Recent Advances in Unification for the $\mathcal{EL}$ Family

Franz Baader, Stefan Borgwardt, and Barbara Morawska

Theoretical Computer Science, TU Dresden, Germany
{baader,stefborg,morawska}@tcs.inf.tu-dresden.de

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. Several algorithms that solve unification in $\mathcal{EL}$ have previously been presented. In this paper, we summarize recent extensions of these algorithms that can deal with general concept inclusion axioms (GCIs), role hierarchies ($\mathcal{H}$), and transitive roles (R^+). For the algorithms to be complete, however, the ontology consisting of the GCIs and role axioms needs to satisfy a certain cycle restriction.

1 Introduction and Preliminaries

The description logic (DL) $\mathcal{EL}$ offers the constructors conjunction ($C \sqcap D$), existential restriction ($\exists r.C$), and the top concept ($\top$) to build concept descriptions, starting with a set of concept names $\mathsf{N_C}$ and role names $\mathsf{N_R}$. Although quite inexpressive compared to other DLs, $\mathcal{EL}$ is used to define biomedical ontologies, such as the large medical ontology SNOMED CT.[1] From the computational point of view, $\mathcal{EL}$ has the advantage over more expressive DLs that important inference problems, such as the subsumption problem, are polynomial, even in the presence of background knowledge formulated using so-called general concept inclusion axioms [12]. The $\mathcal{EL}$ family of description logics consists of several logics that extend $\mathcal{EL}$ by means of expressiveness that are useful for defining medical ontologies, but which do not increase the complexity of reasoning [7].

In all logics of the $\mathcal{EL}$ family, concept descriptions are interpreted by *interpretations* $\mathcal{I}$ as subsets of a *domain* $\Delta^\mathcal{I}$. Each concept name A is assigned a set $A^\mathcal{I} \subseteq \Delta^\mathcal{I}$ and each role name r a binary relation $r^\mathcal{I} \subseteq \Delta^\mathcal{I} \times \Delta^\mathcal{I}$. Complex concept descriptions are then interpreted as follows: $\top^\mathcal{I} = \Delta^\mathcal{I}$, $(C \sqcap D)^\mathcal{I} = C^\mathcal{I} \cap D^\mathcal{I}$, and $(\exists r.C)^\mathcal{I} = \{x \in \Delta^\mathcal{I} \mid \exists y \in \Delta^\mathcal{I} : (x,y) \in r^\mathcal{I} \wedge y \in C^\mathcal{I}\}$. For example, the concept description $\mathsf{Patient} \sqcap \exists \mathsf{finding}.(\mathsf{Injury} \sqcap \exists \mathsf{location}.\mathsf{Head})$ may be used to describe the set of all patients with a head injury.

The most expressive member of the $\mathcal{EL}$ family of description logics for which unification algorithms are available is $\mathcal{ELH}_{R^+}$. The concept descriptions of $\mathcal{ELH}_{R^+}$ are built in the same way as in $\mathcal{EL}$. The logics differ in the kind of axioms that are allowed in the background ontologies. A *general concept inclusion axiom (GCI)* is of the form $C \sqsubseteq D$ for two concept descriptions C, D and is *satisfied* by an interpretation $\mathcal{I}$ if $C^\mathcal{I} \subseteq D^\mathcal{I}$. A *role inclusion axiom* is of the form $r \circ r \sqsubseteq r$ (*transitivity axiom*) or $r_1 \sqsubseteq r_2$ (*role hierarchy axiom*) and is satisfied by $\mathcal{I}$ if $r^\mathcal{I} \circ r^\mathcal{I} \subseteq r^\mathcal{I}$ or $r_1^\mathcal{I} \subseteq r_2^\mathcal{I}$, respectively. An $\mathcal{ELH}_{R^+}$*-ontology* $\mathcal{O}$ is a finite set of such axioms. Such an ontology is an $\mathcal{EL}$*-ontology* if it contains no role inclusions. An interpretation is a *model* of an ontology if it satisfies all its axioms. Ontologies are used to express background knowledge about an application domain. For example, the GCI

$$\exists \mathsf{finding}.\exists \mathsf{severity}.\mathsf{Severe} \sqsubseteq \exists \mathsf{status}.\mathsf{Emergency} \tag{1}$$

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

expresses that every severe finding constitutes an emergency situation and the role inclusion axiom $\mathsf{partOf} \circ \mathsf{partOf} \sqsubseteq \mathsf{partOf}$ says that the role partOf should be interpreted as a transitive binary relation.

In the following, we consider an arbitrary $\mathcal{ELH}_{R^+}$-ontology $\mathcal{O}$. A concept description C is *subsumed* by a concept description D w.r.t. $\mathcal{O}$ (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. $\mathcal{O}$ (written $C \equiv_{\mathcal{O}} D$) 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 $C \equiv_{\mathcal{O}} D$.

Unification

Unification in DLs has been proposed as a tool to detect redundancies in ontologies [11]. For example, assume that the following two concept descriptions were introduced independently into an ontology:

$$\exists \mathsf{finding}.(\mathsf{Head\_injury} \sqcap \exists \mathsf{severity}.\mathsf{Severe}) \tag{2}$$

$$\exists \mathsf{finding}.(\mathsf{Severe\_injury} \sqcap \exists \mathsf{finding\_site}.\mathsf{Head}) \tag{3}$$

The above descriptions are not formally equivalent, nevertheless they are meant to represent the same concept. They can be unified (i.e., made equivalent) by viewing $\mathsf{Head\_injury}$ and $\mathsf{Severe\_injury}$ as variables and substituting them respectively with $\mathsf{Injury} \sqcap \exists \mathsf{finding\_site}.\mathsf{Head}$ and $\mathsf{Injury} \sqcap \exists \mathsf{severity}.\mathsf{Severe}$.

Background knowledge can facilitate unification of concept descriptions. For example, assume that, instead of (3), the concept description

$$\exists \mathsf{finding}.(\mathsf{Severe\_injury} \sqcap \exists \mathsf{finding\_site}.\mathsf{Head}) \sqcap \exists \mathsf{status}.\mathsf{Emergency} \tag{4}$$

occurs in the ontology. The descriptions (2) and (4) are not unifiable. They can, however, be unified (with the same substitution as before) if the GCI (1) is in the background ontology.

To define unification more formally, we assume that the set $\mathsf{N_C}$ is partitioned into *concept variables* ($\mathsf{N_{var}}$) and *concept constants* ($\mathsf{N_{con}}$). A *substitution* σ maps every variable to a concept description and can be extended to concept descriptions in the usual way. A concept description is *ground* if it contains no variables and a substitution is *ground* if all concept descriptions in its range are ground. Similarly, an ontology is *ground* if it contains no variables. In the following, we assume that $\mathcal{O}$ is ground.

A *unification problem* w.r.t. $\mathcal{O}$ is a finite set $\Gamma = \{C_1 \equiv^? D_1, \dots, C_n \equiv^? D_n\}$ of equations between concept descriptions. A substitution σ is a *unifier* of Γ w.r.t. $\mathcal{O}$ if σ *solves* all the equations in Γ w.r.t. $\mathcal{O}$, i.e. if $\sigma(C_1) \equiv_{\mathcal{O}} \sigma(D_1), \dots, \sigma(C_n) \equiv_{\mathcal{O}} \sigma(D_n)$. We say that Γ is *unifiable* w.r.t. $\mathcal{O}$ if it has a unifier w.r.t. $\mathcal{O}$. We call Γ w.r.t. $\mathcal{O}$ an $\mathcal{EL}$- or $\mathcal{ELH}_{R^+}$-unification problem depending on whether $\mathcal{O}$ contains role inclusions.

Connection to E-Unification

We can equivalently express unification w.r.t. $\mathcal{ELH}_{R^+}$-ontologies as unification in the equational theory *SLmO* of semilattices with monotone operators, using additional identities to express GCIs and role inclusions [6, 15]. This unification-theoretic point of view sheds some light on our decision to restrict unification to the case of *ground* ontologies. In fact, if we lifted this restriction, then we would end up with an extension of rigid E-unification [14, 13] by a background theory. To the best of our knowledge, such variants of rigid E-unification have not been considered in the literature, and are probably quite hard to solve.

Cycle-Restricted Ontologies

Unfortunately, our unification algorithms are not complete for general ontologies. We call $\mathcal{O}$ *cycle-restricted* if $C \not\sqsubseteq_{\mathcal{O}} \exists w.C$ for every concept description C and every $w \in \mathsf{N}_\mathsf{R}^+$, where $\exists r_1 \dots r_n$ abbreviates $\exists r_1 . \dots \exists r_n$. We can show that this condition needs to be checked only for the cases where C is a concept name or $\top$. This allows us to decide in polynomial time whether an $\mathcal{ELH}_{R^+}$-ontology is cycle-restricted [6].

The main reason why we need cycle-restrictedness of $\mathcal{O}$ is that it ensures that a substitution always induces a strict partial order on the variables: For a substitution γ and $X, Y \in \mathsf{N}_\mathsf{var}$, we define

$$X >_\gamma Y \quad \text{iff} \quad \gamma(X) \sqsubseteq_{\mathcal{O}} \exists w.\gamma(Y) \text{ for some } w \in \mathsf{N}_\mathsf{R}^+. \tag{5}$$

If $\mathcal{O}$ is cycle-restricted, this defines a strict partial order. This fact turns out to be an important prerequisite for the proofs of completeness of our algorithms.

2 Unification Algorithms

The basis of all $\mathcal{ELH}_{R^+}$-unification algorithms are lemmata that give recursive characterizations of the relation $\sqsubseteq_{\mathcal{O}}$. We have developed two approaches for proving these characterizations: one based on term rewriting [6] and another one based on a sequent calculus for subsumption [3, 5]. Previous algorithms for $\mathcal{EL}$-unification w.r.t. the empty ontology were based on an even simpler characterization of subsumption that only had to take into account the structure of the compared concept descriptions [8, 9, 10]. Each of the following algorithms is based on one of those earlier algorithms and generalizes it using one of the characterizations from [5] and [6].

Before we can describe the algorithms, we need some additional definitions. A *flat atom* is either a concept name or an existential restriction $\exists r.A$, where A is a concept name. We call a concept description *flat* if it is a conjunction of flat atoms. A unification problem or ontology is *flat* if it contains only flat concept descriptions. For every unification problem Γ and ontology $\mathcal{O}$ one can construct a flat unification problem Γ' and a flat ontology $\mathcal{O}'$ in polynomial time such that Γ is unifiable w.r.t. $\mathcal{O}$ iff Γ' is unifiable w.r.t. $\mathcal{O}'$ [6]. Furthermore, it is easy to obtain all unifiers of Γ w.r.t. $\mathcal{O}$ from the unifiers of Γ' w.r.t. $\mathcal{O}'$. Thus, in the following we restrict the unification problem Γ and the ontology $\mathcal{O}$ to be flat.

The Brute-Force Algorithm

The main result underlying all the following $\mathcal{ELH}_{R^+}$-unification algorithms is that $\mathcal{ELH}_{R^+}$-unification is *local*, i.e. every solvable unification problem has a so-called *local unifier*. Let Γ be a flat unification problem and $\mathcal{O}$ be a flat, cycle-restricted $\mathcal{ELH}_{R^+}$-ontology. We will consider the set At_tr, which consists of all flat atoms occurring as subdescriptions in subsumptions in Γ or axioms in $\mathcal{O}$ and some additional flat atoms (see [6] for details). Furthermore, we define the set of *non-variable atoms* by $\mathsf{At}_\mathsf{nv} := \mathsf{At}_\mathsf{tr} \setminus \mathsf{N}_\mathsf{var}$. We call a function S that associates every variable $X \in \mathsf{N}_\mathsf{var}$ with a set $S_X \subseteq \mathsf{At}_\mathsf{nv}$ an *assignment*. For such an assignment S, we define $>_S$ as the transitive closure of $\{(X,Y) \in \mathsf{N}_\mathsf{var} \times \mathsf{N}_\mathsf{var} \mid Y \text{ occurs in an atom of } S_X\}$. We call the assignment S *acyclic* if $>_S$ is irreflexive (and thus a strict partial order). Any acyclic assignment S induces a unique substitution σ_S, which can be defined by induction along $>_S$:

- If X is a minimal element of N_var w.r.t. $>_S$, then we set $\sigma_S(X) := \bigsqcap_{D \in S_X} D$.

- 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 σ *local* if it is of this form, i.e., if there is an acyclic assignment S such that $\sigma = \sigma_S$.

In [3], we have shown that any unifiable $\mathcal{EL}$-unification problem has a local unifier. This also holds for $\mathcal{ELH}_{R^+}$-unification problems [5, 6]. Thus, one can test solvability of $\mathcal{ELH}_{R^+}$-unification problems in nondeterministic polynomial time by guessing an acyclic assignment S and then checking whether the induced substitution σ_S is a unifier, using the polynomial time algorithm for subsumption in $\mathcal{ELH}_{R^+}$ [7]. This is a direct extension of the guess-and-test algorithm for $\mathcal{EL}$ without background ontology from [8]. The following two algorithms try to generate acyclic assignments in a more goal-oriented way instead of blindly guessing arbitrary acyclic assignments.

The Rule-Based Algorithm

In [4] and [5], we extended the rule-based algorithm from [10] to deal with $\mathcal{EL}$- and $\mathcal{ELH}_{R^+}$-ontologies, respectively. The main idea underlying these algorithms is to guide the construction of an acyclic assignment by the equivalences of the unification problem Γ. The algorithm works by exhaustive application of certain rules to Γ. These rules can mark certain parts of Γ as solved, create new equivalences to be solved, and extend the current assignment, which is initially empty. Once Γ is completely solved, the current assignment yields a unifier of the original problem.

We show on a simple example how these rules work. Given the equivalence $\exists r.X \equiv^? \exists r.A$, where $X \in \mathsf{N}_{\mathsf{var}}$ and $A \in \mathsf{N}_{\mathsf{con}}$, we can employ a rule to create the new equivalence $X \equiv^? A$ and mark the old one as solved. Another rule can subsequently solve this smaller equivalence by adding A to S_X. This yields the substitution $X \mapsto A$, which solves the original identity. In contrast, the brute-force algorithm from above would have to guess any local assignment, yielding e.g. the substitution $X \mapsto A \sqcap \exists r.A$, only to realize later that this is not a unifier.

The length of every sequence of rule applications is bounded polynomially in the size of Γ. However, at each point, the algorithm has a nondeterministic choice which rule to apply. To restrict the amount of nondeterminism, several *eager* rules were introduced that are always applied first and leave no choice in their application. All of the rules are triggered by "unsolved parts" of the unification problem, and thus the constructed assignment contains only necessary non-variable atoms. In [4, 5], we added several *Mutation rules* to the original rules from [10] to take into account the axioms of an $\mathcal{ELH}_{R^+}$-ontology.

The Reduction to SAT

In this last approach, we reduce the unification problem to the propositional satisfiability problem [6], which has the advantage that we can employ highly optimized SAT solvers to solve unification problems. Basically, a satisfying propositional valuation yields a local unifier.

The propositional variables are of the form $[C \sqsubseteq D]$ for all atoms C, D of a unification problem Γ. Their intended meaning is as follows: if $[C \sqsubseteq D]$ is true, then the local substitution σ induced by the valuation satisfies $\sigma(C) \sqsubseteq \sigma(D)$. Using these propositional variables, a set of propositional clauses is constructed that (i) encodes Γ, (ii) expresses some relevant properties of subsumption in $\mathcal{ELH}_{R^+}$, and (iii) ensures that the generated assignment is acyclic. A satisfying propositional valuation of these clauses yields an acyclic assignment S, and thus a local substitution, in the following way: S_X contains all non-variable atoms D for which $[X \sqsubseteq D]$

is true. To account for $\mathcal{ELH}_{R^+}$-ontologies, the original reduction from [9] was modified in [6] using the mentioned characterization of subsumption. More precisely, the clauses encoding the properties of subsumption were extended to allow to take GCIs and role inclusions into account.

Consider $\mathcal{O} = \emptyset$ and the example $\exists r.X \equiv^? \exists r.A$ from before. This equivalence is encoded in the clauses $\rightarrow [\exists r.X \sqsubseteq \exists r.A]$ and $\rightarrow [\exists r.A \sqsubseteq \exists r.X]$. The clause $[\exists r.X \sqsubseteq \exists r.A] \rightarrow [X \sqsubseteq A]$ expresses that the subsumption $\exists r.X \sqsubseteq \exists r.A$ can only be solved by *decomposition*, i.e. stripping away the common top-level existential restriction. There are several clauses that prevent $[X \sqsubseteq \exists r.A]$ and $[X \sqsubseteq \exists r.X]$ from being true. Thus, this approach also yields only one unifier, namely $X \mapsto A$.

3 Minimal Unifiers

The brute-force algorithm and the SAT reduction yield *all* local unifiers in the sense that the successful runs of these nondeterministic procedures generate exactly the acyclic assignments that induce unifiers of the unification problem. In contrast, the rule-based algorithm only generates a subset of the local unifiers. However, it is still complete since it generates all *minimal* unifiers. To be more precise, we call a unifier *minimal* if it is minimal w.r.t. the order $\succeq$, where $\sigma \succeq \theta$ iff $\sigma(X) \sqsubseteq_{\mathcal{O}} \theta(X)$ holds for all $X \in \mathsf{N}_{\mathsf{var}}$. In fact, locality of unification w.r.t. $\mathcal{O} = \emptyset$ was first shown in [8] by showing that every solvable unification problem has a minimal unifier and that every minimal unifier is local.

Generating exactly the minimal unifiers would be advantageous since there are considerably fewer minimal unifiers than local ones, and they are usually of smaller size. However, the rule-based algorithm also generates unifiers that are not minimal. If we assume a slight generalization of $\succeq$ to $\succeq_{\mathcal{X}}$, where $\succeq_{\mathcal{X}}$ compares the unifiers only w.r.t. a subset $\mathcal{X} \subseteq \mathsf{N}_{\mathsf{var}}$, we are able to show [2] that there cannot be an NP-procedure that generates exactly the $\succeq_{\mathcal{X}}$-minimal unifiers in the sense that the successful runs of the procedure on a unification problem Γ generate exactly the acyclic assignments that yield $\succeq_{\mathcal{X}}$-minimal unifiers of Γ. It is still open whether this result also holds for the case of $\succeq = \succeq_{\mathsf{N}_{\mathsf{var}}}$.

4 Future Work

The main objective for future work is to find a unification algorithm w.r.t. arbitrary, not necessarily cycle-restricted, $\mathcal{ELH}_{R^+}$-ontologies. We have implemented the rule-based algorithm and the SAT reduction in our system UEL [1] for the case of acyclic terminologies. We have also modified the SAT reduction into a MaxSAT problem that yields only the minimal unifiers of a unification problem [1]. We plan on further optimizing our implementations and extending them to deal with cycle-restricted ontologies. The main difficulty is that the presence of $\mathcal{ELH}_{R^+}$-ontologies other than acyclic terminologies increases the nondeterminism of our decision procedures considerably.

References

[1] Franz Baader, Stefan Borgwardt, Julian Alfredo Mendez, and Barbara Morawska. UEL: Unification solver for $\mathcal{EL}$. In Yevgeny Kazakov, Domenico Lembo, and Frank Wolter, editors, *Proc. of the 25th Int. Workshop on Description Logics (DL'12)*, volume 846 of *CEUR Workshop Proceedings*, pages 26–36, 2012.

[2] Franz Baader, Stefan Borgwardt, and Barbara Morawska. Computing minimal $\mathcal{EL}$-unifiers is hard. In Thomas Bolander, Torben Brauner, Silvio Ghilardi, and Lawrence Moss, editors, *Advances in Modal Logic 9 (AiML'12)*, pages 18–35. College Publications, 2012.

[3] Franz Baader, Stefan Borgwardt, and Barbara Morawska. Extending unification in $\mathcal{EL}$ towards general TBoxes. In Gerhard Brewka, Thomas Eiter, and Sheila A. McIlraith, editors, *Proc. of the 13th Int. Conf. on Principles of Knowledge Representation and Reasoning (KR'12)*, pages 568–572. AAAI Press, 2012. Short paper.

[4] Franz Baader, Stefan Borgwardt, and Barbara Morawska. A goal-oriented algorithm for unification in $\mathcal{EL}$ w.r.t. cycle-restricted TBoxes. In Yevgeny Kazakov, Domenico Lembo, and Frank Wolter, editors, *Proc. of the 25th Int. Workshop on Description Logics (DL'12)*, volume 846 of *CEUR Workshop Proceedings*, pages 37–47, 2012.

[5] Franz Baader, Stefan Borgwardt, and Barbara Morawska. A goal-oriented algorithm for unification in $\mathcal{ELH}_R^+$ w.r.t. cycle-restricted ontologies. In Michael Thielscher and Dongmo Zhang, editors, *Proc. of the 25th Australasian Joint Conf. on Artificial Intelligence (AI'12)*, volume 7691 of *Lecture Notes in Artificial Intelligence*, pages 493–504. Springer-Verlag, 2012.

[6] Franz Baader, Stefan Borgwardt, and Barbara Morawska. SAT encoding of unification in $\mathcal{ELH}_{R^+}$ w.r.t. cycle-restricted ontologies. In Bernhard Gramlich, Dale Miller, and Uli Sattler, editors, *Proc. of the 6th Int. Joint Conf. on Automated Reasoning (IJCAR'12)*, volume 7364 of *Lecture Notes in Artificial Intelligence*, pages 30–44. Springer-Verlag, 2012.

[7] Franz Baader, Sebastian Brandt, and Carsten Lutz. Pushing the $\mathcal{EL}$ envelope. In Leslie Pack Kaelbling and Alessandro Saffiotti, editors, *Proc. of the 19th Int. Joint Conf. on Artificial Intelligence (IJCAI'05)*, pages 364–369. Professional Book Center, 2005.

[8] Franz Baader and Barbara Morawska. Unification in the description logic $\mathcal{EL}$. In Ralf Treinen, editor, *Proc. of the 20th Int. Conf. on Rewriting Techniques and Applications (RTA'09)*, volume 5595 of *Lecture Notes in Computer Science*, pages 350–364. Springer-Verlag, 2009.

[9] Franz Baader and Barbara Morawska. SAT encoding of unification in $\mathcal{EL}$. In Christian G. Fermüller and Andrei Voronkov, editors, *Proc. of the 17th Int. Conf. on Logic for Programming, Artificial Intelligence, and Reasoning (LPAR'10)*, volume 6397 of *Lecture Notes in Computer Science*, pages 97–111. Springer-Verlag, 2010.

[10] Franz Baader and Barbara Morawska. Unification in the description logic $\mathcal{EL}$. *Logical Methods in Computer Science*, 6(3), 2010.

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

[12] Sebastian Brandt. Polynomial time reasoning in a description logic with existential restrictions, GCI axioms, and - what else? In Ramon López de Mántaras and Lorenza Saitta, editors, *Proc. of the 16th Eur. Conf. on Artificial Intelligence (ECAI'04)*, pages 298–302. IOS Press, 2004.

[13] Anatoli Degtyarev and Andrei Voronkov. The undecidability of simultaneous rigid E-unification. *Theoretical Computer Science*, 166(1–2):291–300, 1996.

[14] Jean Gallier, Paliath Narendran, David Plaisted, and Wayne Snyder. Rigid E-unification: NP-completeness and applications to equational matings. *Information and Computation*, 87(1–2):129–195, 1990.

[15] Viorica Sofronie-Stokkermansmans. Locality and subsumption testing in $\mathcal{EL}$ and some of its extensions. In Carlos Areces and Robert Goldblatt, editors, *Advances in Modal Logic 7 (AiML'08)*, pages 315–339. College Publications, 2008.