TECHNISCHE UNIVERSITÄT DRESDEN

Technische Universität Dresden
Institute for Theoretical Computer Science
Chair for Automata Theory

LTCS–Report

Unification in the Description Logic $\mathcal{EL}$ w.r.t. Cycle-Restricted TBoxes

Franz Baader Stefan Borgwardt Barbara Morawska

LTCS-Report 11-05

Postal Address:
Lehrstuhl für Automatentheorie
Institut für Theoretische Informatik
TU Dresden
01062 Dresden

http://lat.inf.tu-dresden.de

Visiting Address:
Nöthnitzer Str. 46
Dresden

Contents

Abstract

Unification in Description Logics (DLs) has been proposed as an inference service that can, for example, be used to detect redundancies in ontologies. The inexpressive Description Logic $\mathcal{EL}$ is of particular interest in this context since, on the one hand, several large biomedical ontologies are defined using $\mathcal{EL}$. On the other hand, unification in $\mathcal{EL}$ has recently been shown to be NP-complete, and thus of significantly lower complexity than unification in other DLs of similarly restricted expressive power. However, the unification algorithms for $\mathcal{EL}$ developed so far cannot deal with general concept inclusion axioms (GCIs). This paper makes a considerable step towards addressing this problem, but the GCIs our new unification algorithm can deal with still need 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 GCIs [11, 4]. 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]

Unification in DLs has been proposed in [8] 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 head injury* as

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

whereas another one represents it as

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

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 Head_injury and Severe_injury as variables, and substituting the first one by $\mathsf{Injury} \sqcap \exists \mathsf{finding\_site}.\mathsf{Head}$ and the second one by $\mathsf{Injury} \sqcap \exists \mathsf{severity}.\mathsf{Severe}$. In this case, we say that the descriptions are unifiable, and call the substitution that makes them equivalent a *unifier*. Intuitively, such a unifier proposes definitions for the concept names that are used as variables: in our example, we know that, if we define Head_injury as $\mathsf{Injury} \sqcap \exists \mathsf{finding\_site}.\mathsf{Head}$ and Severe_injury as $\mathsf{Injury} \sqcap \exists \mathsf{severity}.\mathsf{Severe}$, then the two concept descriptions

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

(1) and (2) are equivalent w.r.t. these definitions. Here equivalence holds without any additional definitions or GCIs.

To motivate our interest in unification w.r.t. GCIs, assume that the second developer uses the description

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

instead of (2). The descriptions (1) and (3) are not unifiable without additional GCIs, but they are unifiable, with the same unifier as above, if the GCI

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

is present in a background ontology.

All previous results on unification in DLs did not consider background GCIs. In [8] 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 [5], we were able to show that unification in $\mathcal{EL}$ is of considerably lower complexity: the decision problem is "only" NP-complete. The original unification algorithm for $\mathcal{EL}$ introduced in [5] was a brutal "guess and then test" NP-algorithm, but we have since then also developed more practical algorithms. On the one hand, in [7] we describe 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. On the other hand, in [6], we present an algorithm that is based on a reduction to satisfiability in propositional logic (SAT). In [7] it was also shown that the approaches for unification of $\mathcal{EL}$-concept descriptions (without any background ontology) can easily be extended to the case of an acyclic TBox as background ontology without really changing the algorithms or increasing their complexity. Basically, by viewing defined concepts as variables, an acyclic TBox can be turned into a unification problem that has as its unique unifier the substitution that replaces the defined concepts by unfolded versions of their definitions.

For GCIs, this simple trick is not possible, and thus handling them requires the development of new algorithms. In this report, we describe two such new algorithms: one that extends the brute-force "guess and then test" NP-algorithm from [5] and a more practical one that extends the goal-oriented algorithm from [7]. Both algorithms are based on a new characterization of subsumption w.r.t. GCIs in $\mathcal{EL}$. Unfortunately, these algorithms are 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 $\exists \mathsf{child.Human} \sqsubseteq \mathsf{Human}$ satisfies this restriction, whereas the cyclic GCI $\mathsf{Human} \sqsubseteq \exists \mathsf{parent.Human}$ does not.

2 The Description Logic $\mathcal{EL}$

We first define the basic syntax and semantics of the description logic $\mathcal{EL}$ and then proceed to more advanced notions.

Let N_C be a set of *concept names* and N_R a set of *role names*. *($\mathcal{EL}$)-concept descriptions* are built from concept names by the constructors *conjunction* ($C \sqcap D$), *existential restriction* ($\exists r.C$ for a role name r), and *top* ($\top$). We say that a concept description C is *built over a signature* $\Sigma \subseteq N_C \cup N_R$ if only concept and role names from Σ occur in it.

An *interpretation* $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ consists of a non-empty domain $\Delta^{\mathcal{I}}$ and an interpretation function that maps concept names to subsets of $\Delta^{\mathcal{I}}$ and role names to binary relations over $\Delta^{\mathcal{I}}$. This function is extended to concept descriptions as follows: $(C \sqcap D)^{\mathcal{I}} := C^{\mathcal{I}} \cap D^{\mathcal{I}}$, $(\exists r.C)^{\mathcal{I}} := \{x \in \Delta^{\mathcal{I}} \mid \exists y \in \Delta^{\mathcal{I}} : r^{\mathcal{I}}(x, y)\}$, $\top^{\mathcal{I}} := \Delta^{\mathcal{I}}$.

The *role depth* $\mathsf{rd}(C)$ of a concept description C is inductively defined as follows: $\mathsf{rd}(A) = \mathsf{rd}(\top) = 0$, $\mathsf{rd}(C \sqcap D) = \max\{\mathsf{rd}(C), \mathsf{rd}(D)\}$, $\mathsf{rd}(\exists r.C) = 1 + \mathsf{rd}(C)$.

2.1 Terminological Axioms

A *concept definition* is of the form $A \equiv C$ for a concept name A and a concept description C. An interpretation $\mathcal{I}$ *satisfies* this concept definition if $A^{\mathcal{I}} = C^{\mathcal{I}}$. A *general concept inclusion (GCI)* is of the form $C \sqsubseteq D$ for concept descriptions C and D and is satisfied by $\mathcal{I}$ if $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$. An *axiom* is a concept definition or a general concept inclusion and a *TBox* is a finite set of axioms.

A *cyclic TBox* contains only concept definitions and may contain at most one concept definition for each concept name. An *acyclic TBox* is a cyclic TBox without cyclic dependencies between concept names.[2] A *general TBox* contains only GCIs. An interpretation is a *model* of a TBox if it satisfies all its axioms.

A concept description C is *subsumed* by a concept description D w.r.t. a TBox $\mathcal{T}$ ($C \sqsubseteq_{\mathcal{T}} D$) if every model of $\mathcal{T}$ satisfies the GCI $C \sqsubseteq D$. We say that C is *equivalent* to D w.r.t. $\mathcal{T}$ ($C \equiv_{\mathcal{T}} D$) if $C \sqsubseteq_{\mathcal{T}} D$ and $D \sqsubseteq_{\mathcal{T}} C$. For the empty TBox, we write $C \sqsubseteq D$ and $C \equiv D$ instead of $C \sqsubseteq_{\emptyset} D$ and $C \equiv_{\emptyset} D$.

Since conjunction is interpreted as intersection, the concept descriptions $(C \sqcap D) \sqcap E$ and $C \sqcap (D \sqcap E)$ are 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^*.

An *atom* is a concept name or an existential restriction. Thus, every concept de-

[2] *A depends on B* if B occurs in the definition of A.

scription C is a conjunction of atoms or $\top$. We call the atoms in this conjunction the *top-level atoms* of C. 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.

Given a concept description C and an acyclic TBox $\mathcal{T}$, the description C can be *expanded* w.r.t. $\mathcal{T}$ by replacing defined concepts by their definitions until no more defined concepts occur. This yields a concept description $C^{\mathcal{T}}$ that is equivalent to C w.r.t. $\mathcal{T}$ and does not contain defined concepts. Expansion can be used to reduce subsumption w.r.t. an acyclic TBox to subsumption w.r.t. the empty TBox, but the expanded description can be exponential in the size of C and $\mathcal{T}$.

2.2 Inseparability

The following definition is useful to compare the expressiveness of different classes of TBoxes, i.e., whether certain kind of TBox can express all restrictions on interpretations expressible in another class.

Definition 1. Let $\Sigma \subseteq N_C \cup N_R$ be a signature. Two TBoxes $\mathcal{T}_1$, $\mathcal{T}_2$ are Σ-*inseparable* if for all concept descriptions C, D built over the signature Σ we have $C \sqsubseteq_{\mathcal{T}_1} D$ iff $C \sqsubseteq_{\mathcal{T}_2} D$.

For a TBox $\mathcal{T}$, let $sig(\mathcal{T}) \subseteq N_C \cup N_R$ denote the set of concept and role names occurring in $\mathcal{T}$.

A class $\mathfrak{T}_2$ of TBoxes is *at least as expressive as* another class $\mathfrak{T}_1$ of TBoxes if for every $\mathcal{T}_1 \in \mathfrak{T}_1$ there is a $\mathcal{T}_2 \in \mathfrak{T}_2$ such that $\mathcal{T}_1$ and $\mathcal{T}_2$ are $sig(\mathcal{T}_1)$-inseparable. $\mathfrak{T}_1$ and $\mathfrak{T}_2$ are *equally expressive* if $\mathfrak{T}_1$ is at least as expressive as $\mathfrak{T}_2$ and $\mathfrak{T}_2$ is at least as expressive as $\mathfrak{T}_1$.

Intuitively, two TBoxes are inseparable if they give the same answers to questions of the form "Does $C \sqsubseteq_{\mathcal{T}} D$ hold?". In this case, a user can use them interchangeably when reasoning about a domain. This notion was introduced in [21] to detect whether changes to a TBox change its behavior w.r.t. subsumption reasoning. Such changes include, e.g., importing of other TBoxes or adding new axioms. Inseparability generalizes the notion of conservative extensions, where one TBox is included in the other [1].

The expressiveness of two classes of TBoxes can be compared using the notion of inseparability. A class $\mathfrak{T}_2$ is at least as expressive as $\mathfrak{T}_1$ if every TBox in $\mathfrak{T}_1$ can be replaced by a TBox of $\mathfrak{T}_2$ without changing any consequences. In the process, the introduction of auxiliary concept names is allowed, i.e., we consider inseparability only w.r.t. the original signature.

We now consider the classes of TBoxes introduced earlier. Every acyclic TBox is obviously a cyclic one, and for every cyclic TBox we obtain an inseparable general TBox by rewriting every concept definition $A \equiv C$ into the GCIs $A \sqsubseteq C$

and $C \sqsubseteq A$. However, these relations do not hold in the other direction, as we will demonstrate in Section 4.

2.3 Flat General TBoxes

To simplify a given general TBox, we will often transform it into a normal form: A general TBox $\mathcal{T}$ is called *flat* if it contains only axioms of the form $A \sqcap B \sqsubseteq C$, where A, B are flat atoms or $\top$ and C is a flat atom.

To flatten $\mathcal{T}$, we employ the procedure described in [12]. This procedure uses rules to transform all axioms of $\mathcal{T}$ into one of the forms $A \sqsubseteq B$, $A_1 \sqcap A_2 \sqsubseteq B$, $A \sqsubseteq \exists r.B$, or $\exists r.A \sqsubseteq B$, where A, A_1, A_2, B are concept names or $\top$. All of these axioms are of the desired form.[3]

The transformation rules employed by this procedure are the following:

- $\widehat{C} \sqcap D \ \rho \ E \longrightarrow \{A \equiv \widehat{C}, A \sqcap D \ \rho \ E\}$
- $C \ \rho \ D \sqcap \widehat{E} \longrightarrow \{C \ \rho \ D \sqcap A, A \equiv \widehat{E}\}$
- $\exists r.\widehat{C} \ \rho \ D \longrightarrow \{A \equiv \widehat{C}, \exists r.A \ \rho \ D\}$
- $C \ \rho \ \exists r.\widehat{D} \longrightarrow \{C \ \rho \ \exists r.A, A \equiv \widehat{D}\}$

In these rules, C, D, E stand for arbitrary concept descriptions, $\widehat{C}$, $\widehat{D}$, $\widehat{E}$ are concept descriptions that are not concept names, $r \in N_R$, and $\rho \in \{\sqsubseteq, \equiv\}$. The concept name A is always a new concept name not occurring in $\mathcal{T}$. Applying a rule $G \longrightarrow \mathcal{S}$ to a TBox $\mathcal{T}$ changes it to $(\mathcal{T} \setminus \{G\}) \cup \mathcal{S}$.

After exhaustively applying these four rules, the TBox consists of flat GCIs of the required form and additional flat concept definitions. The fact that for each definition a new concept name is used ensures that these definitions form an acyclic TBox. In particular, for each newly introduced concept name A we can find a unique concept description C_A occurring in the original TBox such that $A \equiv C_A$ holds in the new TBox. It remains to transform these definitions into GCIs: A definition $A \equiv A_1 \sqcap A_2$ is replaced by $A \sqsubseteq A_1$, $A \sqsubseteq A_2$, and $A_1 \sqcap A_2 \sqsubseteq A$, while any definition of the form $A \equiv \exists r.A'$ is replaced by $A \sqsubseteq \exists r.A'$ and $\exists r.A' \sqsubseteq A$.

The resulting TBox $\mathcal{T}'$ proves the same subsumptions between concepts built over $sig(\mathcal{T})$ as $\mathcal{T}$, i.e., it is $sig(\mathcal{T})$-inseparable from $\mathcal{T}$.

[3] Axioms with $\top$ on the right-hand side are true in all interpretations and can therefore simply be removed. We can further replace $\top$ inside existential restrictions by a new concept name $A_\top$ and introduce the GCI $\top \sqsubseteq A_\top$.

3 Subsumption with General TBoxes

Subsumption w.r.t. a general TBox can be decided in polynomial time [11]. For the purposes of deciding unification, however, we do not simply want a decision procedure for subsumption, but are more interested in a characterization of subsumption that helps us to find unifiers. The following characterization of subsumption w.r.t. the empty TBox has proven useful for $\mathcal{EL}$-unification algorithms before.

Lemma 2 ([7])**.** *Let $A_1, \ldots, A_k, B_1, \ldots, B_l$ be concept names and $C = A_1 \sqcap \ldots \sqcap A_k \sqcap \exists r_1.C_1 \sqcap \ldots \sqcap \exists r_m.C_m$ and $D = B_1 \sqcap \ldots \sqcap B_l \sqcap \exists s_1.D_1 \sqcap \ldots \sqcap \exists s_n.D_n$ concept descriptions. Then $C \sqsubseteq D$ iff $\{B_1, \ldots, B_l\} \subseteq \{A_1, \ldots, A_k\}$ and for every $j \in \{1, \ldots, n\}$ there exists an $i \in \{1, \ldots, m\}$ such that $r_i = s_j$ and $C_i \sqsubseteq D_j$.*

Thus, an atom C is subsumed by an atom D (w.r.t. $\emptyset$) iff $C = D$ is a concept name or $C = \exists r.C'$ and $D = \exists r.D'$ for a role name r and $C' \sqsubseteq D'$.

Lemma 3. *Let C and D be two concept descriptions. Then $C \sqsubseteq D$ iff every top-level atom of D subsumes a top-level atom of C.*

The aim of this section is to provide a characterization of subsumption similar to that of Lemma 2 in the presence of general TBoxes. In the following, let $\mathcal{T}$ be a general TBox. First, we introduce the notion of *structural subsumption* between atoms.

Definition 4. Let C, D be atoms. C is *structurally subsumed* by D w.r.t. $\mathcal{T}$ ($C \sqsubseteq^{\mathsf{s}}_{\mathcal{T}} D$) iff either

- $C = D$ is a concept name or
- $C = \exists r.C'$, $D = \exists r.D'$, and $C' \sqsubseteq_{\mathcal{T}} D'$.

Structural subsumption of C by D is a stronger property than $C \sqsubseteq_{\mathcal{T}} D$ since it additionally requires that C and D have a compatible top-level structure. On the other hand, it is weaker than subsumption $\sqsubseteq$ w.r.t. $\emptyset$, i.e., whenever $C \sqsubseteq D$ holds for two atoms C and D, then $C \sqsubseteq^{\mathsf{s}}_{\mathcal{T}} D$, but not vice versa. Furthermore, it is only defined on atoms. As shown by Lemma 2, if $\mathcal{T} = \emptyset$, then the three relations $\sqsubseteq$, $\sqsubseteq_{\mathcal{T}}$, $\sqsubseteq^{\mathsf{s}}_{\mathcal{T}}$ coincide. Like $\sqsubseteq$ and $\sqsubseteq_{\mathcal{T}}$, $\sqsubseteq^{\mathsf{s}}_{\mathcal{T}}$ is reflexive, transitive, and closed under existential restrictions.

Proposition 5. *Let C, D, E be atoms and r a role name.*

1. *If $C \sqsubseteq D$, then $C \sqsubseteq^{\mathsf{s}}_{\mathcal{T}} D$.*
2. *If $C \sqsubseteq^{\mathsf{s}}_{\mathcal{T}} D$, then $C \sqsubseteq_{\mathcal{T}} D$.*

3. $C \sqsubseteq^{\mathsf{s}}_{\mathcal{T}} C$.

4. *If* $C \sqsubseteq^{\mathsf{s}}_{\mathcal{T}} D$ *and* $D \sqsubseteq^{\mathsf{s}}_{\mathcal{T}} E$, *then* $C \sqsubseteq^{\mathsf{s}}_{\mathcal{T}} E$.

5. *If* $C \sqsubseteq_{\mathcal{T}} D$, *then* $\exists r.C \sqsubseteq^{\mathsf{s}}_{\mathcal{T}} \exists r.D$.

Our aim is to prove the following lemma that characterizes subsumption in the presence of GCIs.

Lemma 6. *Let* $\mathcal{T}$ *be a general TBox and* $C_1, \ldots, C_n$, $D_1, \ldots, D_m$ *atoms. Then* $C_1 \sqcap \cdots \sqcap C_n \sqsubseteq_{\mathcal{T}} D_1 \sqcap \cdots \sqcap D_m$ *iff for every* $j \in \{1, \ldots, m\}$

1. *there is an index* $i \in \{1, \ldots, n\}$ *such that* $C_i \sqsubseteq^{\mathsf{s}}_{\mathcal{T}} D_j$, *or*

2. *there are atoms* $A_1, \ldots, A_k, B$ *of* $\mathcal{T}$ *(*$k \geq 0$*) such that*

   a) $A_1 \sqcap \cdots \sqcap A_k \sqsubseteq_{\mathcal{T}} B$,

   b) *for every* $\eta \in \{1, \ldots, k\}$ *there is* $i \in \{1, \ldots, n\}$ *with* $C_i \sqsubseteq^{\mathsf{s}}_{\mathcal{T}} A_\eta$, *and*

   c) $B \sqsubseteq^{\mathsf{s}}_{\mathcal{T}} D_j$.

If $\mathcal{T} = \emptyset$, this lemma reduces to Lemma 2 since the second condition can never be satisfied.

Note that this lemma does not immediately give rise to an algorithm for checking subsumption in $\mathcal{EL}$ w.r.t. $\mathcal{T}$ since it depends on being able to check subsumptions between conjunctions of atoms of $\mathcal{T}$ in the first place. Since a naive algorithm would have to guess these atoms anyway, such a procedure would not come close to the efficiency of the established subsumption check algorithms [3, 11]. The aim of this characterization is not to provide a fast way to check subsumption, but to help in the design and proof of correctness of the unification algorithm in Section 7.

The following section will provide the proof of Lemma 6.

3.1 Proving Subsumptions by Inference Rules

We will first characterize subsumption w.r.t. $\mathcal{T}$ using a Gentzen-style proof calculus. In [19] a similar calculus was presented and used for a decision procedure for subsumption in $\mathcal{EL}$ with general TBoxes. As said before, the emphasis of the following section is not to prove subsumptions, but to provide a structural characterization of subsumption. Both calculi are sound and complete for subsumption, but are useful in different ways. For now, we assume that $\mathcal{T}$ is a flat general TBox.

Definition 7. We inductively define *proof trees* using the following rules.

($\mathbf{R}_1$) *Introduction of GCIs:* For every $A_1 \sqcap A_2 \sqsubseteq B$ in $\mathcal{T}$,

$$\frac{}{A_1 \sqcap A_2 \vdash_{\mathcal{T}} B}$$

($\mathbf{R}_2$) *Introduction of* $\top$*:* For every $\mathcal{EL}$-concept description C,

$$\frac{}{C \vdash_{\mathcal{T}} \top}$$

($\mathbf{R}_3$) *Reflexive closure:* For every $\mathcal{EL}$-concept description C,

$$\frac{}{C \vdash_{\mathcal{T}} C}$$

($\mathbf{R}_4$) *Idempotency:* For all $\mathcal{EL}$-concept descriptions C, D,

$$\frac{C \sqcap C \vdash_{\mathcal{T}} D}{C \vdash_{\mathcal{T}} D}$$

($\mathbf{R}_5$) *Unit on the right:* For all $\mathcal{EL}$-concept descriptions C, D,

$$\frac{C \vdash_{\mathcal{T}} D \sqcap \top}{C \vdash_{\mathcal{T}} D}$$

($\mathbf{R}_6$) *Unit on the left:* For all $\mathcal{EL}$-concept descriptions C, D,

$$\frac{C \vdash_{\mathcal{T}} \top \sqcap D}{C \vdash_{\mathcal{T}} D}$$

($\mathbf{R}_7$) *Closure under conjunction:* For all $\mathcal{EL}$-concept descriptions C, D, E, F,

$$\frac{C \vdash_{\mathcal{T}} D \qquad E \vdash_{\mathcal{T}} F}{C \sqcap E \vdash_{\mathcal{T}} D \sqcap F}$$

($\mathbf{R}_8$) *Closure under existential restriction:* For all $\mathcal{EL}$-concept descriptions C, D and each $r \in N_R$,

$$\frac{C \vdash_{\mathcal{T}} D}{\exists r.C \vdash_{\mathcal{T}} \exists r.D}$$

($\mathbf{R}_9$) *Transitive closure:* For all $\mathcal{EL}$-concept descriptions C, D, E,

$$\frac{C \vdash_{\mathcal{T}} D \qquad D \vdash_{\mathcal{T}} E}{C \vdash_{\mathcal{T}} E}$$

In each rule, the statements above the line are called *premises* and that below is called its *conclusion*. The rules without premises ($(\mathbf{R}_1)$–$(\mathbf{R}_3)$) are proof trees for their conclusions. If we are given proof trees $\mathbf{T}_1, \ldots, \mathbf{T}_n$ for each of the premises of an instance of a rule $(\mathbf{R}_x)$

$$\frac{C_1 \vdash_{\mathcal{T}} D_1 \qquad \cdots \qquad C_n \vdash_{\mathcal{T}} C_n}{C \vdash_{\mathcal{T}} D}$$

then the following is a proof tree for $C \vdash_{\mathcal{T}} D$:

$$\frac{\mathbf{T}_1 \qquad \cdots \qquad \mathbf{T}_n}{C \vdash_{\mathcal{T}} D}\,(\mathbf{R}_x)$$

If we want to explicitly mark the premises of $(\mathbf{R}_x)$, then we will use the trees $\mathbf{T}_i$ as *lemmata* and write

$$\frac{\dfrac{}{C_1 \vdash_{\mathcal{T}} D_1}\,(\mathbf{T}_1) \qquad \cdots \qquad \dfrac{}{C_n \vdash_{\mathcal{T}} D_n}\,(\mathbf{T}_n)}{C \vdash_{\mathcal{T}} D}\,(\mathbf{R}_x)$$

In the following, we denote by $C \vdash_{\mathcal{T}} D$ the fact that there is a proof tree for $C \vdash_{\mathcal{T}} D$. The *height* $\mathrm{h}(\mathbf{T})$ of a proof tree $\mathbf{T}$ is recursively defined as follows. If $\mathbf{T}_1, \ldots, \mathbf{T}_n$ are the proof trees immediately above the root, then

$$\mathrm{h}(\mathbf{T}) := 1 + \max\{\mathrm{h}(\mathbf{T}_1), \ldots, \mathrm{h}(\mathbf{T}_n)\}.$$

If the root has no premises, then $\mathrm{h}(\mathbf{T}) := 1$.

Using proof trees, we can prove subsumption relationships between $\mathcal{EL}$-concept descriptions w.r.t. $\mathcal{T}$.

Example 8. The following is a proof tree of height 3 for $A_1 \sqcap A_2 \vdash_{\mathcal{T}} \exists r.C$, given the two GCIs $A_1 \sqcap A_2 \sqsubseteq \exists r.B$ and $B \sqsubseteq C$:

$$\frac{\dfrac{}{A_1 \sqcap A_2 \vdash_{\mathcal{T}} \exists r.B}\,(\mathbf{R}_1) \qquad \dfrac{\dfrac{}{B \vdash_{\mathcal{T}} C}\,(\mathbf{R}_1)}{\exists r.B \vdash_{\mathcal{T}} \exists r.C}\,(\mathbf{R}_8)}{A_1 \sqcap A_2 \vdash_{\mathcal{T}} \exists r.C}\,(\mathbf{R}_9)$$

As claimed before, we now show that this proof system is sound and complete for subsumption in $\mathcal{EL}$ w.r.t. $\mathcal{T}$. The proof employs the construction of a *canonical model* for $\mathcal{T}$, which is very similar to the proof of correctness of the classification[4] algorithm in [4]. The algorithm presented there also uses rules that are special instances of our proof trees. They are not as general since the classification algorithm only needs to deal with subsumptions between flat atoms.

[4]Classification is the task of deciding all subsumptions $A \sqsubseteq_{\mathcal{T}} B$ between concept names $A, B \in sig(\mathcal{T})$.

Lemma 9. *Let $\mathcal{T}$ be a flat general TBox and C, D be two $\mathcal{EL}$-concept descriptions. Then $C \vdash_{\mathcal{T}} D$ iff $C \sqsubseteq_{\mathcal{T}} D$.*

Proof. It is easy to verify that the rules $(\mathbf{R}_1)$–$(\mathbf{R}_9)$ are sound, i.e., we have $C \sqsubseteq_{\mathcal{T}} D$ whenever there is a proof tree for $C \vdash_{\mathcal{T}} D$.

If $C \vdash_{\mathcal{T}} D$ does not hold, we can show that $C^{\mathcal{I}} \not\subseteq D^{\mathcal{I}}$ holds in the following *canonical model* $\mathcal{I}$ of $\mathcal{T}$. The domain of $\mathcal{I}$ is the set $\mathfrak{C}$ of all $\mathcal{EL}$-concept descriptions built over N_C and N_R. For every concept name A, we define $A^{\mathcal{I}} := \{E \in \mathfrak{C} \mid E \vdash_{\mathcal{T}} A\}$ and for every role name r, we set $r^{\mathcal{I}} := \{(E, F) \in \mathfrak{C}^2 \mid E \vdash_{\mathcal{T}} \exists r.F\}$. We show by induction on the structure of concept descriptions that the equality $C'^{\mathcal{I}} = \{E \in \mathfrak{C} \mid E \vdash_{\mathcal{T}} C'\}$ holds for each concept description C'.

- If $C' = \top$, then $C'^{\mathcal{I}} = \mathfrak{C} = \{E \in \mathfrak{C} \mid E \vdash_{\mathcal{T}} \top\}$ since $E \vdash_{\mathcal{T}} \top$ holds for all concept descriptions E by rule $(\mathbf{R}_2)$.
- If C' is a concept name, the claim holds by definition of $\mathcal{I}$.
- Let now $C' = C_1 \sqcap C_2$ for two concept descriptions C_1 and C_2 that satisfy the claim. We thus have $C'^{\mathcal{I}} = C_1^{\mathcal{I}} \cap C_2^{\mathcal{I}} = \{E \in \mathfrak{C} \mid E \vdash_{\mathcal{T}} C_1 \text{ and } E \vdash_{\mathcal{T}} C_2\}$. If E is a concept description and $\mathbf{T}_i$ is a proof tree for $E \vdash_{\mathcal{T}} C_i$ $(i = 1, 2)$, then the following is a proof tree for $E \vdash_{\mathcal{T}} C_1 \sqcap C_2$:

$$\dfrac{\dfrac{\dfrac{}{E \vdash_{\mathcal{T}} C_1}\,(\mathbf{T}_1) \qquad \dfrac{}{E \vdash_{\mathcal{T}} C_2}\,(\mathbf{T}_2)}{E \sqcap E \vdash_{\mathcal{T}} C_1 \sqcap C_2}\,(\mathbf{R}_7)}{E \vdash_{\mathcal{T}} C_1 \sqcap C_2}\,(\mathbf{R}_4)$$

  If, on the other hand, we have a proof tree $\mathbf{T}$ for $E \vdash_{\mathcal{T}} C_1 \sqcap C_2$, then the following is a proof tree for $E \vdash_{\mathcal{T}} C_1$:

$$\dfrac{\dfrac{}{E \vdash_{\mathcal{T}} C_1 \sqcap C_2}\,(\mathbf{T}) \qquad \dfrac{\dfrac{\dfrac{}{C_1 \vdash_{\mathcal{T}} C_1}\,(\mathbf{R}_3) \qquad \dfrac{}{C_2 \vdash_{\mathcal{T}} \top}\,(\mathbf{R}_2)}{C_1 \sqcap C_2 \vdash_{\mathcal{T}} C_1 \sqcap \top}\,(\mathbf{R}_7)}{C_1 \sqcap C_2 \vdash_{\mathcal{T}} C_1}\,(\mathbf{R}_5)}{E \vdash_{\mathcal{T}} C_1}\,(\mathbf{R}_9)$$

  Similarly, we can construct a proof tree for $E \vdash_{\mathcal{T}} C_2$, using $(\mathbf{R}_6)$ instead of $(\mathbf{R}_5)$.

  Thus, $(C_1 \sqcap C_2)^{\mathcal{I}} = \{E \in \mathfrak{C} \mid E \vdash_{\mathcal{T}} C_1 \sqcap C_2\}$.
- The last remaining case is that C' is of the form $\exists r.C''$, where C'' satisfies the claim. By the definition of $r^{\mathcal{I}}$, we have $C'^{\mathcal{I}} = (\exists r.C'')^{\mathcal{I}} = \{E \in \mathfrak{C} \mid \exists F \in \mathfrak{C} : E \vdash_{\mathcal{T}} \exists r.F \text{ and } F \vdash_{\mathcal{T}} C''\}$. If $\mathbf{T}_1$ is a proof tree for $E \vdash_{\mathcal{T}} \exists r.F$ and $\mathbf{T}_2$ is a proof tree for $F \vdash_{\mathcal{T}} C''$, then the following is a proof tree for $E \vdash_{\mathcal{T}} \exists r.C''$:

$$\dfrac{\dfrac{}{E \vdash_{\mathcal{T}} \exists r.F}\,(\mathbf{T}_1) \quad \dfrac{\dfrac{}{F \vdash_{\mathcal{T}} C''}\,(\mathbf{T}_2)}{\exists r.F \vdash_{\mathcal{T}} \exists r.C''}\,(\mathbf{R}_8)}{E \vdash_{\mathcal{T}} \exists r.C''}\,(\mathbf{R}_9)$$

If, on the other hand, $E \vdash_{\mathcal{T}} \exists r.C''$ holds, we have $E \in (\exists r.C'')^{\mathcal{I}}$ since $C'' \vdash_{\mathcal{T}} C''$ by rule $(\mathbf{R}_3)$.

To show that $\mathcal{I}$ is a model of $\mathcal{T}$, consider a GCI $A_1 \sqcap A_2 \sqsubseteq B$ in $\mathcal{T}$, $E \in \mathfrak{C}$, and a proof tree $\mathbf{T}$ for $E \vdash_{\mathcal{T}} A_1 \sqcap A_2$. Then the following is a proof tree for $E \vdash_{\mathcal{T}} B$:

$$\dfrac{\dfrac{}{E \vdash_{\mathcal{T}} A_1 \sqcap A_2}\,(\mathbf{T}) \quad \dfrac{}{A_1 \sqcap A_2 \vdash_{\mathcal{T}} B}\,(\mathbf{R}_1)}{E \vdash_{\mathcal{T}} B}\,(\mathbf{R}_9)$$

Thus, $(A_1 \sqcap A_2)^{\mathcal{I}} \subseteq B^{\mathcal{I}}$, i.e., $\mathcal{I}$ is a model of $A_1 \sqcap A_2 \sqsubseteq B$.

To conclude the proof, we notice that $C \in C^{\mathcal{I}}$, since $C \vdash_{\mathcal{T}} C$ holds by rule $(\mathbf{R}_3)$. On the other hand, we assumed that $C \vdash_{\mathcal{T}} D$ does not hold, which implies $C \notin D^{\mathcal{I}}$, and thus $C^{\mathcal{I}} \not\subseteq D^{\mathcal{I}}$. □

We want to emphasize again that this characterization of subsumption does not immediately yield a decision procedure for subsumption in $\mathcal{EL}$ w.r.t. $\mathcal{T}$. The problem is the transitivity rule $(\mathbf{R}_9)$, which makes an efficient proof search infeasible. Contrary to [19], where this rule is unnecessary and the calculus yields a polynomial time decision procedure for subsumption, the aim of our approach does not lie in devising a new subsumption algorithm, but in proving Lemma 6, which is crucial for the unification algorithms and the accompanying proofs presented in Sections 6 and 7.

We can now prove the desired structural characterization of subsumption using the relation $\sqsubseteq^{\mathsf{s}}_{\mathcal{T}}$.

3.2 Proof of Lemma 6

Let $\mathcal{T}$ be a general TBox and $C_1, \ldots, C_n$, $D_1, \ldots, D_m$ be atoms. Observe that if one of the alternatives of the lemma holds for D_j, then clearly D_j subsumes the conjunction $C_1 \sqcap \cdots \sqcap C_n$ w.r.t. $\mathcal{T}$.

For the other direction, assume that $C_1 \sqcap \cdots \sqcap C_n \sqsubseteq_{\mathcal{T}} D_1 \sqcap \cdots \sqcap D_m$ holds. We first reduce the claim to the case of a flat general TBox. We flatten $\mathcal{T}$, which yields a flat TBox $\mathcal{T}'$ that is $sig(\mathcal{T})$-inseparable from $\mathcal{T}$. We additionally take care that the concept names introduced by this process did not already occur in the atoms $C_1, \ldots, C_n, D_1, \ldots, D_m$. In particular, we have that $C_1 \sqcap \cdots \sqcap C_n \sqsubseteq_{\mathcal{T}'} D_1 \sqcap \cdots \sqcap D_m$. Assuming that the claim holds for flat TBoxes, we thus have one of the following cases for each atom D_j:

- There is an index $i \in \{1, \dots, n\}$ such that either $C_i = D_j$ is a concept name or $C_i = \exists r.C'$, $D_j = \exists r.D'$, and $C' \sqsubseteq_{\mathcal{T}'} D'$ hold. Since C' and D' do not contain any of the new concept names in $\mathcal{T}'$, both of these cases also hold with $\mathcal{T}$ instead of $\mathcal{T}'$.

- There are atoms $A_1, \dots, A_k$, B of $\mathcal{T}'$ such that $A_1 \sqcap \dots \sqcap A_k \sqsubseteq_{\mathcal{T}'} B$, $B \sqsubseteq^{\mathsf{s}}_{\mathcal{T}'} D$, and for every $\eta \in \{1, \dots, k\}$ there is $i \in \{1, \dots, n\}$ with $C_i \sqsubseteq^{\mathsf{s}}_{\mathcal{T}'} A_\eta$. Since every atom $A_1, \dots, A_k$, B is in a structural subsumption relationship with an atom that does not contain any of the new concept names, these atoms are either already concept names of $sig(\mathcal{T})$ or of the form $\exists r.A$ for some (old or new) concept name A.

  If A is an old concept name, $\exists r.A$ is already an atom of $\mathcal{T}$. Otherwise, by construction of $\mathcal{T}'$, there is a concept description C_A occurring in $\mathcal{T}$ such that $A \equiv_{\mathcal{T}'} C_A$. Replacing $\exists r.A$ by the equivalent $\exists r.C_A$ does not invalidate any of the subsumption relations that hold for this atom. For example, if $C_i \sqsubseteq^{\mathsf{s}}_{\mathcal{T}'} \exists r.A$ holds, then $C_i \sqsubseteq^{\mathsf{s}}_{\mathcal{T}} \exists r.C_A$ holds after the replacement.

  The above arguments show that we can find atoms $A'_1, \dots, A'_k$, B' of $\mathcal{T}$ in place of $A_1, \dots, A_k$, B for which all the above subsumptions and structural subsumptions hold w.r.t. $\mathcal{T}$ instead of $\mathcal{T}'$.

It remains to prove Lemma 6 for the case of a flat general TBox $\mathcal{T}$. For every subsumption $C_1 \sqcap \dots \sqcap C_n \sqsubseteq_{\mathcal{T}} D_1 \sqcap \dots \sqcap D_m$, by Lemma 9, there must be a proof tree $\mathbf{T}$ for $C_1 \sqcap \dots \sqcap C_n \vdash_{\mathcal{T}} D_1 \sqcap \dots \sqcap D_m$. We prove by induction on the height of $\mathbf{T}$ that for every atom D_j on the right-hand side one of the alternatives from Lemma 6 holds. Consider the rule applied at the root of $\mathbf{T}$.

- If ($\mathbf{R}_1$) has been applied, then $n = 2$, $m = 1$, and C_1, C_2 are atoms of $\mathcal{T}$ or $\top$ and D_1 is also an atom of $\mathcal{T}$. D_1 cannot be $\top$ since $\mathcal{T}$ is flat. Let $A_1, \dots, A_k$ be the atoms in $\{C_1, C_2\}$, i.e., we have $k \leq 2$. By Lemma 9, the subsumption $A_1 \sqcap \dots \sqcap A_k \sqsubseteq_{\mathcal{T}} D_1$ holds and for every $\eta \in \{1, \dots, k\}$ we have either $C_1 \sqsubseteq^{\mathsf{s}}_{\mathcal{T}} A_\eta$ or $C_2 \sqsubseteq^{\mathsf{s}}_{\mathcal{T}} A_\eta$ by reflexivity of $\sqsubseteq^{\mathsf{s}}_{\mathcal{T}}$. Similarly, we have $D_1 \sqsubseteq^{\mathsf{s}}_{\mathcal{T}} D_1$, and thus the second alternative of Lemma 6 holds for D_1.

- If ($\mathbf{R}_2$) has been applied, then $m = 1$ and $D_1 = \top$ and there is nothing to show since D_1 is not an atom.

- If ($\mathbf{R}_3$) has been applied, then $n = m$ and $C_i = D_i$ for every $i \in \{1, \dots, n\}$. By reflexivity of $\sqsubseteq^{\mathsf{s}}_{\mathcal{T}}$, we have $C_j \sqsubseteq^{\mathsf{s}}_{\mathcal{T}} D_j$ for every j with $D_j \neq \top$, and thus the first alternative holds for these atoms.

- If ($\mathbf{R}_4$) has been applied, then there is a proof tree $\mathbf{T}'$ for $C \sqcap C \vdash_{\mathcal{T}} D$ of height smaller than $\mathrm{h}(\mathbf{T})$. By induction, for every atom D_j one of the alternatives of Lemma 6 holds w.r.t. the left-hand side $C \sqcap C$. Since the top-level atoms of $C \sqcap C$ are exactly the top-level atoms of C, the same holds when considering C on the left-hand side.

- If $(\mathbf{R}_5)$ or $(\mathbf{R}_6)$ have been applied, then there is a proof tree $\mathbf{T}'$ for $C \vdash_\mathcal{T} D \sqcap \top$ or $C \vdash_\mathcal{T} \top \sqcap D$ of height smaller than $\mathrm{h}(\mathbf{T})$. By induction, for every atom D_j one of the alternatives holds.

- If $(\mathbf{R}_7)$ has been applied, then the two premises are of the form $E \vdash_\mathcal{T} F$ and $G \vdash_\mathcal{T} H$, where $E \sqcap G = C_1 \sqcap \cdots \sqcap C_n$ and $F \sqcap H = D_1 \sqcap \cdots \sqcap D_m$. Let $j \in \{1, \ldots, m\}$ and consider the atom $D_j \neq \top$. This atom must be a top-level atom of F or H; assume w.l.o.g. that it occurs in F. By induction, one of the alternatives holds for D_j w.r.t. the left-hand side E. Since every top-level atom of E is of the form C_i for some $i \in \{1, \ldots, n\}$, the same holds when considering C on the left-hand side.

- If $(\mathbf{R}_8)$ has been applied, then $n = m = 1$, $C_1 = \exists r.C'$ and $D_1 = \exists r.D'$ for some $r \in N_R$, and $C' \vdash_\mathcal{T} D'$. By Lemma 9, we have $C' \sqsubseteq_\mathcal{T} D'$ and thus, $C_1 \sqsubseteq^{\mathsf{s}}_\mathcal{T} D_1$, i.e., the first alternative holds for D_1.

- If $(\mathbf{R}_9)$ has been applied, then the premises are $C_1 \sqcap \cdots \sqcap C_n \vdash_\mathcal{T} E_1 \sqcap \cdots \sqcap E_k$ and $E_1 \sqcap \cdots \sqcap E_k \vdash_\mathcal{T} D_1 \sqcap \cdots \sqcap D_m$, where $E_1, \ldots, E_k$ are atoms or $\top$. By induction, we can distinguish several cases for every atom $D_j \neq \top$:

  1. There is $l \in \{1, \ldots, k\}$ such that $E_l \sqsubseteq^{\mathsf{s}}_\mathcal{T} D_j$. By definition of $\sqsubseteq^{\mathsf{s}}_\mathcal{T}$, this implies that $E_l \neq \top$. We again distinguish the following cases for E_l:
     1'. There is $i \in \{1, \ldots, k\}$ such that $C_i \sqsubseteq^{\mathsf{s}}_\mathcal{T} E_l$. By transitivity of $\sqsubseteq^{\mathsf{s}}_\mathcal{T}$, we have $C_i \sqsubseteq^{\mathsf{s}}_\mathcal{T} D_j$, i.e., the first alternative holds for D_j.
     2'. There are atoms $A_1, \ldots, A_\alpha, B$ of $\mathcal{T}$ with $A_1 \sqcap \cdots \sqcap A_\alpha \sqsubseteq_\mathcal{T} B$, $B \sqsubseteq^{\mathsf{s}}_\mathcal{T} E_l$, and for every $\eta \in \{1, \ldots, \alpha\}$ there is $i \in \{1, \ldots, n\}$ such that $C_i \sqsubseteq^{\mathsf{s}}_\mathcal{T} A_\eta$. By transitivity of $\sqsubseteq^{\mathsf{s}}_\mathcal{T}$, we have $B \sqsubseteq^{\mathsf{s}}_\mathcal{T} D_j$, and thus the second alternative holds for D_j.
  2. There are atoms $F_1, \ldots, F_\mu, G$ of $\mathcal{T}$ such that $F_1 \sqcap \cdots \sqcap F_\mu \sqsubseteq_\mathcal{T} G$, $G \sqsubseteq^{\mathsf{s}}_\mathcal{T} D_j$, and for every $\nu \in \{1, \ldots, \mu\}$ there is $l_\nu \in \{1, \ldots, k\}$ such that $E_{l_\nu} \sqsubseteq^{\mathsf{s}}_\mathcal{T} F_\nu$. We will replace every F_ν by a conjunction of atoms $A^\nu_1, \ldots, A^\nu_{\alpha_\nu}$ of $\mathcal{T}$ such that $A^\nu_1 \sqcap \cdots \sqcap A^\nu_{\alpha_\nu} \sqsubseteq_\mathcal{T} F_\nu$ and for every $\eta \in \{1, \ldots, \alpha_\nu\}$ there is $i \in \{1, \ldots, n\}$ such that $C_i \sqsubseteq^{\mathsf{s}}_\mathcal{T} A^\nu_\eta$. Since this implies that the subsumption $\bigsqcap_{\nu=1}^{\mu} A^\nu_1 \sqcap \cdots \sqcap A^\nu_{\alpha_\nu} \sqsubseteq_\mathcal{T} G$ holds, the second alternative holds for D_j.

     It remains to show how to replace F_ν for each $\nu \in \{1, \ldots, \mu\}$. Since $E_{l_\nu} \sqsubseteq^{\mathsf{s}}_\mathcal{T} F_\nu$, we know that $E_{l_\nu} \neq \top$. By induction, one of the following cases must hold for E_{l_ν}:
     1'. There is $i \in \{1, \ldots, n\}$ such that $C_i \sqsubseteq^{\mathsf{s}}_\mathcal{T} E_{l_\nu} \sqsubseteq^{\mathsf{s}}_\mathcal{T} F_\nu$. In this case, we do not need to replace F_ν, since it already has the desired property.
     2'. There are atoms $A^\nu_1, \ldots, A^\nu_{\alpha_\nu}, B$ of $\mathcal{T}$ with $A^\nu_1 \sqcap \cdots \sqcap A^\nu_{\alpha_\nu} \sqsubseteq_\mathcal{T} B$, $B \sqsubseteq^{\mathsf{s}}_\mathcal{T} E_{l_\nu}$, and for every $\eta \in \{1, \ldots, \alpha_\nu\}$ there is $i \in \{1, \ldots, n\}$ such that $C_i \sqsubseteq^{\mathsf{s}}_\mathcal{T} A_\eta$. This implies $A^\nu_1 \sqcap \cdots \sqcap A^\nu_{\alpha_\nu} \sqsubseteq_\mathcal{T} F_\nu$ and thus, we can replace F_ν by $A^\nu_1 \sqcap \cdots \sqcap A^\nu_{\alpha_\nu}$. $\square$

4 Cycle-Restricted TBoxes

We now present a restricted form of general TBoxes, in which we do not allow cyclic subsumptions of a certain form to occur. We will later show that, while these TBoxes are more expressive than acyclic TBoxes, they cannot express some cyclic TBoxes.

Definition 10. The general TBox $\mathcal{T}$ is called *cycle-restricted* iff there is no nonempty word $w \in N_R^+$ and $\mathcal{EL}$-concept description C such that $C \sqsubseteq_\mathcal{T} \exists w.C$.

We first show that for flat general TBoxes it suffices to consider cycles involving concept names and $\top$.

Lemma 11. *Let $\mathcal{T}$ be a flat general TBox. Then $\mathcal{T}$ is cycle-restricted iff there is no nonempty word $w \in N_R^+$ such that $\top \sqsubseteq_\mathcal{T} \exists w.\top$ or $A \sqsubseteq_\mathcal{T} \exists w.A$ for a concept name $A \in sig(\mathcal{T})$.*

Proof. The 'only if'-direction is trivial. We prove the other direction by induction on the structure of C, which can be $\top$, a concept name, an existential restriction, or a conjunction of several atoms and $\top$. If C is $\top$ or a concept name, the claim follows from the assumption.

If $C = \exists r.D$ for a role name r and a concept description D, assume that $\exists r.D \sqsubseteq_\mathcal{T} \exists w.\exists r.D$ holds for some $w \in N_R^+$. By Lemma 6, we either have $w = rw'$ and $D \sqsubseteq_\mathcal{T} \exists w'r.D$, which immediately contradicts the induction hypothesis, or there are atoms $\exists r.A_1, \ldots, \exists r.A_k, \exists s.B$ of $\mathcal{T}$ such that for every $\eta \in \{1, \ldots, k\}$ we have $D \sqsubseteq_\mathcal{T} A_\eta$, $\exists r.A_1 \sqcap \cdots \sqcap \exists r.A_k \sqsubseteq_\mathcal{T} \exists s.B$, and $w = sw'$ and $B \sqsubseteq_\mathcal{T} \exists w'r.D$. This implies that $B \sqsubseteq_\mathcal{T} \exists w'r.D \sqsubseteq_\mathcal{T} \exists w'.(\exists r.A_1 \sqcap \cdots \sqcap \exists r.A_k) \sqsubseteq_\mathcal{T} \exists w's.B$ holds. Since $\mathcal{T}$ is flat, B is a concept name or $\top$, and thus this subsumption contradicts the assumption.

If $C = C_1 \sqcap \cdots \sqcap C_n$, where $C_1, \ldots, C_n$ are atoms or $\top$, assume that $C_1 \sqcap \cdots \sqcap C_n \sqsubseteq_\mathcal{T} \exists w.(C_1 \sqcap \cdots \sqcap C_n)$ holds for some $w \in N_R^+$. By Lemma 6, there are two possibilities:

1. We have $C_i \sqsubseteq_\mathcal{T} \exists w.(C_1 \sqcap \cdots \sqcap C_n) \sqsubseteq_\mathcal{T} \exists w.C_i$ for some $i \in \{1, \ldots, n\}$, which contradicts the induction hypothesis.

2. There are atoms $A_1, \ldots, A_k, \exists s.B$ of $\mathcal{T}$ such that for every $\eta \in \{1, \ldots, k\}$ there is $i \in \{1, \ldots, n\}$ with $C_i \sqsubseteq_\mathcal{T}^s A_\eta$, $A_1 \sqcap \cdots \sqcap A_k \sqsubseteq_\mathcal{T} \exists s.B$, and $w = sw'$ and $B \sqsubseteq_\mathcal{T} \exists w'.(C_1 \sqcap \cdots \sqcap C_n)$. This implies that $B \sqsubseteq_\mathcal{T} \exists w'.(A_1 \sqcap \cdots \sqcap A_k) \sqsubseteq_\mathcal{T} \exists w's.B$, which again contradicts the assumption. □

The condition in Definition 10 can be tested by the following procedure, which is based on Lemma 6.

Lemma 12. *Let $\mathcal{T}$ be a general TBox. It can be decided in time polynomial in the size of $\mathcal{T}$ whether $\mathcal{T}$ is cycle-restricted or not.*

Proof. We first flatten $\mathcal{T}$ as described in Section 2.3. The resulting TBox $\mathcal{T}'$ has a larger signature than $\mathcal{T}$, but each new concept name A is equivalent to a concept description C_A over the signature of the original TBox. Furthermore, we can show that $\mathcal{T}'$ is cycle-restricted iff $\mathcal{T}$ is. Assume first that $\mathcal{T}$ is not cycle-restricted, i.e., there is a concept description C over $sig(\mathcal{T})$ and $w \in N_R^+$ such that $C \sqsubseteq_\mathcal{T} \exists w.C$. Since $\mathcal{T}'$ is $sig(\mathcal{T})$-inseparable from $\mathcal{T}$, the same holds w.r.t. $\mathcal{T}'$, which shows that $\mathcal{T}'$ is not cycle-restricted. On the other hand, if $C \sqsubseteq_{\mathcal{T}'} \exists w.C$ for $w \in N_R^+$ and a concept description C over $sig(\mathcal{T}')$, then we can replace each new concept name A by the equivalent C_A. The resulting concept description C' is built over $sig(\mathcal{T})$, and thus $C' \sqsubseteq_\mathcal{T} \exists w.C'$, i.e., $\mathcal{T}$ is not cycle-restricted.

Thus, we can assume in the following that $\mathcal{T}$ is flat. By Lemma 11, we only have to test for cycles involving concept names and $\top$. We first characterize such cycles in a convenient way. Let A be a concept name or $\top$. By Lemma 6, $A \sqsubseteq_\mathcal{T} \exists r w'.A$ holds for $w' \in N_R^*$ iff one of the two alternatives of this lemma holds. The first alternative cannot hold since $\exists r w'.A$ and A have an incompatible top-level structure – one is an existential restriction, the other is a concept name. Thus, we have $A \sqsubseteq_\mathcal{T} \exists r w'.A$ iff there atoms $A'_1, \dots, A'_k, \exists r.B$ of $\mathcal{T}$ such that $A \sqsubseteq^{\mathsf{s}}_\mathcal{T} A'_\eta$ holds for all $\eta \in \{1, \dots, k\}$, $A'_1 \sqcap \dots \sqcap A'_k \sqsubseteq_\mathcal{T} \exists r.B$, and $B \sqsubseteq_\mathcal{T} \exists w'.A$.

If $A = \top$, then k must be 0 since $\top \sqsubseteq^{\mathsf{s}}_\mathcal{T} A'_\eta$ cannot hold. This implies $A = \top \sqsubseteq_\mathcal{T} \exists r.B$. If A is a concept name, then all A'_η must be equal to A, and again we have $A \sqsubseteq_\mathcal{T} \exists r.B$. If w' is not empty, we can apply the same argument to the subsumption $B \sqsubseteq_\mathcal{T} \exists w'.A$ since B is either a concept name or $\top$ since $\mathcal{T}$ is flat. We can iterate this argument until only the empty word remains, which yields a sequence of subsumptions $A \sqsubseteq_\mathcal{T} \exists r.B, B \sqsubseteq_\mathcal{T} \exists r_2.B_2, \dots, B_{n-1} \sqsubseteq_\mathcal{T} \exists r_n.B_n, B_n \sqsubseteq_\mathcal{T} A$ that hold between atoms of $\mathcal{T}$ (or $\top$).

Since subsumption w.r.t. $\mathcal{T}$ can be checked in polynomial time, we can construct the following graph in polynomial time: The nodes are the concept names of $\mathcal{T}$ and $\top$. There is an edge labeled by r from A to B iff $A \sqsubseteq_\mathcal{T} \exists r.B$ and an edge labeled by ε from A to B iff $A \sqsubseteq_\mathcal{T} B$. The size of this graph is polynomial in the size of $\mathcal{T}$.

To check whether $\mathcal{T}$ contains cycles it suffices to check for cycles in this graph that contain at least one edge labeled by a role name. This can be checked in polynomial time in the size of this graph. □

Example 13. Consider the general TBox $\{\exists r.A \sqsubseteq A, A \sqsubseteq \exists s.B\}$. The graph constructed in Lemma 12 has the tree nodes A, B, and $\top$. It contains s-edges from A to B and from A to $\top$ and ε-edges from A to $\top$ and from B to $\top$. Since these edges form no cycles, the TBox is a cycle-restricted TBox.

4.1 Relationship to Other Classes

We now analyze the expressiveness of cycle-restricted TBoxes in relation to the previously mentioned classes of TBoxes. Of course, every cycle-restricted TBox is also a general TBox.

Lemma 14. *For every acyclic TBox $\mathcal{T}'$ there is a cycle-restricted TBox $\mathcal{T}$ that is $sig(\mathcal{T}')$-inseparable from $\mathcal{T}'$.*

Proof. By replacing all definitions $A \equiv C$ of $\mathcal{T}'$ by the two equivalent subsumptions $A \sqsubseteq C$ and $C \sqsubseteq A$, we obtain a general TBox $\mathcal{T}$ that is $sig(\mathcal{T}')$-inseparable from $\mathcal{T}'$. To show that this is even a cycle-restricted TBox, assume that $A \sqsubseteq_{\mathcal{T}'} \exists w.A$ holds for some concept name A of $sig(\mathcal{T}) = sig(\mathcal{T}')$ and $w \in N_R^+$.

We can *expand* A by exhaustively replacing defined concept names by their definitions in $\mathcal{T}'$. Since this TBox is acyclic, this process terminates in a concept description $C_A \equiv_{\mathcal{T}} A$ that contains only concept names without definition. Thus, the subsumption $C_A \sqsubseteq \exists w.C_A$ must hold w.r.t. the empty TBox. However, it is a consequence of Lemma 2 that whenever $C \sqsubseteq D$, then the role depth of C must be greater than or equal to the role depth of D. This contradicts $C_A \sqsubseteq \exists w.C_A$ and the assumption that $w \in N_R^+$. □

Thus, every acyclic TBox can be expressed by a cycle-restricted TBox. On the other hand, it turns out that there are some cycle-restricted TBoxes whose restrictions cannot even expressed by a cyclic TBox.

To show this, we use a characterization of subsumption w.r.t. cyclic TBoxes from [3]. First, we have to introduce some preliminary notions. A cyclic TBox $\mathcal{T}$ is said to be *normalized* if all its definitions are of the form $A \equiv P_1 \sqcap \ldots P_n \sqcap \exists r_1.B_1 \sqcap \cdots \sqcap \exists r_m.B_m$, where $P_1, \ldots, P_n$ are *primitive concepts*, i.e., have no definitions, $r_1, \ldots, r_m$ are role names, and $B_1, \ldots, B_m$ are *defined concepts*. Every cyclic TBox can be transformed into a normalized TBox that is inseparable from the original one w.r.t. the original signature. This is due to the fact that the normalization procedure described in [2] only employs the following operations, none of which affect the subsumption relationships between concepts built over the original signature:

- Introduction of auxiliary definitions of the form $B \equiv \exists r.A$ for a new concept name B.
- Merging of equivalent concept names.
- Introduction of new concept names to transform subsumptions like $A \sqsubseteq \exists r.B$ into definitions $A \equiv A' \sqcap \exists r.B$.

The *description graph* $\mathcal{G}_\mathcal{T}$ of a cyclic TBox $\mathcal{T}$ consists of a node for each defined concept and an edge from A to B labeled by r whenever $\exists r.B$ is a conjunct in the definition of A.[5] A *simulation* from $\mathcal{G}_\mathcal{T}$ to $\mathcal{G}_\mathcal{T}$ is a binary relation S on the set of all defined concepts that satisfies the following condition: If $(A, B) \in S$ and there is an edge from A to A' in $\mathcal{G}_\mathcal{T}$ labeled by r, then there has to be a defined concept B' such that $(A', B') \in S$ and there is an edge from B to B' labeled by r. A useful consequence of one of the main results of [3] is the following: Whenever $A \sqsubseteq_\mathcal{T} B$ holds between two defined concepts, then there is a simulation S from $\mathcal{G}_\mathcal{T}$ to $\mathcal{G}_\mathcal{T}$ with $(B, A) \in S$.

Lemma 15. *There is no cyclic TBox $\mathcal{T}$ that is $\{r, s, A, B\}$-inseparable from the set $\{\exists r.A \sqsubseteq A, A \sqsubseteq \exists s.B\}$, which is a cycle-restricted TBox (see Example 13).*

Proof. Assume that $\mathcal{T}$ is a cyclic TBox that is $\{r, s, A, B\}$-inseparable from $\{\exists r.A \sqsubseteq A, A \sqsubseteq \exists s.B\}$. As described before, we can assume that $\mathcal{T}$ is normalized in the sense of [3]. We introduce two new concept names A' and B' with new definitions $A' \equiv \exists r.A$ and $B' \equiv \exists s.B$. The resulting description graph has two additional nodes for A' and B' and two additional edges: one from A' to A labeled by r and one from B' to B labeled by s. Since A' and B' are new concept names, they have no influence on the subsumptions holding between concepts built over the signature $\{r, s, A, B\}$, and thus the resulting TBox $\mathcal{T}'$ is still $\{r, s, A, B\}$-inseparable from $\{\exists r.A \sqsubseteq A, A \sqsubseteq \exists s.B\}$. In particular, we have $A' \equiv_{\mathcal{T}'} \exists r.A \sqsubseteq_{\mathcal{T}'} A$ and $A \sqsubseteq_{\mathcal{T}'} \exists s.B \equiv_{\mathcal{T}'} B'$.

From the first subsumption we can deduce that there is a simulation S from $\mathcal{G}_{\mathcal{T}'}$ to $\mathcal{G}_{\mathcal{T}'}$ with $(A, A') \in S$. Thus, any edge starting from A must be simulated by an edge starting from A'. Since the only edge starting in A' is labeled by r, every edge starting in A must also be labeled by r. From the second subsumption it follows that there is a simulation S' with $(B', A) \in S'$. Thus, the edge from B' to B must be simulated by an edge starting in A, i.e., there must be an edge starting in A that is labeled by s. Together, the two subsumptions $A' \sqsubseteq_{\mathcal{T}'} A$ and $A \sqsubseteq_{\mathcal{T}'} B'$ thus lead to a contradiction. □

But cycle-restricted TBoxes cannot express all cyclic TBoxes–there is a simple example of a cyclic TBox that cannot be expressed by a cycle-restricted TBox.

Lemma 16. *There is no cycle-restricted TBox $\mathcal{T}$ that is $\{r, A\}$-inseparable from $\{A \equiv \exists r.A\}$, which is a cyclic TBox.*

Proof. For any such TBox $\mathcal{T}$ we would have $A \sqsubseteq_\mathcal{T} \exists r.A$, which directly contradicts Definition 10. □

Thus, we can summarize the relationships between the four discussed classes of TBoxes as depicted in Figure 1. By considering cycle-restricted TBoxes we are

[5] We ignore the node labels from [3] since they are not important for our arguments.

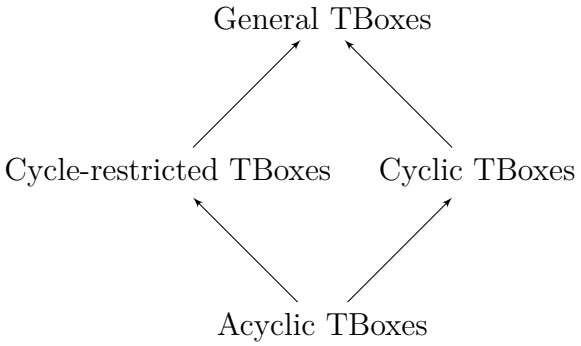

Figure 1: The relations between the expressiveness of the four classes of TBoxes discussed here. An arrow depicts a strict increase in expressiveness.

more expressive than acyclic TBoxes, but do not reach the full expressivity of general TBoxes or even cyclic TBoxes.

This concludes the first part of this report. We now present unification in $\mathcal{EL}$ and two algorithms to solve $\mathcal{EL}$-unification w.r.t. cycle-restricted TBoxes.

5 Unification

From now on, we assume that the set N_C is partitioned into *concept variables* (N_v) and *concept constants* (N_c). A *substitution* σ maps every variable to a concept description and can be extended to concept descriptions in the usual way. A concept description C is *ground* if it contains no variables and a substitution is *ground* if all concept descriptions in its range are ground. Similarly, a TBox is *ground* if it contains no variables.

Definition 17. Let $\mathcal{T}$ be a general TBox that is ground. An *$\mathcal{EL}$-unification problem* w.r.t. $\mathcal{T}$ is a finite set $\Gamma = \{C_1 \sqsubseteq^? D_1, \ldots, C_n \sqsubseteq^? D_n\}$ of subsumptions. A substitution σ is a *unifier* of Γ w.r.t. $\mathcal{T}$ if σ *solves* all the subsumptions in Γ, i.e., if $\sigma(C_1) \sqsubseteq_\mathcal{T} \sigma(D_1), \ldots, \sigma(C_n) \sqsubseteq_\mathcal{T} \sigma(D_n)$. We say that Γ is *unifiable* w.r.t. $\mathcal{T}$ if it has a unifier.

Three remarks regarding this definition are in order. First, note that the previous papers on unification in DLs used 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 $C \sqcap D \equiv^? C$.

Second, note that we have restricted the background general TBox $\mathcal{T}$ to be ground. This is not without loss of generality. In fact, if $\mathcal{T}$ contained variables, then we would need to apply the substitution also to its axioms, and instead of requiring $\sigma(C_i) \sqsubseteq_\mathcal{T} \sigma(D_i)$ we would thus need to require $\sigma(C_i) \sqsubseteq_{\sigma(\mathcal{T})} \sigma(D_i)$, which

would change the nature of the problem considerably. The treatment of unification w.r.t. acyclic TBoxes in [7] actually considers a more general setting, where some of the primitive concepts occurring in the TBox may be variables. The restriction to ground general TBoxes is, however, appropriate for the application scenario sketched in the introduction. In this scenario, there is a fixed background ontology, given by a general TBox, which is extended with definitions of new concepts by several knowledge engineers. Unification w.r.t. the background ontology is used to check whether some of these new definitions actually are redundant, i.e., define the same intuitive concept. Here, some of the primitive concepts newly introduced by one knowledge engineer may be further defined by another one, but we assume that the knowledge engineers use the vocabulary from the background ontology unchanged, i.e., they define *new* concepts rather than adding definitions for concepts that already occur in the background ontology. An instance of this scenario can, e.g., be found in [13], where different extensions of SNOMED CT are checked for overlaps, albeit not by using unification, but by simply testing for equivalence.

Third, though we allow for arbitrary substitutions σ in the definition of a unifier, it is actually sufficient to consider ground substitutions such that all concept descriptions $\sigma(X)$ in the range of σ contain only concept and role names occurring in Γ or $\mathcal{T}$. It is an easy consequence of well-known results from unification theory [10] that Γ has a unifier w.r.t. $\mathcal{T}$ iff it has such a ground unifier.

Given a general TBox $\mathcal{T}$ and a unification problem Γ, we will first flatten $\mathcal{T}$ as described in Section 2.3. Since the signature of the TBox is changed by this process, this has consequences for the unifiers: when looking for a unifier w.r.t. a given TBox one does not want this unifier to use auxiliary concept names introduced in a preprocessing step of the unification algorithm. The next lemma shows, however, that unifiability of a unification problem is not influenced by flattening the TBox. The proof of this lemma also shows how to remove unwanted auxiliary concept names from a unifier.

Lemma 18. *Let Γ be a unification problem, $\mathcal{T}$ be a general TBox, and $\mathcal{T}'$ be the result of applying the normalization procedure from [12] to $\mathcal{T}$. Then Γ is unifiable w.r.t. $\mathcal{T}$ iff it is unifiable w.r.t. $\mathcal{T}'$.*

Proof. Any unifier σ of Γ w.r.t. $\mathcal{T}$ is also a unifier of Γ w.r.t. $\mathcal{T}'$ since the equivalences holding over $sig(\mathcal{T})$ w.r.t. $\mathcal{T}$ also hold w.r.t. $\mathcal{T}'$. If, on the other hand, σ' is a unifier of Γ w.r.t. $\mathcal{T}'$, then its range may contain some of the newly introduced concept names. However, each of these new concept names A is equivalent to a concept description C_A from $\mathcal{T}$. We now define the substitution σ by replacing all occurrences of the new concept names A by the concept descriptions C_A. Since equivalences are preserved under replacing subdescriptions by equivalent concept descriptions, σ is still a unifier of Γ w.r.t. $\mathcal{T}'$. Since it does not contain any concept names introduced by the flattening procedure, it is also a unifier of Γ w.r.t. $\mathcal{T}$ over the original signature. □

Without loss of generality, we can also assume that the unification problem Γ is *flat* in the sense that it contains only subsumptions of the form $C_1 \sqcap \cdots \sqcap C_n \sqsubseteq^? D$, where $C_1, \ldots, C_n$, and D are flat atoms.[6] This normal form can be achieved by introducing auxiliary variables and splitting any subsumption with a conjunction on the right-hand side into several smaller subsumptions.

5.1 Unifiers versus Acyclic TBoxes

There is a close relationship between ground substitutions and acyclic TBoxes. Given a ground substitution σ, we can build the TBox $\mathcal{T}_\sigma := \{X \equiv \sigma(X) \mid X \in N_v\}$. Since σ is ground, this is indeed an acyclic TBox, and expansion w.r.t. $\mathcal{T}_\sigma$ corresponds to applying σ, i.e., for every concept description C we have $\sigma(C) = C^{\mathcal{T}_\sigma}$. As an easy consequence of this observation we have for any ground general TBox $\mathcal{T}$:

$$\sigma(C) \sqsubseteq_{\mathcal{T}} \sigma(D) \text{ iff } C^{\mathcal{T}_\sigma} \sqsubseteq_{\mathcal{T}} D^{\mathcal{T}_\sigma} \text{ iff } C \sqsubseteq_{\mathcal{T} \cup \mathcal{T}_\sigma} D.$$

Conversely, any acyclic TBox $\mathcal{S}$ whose defined concepts are the variables in N_v yields a ground substitution $\sigma_\mathcal{S}$, which is defined by setting $\sigma_\mathcal{S}(X) = X^\mathcal{S}$ for all variables X. Again, expansion w.r.t. the acyclic TBox corresponds to applying the substitution, i.e., $C^\mathcal{S} = \sigma_\mathcal{S}(C)$, and thus

$$C \sqsubseteq_{\mathcal{T} \cup \mathcal{S}} D \text{ iff } C^\mathcal{S} \sqsubseteq_{\mathcal{T}} D^\mathcal{S} \text{ iff } \sigma_\mathcal{S}(C) \sqsubseteq_{\mathcal{T}} \sigma_\mathcal{S}(D).$$

This yields another view on what unification is trying to compute, and thus another potential application scenario: the extraction of concept definitions that imply a given set of GCIs w.r.t. a background ontology.

Proposition 19. *Let $\mathcal{T}$ be a ground general TBox and $\mathcal{T}'$ an arbitrary general TBox. Then $\Gamma' := \{C \sqsubseteq^? D \mid C \sqsubseteq D \in \mathcal{T}'\}$ has a unifier w.r.t. $\mathcal{T}$ iff there is an acyclic TBox $\mathcal{S}$ whose defined concepts are the variables in N_v such that every GCI in $\mathcal{T}'$ follows from $\mathcal{T} \cup \mathcal{S}$.*

5.2 Relationship to Equational Unification

Unification was originally not introduced for Description Logics, but for equational theories [10]. In [23, 7] it was shown that equivalence and unification in $\mathcal{EL}$ are the same as the word problem and unification, respectively, in the equational theory $SLmO$ of semilattices with monotone operators. The signature Σ_{SLmO} of this theory consists of a binary function symbol $\wedge$, a constant symbol 1, and finitely many unary function symbols $f_1, \ldots, f_n$. Terms can be built using these symbols and additional variable symbols and free constant symbols.

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

Definition 20. The equational theory of *semilattices with monotone operators* is defined by the following identities:

$$\begin{aligned} SLmO := \{&x \wedge (y \wedge z) = (x \wedge y) \wedge z,\ x \wedge y = y \wedge x,\ x \wedge x = x,\ x \wedge 1 = x\} \\ &\cup \{f_i(x \wedge y) \wedge f_i(y) = f_i(x \wedge y) \mid 1 \leq i \leq n\} \end{aligned}$$

Any $\mathcal{EL}$-concept description C using only the roles $r_1, \ldots, r_n$ can be translated into a term t_C over the signature Σ_{SLmO} by replacing each concept constant A by a free constant a, each concept variable X by a variable x, $\top$ by 1, $\sqcap$ by $\wedge$, and $\exists r_i$ by f_i. For example, the $\mathcal{EL}$-concept description $C = A \sqcap \exists r_1.\top \sqcap \exists r_3(X \sqcap B)$ is translated into $t_C = a \wedge f_1(1) \wedge f_3(x \wedge b)$. Conversely, any term t over the signature Σ_{SLmO} can be translated back into an $\mathcal{EL}$-concept description C_t. As shown in [23], the word problem in the theory $SLmO$ is the same as the equivalence problem for $\mathcal{EL}$-concept descriptions.

Lemma 21. *Let C, D be $\mathcal{EL}$-concept descriptions using only roles $r_1, \ldots, r_n$. Then $C \equiv D$ iff $t_C =_{SLmO} t_D$.*

As an immediate consequence of this lemma, every $\mathcal{EL}$-unification problem can be translated into an $SLmO$-unification problem that, modulo the translation between concept descriptions and terms, has the same unifiers.

Using this translation, any ground general TBox $\mathcal{T}$ can be translated into a finite set $G_{\mathcal{T}}$ of ground identities by replacing each GCI $C \sqsubseteq D$ by the equation $t_C \wedge t_D = t_C$. Conversely, a set G of ground identities can be translated back into a ground general TBox $\mathcal{T}_G$ by replacing every ground identity $s = t$ by the GCIs $C_s \sqsubseteq C_t$ and $C_t \sqsubseteq C_s$. Lemma 21 can easily be extended to account for an additional ground general TBox.

Proposition 22. *Let $\mathcal{T}$ be a ground general TBox and C, D be $\mathcal{EL}$-concept descriptions using only roles $r_1, \ldots, r_n$. Then $C \equiv_{\mathcal{T}} D$ iff $t_C =_{SLmO \cup G_{\mathcal{T}}} t_D$.*

Unification in $\mathcal{EL}$ w.r.t. a ground general TBox, as introduced in Definition 17, thus corresponds to unification in $SLmO$ extended with a finite set of ground identities. From a unification theory point of view, we are thus dealing with an instance of the following general question:

Problem. For which equational theories E does decidability and/or complexity transfer from E to all extensions of E by finite sets of ground identities?

The connection to equational unification also sheds some light on our decision to restrict unification to the case of general TBoxes that are ground. If we would lift this restriction, the background general TBox $\mathcal{T}$ would contain variables, which are subject to substitution. For a substitution σ, we define $\sigma(\mathcal{T})$ to be the set

of all GCIs $\sigma(C) \sqsubseteq \sigma(D)$ for all GCIs $C \sqsubseteq D$ in $\mathcal{T}$. Consider now the following generalization of Definition 17:[7]

Problem ($\mathcal{EL}$-unification w.r.t. a non-ground general TBox). Given a general TBox $\mathcal{T}$ and an $\mathcal{EL}$-unification problem $\Gamma = \{C_1 \equiv^? D_1, \ldots, C_n \equiv^? D_n\}$, is there a substitution σ that satisfies $\sigma(C_i) \equiv_{\sigma(\mathcal{T})} \sigma(D_i)$ for all $i \in \{1, \ldots, n\}$?

According to the above translations, this is equivalent to finding a substitution σ with $\sigma(t_{C_i}) =_{SLmO \cup \sigma(G_\mathcal{T})} \sigma(t_{D_i})$ for all $i \in \{1, \ldots, n\}$, where the variables in $\sigma(G_\mathcal{T})$ are viewed as free constant symbols instead of proper (i.e., universally quantified) variables. This problem is related to the following problem [16, 15]:

Problem (Simultaneous rigid E-unification). Given finitely many equational theories $E_1, \ldots, E_n$ and terms $s_1, \ldots, s_n$, $t_1, \ldots, t_n$, is there a substitution σ that satisfies $\sigma(s_i) =_{\sigma(E_i)} \sigma(t_i)$ for all $i \in \{1, \ldots, n\}$, where the variables in $\sigma(E_i)$ are treated as free constant symbols?

Rigid E-unification is the special case where $n = 1$. In general, simultaneous rigid E-unification is undecidable [15], even in the case $n = 3$ with only 2 variables and ground terms s_1, s_2, s_3 [24]. For the case of only monadic function symbols, the problem is known to be PSPACE-hard [17], and in PSPACE if there is only one variable [18]. If there is only one variable, but arbitrary function symbols, then the problem is EXPTIME-complete [14]. The restricted problem of (non-simultaneous) rigid E-unification is decidable (more precisely, NP-complete) [16]. If there is only one variable, then the problem is P-complete [14].

Our problem is a generalization of rigid E-unification rather than simultaneous rigid E-unification since we use only one general TBox $\mathcal{T}$ rather than a different one for every equivalence. The main generalization is that we have $SLmO$ as additional non-ground background theory. Whether the fact that we have several equivalences rather than a single one is relevant or not is not so clear. In fact, if $\mathcal{T}$ is ground, then several equivalences can be encoded into a single one if sufficiently many *free* role names are available, i.e., role names that do not occur in $\mathcal{T}$. The following is an easy consequence of Lemma 6: if $r_1, \ldots, r_n$ are distinct free role names, then σ is a unifier of $\{C_1 \equiv^? D_1, \ldots, C_n \equiv^? D_n\}$ w.r.t. $\mathcal{T}$ iff it is a unifier of $\{\exists r_1.C_1 \sqcap \cdots \sqcap \exists r_n.C_n \equiv^? \exists r_1.D_1 \sqcap \cdots \sqcap \exists r_n.D_n\}$ w.r.t. $\mathcal{T}$. If $\mathcal{T}$ is not ground, this trick does not necessarily work since, even if $r_1, \ldots, r_n$ are free w.r.t. $\mathcal{T}$, they may no longer be free w.r.t. $\sigma(\mathcal{T})$.

To sum up, $\mathcal{EL}$-unification w.r.t. non-ground general TBoxes is an instance of the following generalization of simultaneous rigid E-unification:

Problem (Simultaneous rigid E-unification with background theory E'). Given finitely many equational theories $E_1, \ldots, E_n, E'$ and terms $s_1, \ldots, s_n$, $t_1, \ldots, t_n$,

[7] We use equivalences rather than subsumptions in this definition to have a more direct connection to equational unification problems. As noted above, equivalences can be translated into subsumptions and vice versa.

is there a substitution σ that satisfies $\sigma(s_i) =_{E' \cup \sigma(E_i)} \sigma(t_i)$ for all $i \in \{1, \ldots, n\}$, where the variables in $\sigma(E_i)$ are treated as free constant symbols?

The non-simultaneous version of this problem considers the case where $n = 1$. To the best of our knowledge, the problem of simultaneous or non-simultaneous rigid E-unification with background theory has not yet been considered in the literature, and it is probably quite hard to solve even in the non-simultaneous case. This is one of our reasons for restricting our attention to the case of a single ground general TBox.

6 A Brute-Force NP-Algorithm

Since deciding unifiability of $\mathcal{EL}$-unification problems is NP-complete w.r.t. the empty TBox [5], NP-hardness also holds in our more general setting. In this section, we provide a short proof for the fact that unifiability is still in NP w.r.t. cycle-restricted TBoxes.

As we have already shown, we can assume without loss of generality that the unification problem Γ and the cycle-restricted TBox $\mathcal{T}$ are flat. We denote by At the set of atoms occurring as subdescriptions in subsumptions in Γ or axioms in $\mathcal{T}$. Furthermore, we define the set of *non-variable atoms* by $\mathrm{At}_{\mathsf{nv}} := \mathrm{At} \setminus N_v$.

6.1 Local unifiers

The main idea underlying the "in NP" result in [5] is to show that any $\mathcal{EL}$-unification problem that is unifiable w.r.t. the empty TBox has a so-called local unifier. Here, we generalize the notion of a local unifier to the case of unification w.r.t. cycle-restricted TBoxes, and show that a similar locality result holds in this case.

Every assignment S of subsets S_X of $\mathrm{At}_{\mathsf{nv}}$ to the variables X in N_v induces a unique TBox

$$\mathcal{T}_S := \{X \equiv \bigsqcap_{D \in S_X} D \mid X \in N_v\}.$$

We call the assignment S *acyclic* if $\mathcal{T}_S$ is acyclic. Thus, if S is acyclic, the TBox $\mathcal{T}_S$ induces a unique substitution $\sigma_{\mathcal{T}_S}$. To simplify the notation, we write this substitution as σ_S. It is easy to see that this substitution satisfies

$$\sigma_S(X) = \bigsqcap_{D \in S_X} \sigma_S(D)$$

for all $X \in N_v$. We call a substitution σ *local* if it is of this form, i.e., if there is an acyclic assignment S such that $\sigma = \sigma_S$.

Theorem 23. *Let $\mathcal{T}$ be a flat cycle-restricted TBox and Γ a flat unification problem. If Γ has a unifier w.r.t. $\mathcal{T}$, then it also has a local unifier w.r.t. $\mathcal{T}$.*

This theorem immediately implies that unification in $\mathcal{EL}$ w.r.t. cycle-restricted TBoxes is decidable within NP. In fact, one can guess an acyclic assignment S in polynomial time. To check whether the induced local substitution σ_S is a unifier of Γ w.r.t. $\mathcal{T}$, one builds the TBox $\mathcal{T}_S$ and then checks in polynomial time whether $C \sqsubseteq_{\mathcal{T} \cup \mathcal{T}_S} D$ holds for all subsumptions $C \sqsubseteq^? D$ in Γ.

Corollary 24. *Unification in $\mathcal{EL}$ w.r.t. cycle-restricted TBoxes is in NP.*

6.2 Proof of Theorem 23

Assume that γ is a unifier of Γ w.r.t. $\mathcal{T}$. We define the assignment S^γ induced by γ as

$$S^\gamma_X := \{D \in \mathrm{At}_{\mathsf{nv}} \mid \gamma(X) \sqsubseteq_{\mathcal{T}} \gamma(D)\}.$$

The following lemma is the only place in the proof of Theorem 23 where cycle-restrictedness of $\mathcal{T}$ is needed. Later we will give an example that demonstrates that the theorem actually does not hold if this restriction is removed.

Lemma 25. *The assignment S^γ is acyclic.*

Proof. Assume that S^γ is cyclic. Then there are variables $X_1, \ldots, X_n$ and role names $r_1, \ldots, r_{n-1}$ ($n \geq 2$) such that $X_1 = X_n$ and $\exists r_i.X_{i+1} \in S^\gamma(X_i)$ ($i = 1, \ldots, n-1$). But then we have $\gamma(X_i) \sqsubseteq_{\mathcal{T}} \exists r_i.\gamma(X_{i+1})$ for $i = 1, \ldots, n-1$, which yields $\gamma(X_1) \sqsubseteq_{\mathcal{T}} \exists r_1.\gamma(X_2) \sqsubseteq_{\mathcal{T}} \exists r_1.\exists r_2.\gamma(X_3) \sqsubseteq_{\mathcal{T}} \cdots \sqsubseteq_{\mathcal{T}} \exists r_1.\cdots\exists r_{n-1}.\gamma(X_n)$. Since $X_1 = X_n$ and $n \geq 2$, this contradicts our assumption that $\mathcal{T}$ is cycle-restricted. Thus, S^γ must be acyclic. □

Since S^γ is acyclic, it induces a substitution σ_{S^γ}. To simplify the notation, we call this substitution in the following σ^γ. The following lemma implies that σ^γ is a unifier of Γ w.r.t. $\mathcal{T}$, and thus proves Theorem 23.

Lemma 26. *Let $C_1, \ldots, C_n, D \in \mathrm{At}$. Then $\gamma(C_1) \sqcap \ldots \sqcap \gamma(C_n) \sqsubseteq_{\mathcal{T}} \gamma(D)$ implies $\sigma^\gamma(C_1) \sqcap \ldots \sqcap \sigma^\gamma(C_n) \sqsubseteq_{\mathcal{T}} \sigma^\gamma(D)$.*

Proof. We prove the lemma by induction over

$$max\{\mathsf{rd}(\sigma^\gamma(E)) \mid E \in \{C_1, \ldots, C_n, D\} \wedge E \text{ not ground}\}.$$

First, assume that $D = Y \in N_v$, and let $S^\gamma_Y = \{D_1, \ldots, D_m\}$. By the definition of S^γ, this implies $\gamma(Y) \sqsubseteq_{\mathcal{T}} \gamma(D_1) \sqcap \ldots \sqcap \gamma(D_m)$, and thus

$$\gamma(C_1) \sqcap \ldots \sqcap \gamma(C_n) \sqsubseteq_{\mathcal{T}} \gamma(D_1) \sqcap \ldots \sqcap \gamma(D_m).$$

We apply Lemma 6 to this subsumption. Consider $\gamma(D_j)$ for some $j, 1 \leq j \leq m$. Since D_j is a non-variable atom, $\gamma(D_j)$ is an atom, and thus the first or the second case of the lemma holds.

1. In the first case, there is an $i, 1 \leq i \leq n$, such that one of the following two cases holds:

   (i) *C_i is a non-variable atom and $\gamma(C_i) \sqsubseteq^s_{\mathcal{T}} \gamma(D_j)$.*
   By the definition of $\sqsubseteq^s_{\mathcal{T}}$, there are two possible cases. Either both concept descriptions are the same concept name A, or both are existential restrictions for the same role name r. In the first case, $C_i = A = D_j$, and thus $\sigma^\gamma(C_i) = A = \sigma^\gamma(D_j)$. In the second case, $C_i = \exists r.C'_i$, $D_j = \exists r.D'_j$, and $\gamma(C'_i) \sqsubseteq_{\mathcal{T}} \gamma(D'_j)$. Both C'_i and D'_j are elements of At. The role depth of $\sigma^\gamma(C'_i)$ is obviously smaller than the role depth of $\sigma^\gamma(C_i)$. For the same reason, the role depth of $\sigma^\gamma(D'_j)$ is smaller that the one of $\sigma^\gamma(D_j)$. Since $\sigma^\gamma(D_j)$ is a top-level conjunct in $\sigma^\gamma(D)$, the role depth of $\sigma^\gamma(D'_j)$ is also smaller than the ones of $\sigma^\gamma(D)$. Consequently, if C_i or D_j is non-ground, induction yields $\sigma^\gamma(C'_i) \sqsubseteq_{\mathcal{T}} \sigma^\gamma(D'_j)$, and thus also $\sigma^\gamma(C_i) = \exists r.\sigma^\gamma(C'_i) \sqsubseteq_{\mathcal{T}} \exists r.\sigma^\gamma(D'_j) = \sigma^\gamma(D_j)$. If C_i, D_j are both ground, then $\sigma^\gamma(C_i) = C_i = \gamma(C_i) \sqsubseteq_{\mathcal{T}} \gamma(D_j) = D_j = \sigma^\gamma(D_j)$.

   (ii) *$C_i = X$ is a variable and the top-level conjunction of $\gamma(X)$ contains an atom E such that $E \sqsubseteq^s_{\mathcal{T}} \gamma(D_j)$.*
   Then we have $\gamma(X) \sqsubseteq_{\mathcal{T}} E \sqsubseteq_{\mathcal{T}} \gamma(D_j)$, and thus $D_j \in S^\gamma_X$. By the definition of σ^γ, this implies $\sigma^\gamma(C_i) = \sigma^\gamma(X) \sqsubseteq_{\mathcal{T}} \sigma^\gamma(D_j)$.

   Both (i) and (ii) yield $\sigma^\gamma(C_1) \sqcap \ldots \sqcap \sigma^\gamma(C_n) \sqsubseteq_{\mathcal{T}} \sigma^\gamma(D_j)$.

2. In the second case, there are atoms $A_1, \ldots, A_k, B$ of $\mathcal{T}$ such that

   a) $A_1 \sqcap \ldots \sqcap A_k \sqsubseteq_{\mathcal{T}} B$,

   b) for every $\eta, 1 \leq \eta \leq k$, there is $i, 1 \leq i \leq n$, such that one of the following two cases holds:

      (i) C_i is a non-variable atom and $\gamma(C_i) \sqsubseteq^s_{\mathcal{T}} A_\eta$,

      (ii) $C_i = X$ is a variable and the top-level conjunction of $\gamma(X)$ contains an atom E such that $E \sqsubseteq^s_{\mathcal{T}} A_\eta$;

   c) $B \sqsubseteq^s_{\mathcal{T}} \gamma(D_j)$.

   In case (i) we have that either $C_i = A = A_\eta$ is a concept name, or both concept descriptions are existential restrictions $C_i = \exists r.C'_i$ and $A_\eta = \exists r.A'_\eta$ with $\gamma(C'_i) \sqsubseteq_{\mathcal{T}} A'_\eta$. In the first case, we have $\sigma^\gamma(C_i) = A = A_\eta$. In the second case, the case where C_i is ground is again trivial. Otherwise, we can apply induction since $C'_i, A'_\eta \in$ At, the role depth of $\sigma^\gamma(C'_i)$ is smaller than the one of $\sigma^\gamma(C_i)$, and the role depth of A'_η is not counted since it is ground. Thus, we have $\sigma^\gamma(C'_i) \sqsubseteq_{\mathcal{T}} A'_\eta$, which yields $\sigma^\gamma(C_i) \sqsubseteq_{\mathcal{T}} A_\eta$.

In case (ii), we again have $\gamma(X) \sqsubseteq_{\mathcal{T}} E \sqsubseteq_{\mathcal{T}} A_\eta$, and thus $A_\eta \in S^\gamma_X$. This yields $\sigma^\gamma(C_i) = \sigma^\gamma(X) \sqsubseteq_{\mathcal{T}} A_\eta$.

For similar reasons as before, we can again show that $B \sqsubseteq^s_{\mathcal{T}} \gamma(D_j)$ implies $B \sqsubseteq_{\mathcal{T}} \sigma^\gamma(D_j)$.

To sum up, we thus have also in this case $\sigma^\gamma(C_1) \sqcap \ldots \sqcap \sigma^\gamma(C_n) \sqsubseteq_{\mathcal{T}} A_1 \sqcap \ldots \sqcap A_k \sqsubseteq_{\mathcal{T}} B \sqsubseteq_{\mathcal{T}} \sigma^\gamma(D_j)$.

Hence, we have shown that, for all $j, 1 \leq j \leq m$, we have $\sigma^\gamma(C_1) \sqcap \ldots \sqcap \sigma^\gamma(C_n) \sqsubseteq_{\mathcal{T}} \sigma^\gamma(D_j)$, which yields $\sigma^\gamma(C_1) \sqcap \ldots \sqcap \sigma^\gamma(C_n) \sqsubseteq_{\mathcal{T}} \sigma^\gamma(D_1) \sqcap \ldots \sqcap \sigma^\gamma(D_m) = \sigma(D)$. The last identity holds since $D = Y$ and $S^\gamma_Y = \{D_1, \ldots, D_m\}$.

It remains to consider the case where D is a non-variable atom. But then we have

$$\gamma(C_1) \sqcap \ldots \sqcap \gamma(C_n) \sqsubseteq_{\mathcal{T}} \gamma(D),$$

and $\gamma(D)$ is an atom. As for $\gamma(D_j)$ above, we can use Lemma 6 to show that this implies $\sigma^\gamma(C_1) \sqcap \ldots \sqcap \sigma^\gamma(C_n) \sqsubseteq_{\mathcal{T}} \sigma^\gamma(D)$. □

6.3 Cycle-Restrictedness is Needed

We show that Theorem 23 does not hold for arbitrary general TBoxes. To this purpose, consider the general TBox $\mathcal{T} = \{B \sqsubseteq \exists s.D,\ D \sqsubseteq B\}$, which is not cycle-restricted, and the unification problem

$$\Gamma = \{A_1 \sqcap B \equiv^? Y_1,\ A_2 \sqcap B \equiv^? Y_2,\ \exists s.Y_1 \sqsubseteq^? X,\ \exists s.Y_2 \sqsubseteq^? X,\ X \sqsubseteq^? \exists s.X\}.$$

This problem has the unifier $\gamma := \{Y_1 \mapsto A_1 \sqcap B, Y_2 \mapsto A_2 \sqcap B, X \mapsto \exists s.B\}$. However, the induced assignment S^γ is cyclic since $\gamma(X) = \exists s.B \sqsubseteq_{\mathcal{T}} \exists s.\exists s.B = \gamma(\exists s.X)$ yields $\exists s.X \in S^\gamma_X$. Thus, γ does not induce a local unifier.

We claim that Γ actually does not have any local unifier w.r.t. $\mathcal{T}$. Assume to the contrary that σ is a local unifier of Γ w.r.t. $\mathcal{T}$. Then $\sigma(X)$ cannot be $\top$ since $\top \not\sqsubseteq_{\mathcal{T}} \exists s.\top$. Thus, $\sigma(X)$ must contain a top-level atom of the form $\sigma(E)$ for $E \in \mathrm{At}_{\mathsf{nv}}$. This atom cannot be $\sigma(\exists s.Y_i) \equiv_{\mathcal{T}} \exists s.(A_i \sqcap B)$ for $i \in \{1, 2\}$ since then $\sigma(\exists s.Y_j) \sqsubseteq_{\mathcal{T}} \sigma(E)$ for $j \in \{1, 2\} \setminus \{i\}$ would not hold, contradicting the assumption that σ solves $\exists s.Y_j \sqsubseteq^? X$ w.r.t. $\mathcal{T}$. Since local unifiers are induced by acyclic assignments, E cannot be $\sigma(\exists s.X)$, and thus E must be an atom of $\mathcal{T}$. However, none of the atoms $B, D, \exists s.D$ subsume $\exists s.(A_j \sqcap B)$ w.r.t. $\mathcal{T}$, again contradicting the assumption that σ solves $\exists s.Y_j \sqsubseteq^? X$ w.r.t. $\mathcal{T}$.

7 A Goal-Oriented Unification Algorithm

The algorithm presented in the previous section is not practical since it blindly guesses an acyclic assignment and only afterwards checks whether the guessed

assignment induces a unifier. In this section, we 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 produces a unifier. This goal-oriented algorithm in principle generalizes the goal-oriented algorithm for unification in $\mathcal{EL}$ w.r.t. the empty TBox introduced in [7], though the rules look quite different because here we consider unification problems that consist of subsumptions whereas in [7] we considered equivalences.

As in the previous section, we assume without loss of generality that the TBox $\mathcal{T}$ and the input unification problem Γ_0 are flat. Given $\mathcal{T}$ and Γ_0, the sets At of atoms of Γ_0 and $\mathcal{T}$ and $\mathrm{At}_{\mathsf{nv}}$ of non-variable atoms of Γ_0 and $\mathcal{T}$ are defined as above.

We will first consider unification w.r.t. the empty TBox and present a modified version of the algorithm in [7] to explain the basic principles at work. Afterwards, we will add rules to enable the algorithm to deal with a cycle-restricted TBox.

7.1 Unification with the Empty TBox Once More

Starting with Γ_0, the algorithm maintains a current unification problem Γ and a current acyclic assignment S, which initially assigns the empty set to all variables. In addition, for each subsumption in Γ it maintains the information on whether it is *solved* or not. Initially, all subsumptions of Γ_0 are unsolved, except those with a variable on the right-hand side. Rules are applied only to unsolved subsumptions. A (non-failing) application of a rule of our algorithm does the following:

- it solves exactly one unsolved subsumption,
- it may extend the current assignment S by adding elements of $\mathrm{At}_{\mathsf{nv}}$ to S_X for some variable X, and
- it may introduce new flat subsumptions built from elements of At.

Each rule application that extends S additionally *expands* Γ *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 Γ for every $A \in S_X$.

Subsumptions are only added by a rule application or by expansion if they are not already present in Γ. If a new subsumption is added to Γ, either by a rule application or by expansion of Γ, then it is initially designated unsolved, except if it has a variable on the right-hand side. Once a subsumption is in Γ, it will not

be removed. Likewise, if a subsumption in Γ is marked as solved, then it will not become unsolved later.

If a subsumption is marked as solved, this does not necessarily mean that it is indeed already solved by the substitution induced by the current assignment. Instead, 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 certain well-defined sense. A subsumption whose right-hand side is a variable is always marked as solved since the task of solving it 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 2), and several *nondeterministic* rules (see Figure 3). Eager rules are applied with higher priority than nondeterministic rules, and among the eager rules, Eager Ground Solving has the highest priority, then comes Eager Solving, and then Eager Extension.

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 by *Eager Ground Solving*, either holds, in which case any substitution solves it, or it does not, in which case it does not have a solution. In the case considered by *Eager Solving*, the substitution induced by the current assignment 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 substitution σ' satisfies $\sigma'(X) \equiv \sigma'(C_1) \sqcap \cdots \sqcap \sigma'(C_n)$. Thus, if S'_X contains D, then $\sigma'(X) \sqsubseteq \sigma'(D)$, and σ' solves the subsumption. Conversely, if σ' solves the subsumption, then $\sigma'(X) \sqsubseteq \sigma'(D)$, 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 Lemma 2. Assume that γ is induced by an acyclic assignment S. To satisfy the lemma, the atom $\gamma(D)$ must 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 a non-variable atom C_i, or of the form $\gamma(C)$ for $C \in S_{C_i}$ and a variable C_i. In the first case, either $\gamma(C_i) = A = \gamma(D)$ is a concept constant, in which case the subsumption is solved by one of the Solving rules, or $\gamma(C_i) = \exists r.C'$, $\gamma(D) = \exists r.D'$ and $C' \sqsubseteq D'$, which is covered by the *Decomposition* rule. In the second case, the atom C is either already in S_{C_i} or it can be put into S_{C_i} by an application of the *Extension* rule.

To sum up, the unification algorithm works as follows.

Algorithm 27. Let Γ_0 be a flat $\mathcal{EL}$-unification problem. We initialize $\Gamma := \Gamma_0$ and $S_X := \emptyset$ for all variables $X \in N_v$. While Γ contains an unsolved subsumption, do the following:

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 D does not occur on the left-hand side of $\mathfrak{s}$, 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 there is an index $i \in \{1, \ldots, n\}$ such that $C_i = D$ or $C_i = X \in N_v$ and $D \in S_X$.
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 an index $i \in \{1, \ldots, n\}$ 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, Γ is expanded w.r.t. X and $\mathfrak{s}$ is marked as *solved*.

Figure 2: The eager rules for Algorithm 27 w.r.t. the empty TBox.

Decomposition:

Condition: This rule applies to $\mathfrak{s} = C_1 \sqcap \cdots \sqcap C_n \sqsubseteq^? \exists s.D'$ if there is at least one index $i \in \{1, \ldots, n\}$ with $C_i = \exists s.C'$.
Action: Its application chooses such an index i, adds the subsumption $C' \sqsubseteq^? D'$ to Γ, expands it w.r.t. D' if D' is a variable, 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 at least one index $i \in \{1, \ldots, n\}$ with $C_i \in N_v$.
Action: Its application chooses such an index i and adds D to S_{C_i}. If this makes S cyclic, the rule application fails. Otherwise, Γ is expanded w.r.t. C_i and $\mathfrak{s}$ is marked as *solved*.

Figure 3: The nondeterministic rules for Algorithm 27 w.r.t. the empty TBox.

(1) **Eager rule application:** If some eager rules apply to an unsolved subsumption $\mathfrak{s}$ in Γ, apply the one with the highest priority and mark $\mathfrak{s}$ as solved. If the rule application fails, return "not unifiable".

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

Once all subsumptions in Γ are solved, return the substitution σ that is 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 of polynomially many data.

The algorithm presented so far is sound and complete for unification w.r.t. the empty TBox and always terminates [7]. We will now modify it to be able to deal with a cycle-restricted TBox.

7.2 Unification with a Cycle-restricted TBox

Let now $\mathcal{T}$ be a non-empty flat cycle-restricted TBox. The subsumptions in Γ can now be composed not only of non-variable atoms of Γ_0, but also of atoms of $\mathcal{T}$. Other than that, the only changes to Algorithm 27 are the modification of the Solving rules (see Figure 4) and the addition of the nondeterministic rules Mutation 1–4 (see Figure 5). The name "Mutation" is due to approaches to unification w.r.t. equational theories that use similar rules [20].

The modified Solving rules generalize the rules from Figure 2 to allow for ground subsumptions that hold because of $\mathcal{T}$. Such subsumptions can be checked in polynomial time [11]. For example, the new Eager Solving rule additionally considers the case that the current "ground part" of the subsumption already suffices to solve it.

The new Mutation rules allow the algorithm to solve a given subsumption according to the second case of Lemma 6. Each rule covers a different case: For example, Mutation 1 is applicable only to subsumptions with more that one conjunct on the left-hand side. It solves the subsumption by making sure that all conditions of the second alternative of Lemma 6 are satisfied. Whenever a structural subsumption $\gamma(E) \sqsubseteq^{\mathsf{s}}_{\mathcal{T}} \gamma(F)$ is required to hold for a (hypothetical) unifier γ of Γ, 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

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 $C_1 \sqcap \cdots \sqcap C_n \sqsubseteq_{\mathcal{T}} D$ 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 an index $i \in \{1, \ldots, n\}$ such that $C_i = D$ or $C_i = X \in N_v$ and $D \in S_X$, or
- D is ground and $\prod \mathcal{G} \sqsubseteq_{\mathcal{T}} D$ 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 an index $i \in \{1, \ldots, n\}$ 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, Γ is expanded w.r.t. X and $\mathfrak{s}$ is marked as *solved*.

Figure 4: The Eager rules of Algorithm 27 w.r.t. a cycle-restricted TBox $\mathcal{T}$.

conditions of the lemma.[8] To check the subsumption $A_1 \sqcap \cdots \sqcap A_k \sqsubseteq_{\mathcal{T}} B$, the rule again employs a polynomial-time subsumption check algorithm.

The other Mutation rules follow the same idea, but they implicitly apply one or more Decomposition or Eager Extension rules after mutation. This ensures that the newly generated subsumptions are "smaller" than the subsumption that triggers their introduction. One could also combine these rules into a single Mutation rule that covers all the cases. However, this rule would be very complicated and essentially do the same as Mutation 1–4.

Example 28. Consider the cycle-restricted TBox $\mathcal{T} = \{\exists r.A \sqsubseteq A,\ A \sqsubseteq \exists s.B\}$ and the subsumptions

$$\mathfrak{s}_1 = \exists r.Y \sqsubseteq^? X,\ \mathfrak{s}_2 = \exists s.Y \sqcap \exists r.X \sqsubseteq^? A,$$

$$\mathfrak{s}_3 = Z \sqsubseteq^? \exists r.X,\ \mathfrak{s}_4 = X \sqcap Y \sqsubseteq^? \exists s.Z.$$

Unification w.r.t. the empty TBox would fail since, e.g., the subsumption $\mathfrak{s}_3$

[8]Note that, for this direction of the lemma it is not important that this is a structural subsumption.

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 $\mathcal{T}$ such that $A_1 \sqcap \cdots \sqcap A_k \sqsubseteq_\mathcal{T} B$ 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 new subsumption $C_i \sqsubseteq^? A_\eta$ to Γ,
- it adds the subsumption $B \sqsubseteq^? D$ to Γ.

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 $\exists r.A_1, \ldots, \exists r.A_k$ of $\mathcal{T}$ such that $\exists r.A_1 \sqcap \cdots \sqcap \exists r.A_k \sqsubseteq_\mathcal{T} D$ holds.
Action: Its application chooses such atoms, adds $A_1, \ldots, A_k$ to S_X, expands Γ w.r.t. X, 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 $\exists r.A_1, \ldots, \exists r.A_k, \exists s.B$ of $\mathcal{T}$ such that $\exists r.A_1 \sqcap \cdots \sqcap \exists r.A_k \sqsubseteq_\mathcal{T} \exists s.B$ holds.
Action: Its application chooses such atoms, marks $\mathfrak{s}$ as *solved*, and generates the following subsumptions:

- it adds $A_1, \ldots, A_k$ to S_X and expands Γ w.r.t. X,
- it adds the subsumption $B \sqsubseteq^? Y$ to Γ and expands it w.r.t. Y.

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 $\exists s.B$ of $\mathcal{T}$ such that $C \sqsubseteq_\mathcal{T} \exists s.B$ holds.
Action: Its application chooses such an atom, adds the new subsumption $B \sqsubseteq^? Y$ to Γ, expands it w.r.t. Y, and marks $\mathfrak{s}$ as *solved*.

Figure 5: The nondeterministic rules of Algorithm 27 w.r.t. a cycle-restricted TBox $\mathcal{T}$.

cannot be solved by any of the original rules. However, we will demonstrate that the additional rules that are aware of the TBox help to solve the subsumptions.

Initially, the sets S_X, S_Y, and S_Z are empty. First, the algorithm solves $\mathfrak{s}_3$ by Eager Extension, resulting in the set $S_Z = \{\exists r.X\}$. No eager rules are applicable at this point, so we have to choose a nondeterministic rule to solve a subsumption. For example, $\mathfrak{s}_2$ can be solved by Mutation 1 since $\exists r.A \sqsubseteq_{\mathcal{T}} A$. This results in the new subsumptions $\mathfrak{s}_5 = \exists r.X \sqsubseteq^? \exists r.A$ and $\mathfrak{s}_6 = A \sqsubseteq^? A$. The latter is immediately solved by Eager Ground Solving, while the former can be solved by Decomposition, resulting in $\mathfrak{s}_7 = X \sqsubseteq^? A$. Eager Extension then yields the new assignment $S_X = \{A\}$. By expansion of $\mathfrak{s}_1$, this yields the new subsumption $\mathfrak{s}_8 = \exists r.Y \sqsubseteq^? A$. $\mathfrak{s}_8$ can be solved by Mutation 2 since $\exists r.A \sqsubseteq_{\mathcal{T}} A$ holds, resulting in $S_Y = \{A\}$. For the remaining subsumption $\mathfrak{s}_4$, there are several possibilities:

- Applying Extension to add $\exists s.Z$ to X would fail since this would make S cyclic. The resulting substitution σ would have to satisfy $\sigma(X) \sqsubseteq_{\mathcal{T}} \exists sr.\sigma(X)$, which is impossible by Lemma 11.
- Applying Extension to add $\exists s.Z$ to Y succeeds, resulting in $S_Y = \{A, \exists s.Z\}$. The final substitution σ then maps X to A, Z to $\exists r.A$, and Y to $A \sqcap \exists sr.A$.
- One way of applying Mutation 1 to $\mathfrak{s}_4$ would use $\exists r.A \sqsubseteq_{\mathcal{T}} \exists s.B$ to create the new subsumptions $\mathfrak{s}_9 = \exists s.B \sqsubseteq^? \exists s.Z$ and $\mathfrak{s}_{10} = X \sqsubseteq^? \exists r.A$. The second subsumption is solved by Eager Extension and then we have $S_X = \{A, \exists r.A\}$ and $\mathfrak{s}_{11} = \exists r.Y \sqsubseteq^? \exists r.A$ by expansion of $\mathfrak{s}_1$. The remaining subsumptions $\mathfrak{s}_9$ and $\mathfrak{s}_{11}$ can easily be solved by Decomposition and Eager Solving. The final substitution σ maps X to $A \sqcap \exists r.A \equiv_{\mathcal{T}} \exists r.A$, Y to A, and Z to $\exists r.(A \sqcap \exists r.A) \equiv_{\mathcal{T}} \exists rr.A$.
- There are other ways of applying Mutation 1 to $\mathfrak{s}_4$, which result in different substitutions.

It is easy to verify that all substitutions computed in this way are actually unifiers of the subsumptions $\mathfrak{s}_1$–$\mathfrak{s}_4$ w.r.t. $\mathcal{T}$.

We will now formally show that the algorithm is an NP-decision procedure that is sound and complete for $\mathcal{EL}$-unification w.r.t. cycle-restricted TBoxes.

7.3 Soundness

We will show that, if Algorithm 27 returns a substitution σ on input Γ_0, then σ is a unifier of Γ_0 w.r.t. $\mathcal{T}$. In the following, let S be the final assignment computed by Algorithm 27 on input Γ_0 and σ be the substitution induced by S. With $\widehat{\Gamma}$ we denote the final set of subsumptions computed by this run, i.e.,

the original subsumptions of Γ_0 together with the new ones generated by rule applications. To show that σ solves $\widehat{\Gamma}$, we use well-founded induction [9] on the following well-founded order $\succ$ on $\widehat{\Gamma}$.

Definition 29. Let $\mathfrak{s} = C_1 \sqcap \cdots \sqcap C_n \sqsubseteq^? C_{n+1} \in \widehat{\Gamma}$.

- $\mathfrak{s}$ is *small* if $n = 1$ and C_1 is ground or C_{n+1} is ground.
- We define $m(\mathfrak{s}) := (m_1(\mathfrak{s}), m_2(\mathfrak{s}), m_3(\mathfrak{s}))$, where
  - $m_1(\mathfrak{s}) := 0$ if $\mathfrak{s}$ is small, and $m_1(\mathfrak{s}) := 1$ otherwise.
  - $m_2(\mathfrak{s}) := X$ if $C_{n+1} = X$ or $C_{n+1} = \exists r.X$ for a variable X and some $r \in N_R$, and $m_2(\mathfrak{s}) := \bot$ otherwise.
  - $m_3(\mathfrak{s}) := \max\{\mathsf{rd}(\sigma(C_i)) \mid i \in \{1, \ldots, n+1\}\}$.
- The strict partial order $\succ$ on such triples is the lexicographic order, where the first and the third component are compared w.r.t. the normal order $>$ on natural numbers. The variables in the second component are compared w.r.t. the depends on relation induced by $\mathcal{T}_S$, i.e., X is larger than all variables it depends on, and $\bot$ is smaller than any variable.
- We extend $\succ$ to $\widehat{\Gamma}$ by setting $\mathfrak{s}_1 \succ \mathfrak{s}_2$ iff $m(\mathfrak{s}_1) \succ m(\mathfrak{s}_2)$.

As the lexicographic product of well-founded strict partial orders is again well-founded [9], $\succ$ is a well-founded strict partial order on $\widehat{\Gamma}$.

Lemma 30. *σ is an $\mathcal{EL}$-unifier of $\widehat{\Gamma}$ w.r.t. $\mathcal{T}$, and thus also of its subset Γ_0.*

Proof. Let $\mathfrak{s} \in \widehat{\Gamma}$ and assume that σ solves all subsumptions $\mathfrak{s}' \in \widehat{\Gamma}$ with $\mathfrak{s}' \prec \mathfrak{s}$. Consider the following cases for $\mathfrak{s}$.

- If $\mathfrak{s}$ has a *non-variable atom on the right-hand side*, then it was initially marked as unsolved and must have been solved by a successful rule application. We consider the rule that was applied.
  - Eager Ground Solving: Then $\mathfrak{s}$ is ground and holds under any substitution w.r.t. $\mathcal{T}$.
  - Eager Solving: Then $\mathfrak{s}$ is of the form $C_1 \sqcap \cdots \sqcap C_n \sqsubseteq^? D$ and either $\sigma(D)$ occurs on the top-level of $\sigma(C_1) \sqcap \cdots \sqcap \sigma(C_n)$ or D is ground and subsumes the conjunction of all ground atoms on the top-level of $\sigma(C_1) \sqcap \cdots \sqcap \sigma(C_n)$ w.r.t. $\mathcal{T}$. In both cases, σ solves the subsumption by monotonicity of $\sqsubseteq_{\mathcal{T}}$.
  - (Eager) Extension: Then $\mathfrak{s}$ is of the form $X \sqcap C_1 \sqcap \cdots \sqcap C_n \sqsubseteq^? D$ for a variable X and $D \in S_X$. By definition of σ, we have $\sigma(X) \sqsubseteq \sigma(D)$, and thus σ solves $\mathfrak{s}$.

– Decomposition: Then $\mathfrak{s}$ is of the form $C_1 \sqcap \cdots \sqcap C_n \sqsubseteq^? \exists s.D'$ with $C_i = \exists s.C'$ for some $i \in \{1, \ldots, n\}$ and we have $\mathfrak{s}' = C' \sqsubseteq^? D' \in \widehat{\Gamma}$. We will show that $\mathfrak{s} \succ \mathfrak{s}'$ holds. By induction, this implies that σ solves $\mathfrak{s}'$, and by Lemma 6 thus also $\mathfrak{s}$.

  To compare $m(\mathfrak{s})$ and $m(\mathfrak{s}')$, observe first that $m_2(\mathfrak{s}) = m_2(\mathfrak{s}')$ since either $\exists s.D'$ and D' contain the same variable or both are ground. We now make a case distinction based on $m_1(\mathfrak{s}')$.

  If $\mathfrak{s}'$ is small, then $\mathfrak{s}$ is either non-small, i.e., $m_1(\mathfrak{s}) > m_1(\mathfrak{s}')$, or small and of the form $\exists s.C' \sqsubseteq^? \exists s.D'$. In the second case, we have $m_1(\mathfrak{s}) = m_1(\mathfrak{s}')$ and $m_3(\mathfrak{s}) > m_3(\mathfrak{s}')$.

  If $\mathfrak{s}'$ is non-small, then both C' and D' are variables, and thus $\mathfrak{s}$ is also non-small, which yields $m_1(\mathfrak{s}) = m_1(\mathfrak{s}')$. Furthermore,

$$\begin{aligned} m_3(\mathfrak{s}) &\geq \max\{\mathsf{rd}(\sigma(\exists s.C')), \mathsf{rd}(\sigma(\exists s.D'))\} \\ &= \max\{\mathsf{rd}(\sigma(C')), \mathsf{rd}(\sigma(D'))\} + 1 \\ &> \max\{\mathsf{rd}(\sigma(C')), \mathsf{rd}(\sigma(D'))\} \\ &= m_3(\mathfrak{s}'). \end{aligned}$$

  In all cases we have shown $m(\mathfrak{s}) \succ m(\mathfrak{s}')$, i.e., $\mathfrak{s} \succ \mathfrak{s}'$.

– Mutation 1: Then $\mathfrak{s}$ is of the form $C_1 \sqcap \cdots \sqcap C_n \sqsubseteq^? D$ with $n > 1$ and there are atoms $A_1, \ldots, A_k, B$ of $\mathcal{T}$ with $A_1 \sqcap \cdots \sqcap A_k \sqsubseteq_\mathcal{T} B$. Furthermore, for every $\eta \in \{1, \ldots, k\}$ there is a subsumption $\mathfrak{s}_\eta = C_i \sqsubseteq^? A_\eta \in \widehat{\Gamma}$ for some $i \in \{1, \ldots, n\}$ and the subsumption $\mathfrak{s}' = B \sqsubseteq^? D$ is also in $\widehat{\Gamma}$.

  Since $\mathfrak{s}$ is not small and all the subsumptions $\mathfrak{s}_1, \ldots, \mathfrak{s}_k$, $\mathfrak{s}'$ are small, they are smaller than $\mathfrak{s}$ w.r.t. $\succ$. By induction, σ solves those small subsumptions and we have $\sigma(C_1) \sqcap \cdots \sqcap \sigma(C_n) \sqsubseteq_\mathcal{T} A_1 \sqcap \cdots \sqcap A_k \sqsubseteq_\mathcal{T} B \sqsubseteq_\mathcal{T} \sigma(D)$, i.e., σ solves $\mathfrak{s}$.

– Mutation 2: Then $\mathfrak{s}$ is of the form $\exists r.X \sqsubseteq^? D$, D is ground, and $\exists r.A_1 \sqcap \cdots \sqcap \exists r.A_k \sqsubseteq_\mathcal{T} D$ holds for atoms $\exists r.A_1, \ldots, \exists r.A_k$ of $\mathcal{T}$. Since $A_1, \ldots, A_k \in S_X$, we have $\sigma(\exists r.X) \sqsubseteq_\mathcal{T} \exists r.A_1 \sqcap \cdots \sqcap \exists r.A_k \sqsubseteq_\mathcal{T} D$.

– Mutation 3: Then $\mathfrak{s}$ is of the form $\exists r.X \sqsubseteq^? \exists s.Y$ and the subsumption $\exists r.A_1 \sqcap \cdots \sqcap \exists r.A_k \sqsubseteq_\mathcal{T} \exists s.B$ holds between atoms of $\mathcal{T}$. Furthermore, the subsumption $\mathfrak{s}' = B \sqsubseteq^? Y$ is in $\widehat{\Gamma}$. Since $\mathfrak{s}$ is non-small and $\mathfrak{s}'$ is small, by induction we have $B \sqsubseteq_\mathcal{T} \sigma(Y)$. Since $A_1, \ldots, A_k \in S_X$, we can conclude that $\sigma(\exists r.X) \sqsubseteq_\mathcal{T} \exists r.A_1 \sqcap \cdots \sqcap \exists r.A_k \sqsubseteq_\mathcal{T} \exists s.B \sqsubseteq_\mathcal{T} \sigma(\exists s.Y)$.

– Mutation 4: Then $\mathfrak{s}$ is of the form $C \sqsubseteq^? \exists s.Y$, C is ground, and $C \sqsubseteq_\mathcal{T} \exists s.B$ holds for an atom $\exists s.B$ of $\mathcal{T}$. Furthermore, the subsumption $\mathfrak{s}' = B \sqsubseteq^? Y$ is in $\widehat{\Gamma}$. Both $\mathfrak{s}$ and $\mathfrak{s}'$ are small and $m_2(\mathfrak{s}) = Y = m_2(\mathfrak{s}')$. Since B is a constant, i.e., has depth 0, and C has at most depth 1, we have $m_3(\mathfrak{s}) = \mathsf{rd}(\sigma(\exists s.Y)) > \mathsf{rd}(\sigma(Y)) = m_3(\mathfrak{s}')$, and thus $\mathfrak{s} \succ \mathfrak{s}'$.

By induction, $B \sqsubseteq_{\mathcal{T}} \sigma(Y)$ holds, which implies that $C \sqsubseteq_{\mathcal{T}} \exists s.B \sqsubseteq_{\mathcal{T}} \sigma(\exists s.Y)$.

- If $\mathfrak{s}$ has a *variable as its right-hand side*, then it is of the form $C_1 \sqcap \cdots \sqcap C_n \sqsubseteq^? X$ and for every $A \in S_X$ there is a subsumption $\mathfrak{s}_A = C_1 \sqcap \cdots \sqcap C_n \sqsubseteq^? A$ in $\widehat{\Gamma}$.

  If $\mathfrak{s}$ is small, then $n = 1$ and C_1 is ground, and thus the subsumptions $\mathfrak{s}_A$ are also small. Thus, we have $m_1(\mathfrak{s}) \geq m_1(\mathfrak{s}_A)$ for every $A \in S_X$. Furthermore, we have $m_2(\mathfrak{s}) > m_2(\mathfrak{s}_A)$ since either A is ground or contains a variable on which X depends. This yields $\mathfrak{s} \succ \mathfrak{s}_A$, and thus by induction all subsumptions $\mathfrak{s}_A$ are solved by σ. Hence, $\sigma(C_1) \sqcap \cdots \sqcap \sigma(C_n) \sqsubseteq_{\mathcal{T}} \sigma(A)$ for every $A \in S_X$, which implies that $\sigma(C_1) \sqcap \cdots \sqcap \sigma(C_n) \sqsubseteq_{\mathcal{T}} \sigma(X)$ by the definition of σ. This shows that σ solves $\mathfrak{s}$. □

7.4 Completeness

Assume that Γ_0 is unifiable w.r.t. $\mathcal{T}$ and let γ be a ground unifier of Γ_0 w.r.t. $\mathcal{T}$. We can use this unifier to guide the application of the nondeterministic rules such that Algorithm 27 does not fail. The following invariants for the current set of subsumptions Γ and the current assignment S will be maintained:

(I) γ is a unifier of Γ.

(II) For all $B \in S_X$ we have $\gamma(X) \sqsubseteq_{\mathcal{T}} \gamma(B)$.

Since S_X is initialized to $\emptyset$ for all variables $X \in N_v$ and Γ is initialized to Γ_0, these invariants are satisfied after the initialization of the algorithm.

The second invariant immediately rules out one possible cause of failure for the algorithm, namely that the current assignment may become cyclic.

Lemma 31. *If invariant (II) is satisfied, then the current assignment S is acyclic.*

Proof. Assume that S is cyclic. Then there are variables $X_1, \ldots, X_n$ such that $X_n = X_1$ and for every $i \in \{1, \ldots, n-1\}$ we have $X_i > X_{i+1}$, i.e., there is a non-variable atom $\exists r_i.X_{i+1} \in S_{X_i}$. By invariant (II), we have $\gamma(X_i) \sqsubseteq_{\mathcal{T}} \gamma(\exists r_i.X_{i+1})$ for every such i, and thus $\gamma(X_1) \sqsubseteq_{\mathcal{T}} \exists r_1 \ldots r_{n-1}.\gamma(X_1)$, which is impossible by Lemma 11. □

To prove completeness, we first show that expansion of Γ does not violate the invariants.

Lemma 32. *The invariants are maintained by the operation of expanding Γ.*

Proof. Since expansion does not change the current assignment S, we only have to show that invariant (I) still holds afterwards.

Consider a subsumption $\mathfrak{s} = C_1 \sqcap \cdots \sqcap C_n \sqsubseteq^? X$ for which a new subsumption $\mathfrak{s}' = C_1 \sqcap \cdots \sqcap C_n \sqsubseteq^? A$ is created with $A \in S_X$. By the invariants, $\mathfrak{s}$ is solved by γ and we have $\gamma(X) \sqsubseteq_{\mathcal{T}} \gamma(A)$. By transitivity of $\sqsubseteq_{\mathcal{T}}$, this implies that γ also solves $\mathfrak{s}'$. □

Another reason why Algorithm 27 might fail is that the application of an eager rule fails. This is ruled out by the following lemma.

Lemma 33. *The application of an eager rule never fails and maintains the invariants.*

Proof. Consider the application of Eager Ground Solving to an unsolved ground subsumption $\mathfrak{s} = C_1 \sqcap \cdots \sqcap C_n \sqsubseteq^? D \in \Gamma$. By invariant (I), $\mathfrak{s}$ is solved by γ, and thus the rule application does not fail. The invariants are not affected by this rule since neither Γ nor S are changed.

The application of Eager Solving cannot fail and does not affect the invariants.

Finally, consider the case that the Eager Extension rule is applied to an unsolved subsumption of the form $C_1 \sqcap \cdots \sqcap C_n \sqsubseteq^? D \in \Gamma$ with $C_i = X$ and $\{C_1, \ldots, C_n\} \setminus \{C_i\} \subseteq S_X$ for some $i \in \{1, \ldots, n\}$. By invariant (II), we know that $\gamma(C_1) \sqcap \cdots \sqcap \gamma(C_n) \equiv_{\mathcal{T}} \gamma(X)$ holds. By invariant (I), γ solves $\mathfrak{s}$, which implies $\gamma(X) \sqsubseteq_{\mathcal{T}} \gamma(D)$. Thus, invariant (II) still holds after adding D to S_X. By Lemma 31, S is acyclic and the rule application does not fail. Invariant (I) is not affected by this rule. □

So far, we have ruled out all causes of failure except for one: If no eager rules are applicable to subsumptions in Γ and there is still an unsolved subsumption left, then the algorithm has to find an applicable nondeterministic rule that does not fail.

Lemma 34. *Let $\mathfrak{s}$ be an unsolved subsumption of Γ to which no eager rule applies. Then there is a nondeterministic rule that can be successfully applied to $\mathfrak{s}$ while maintaining the invariants.*

Proof. $\mathfrak{s}$ must be of the form $C_1 \sqcap \cdots \sqcap C_n \sqsubseteq^? D$, where $C_1, \ldots, C_n$ are flat atoms or $\top$ and D is a flat non-variable atom of $\Gamma_0 \cup \mathcal{T}$.

By invariant (I), γ solves $\mathfrak{s}$, i.e., we have $\gamma(C_1) \sqcap \cdots \sqcap \gamma(C_n) \sqsubseteq_{\mathcal{T}} \gamma(D)$. By Lemma 6, one of the following alternatives holds:

1. There is an index $i \in \{1, \ldots, n\}$ such that $E \sqsubseteq^{\mathsf{s}}_{\mathcal{T}} \gamma(D)$ for a top-level atom E of $\gamma(C_i)$. We consider the following cases for C_i, which can obviously not be $\top$.

- If C_i is a concept name, then $C_i = E = D$ and Eager Solving is applicable to $\mathfrak{s}$, which contradicts the assumption.
- If $C_i = \exists r.C'$, then $\gamma(C_i) = \exists r.\gamma(C') = E$, and thus $D = \exists r.D'$ and $\gamma(C') \sqsubseteq_\mathcal{T} \gamma(D')$. This shows that the Decomposition rule can be successfully applied to $\mathfrak{s}$ and results in a new subsumption $C' \sqsubseteq^? D'$ that is solved by γ.
- If $C_i = X$ is a variable, then invariant (II) is preserved by adding D to S_X since $\gamma(X) \sqsubseteq E \sqsubseteq_\mathcal{T} \gamma(D)$. By Lemma 31, S stays acyclic, and thus we can successfully apply Extension to $\mathfrak{s}$.

2. There are atoms $A_1, \dots, A_k, B$ of $\mathcal{T}$ such that for all $\eta \in \{1, \dots, k\}$ there is $i \in \{1, \dots, n\}$ with $\gamma(C_i) \sqsubseteq_\mathcal{T} E \sqsubseteq^\mathsf{s}_\mathcal{T} A_\eta$ for a top-level atom E of $\gamma(C_i)$, $A_1 \sqcap \dots \sqcap A_k \sqsubseteq_\mathcal{T} B$, and $B \sqsubseteq^\mathsf{s}_\mathcal{T} \gamma(D)$. If $n > 1$, we can apply Mutation 1 in such a way that all created subsumptions are solved by γ.

   If $n = 1$, we distinguish several cases for C_1 and D:

   - If both C_1 and D are ground, then the Eager Ground Solving rule is applicable to $\mathfrak{s}$, which contradicts our assumption.
   - If $C_1 = X$ is a variable, then the Eager Extension rule is applicable to $\mathfrak{s}$, which again contradicts our assumption.
   - If $C_1 = \exists r.X$ for a variable X, then we have $A_\eta = \exists r.A'_\eta$ and $\gamma(X) \sqsubseteq_\mathcal{T} A'_\eta$ for every $\eta \in \{1, \dots, k\}$. Thus, we can add $A'_1, \dots, A'_k$ to S_X without violating invariant (II). If D is ground, we have $A_1 \sqcap \dots \sqcap A_k \sqsubseteq_\mathcal{T} B \sqsubseteq_\mathcal{T} D$, and thus we can successfully apply Mutation 2 to $\mathfrak{s}$. Otherwise, $D = \exists s.Y$ for a variable Y, which implies that $B = \exists s.B'$ and $B' \sqsubseteq_\mathcal{T} \gamma(Y)$. In this case, we can apply Mutation 3 to $\mathfrak{s}$ while preserving the invariants.
   - If C is ground and $D = \exists s.Y$ for a variable Y, then we have $C \sqsubseteq_\mathcal{T} A_1 \sqcap \dots \sqcap A_k \sqsubseteq_\mathcal{T} B = \exists s.B'$ and $B' \sqsubseteq_\mathcal{T} \gamma(Y)$. Thus, we can successfully apply Mutation 4 to $\mathfrak{s}$. □

As an immediate consequence of these lemmata we obtain that for any unifiable input problem Γ_0 there is a non-failing run of Algorithm 27 on Γ_0 during which the invariants (I) and (II) are satisfied. Together with the fact that any run of the algorithm terminates (see below), this shows completeness, i.e., whenever Γ_0 has a unifier w.r.t. $\mathcal{T}$, the algorithm computes one.

7.5 Termination and Complexity

Consider a run of Algorithm 27. We will show that any subsumption encountered during this run falls into one of the following categories:

1) subsumptions from Γ_0;

2) subsumptions created by expansion of Γ_0: these are of the form $C_1 \sqcap \cdots \sqcap C_n \sqsubseteq^? A$ for a subsumption $C_1 \sqcap \cdots \sqcap C_n \sqsubseteq^? X \in \Gamma_0$ and $A \in \mathrm{At}_{\mathsf{nv}}$;

3) subsumptions of the form $C \sqsubseteq^? D$ for $C, D \in \mathrm{At}$.

Of course, initially all subsumptions are of the form 1).

Lemma 35. *If the subsumptions in Γ are of the forms 1)–3), then this is still the case after one rule application.*

Proof. If a rule directly creates a subsumption, then it is of the form 3) (Decomposition rule and Mutation rules 1, 3, and 4).

If the current assignment S is extended by a rule, then a subsumption $\mathfrak{s} \in \Gamma$ may be expanded w.r.t. S. If $\mathfrak{s}$ is of the form 1), then the resulting subsumptions will be of the form 2). If it is of the form 3), then the resulting subsumptions will also be of this form.

Other operations executed by a rule application do not affect the invariant. □

This leads us to the following conclusion.

Lemma 36. *Every run of Algorithm 27 on input Γ_0 terminates in time polynomial in the size of $\Gamma_0 \cup \mathcal{T}$.*

Proof. There are only polynomially many subsumptions of the forms 1)–3). Since every rule application solves at least one subsumption, by Lemma 35 the algorithm can apply at most polynomially many rules.

Checking applicability of the rules is thus done only polynomially often and only executes a subset of the following operations, all of which can be done in time polynomial in the size of $\Gamma_0 \cup \mathcal{T}$:

- Guessing a set of atoms of $\mathcal{T}$.
- Checking a subsumption between ground atoms of $\Gamma_0 \cup \mathcal{T}$ (see [11]).
- Checking whether the atoms of the subsumption are of a specific form, e.g., ground or existential restrictions.
- Looking up whether an atom is in the current assignment S_X of a variable X that occurs on the left-hand side of the subsumption.

Additionally, the application of a rule can execute the following polynomial operations:

- Guessing polynomially many atoms from the left-hand side of a subsumption.

- Adding polynomially many subsumptions to Γ.
- Adding polynomially many atoms to the current assignment.
- Checking whether the current assignment is cyclic: It has to be checked whether there is a sequence $X_1, \ldots, X_n$ of variables such that $X_1 = X_n$ and X_i directly depends on X_{i+1} for each $i \in \{1, \ldots, n-1\}$ w.r.t. the current assignment S. This can be expressed as a reachability problem in a graph of polynomial size, and thus can be solved in polynomial time. □

This concludes the analysis of Algorithm 27 and the proof of our main result.

Theorem 37. *Algorithm 27 is an* NP*-decision procedure for unifiability in* $\mathcal{EL}$ *w.r.t. cycle-restricted TBoxes.*

This again shows that $\mathcal{EL}$-unification w.r.t. cycle-restricted TBoxes is in NP. Furthermore, it also yields a different proof of the locality result of Theorem 23: If Γ is unifiable w.r.t. $\mathcal{T}$, by Section 7.4, Algorithm 27 has a successful run on input Γ. By Section 7.3, there is a unifier of Γ w.r.t. $\mathcal{T}$ that is induced by an acyclic assignment.

8 Conclusions

We have shown that unification in $\mathcal{EL}$ stays in NP in the presence of a cycle-restricted TBox, by giving a brute-force NP-algorithm that tries to guess a local unifier. On the one hand, this algorithm is interesting since it provides a quite simple, self-contained proof for the complexity upper-bound. On the other hand, it can be seen as a first step towards a polytime reduction of the unification problem to a propositional satisfiability problem (SAT), similar to the reduction in [6]. However, we would also need to encode subsumption w.r.t. a cycle-restricted TBox into the SAT problem in order to avoid generating solutions of the SAT problem that do not correspond to solutions of the unification problem. While Lemma 6 can probably be used to obtain such an encoding it is not clear how to do this in polynomial time. In fact, there may exist exponentially many subsumptions $A_1 \sqcap \cdots \sqcap A_k \sqsubseteq_{\mathcal{T}} B$ for atoms $A_1, \ldots, A_k, B$ of $\mathcal{T}$. The goal-oriented algorithm already has the property that it only generates substitutions that are unifiers, but a practical implementation still needs to be optimized by including some of the ideas underlying modern SAT solvers.

On the theoretical side, the main topic for future research is to consider unification w.r.t. unrestricted general TBoxes. 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, the idea would 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 [22].

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 TBox $\mathcal{T}$ together with the cyclic TBox $\mathcal{T}_S$ induced by S satisfies $C \sqsubseteq_{\mathcal{T} \cup \mathcal{T}_S} D$ for all subsumptions $C \sqsubseteq^? D$ in Γ_0.

References

[1] Grigoris Antoniou and Athanasios Kehagias. A note on the refinement of ontologies. *International Journal of Intelligent Systems*, 15:623–632, 2000.

[2] Franz Baader. Terminological cycles in a description logic with existential restrictions. LTCS-Report 02-02, TU Dresden, 2002. See http://lat.inf.tu-dresden.de/research/reports.html.

[3] Franz Baader. Terminological cycles in a description logic with existential restrictions. In Georg Gottlob and Toby Walsh, editors, *Proc. of the 18th Int. Joint Conf. on Artificial Intelligence (IJCAI'03)*, pages 325–330. Morgan Kaufmann, 2003.

[4] 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.

[5] 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.

[6] 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.

[7] Franz Baader and Barbara Morawska. Unification in the description logic $\mathcal{EL}$. *Logical Methods in Computer Science*, 6(3), 2010. Special Issue: 20th Int. Conf. on Rewriting Techniques and Applications (RTA'09).

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

[9] Franz Baader and Tobias Nipkow. *Term Rewriting and All That*. Cambridge University Press, United Kingdom, 1998.

[10] Franz Baader and Wayne Snyder. Unification theory. In John Alan Robinson and Andrei Voronkov, editors, *Handbook of Automated Reasoning*, pages 445–532. The MIT Press, 2001.

[11] 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.

[12] Sebastian Brandt. Reasoning in $\mathcal{ELH}$ w.r.t. general concept inclusion axioms. LTCS-Report 04-03, Chair for Automata Theory, Institute for Theoretical Computer Science, Dresden University of Technology, Germany, 2004. See http://lat.inf.tu-dresden.de/research/reports.html.

[13] James R. Campbell, Alejandro Lopez Osornio, Fernan de Quiros, Daniel Luna, and Guillermo Reynoso. Semantic interoperability and SNOMED CT: A case study in clinical problem lists. In K.A. Kuhn, J.R. Warren, and T.-Y. Leong, editors, *Proc. of the 12th World Congress on Health (Medical) Informatics (MEDINFO 2007)*, pages 2401–2402. IOS Press, 2007.

[14] Anatoli Degtyarev, Yuri Gurevich, Paliath Narendran, Margus Veanes, and Andrei Voronkov. Decidability and complexity of simultaneous rigid E-unification with one variable and related results. *Theoretical Computer Science*, 243(1-2):167–184, 2000.

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

[16] 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.

[17] Jean Goubault. Rigid $\vec{E}$-unifiability is DEXPTIME-complete. In *Proc. of the 9th Annual IEEE Symp. on Logic in Computer Science (LICS'94)*, pages 498–506. IEEE Computer Society Press, 1994.

[18] Yuri Gurevich and Andrei Voronkov. Monadic simultaneous rigid E-unification. *Theoretical Computer Science*, 222(1-2):133–152, 1999.

[19] Martin Hofmann. Proof-theoretic approach to description-logic. In *Proc. of the 20th Annual IEEE Symp. on Logic in Computer Science (LICS'05)*, pages 229–237, 2005.

[20] Jean-Pierre Jouannaud and Claude Kirchner. Solving equations in abstract algebras: A rule-based survey of unification. In Jean-Louis Lassez and Gordon Plotkin, editors, *Computational Logic: Essays in honor of Alan Robinson*, chapter 8, pages 257–321. The MIT Press, 1991.

[21] Carsten Lutz and Frank Wolter. Deciding inseparability and conservative extensions in the description logic $\mathcal{EL}$. *Journal of Symbolic Computation*, 45(2):194–228, 2010.

[22] Barbara Morawska. General E-unification with eager variable elimination and a nice cycle rule. *J. of Automated Reasoning*, 39(1):77–106, 2007.

[23] Viorica Sofronie-Stokkermans. 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.

[24] Margus Veanes. *On Simultaneous Rigid E-Unification*. PhD thesis, Uppsala University, 1997.