma'$ has a unifier w.r.t. $\mathcal{O}'$. In addition, if $\mathcal{O}$ was cycle-restricted, then so is $\mathcal{O}'$ (see [6] for details). Thus, we can assume without loss of generality that the input unification problem and ontology are flat.

**Local Unifiers** The main idea underlying the "in NP" results in [8,2] is to show that any unification problem that is unifiable has a so-called local unifier.

We denote by At the set of atoms occurring as subdescriptions in subsumptions in $\Gamma$ or axioms in $\mathcal{O}$ and define

$$\mathrm{At}_{\mathsf{tr}} := \mathrm{At} \cup \{\exists t.D' \mid \exists s.D' \in \mathrm{At},\ t \trianglelefteq_{\mathcal{O}} s,\ t \text{ transitive}\}.$$

Furthermore, we define the set of *non-variable atoms* by $\mathrm{At}_{\mathsf{nv}} := \mathrm{At}_{\mathsf{tr}} \setminus N_v$. Though the elements of $\mathrm{At}_{\mathsf{nv}}$ cannot be variables, they may contain variables if they are of the form $\exists r.X$ for some role $r$ and a variable $X$.

We call a function $S$ that associates every variable $X \in N_v$ with a set $S_X \subseteq \mathrm{At}_{\mathsf{nv}}$ an *assignment*. Such an assignment induces the following relation $>_S$ on $N_v$: $>_S$ is the transitive closure of

$$\{(X, Y) \in N_v \times N_v \mid Y \text{ occurs in an element of } S_X\}.$$

We call the assignment $S$ *acyclic* if $>_S$ is irreflexive (and thus a strict partial order). Any acyclic assignment $S$ induces a unique substitution $\sigma_S$, which can be defined by induction along $>_S$:

[4] If $n = 0$, then we have an empty conjunction on the left-hand side, which as usual stands for $\top$.

– If $X \in N_v$ is minimal w.r.t. $>_S$, then we define $\sigma_S(X) := \bigsqcap_{D \in S_X} D$.
– Assume that $\sigma(Y)$ is already defined for all $Y$ such that $X >_S Y$. Then we define $\sigma_S(X) := \bigsqcap_{D \in S_X} \sigma_S(D)$.

We call a substitution $\sigma$ *local* if it is of this form, i.e., if there is an acyclic assignment $S$ such that $\sigma = \sigma_S$. If the unifier $\sigma$ of $\Gamma$ w.r.t. $\mathcal{O}$ is a local substitution, then we call it a *local unifier* of $\Gamma$ w.r.t. $\mathcal{O}$.

The main technical result shown in [2] is that any unifiable $\mathcal{EL}$-unification problem w.r.t. a cycle-restricted ontology has a local unifier. This yields the following brute-force unification algorithm for $\mathcal{EL}$ w.r.t. cycle-restricted ontologies: first guess an acyclic assignment $S$, and then check whether the induced local substitution $\sigma_S$ solves $\Gamma$. As shown in [2], this algorithm runs in nondeterministic polynomial time. NP-hardness follows from the fact that already unification in $\mathcal{EL}$ w.r.t. the empty ontology is NP-hard [8]. In [2] it is also shown why cycle-restrictedness is needed: there is a non-cycle-restricted $\mathcal{EL}$-ontology $\mathcal{O}$ and an $\mathcal{EL}$-unification problem $\Gamma$ such that $\Gamma$ has a unifier w.r.t. $\mathcal{O}$, but it does not have a local unifier.

## 4 A Goal-Oriented Unification Algorithm

The brute-force algorithm is not practical since it blindly guesses an acyclic assignment and only afterwards checks whether the guessed assignment induces a unifier. We now introduce a more goal-oriented unification algorithm, in which nondeterministic decisions are only made if they are triggered by "unsolved parts" of the unification problem. In addition, failure due to wrong guesses can be detected early. Any non-failing run of the algorithm produces a unifier, i.e., there is no need for checking whether the assignment computed by this run really induces a unifier. This goal-oriented algorithm generalizes the algorithm for unification in $\mathcal{EL}$ (without background ontology) introduced in [10], though the rules look quite different because in the present paper we consider unification problems that consist of subsumptions whereas in [10] we considered equivalences. We assume without loss of generality that the cycle-restricted $\mathcal{ELH}_{R^+}$-ontology $\mathcal{O}$ and the unification problem $\Gamma_0$ are flat. Given $\mathcal{O}$ and $\Gamma_0$, the sets $\mathrm{At}$, $\mathrm{At}_{\mathsf{tr}}$, and $\mathrm{At}_{\mathsf{nv}}$ are defined as above. Starting with $\Gamma_0$, the algorithm maintains a current unification problem $\Gamma$ and a current acyclic assignment $S$, which initially assigns the empty set to all variables. In addition, for each subsumption in $\Gamma$ it maintains the information on whether it is *solved* or not. Initially, all subsumptions are unsolved, except those with a variable on the right-hand side. Rules are applied only to unsolved subsumptions. A (non-failing) rule application does the following:

– it solves exactly one unsolved subsumption,
– it may extend the current assignment $S$, and
– it may introduce new flat subsumptions built from elements of $\mathrm{At}_{\mathsf{tr}}$.

Each rule application that extends $S_X$ additionally *expands* $\Gamma$ *w.r.t.* $X$ as follows: every subsumption $\mathfrak{s} \in \Gamma$ of the form $C_1 \sqcap \cdots \sqcap C_n \sqsubseteq^? X$ is *expanded* by adding the subsumption $C_1 \sqcap \cdots \sqcap C_n \sqsubseteq^? A$ to $\Gamma$ for every $A \in S_X$.

**Eager Ground Solving:**

**Condition:** This rule applies to $\mathfrak{s} = C_1 \sqcap \cdots \sqcap C_n \sqsubseteq^? D$ if it is ground.
**Action:** If $C_1 \sqcap \cdots \sqcap C_n \sqsubseteq_{\mathcal{O}} D$ does not hold, the rule application fails. Otherwise, $\mathfrak{s}$ is marked as *solved*.

**Eager Solving:**

**Condition:** This rule applies to $\mathfrak{s} = C_1 \sqcap \cdots \sqcap C_n \sqsubseteq^? D$ if either
- there is $i \in \{1, \ldots, n\}$ such that $C_i = D$ or $C_i = X \in N_v$ and $D \in S_X$, or
- $D$ is ground and $\bigsqcap \mathcal{G} \sqsubseteq_{\mathcal{O}} D$ holds, where $\mathcal{G}$ is the set of all ground atoms in $\{C_1, \ldots, C_n\} \cup \bigcup_{X \in \{C_1, \ldots, C_n\} \cap N_v} S_X$.

**Action:** Its application marks $\mathfrak{s}$ as *solved*.

**Eager Extension:**

**Condition:** This rule applies to $\mathfrak{s} = C_1 \sqcap \cdots \sqcap C_n \sqsubseteq^? D$ if there is $i \in \{1, \ldots, n\}$ with $C_i = X \in N_v$ and $\{C_1, \ldots, C_n\} \setminus \{X\} \subseteq S_X$.
**Action:** Its application adds $D$ to $S_X$. If this makes $S$ cyclic, the rule application fails. Otherwise, $\Gamma$ is expanded w.r.t. $X$ and $\mathfrak{s}$ is marked as *solved*.

**Figure 1.** The eager rules of the unification algorithm.

Subsumptions are only added if they are not already present in $\Gamma$. If a new subsumption is added to $\Gamma$, either by a rule application or by expansion of $\Gamma$, then it is initially designated unsolved, except if it has a variable on the right-hand side. Once a subsumption is in $\Gamma$, it will not be removed. Likewise, if a subsumption in $\Gamma$ is marked as solved, then it will not become unsolved later.

If a subsumption is marked as solved, this does not mean that it is already solved by the substitution induced by the current assignment. It may be the case that the task of satisfying the subsumption was deferred to solving other subsumptions which are "smaller" than the given subsumption in a well-defined sense. The task of solving a subsumption whose right-hand side is a variable is deferred to solving the subsumptions introduced by expansion.

The rules of the algorithm consist of the three *eager* rules Eager Ground Solving, Eager Solving, and Eager Extension (see Figure 1), and several *nondeterministic* rules (see Figures 2 and 3). Eager rules are applied with higher priority than nondeterministic rules. Among the eager rules, Eager Ground Solving has the highest priority, then comes Eager Solving, and then Eager Extension.

**Algorithm 4.** Let $\Gamma_0$ be a flat $\mathcal{EL}$-unification problem. We set $\Gamma := \Gamma_0$ and $S_X := \emptyset$ for all $X \in N_v$. While $\Gamma$ contains an unsolved subsumption, apply the steps (1), (2), and (3).

(1) **Eager rule application:** If some eager rules apply to an unsolved subsumption $\mathfrak{s}$ in $\Gamma$, apply one of highest priority. If the rule application fails, then return "not unifiable".
(2) **Nondeterministic rule application:** If no eager rule is applicable, let $\mathfrak{s}$ be an unsolved subsumption in $\Gamma$. If one of the nondeterministic rules applies to $\mathfrak{s}$, nondeterministically choose one of these rules and apply it. If none of these rules apply to $\mathfrak{s}$ or the rule application fails, then return "not unifiable".

**Decomposition 1:**

**Condition:** This rule applies to $\mathfrak{s} = C_1 \sqcap \cdots \sqcap C_n \sqsubseteq^? \exists s.D'$ if there is an index $i \in \{1, \ldots, n\}$ with $C_i = \exists r.C'$ and $r \trianglelefteq_{\mathcal{O}} s$.
**Action:** Its application chooses such an index $i$, adds the subsumption $C' \sqsubseteq^? D'$ to $\Gamma$, expands it w.r.t. $D'$ if $D'$ is a variable, and marks $\mathfrak{s}$ as *solved*.

**Decomposition 2:**

**Condition:** This rule applies to $\mathfrak{s} = C_1 \sqcap \cdots \sqcap C_n \sqsubseteq^? \exists s.D'$ if there is an index $i \in \{1, \ldots, n\}$ and a transitive role $t$ with $C_i = \exists r.C'$ and $r \trianglelefteq_{\mathcal{O}} t \trianglelefteq_{\mathcal{O}} s$.
**Action:** Its application chooses such an index $i$, adds the subsumption $C' \sqsubseteq^? \exists t.D'$ to $\Gamma$ and marks $\mathfrak{s}$ as *solved*.

**Extension:**

**Condition:** This rule applies to $\mathfrak{s} = C_1 \sqcap \cdots \sqcap C_n \sqsubseteq^? D$ if there is an index $i \in \{1, \ldots, n\}$ with $C_i \in N_v$.
**Action:** Its application chooses such an $i$ and adds $D$ to $S_{C_i}$. If this makes $S$ cyclic, the rule application fails. Otherwise, $\Gamma$ is expanded w.r.t. $C_i$ and $\mathfrak{s}$ is marked as *solved*.

**Figure 2.** The nondeterministic rules *Decomposition 1 and 2* and *Extension*.

(3) **Eager application of Decomposition:** If in the previous step one of the rules *Mutation 2 or 3* was applied, do the following for all subsumptions $\mathfrak{s}'$ added to $\Gamma$ by this rule application: If one of the rules *Decomposition 1 or 2* applies to $\mathfrak{s}'$, nondeterministically choose one of the applicable decomposition rules and apply it to $\mathfrak{s}'$.[5]

Once all subsumptions are solved, return the substitution $\sigma$ induced by the current assignment.

In step (2), the choice which unsolved subsumption to consider next is don't care nondeterministic. However, choosing which rule to apply to the chosen subsumption is don't know nondeterministic. Additionally, the application of nondeterministic rules requires don't know nondeterministic guessing.

The *eager rules* are mainly there for optimization purposes, i.e., to avoid nondeterministic choices if a deterministic decision can be made. For example, a ground subsumption, as considered in the *Eager Ground Solving* rule, either follows from the ontology, in which case any substitution solves it, or it does not, in which case it does not have a solution. This condition can be checked in polynomial time using the polynomial time subsumption algorithm for $\mathcal{ELH}_{R^+}$ [7]. In the case considered in the *Eager Solving* rule, the substitution induced by the current assignment obviously already solves the subsumption. The *Eager Extension* rule solves a subsumption that contains only a variable $X$ and some elements of $S_X$ on the left-hand side. The rule is motivated by the following observation: for any assignment $S'$ extending the current assignment, the induced

[5] Note that *Decomposition 1* always applies to the new subsumptions. Whether *Decomposition 2* is also applicable depends on the existence of an appropriate transitive role $t$.

**Mutation 1:**

**Condition:** This rule applies to $\mathfrak{s} = C_1 \sqcap \cdots \sqcap C_n \sqsubseteq^? D$ if $n > 1$ and there are atoms $A_1, \ldots, A_k, B$ of $\mathcal{O}$ such that $A_1 \sqcap \cdots \sqcap A_k \sqsubseteq_\mathcal{O} B$ holds.
**Action:** Its application chooses such atoms, marks $\mathfrak{s}$ as *solved*, and generates the following subsumptions:

- it chooses for each $\eta \in \{1, \ldots, k\}$ an $i \in \{1, \ldots, n\}$ and adds the subsumption $C_i \sqsubseteq^? A_\eta$ to $\Gamma$,
- it adds the subsumption $B \sqsubseteq^? D$ to $\Gamma$.

**Mutation 2:**

**Condition:** This rule applies to $\mathfrak{s} = \exists r.X \sqsubseteq^? D$ if $X$ is a variable, $D$ is ground, and there are atoms $\exists r_1.A_1, \ldots, \exists r_k.A_k$ of $\mathcal{O}$ such that $r \trianglelefteq_\mathcal{O} r_1$, …, $r \trianglelefteq_\mathcal{O} r_k$, and $\exists r_1.A_1 \sqcap \cdots \sqcap \exists r_k.A_k \sqsubseteq_\mathcal{O} D$ hold.
**Action:** Its application chooses such atoms, adds the subsumptions $\exists r.X \sqsubseteq^? \exists r_1.A_1$, …, $\exists r.X \sqsubseteq^? \exists r_k.A_k$ to $\Gamma$, and marks $\mathfrak{s}$ as *solved*.

**Mutation 3:**

**Condition:** This rule applies to $\mathfrak{s} = \exists r.X \sqsubseteq^? \exists s.Y$ if $X$ and $Y$ are variables, and there are atoms $\exists r_1.A_1, \ldots, \exists r_k.A_k, \exists u.B$ of $\mathcal{O}$ such that $r \trianglelefteq_\mathcal{O} r_1$, …, $r \trianglelefteq_\mathcal{O} r_k$, $u \trianglelefteq_\mathcal{O} s$, and $\exists r_1.A_1 \sqcap \cdots \sqcap \exists r_k.A_k \sqsubseteq_\mathcal{O} \exists u.B$ hold.
**Action:** Its application chooses such atoms, adds the subsumptions $\exists r.X \sqsubseteq^? \exists r_1.A_1$, …, $\exists r.X \sqsubseteq^? \exists r_k.A_k$, $\exists u.B \sqsubseteq^? \exists s.Y$ to $\Gamma$, and marks $\mathfrak{s}$ as *solved*.

**Mutation 4:**

**Condition:** This rule applies to $\mathfrak{s} = C \sqsubseteq^? \exists s.Y$ if $C$ is a ground atom or $\top$, $Y$ is a variable, and there is an atom $\exists u.B$ of $\mathcal{O}$ such that either

- $C \sqsubseteq_\mathcal{O} \exists u.B$ and $u \trianglelefteq_\mathcal{O} s$, or
- $C \sqsubseteq_\mathcal{O} \exists t.B$ for a transitive role $t$ with $u \trianglelefteq_\mathcal{O} t \trianglelefteq_\mathcal{O} s$.

**Action:** Its application chooses such an atom, adds the subsumption $B \sqsubseteq^? Y$ to $\Gamma$, and marks $\mathfrak{s}$ as *solved*.

**Figure 3.** The nondeterministic *Mutation* rules of the unification algorithm.

substitution $\sigma'$ satisfies $\sigma'(X) \equiv \sigma'(C_1) \sqcap \ldots \sqcap \sigma'(C_n)$. Thus, if $S'_X$ contains $D$, then $\sigma'(X) \sqsubseteq_\mathcal{O} \sigma'(D)$, and $\sigma'$ solves the subsumption. Conversely, if $\sigma'$ solves the subsumption, then $\sigma'(X) \sqsubseteq_\mathcal{O} \sigma'(D)$, and thus adding $D$ to $S'_X$ yields an equivalent induced substitution.

The *nondeterministic rules* only come into play if no eager rules can be applied. In order to solve an unsolved subsumption $\mathfrak{s} = C_1 \sqcap \cdots \sqcap C_n \sqsubseteq^? D$, we consider the two conditions of Lemma 1. Regarding the first condition, which is addressed by the rules *Decomposition 1 and 2* and *Extension*, assume that $\gamma$ is induced by an acyclic assignment $S$. To satisfy the first condition of the lemma with $\gamma$, the atom $\gamma(D)$ must structurally subsume a top-level atom in $\gamma(C_1) \sqcap \cdots \sqcap \gamma(C_n)$. This atom can either be of the form $\gamma(C_i)$ for an atom $C_i$, or it can be of the form $\gamma(C)$ for an atom $C \in S_{C_i}$ and a variable $C_i$. In the second case, the atom $C$ can either already be in $S_{C_i}$ or it can be put into $S_{C_i}$ by an application of the Extension rule. The two versions of *Decomposition* correspond to the cases (2) and (3) in the definition of structural subsumption.

The *Mutation rules* cover the second condition in Lemma 1. For example, let us analyze how *Mutation 1* ensures that all the requirements of this condition are satisfied. The rule guesses atoms $A_1, \ldots, A_k, B$ such that $A_1 \sqcap \cdots \sqcap A_k \sqsubseteq_{\mathcal{O}} B$ holds. This can be checked using the polynomial-time subsumption algorithm for $\mathcal{ELH}_{R^+}$. Whenever the second condition of Lemma 1 requires a structural subsumption $\gamma(E) \sqsubseteq^{\mathsf{s}}_{\mathcal{O}} \gamma(F)$ to hold for a (hypothetical) unifier $\gamma$ of $\Gamma$, the rule creates the new subsumption $E \sqsubseteq^{?} F$, which has to be solved later on. This way, the rule ensures that the substitution built by the algorithm actually satisfies the conditions of the lemma. The *other mutation rules* follow the same idea, but they consider cases where only a single atom occurs on the left-hand side of the subsumption to be solved. The reason for considering these cases separately is that in the proof of soundness we need the newly introduced subsumptions to be "smaller" than the subsumption that triggered their introduction. For *Mutation 1* this is the case due to the smaller left-hand side (only one atom), whereas for the other mutation rules this is not so clear. Actually, for *Mutation 2 and 3*, the new subsumptions turn out to be smaller only after *Decomposition* is applied to them. *Mutation 4* implicitly applies a form of decomposition.

Due to the space restrictions, we cannot give more details on how to prove that the algorithm is correct. Complete proofs of soundness, completeness and termination can be found in [3].

**Theorem 5.** *Algorithm 4 is an* NP*-decision procedure for testing solvability of* $\mathcal{ELH}_{R^+}$*-unification problems w.r.t. cycle-restricted ontologies.*

## 5 Conclusions

Above, we have presented a goal-oriented NP-algorithm for unification in $\mathcal{ELH}_{R^+}$ w.r.t. cycle-restricted ontologies. In [5], we have developed a reduction of this problem to SAT, which is based on a characterization of subsumption different from the one in Lemma 1. Though clearly better than the brute-force algorithm introduced in [2], both algorithms suffer from a high degree of nondeterminism due to having to guess true subsumptions between concepts built from atoms of the background cycle-restricted ontology. We must find optimizations to tackle this problem before an implementation becomes feasible.

On the theoretical side, the main topic for future research is to consider unification w.r.t. unrestricted $\mathcal{ELH}_{R^+}$-ontologies. In order to generalize the brute-force algorithm in this direction, we need to find a more general notion of locality. Starting with the goal-oriented algorithm, one idea could be not to fail when a cyclic assignment is generated, but rather to add rules that can break such cycles, similar to what is done in procedures for general $E$-unification [16].

Another idea could be to use just the rules of our goal-oriented algorithm, and not fail when a cyclic assignment $S$ is generated. Our conjecture is that then the background ontology $\mathcal{O}$ together with the cyclic TBox $\mathcal{T}_S := \{X \equiv \prod_{C \in S_X} C \mid X \in N_v\}$ induced by $S$ satisfies $C \sqsubseteq_{\mathcal{O} \cup \mathcal{T}_S} D$ for all subsumptions $C \sqsubseteq^{?} D$ in $\Gamma_0$ if an appropriate hybrid semantics [13] for the combined ontology $\mathcal{O} \cup \mathcal{T}_S$ is used.

All the results on unification in Description Logics mentioned in this paper are restricted to relatively inexpressive logics that do not support all Boolean operators. If we close $\mathcal{EL}$ under negation, then we obtain the DL $\mathcal{ALC}$, which corresponds to the modal logic K [17]. Whether unification in K is decidable is a long-standing open problem. It is only known that relatively minor extensions of K have an undecidable unification problem [18].

## References


1. Baader, F., Borgwardt, S., Mendez, J., Morawska, B.: UEL: Unification solver for $\mathcal{EL}$. In: Proc. DL'12. CEUR Workshop Proceedings, vol. 846 (2012)
2. Baader, F., Borgwardt, S., Morawska, B.: Extending unification in $\mathcal{EL}$ towards general TBoxes. In: Proc. KR'12. pp. 568–572. AAAI Press (2012), short paper.
3. Baader, F., Borgwardt, S., Morawska, B.: A goal-oriented algorithm for unification in $\mathcal{ELH}_{R^+}$ w.r.t. cycle-restricted ontologies. LTCS-Report 12-05, TU Dresden, Germany (2012), see http://lat.inf.tu-dresden.de/research/reports.html.
4. Baader, F., Borgwardt, S., Morawska, B.: A goal-oriented algorithm for unification in $\mathcal{EL}$ w.r.t. cycle-restricted TBoxes. In: Proc. DL'12. CEUR Workshop Proceedings, vol. 846 (2012)
5. Baader, F., Borgwardt, S., Morawska, B.: SAT encoding of unification in $\mathcal{ELH}_{R^+}$ w.r.t. cycle-restricted ontologies. In: Proc. IJCAR'12. LNCS, vol. 7364, pp. 30—44. Springer (2012)
6. Baader, F., Borgwardt, S., Morawska, B.: SAT encoding of unification in $\mathcal{ELH}_{R^+}$ w.r.t. cycle-restricted ontologies. LTCS-Report 12-02, TU Dresden, Germany (2012), see http://lat.inf.tu-dresden.de/research/reports.html.
7. Baader, F., Brandt, S., Lutz, C.: Pushing the $\mathcal{EL}$ envelope. In: Proc. IJCAI'05. pp. 364–369. Morgan-Kaufmann (2005)
8. Baader, F., Morawska, B.: Unification in the description logic $\mathcal{EL}$. In: Proc. RTA'09. LNCS, vol. 5595, pp. 350–364. Springer (2009)
9. Baader, F., Morawska, B.: SAT encoding of unification in $\mathcal{EL}$. In: Proc. LPAR'10. LNCS, vol. 6397, pp. 97–111. Springer (2010)
10. Baader, F., Morawska, B.: Unification in the description logic $\mathcal{EL}$. Log. Meth. Comput. Sci. 6(3) (2010)
11. Baader, F., Narendran, P.: Unification of concept terms in description logics. J. Symb. Comput. 31(3), 277–305 (2001)
12. Brandt, S.: Polynomial time reasoning in a description logic with existential restrictions, GCI axioms, and—what else? In: Proc. ECAI'04. pp. 298–302 (2004)
13. Brandt, S., Model, J.: Subsumption in $\mathcal{EL}$ w.r.t. hybrid TBoxes. In: Proc. KI'05. LNCS, vol. 3698, pp. 34–48. Springer (2005)
14. Gomes, C.P., Kautz, H., Sabharwal, A., Selman, B.: Satisfiability solvers. In: Handbook of Knowledge Representation, pp. 89–134. Elsevier (2008)
15. Hofmann, M.: Proof-theoretic approach to description-logic. In: Proc. LICS'05. pp. 229–237. IEEE Press (2005)
16. Morawska, B.: General $E$-unification with eager variable elimination and a nice cycle rule. J. Autom. Reasoning 39(1), 77–106 (2007)
17. Schild, K.: A correspondence theory for terminological logics: Preliminary report. In: Proc. IJCAI'91. pp. 466–471 (1991)
18. Wolter, F., Zakharyaschev, M.: Undecidability of the unification and admissibility problems for modal and description logics. ACM Trans. Comput. Log. 9(4) (2008)