Hybrid Unification in the Description Logic $\mathcal{EL}$

Franz Baader, Oliver Fernández Gil, and Barbara Morawska*

Theoretical Computer Science, TU Dresden, Germany
{baader,morawska}@tcs.inf.tu-dresden.de,
fernandez@informatik.uni-leipzig.de

Abstract. Unification in Description Logics (DLs) has been proposed as an inference service that can, for example, be used to detect redundancies in ontologies. For the DL $\mathcal{EL}$, which is used to define several large biomedical ontologies, unification is NP-complete. However, the unification algorithms for $\mathcal{EL}$ developed until recently could not deal with ontologies containing general concept inclusions (GCIs). In a series of recent papers we have made some progress towards addressing this problem, but the ontologies the developed unification algorithms can deal with need to satisfy a certain cycle restriction. In the present paper, we follow a different approach. Instead of restricting the input ontologies, we generalize the notion of unifiers to so-called hybrid unifiers. Whereas classical unifiers can be viewed as acyclic TBoxes, hybrid unifiers are cyclic TBoxes, which are interpreted together with the ontology of the input using a hybrid semantics that combines fixpoint and descriptive semantics. We show that hybrid unification in $\mathcal{EL}$ is NP-complete and introduce a goal-oriented algorithm for computing hybrid unifiers.

1 Introduction

Description logics [5] are a well-investigated family of logic-based knowledge representation formalisms. They can be used to represent the relevant concepts of an application domain using concept descriptions, which are built from concept names and role names using certain concept constructors. 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 [10]. 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] From a semantic point of view, concept names and concept descriptions represent sets of individuals, whereas role names represent binary relations between individuals. For example, using the concept names Head_injury and Severe, and the role names finding and status, we can describe the concept of a *patient with severe head injury* as

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

* Supported by DFG under grant BA 1122/14-2.

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

P. Fontaine, C. Ringeissen, and R.A. Schmidt (Eds.): FroCoS 2013, LNAI 8152, pp. 295–310, 2013.

In a DL ontology, one can use *concept definitions* to introduce abbreviations for concept descriptions. For example, we could use the definition $\mathsf{Head\_injury} \equiv \mathsf{Injury} \sqcap \exists \mathsf{finding\_site.Head}$ to define $\mathsf{Head\_injury}$ as an injury that is located at the head. More generally, GCIs can be used to require that certain inclusions hold in all models of the ontology. For example,

$$\exists \mathsf{finding}.\exists \mathsf{status.Severe} \sqsubseteq \exists \mathsf{status.Emergency} \tag{2}$$

is a GCI that says that a severe finding entails an emergency status.

Knowledge representation systems based on DLs provide their users with various inference services that allow them to deduce implicit knowledge from the explicitly represented knowledge. For instance, the subsumption algorithm allows one to determine subconcept-superconcept relationships. For example, the concept description (1) is subsumed by (i.e., is a subconcept of) the concept description $\exists \mathsf{finding}.\exists \mathsf{status.Severe}$. With respect to the GCI (2), it is thus also subsumed by $\exists \mathsf{status.Emergency}$, i.e., in all models of this GCI, patients with severe head injury have an emergency status.

Unification in DLs has been proposed in [9] 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 describes the concept of a *patient with severe head injury* using the concept description (1), whereas another one represents it as

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

These two concept descriptions are not equivalent, but they are nevertheless meant to represent the same concept. They can obviously be made equivalent by introducing definitions for the concept names $\mathsf{Head\_injury}$ and $\mathsf{Severe\_injury}$: if we define $\mathsf{Head\_injury} \equiv \mathsf{Injury} \sqcap \exists \mathsf{finding\_site.Head}$ and $\mathsf{Severe\_injury} \equiv \mathsf{Injury} \sqcap \exists \mathsf{status.Severe}$, then the two concept descriptions (1) and (3) are equivalent w.r.t. these definitions. If such definitions exist, we say that the descriptions are unifiable, and call the TBox consisting of these definitions a *unifier*. More precisely, it is required that this TBox is acyclic, i.e., there are no cyclic dependencies between the definitions.

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

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

instead of (3). The descriptions (1) and (4) are not unifiable without additional GCIs, but they are unifiable, with the same unifier as above, if the GCI (2) is present in a background ontology.

In [7], we were able to show that unification in the DL $\mathcal{EL}$ (without background ontology) is NP-complete. In addition to a brute-force "guess and then test" NP-algorithm [7], we have also developed a goal-oriented unification algorithm for $\mathcal{EL}$, in which nondeterministic decisions are only made if they are triggered by "unsolved parts" of the unification problem [8]. In [8] it was also shown that

these two 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. For more general GCIs, such a simple solution is no longer possible.

In [3], we extended the brute-force "guess and then test" NP-algorithm from [7] to the case of GCIs. Unfortunately, the algorithm is complete only for ontologies that satisfy a certain restriction on cycles, which, however, does not prevent all cycles. For example, the cyclic GCI $\exists\mathsf{child}.\mathsf{Human} \sqsubseteq \mathsf{Human}$ satisfies this restriction, whereas the cyclic GCI $\mathsf{Human} \sqsubseteq \exists\mathsf{parent}.\mathsf{Human}$ does not. In [4], we introduced a more practical, goal-oriented unification algorithm that can also deal with role hierarchies and transitive roles, but still needs the ontology (now consisting of GCIs and role axioms) to be cycle-restricted. At the moment, it is not clear how similar brute-force or goal-oriented algorithms could be obtained for the general case without cycle-restriction.

In this paper, we follow another line of attack on this problem. Instead of restricting the input ontology, we allow cyclic TBoxes to be used as unifiers. Subsumption w.r.t. cyclic TBoxes in $\mathcal{EL}$ has been investigated in detail in [1]. In addition to the classical descriptive semantics, it also makes sense to use *greatest fixpoint semantics (gfp-semantics)* for such TBoxes. For example, w.r.t. this semantics, the definition $X \equiv \exists\mathsf{parent}.X$ describes exactly those domain elements that are the origin of an infinite parent-chain, whereas descriptive semantics would also allow the empty set to be an interpretation of X, even if there are infinite parent-chains. *Hybrid semantics* deals with the case where a TBox interpreted with gfp-semantics is combined with GCIs that are interpreted with descriptive semantics [11,14,13]. Its introduction was originally motivated by the fact that the least common subsumer (lcs) w.r.t. a set of GCIs interpreted with descriptive semantics need not exist. For example, w.r.t. the GCIs

$$\mathsf{Human} \sqsubseteq \exists\mathsf{parent}.\mathsf{Human} \text{ and } \mathsf{Horse} \sqsubseteq \exists\mathsf{parent}.\mathsf{Horse}, \tag{5}$$

there is no least concept description (w.r.t. subsumption) that subsumes both Human and Horse. What elements of these two concepts have in common is that they are the origin of an infinite parent-chain, and thus the concept X with definition $X \equiv \exists\mathsf{parent}.X$ is their lcs, if we interpret this definition with gfp-semantics, but the GCIs (5) still with descriptive semantics. A hybrid unifier is a cyclic TBox that, together with the background ontology consisting of GCIs, entails the unification problem w.r.t. hybrid semantics. We will show that hybrid unification in $\mathcal{EL}$, i.e., the problem of testing whether a hybrid unifier exists, is NP-complete. In addition, we will introduce a goal-oriented algorithm for computing hybrid unifiers. The proofs, which can be found in [6], are based on the proof system for hybrid subsumption introduced in [14,13].

2 The Description Logic $\mathcal{EL}$

The expressiveness of a DL is determined both by the formalism for describing concepts (the concept description language) and the terminological formalism,

which can be used to state additional constraints on the interpretation of concepts in a so-called ontology.

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

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

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

Classical Ontologies and Subsumption. A *concept definition* is an expression of the form $X \equiv C$ where X is a concept name and C is a concept description, and a *general concept inclusion* (GCI) is an expression of the form $C \sqsubseteq D$, where C, D are concept descriptions. An interpretation $\mathcal{I}$ is a *model* of this concept definition (this GCI) if it satisfies $X^{\mathcal{I}} = C^{\mathcal{I}}$ ($C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$). This semantics for GCIs and concept definitions is usually called *descriptive semantics*.

A *TBox* is a finite set $\mathcal{T}$ of concept definitions that does not contain multiple definitions, i.e., $\{X \equiv C, X \equiv D\} \subseteq \mathcal{T}$ implies $C = D$. Note that we do *not* prohibit cyclic dependencies among the concept definitions in a TBox, i.e., when defining a concept X we may (directly or indirectly) refer to X. An *acyclic TBox* is a TBox without cyclic dependencies. An *ontology* is a finite set of GCIs. The interpretation $\mathcal{I}$ is a *model* of a TBox (ontology) iff it is a model of all concept definitions (GCIs) contained in it.

A concept description C is *subsumed* by a concept description D w.r.t. an ontology $\mathcal{O}$ (written $C \sqsubseteq_{\mathcal{O}} D$) if every model of $\mathcal{O}$ is also a model of the GCI $C \sqsubseteq D$. We say that C is *equivalent* to D w.r.t. $\mathcal{O}$ ($C \equiv_{\mathcal{O}} D$) if $C \sqsubseteq_{\mathcal{O}} D$ and $D \sqsubseteq_{\mathcal{O}} C$. As shown in [10], subsumption w.r.t. $\mathcal{EL}$-ontologies is decidable in polynomial time.

Note that TBoxes can be seen as special kinds of ontologies since concept definitions $X \equiv C$ can of course be expressed by GCIs $X \sqsubseteq C, C \sqsubseteq X$. Thus, the above definition of subsumption also applies to TBoxes. However, in our hybrid ontologies we will interpret concept definitions using greatest fixpoint semantics rather than descriptive semantics.

Hybrid Ontologies. We assume in the following that the set of concept names N_C is partitioned into the set of *primitive concepts* N_{prim} and the set of *defined concepts* N_{def}. In a hybrid TBox, concept names occurring on the left-hand side of a concept definition are required to come from the set N_{def}, whereas GCIs must not contain concept names from N_{def}.

Definition 1 (Hybrid $\mathcal{EL}$-ontologies). *A* hybrid $\mathcal{EL}$-ontology *is a pair* $(\mathcal{O}, \mathcal{T})$*, where* $\mathcal{O}$ *is an* $\mathcal{EL}$*-ontology containing only concept names from* N_{prim}*, and* $\mathcal{T}$ *is a (possibly cyclic)* $\mathcal{EL}$*-TBox such that* $X \equiv C \in \mathcal{T}$ *for some concept description* C *iff* $X \in N_{def}$*.*

The idea underlying the definition of hybrid ontologies is the following: $\mathcal{O}$ can be used to constrain the interpretation of the primitive concepts and roles, whereas $\mathcal{T}$ tells us how to interpret the defined concepts occurring in it, once the interpretation of the primitive concepts and roles is fixed.

A *primitive interpretation* $\mathcal{J}$ is defined like an interpretation, with the only difference that it does not provide an interpretation for the defined concepts. A primitive interpretation can thus interpret concept descriptions built over N_{prim} and N_R, but it cannot interpret concept descriptions containing elements of N_{def}. Given a primitive interpretation $\mathcal{J}$, we say that the (full) interpretation $\mathcal{I}$ is *based on* $\mathcal{J}$ if it has the same domain as $\mathcal{J}$ and its interpretation function coincides with $\mathcal{J}$ on N_{prim} and N_R.

Given two interpretations $\mathcal{I}_1$ and $\mathcal{I}_2$ based on the same primitive interpretation $\mathcal{J}$, we define $\mathcal{I}_1 \preceq_{\mathcal{J}} \mathcal{I}_2$ iff $X^{\mathcal{I}_1} \subseteq X^{\mathcal{I}_2}$ for all $X \in N_{def}$.

It is easy to see that the relation $\preceq_{\mathcal{J}}$ is a partial order on the set of interpretations based on $\mathcal{J}$. In [1] the following was shown: given an $\mathcal{EL}$-TBox $\mathcal{T}$ and a primitive interpretation $\mathcal{J}$, there exists a unique model $\mathcal{I}$ of $\mathcal{T}$ such that

- $\mathcal{I}$ is based on $\mathcal{J}$;
- $\mathcal{I}' \preceq_{\mathcal{J}} \mathcal{I}$ for all models $\mathcal{I}'$ of $\mathcal{T}$ that are based on $\mathcal{J}$.

We call such a model $\mathcal{I}$ a *gfp-model* of $\mathcal{T}$.

Definition 2 (Semantics of hybrid $\mathcal{EL}$-ontologies). *The interpretation* $\mathcal{I}$ *is a* hybrid model *of the hybrid* $\mathcal{EL}$*-ontology* $(\mathcal{O}, \mathcal{T})$ *iff* $\mathcal{I}$ *is a gfp-model of* $\mathcal{T}$ *and the primitive interpretation* $\mathcal{J}$ *it is based on is a model of* $\mathcal{O}$*.*

It is well-known that gfp-semantics coincides with descriptive semantics for acyclic TBoxes. Thus, if $\mathcal{T}$ is actually acyclic, then $\mathcal{I}$ is a hybrid model of $(\mathcal{O}, \mathcal{T})$ according to the semantics introduced in Definition 2 iff it is a model of $\mathcal{T} \cup \mathcal{O}$ w.r.t. descriptive semantics, i.e., iff $\mathcal{I}$ is a model of every GCI in $\mathcal{O}$ and of every concept definition in $\mathcal{T}$.

Subsumption w.r.t. Hybrid $\mathcal{EL}$-Ontologies. Let $(\mathcal{O}, \mathcal{T})$ be a hybrid $\mathcal{EL}$-ontology and C, D $\mathcal{EL}$-concept descriptions. Then C *is subsumed by* D *w.r.t.* $(\mathcal{O}, \mathcal{T})$ (written $C \sqsubseteq_{gfp,\mathcal{O},\mathcal{T}} D$) iff every hybrid model of $(\mathcal{O}, \mathcal{T})$ is also a model of the GCI $C \sqsubseteq D$. As shown in [11,14,13], subsumption w.r.t. hybrid $\mathcal{EL}$-ontologies is also decidable in polynomial time.

Here, we sketch the proof-theoretic approach for deciding subsumption from [14,13] since our algorithms for hybrid unification in $\mathcal{EL}$ are based on it. The proof calculus is parametrized with a hybrid $\mathcal{EL}$-ontology $(\mathcal{O}, \mathcal{T})$ and a finite set of GCIs Δ for which we want to decide subsumption. A *sequent for* $(\mathcal{O}, \mathcal{T})$ *and* Δ is of the form $C \sqsubseteq_n D$, where C, D are sub-descriptions of concept descriptions

$$C \sqsubseteq_n C \quad \text{(Refl)} \qquad C \sqsubseteq_n \top \quad \text{(Top)} \qquad C \sqsubseteq_0 D \quad \text{(Start)}$$

$$\frac{C \sqsubseteq_n E}{C \sqcap D \sqsubseteq_n E} \quad \text{(AndL1)} \qquad \frac{D \sqsubseteq_n E}{C \sqcap D \sqsubseteq_n E} \quad \text{(AndL2)} \qquad \frac{C \sqsubseteq_n D \quad C \sqsubseteq_n E}{C \sqsubseteq_n D \sqcap E} \quad \text{(AndR)}$$

$$\frac{C \sqsubseteq_n D}{\exists r.C \sqsubseteq_n \exists r.D} \quad \text{(Ex)}$$

$$\frac{C \sqsubseteq_n D}{X \sqsubseteq_n D} \quad \text{(DefL)} \quad \text{for } X \equiv C \in \mathcal{T} \qquad \frac{D \sqsubseteq_n C}{D \sqsubseteq_{n+1} X} \quad \text{(DefR)} \quad \text{for } X \equiv C \in \mathcal{T} \qquad \frac{C \sqsubseteq_n E \quad F \sqsubseteq_n D}{C \sqsubseteq_n D} \quad \text{(GCI)} \quad \text{for } E \sqsubseteq F \in \mathcal{O}$$

Fig. 1. The calculus $\mathrm{HC}(\mathcal{O}, \mathcal{T}, \Delta)$

occurring in $\mathcal{O}$, $\mathcal{T}$, and Δ, and $n \geq 0$. If $(\mathcal{O}, \mathcal{T})$ and Δ are clear from the context, we will sometimes simply say sequent without specifying $(\mathcal{O}, \mathcal{T})$ and Δ explicitly.

The rules of the **H**ybrid $\mathcal{EL}$-ontology **C**alculus $\mathrm{HC}(\mathcal{O}, \mathcal{T}, \Delta)$ are depicted in Fig. 1. Again, if $(\mathcal{O}, \mathcal{T})$ and Δ are clear from the context, we will sometimes dispense with specifying them explicitly and just talk about the calculus HC. The rules of this calculus can be used to derive new sequents from sequents that have already been derived. For example, the sequents in the first row of the figure can always be derived without any prerequisites, using the rules (Refl), (Top), and (Start), respectively. Using the rule (AndR), the sequent $C \sqsubseteq_n D \sqcap E$ can be derived in case both $C \sqsubseteq_n D$ and $C \sqsubseteq_n E$ have already been derived. Note that the rule Start applies only for $n = 0$. Also note that, in the rule (DefR), the index is incremented when going from the prerequisite to the consequent.

A derivation in $\mathrm{HC}(\mathcal{O}, \mathcal{T}, \Delta)$ can be represented in an obvious way by a proof tree whose nodes are sequents: a proof tree for $C \sqsubseteq_n D$ has this sequent as its root, instances of the rules Refl, Top, and Start as leaves, and each parent-child relation corresponds to an instance of a rule of HC other than Refl, Top, and Start (see [14,13] for more details)

Definition 3. *Let C, D be sub-descriptions of concept descriptions occurring in $\mathcal{O}, \mathcal{T}$, and Δ. Then we say that $C \sqsubseteq_\infty D$* can be derived *in $\mathrm{HC}(\mathcal{O}, \mathcal{T}, \Delta)$ if all sequents $C \sqsubseteq_n D$ for $n \geq 0$ can be derived using the rules of $\mathrm{HC}(\mathcal{O}, \mathcal{T}, \Delta)$.*

The calculus HC is sound and complete for subsumption w.r.t. hybrid $\mathcal{EL}$-ontologies in the following sense.

Theorem 4 (Soundness and Completeness of HC). *Let $(\mathcal{O}, \mathcal{T})$ be a hybrid $\mathcal{EL}$-TBox, Δ a finite set of GCIs, and C, D sub-descriptions of concept descriptions occurring in $\mathcal{O}, \mathcal{T}$, and Δ. Then $C \sqsubseteq_{gfp,\mathcal{O},\mathcal{T}} D$ iff $C \sqsubseteq_\infty D$ can be derived in $\mathrm{HC}(\mathcal{O}, \mathcal{T}, \Delta)$.*

In [13], soundness and completeness of HC is actually formulated for a restricted setting where Δ is empty and C, D are elements of N_{def} that occur as left-hand sides in $\mathcal{T}$. It is, however, easy to see that the proof given in [13] generalizes to the above theorem.

For $n \in \mathbb{N} \cup \{\infty\}$, we collect the GCIs $C \sqsubseteq D$ such that $C \sqsubseteq_n D$ is derivable in $\mathrm{HC}(\mathcal{O}, \mathcal{T}, \Delta)$ in the set $\mathcal{D}_n(\mathcal{O}, \mathcal{T}, \Delta)$. Obviously, $\mathcal{D}_0(\mathcal{O}, \mathcal{T}, \Delta)$ consists of all GCIs built from sub-descriptions of concept descriptions occurring in $\mathcal{O}, \mathcal{T}$, and Δ, and it is not hard to show that $\mathcal{D}_{n+1}(\mathcal{O}, \mathcal{T}, \Delta) \subseteq \mathcal{D}_n(\mathcal{O}, \mathcal{T}, \Delta)$ holds for all $n \geq 0$ [14,13]. Thus, to compute $\mathcal{D}_\infty(\mathcal{O}, \mathcal{T}, \Delta)$, one can start with $\mathcal{D}_0(\mathcal{O}, \mathcal{T}, \Delta)$, and then compute $\mathcal{D}_1(\mathcal{O}, \mathcal{T}, \Delta), \mathcal{D}_2(\mathcal{O}, \mathcal{T}, \Delta), \ldots$, until $\mathcal{D}_{m+1}(\mathcal{O}, \mathcal{T}, \Delta) = \mathcal{D}_m(\mathcal{O}, \mathcal{T}, \Delta)$ holds for some $m \geq 0$, and thus $\mathcal{D}_m(\mathcal{O}, \mathcal{T}, \Delta) = \mathcal{D}_\infty(\mathcal{O}, \mathcal{T}, \Delta)$. Since the cardinality of the set of sub-descriptions is polynomial in the size of the input $\mathcal{O}, \mathcal{T}$, and Δ, the computation of each set $\mathcal{D}_n(\mathcal{O}, \mathcal{T}, \Delta)$ can be done in polynomial time, and we can be sure that only polynomially many such sets need to be computed until an m with $\mathcal{D}_{m+1}(\mathcal{O}, \mathcal{T}, \Delta) = \mathcal{D}_m(\mathcal{O}, \mathcal{T}, \Delta)$ is reached. This shows that the calculus $\mathrm{HC}(\mathcal{O}, \mathcal{T}, \Delta)$ indeed yields a polynomial-time subsumption algorithm (see [14,13] for details).

3 Hybrid Unification in $\mathcal{EL}$

We will first introduce the new notion of hybrid unification and then relate it to the notion of unification in $\mathcal{EL}$ w.r.t. background ontologies considered in [3,4].

Definition 5. *Let $\mathcal{O}$ be an $\mathcal{EL}$-ontology containing only concept names from N_{prim}. An* $\mathcal{EL}$-unification problem *w.r.t. $\mathcal{O}$ is a finite set of GCIs $\Gamma = \{C_1 \sqsubseteq D_1, \ldots, C_n \sqsubseteq D_n\}$ (which may also contain concept names from N_{def}). The TBox $\mathcal{T}$ is a* hybrid unifier *of Γ w.r.t. $\mathcal{O}$ if $(\mathcal{O}, \mathcal{T})$ is a hybrid $\mathcal{EL}$-ontology that entails all the GCIs in Γ, i.e. , $C_1 \sqsubseteq_{gfp,\mathcal{O},\mathcal{T}} D_1, \ldots, C_n \sqsubseteq_{gfp,\mathcal{O},\mathcal{T}} D_n$. We call such a TBox $\mathcal{T}$ a* classical unifier *of Γ w.r.t. $\mathcal{O}$ if it is acyclic.*

It is easy to see that the notion of a classical unifier indeed corresponds to the notion of a unifier introduced in [3,4]. In fact, N_{prim} and N_{def} respectively correspond to the sets of concept constants and concept variables in previous papers on unification in DLs. Using acyclic TBoxes rather than substitutions as unifiers is also not a relevant difference. As explained in [2], by unfolding concept definitions, the acyclic TBox $\mathcal{T}$ can be transformed into a substitution $\sigma_\mathcal{T}$ such that $C_i \sqsubseteq_{\mathcal{T} \cup \mathcal{O}} D_i$ iff $\sigma_\mathcal{T}(C_i) \sqsubseteq_\mathcal{O} \sigma_\mathcal{T}(D_i)$. Conversely, replacements $X \mapsto E$ of a substitution σ can be expressed as concept definitions $X \equiv E$ in a corresponding acyclic TBox. In contrast, hybrid unifiers cannot be translated into substitutions since the unfolding process would not terminate for a cyclic TBox.

Obviously, any classical unifier is a hybrid unifier, but the converse need not hold. The following is an example of an $\mathcal{EL}$-unification problem w.r.t. a background ontology that has a hybrid unifier, but no classical unifier.

Example 6. Let $\mathcal{O}$ be the ontology consisting of the GCIs (5), and

$$\Gamma := \{\mathsf{Human} \sqsubseteq X, \mathsf{Horse} \sqsubseteq X, X \sqsubseteq \exists \mathsf{parent}.X\},$$

where $X \in N_{def}$ and $\mathsf{Human}, \mathsf{Horse} \in N_{prim}$. Intuitively, this unification problem asks for a concept such that all horses and humans belong to this concept and every element of it has a parent also belonging to it. It is easy to see that $\mathcal{T} := \{X \equiv \exists \mathsf{parent}.X\}$ is a hybrid unifier of Γ w.r.t. $\mathcal{O}$. In fact, we have already mentioned in the introduction that X is then the lcs of Human and Horse, and obviously the hybrid ontology $(\mathcal{O}, \mathcal{T})$ also entails the third GCI in Γ. It is also not hard to show that this unification problem does not have a classical unifier, basically for the same reasons that Human and Horse do not have an $\mathcal{EL}$-concept description as lcs (see [6] for details).

Flat Unification Problems. To simplify the technical development, it is convenient to normalize the unification problem appropriately. To introduce this normal form, we need the notion of an atom. An *atom* is a concept name or an existential restriction. Obviously, every $\mathcal{EL}$-concept description C is a finite conjunction of atoms, where $\top$ is considered to be the empty conjunction. An atom is called *flat* if it is a concept name or an existential restriction of the form $\exists r.A$ for a concept name A.

The GCI $C \sqsubseteq D$ is called *flat* if C is a conjunction of $n \geq 0$ flat atoms and D is a flat atom. The unification problem Γ w.r.t. the ontology $\mathcal{O}$ is called *flat* if both Γ and $\mathcal{O}$ consist of flat GCIs.

Given a unification problem Γ w.r.t. an ontology $\mathcal{O}$, we can compute in polynomial time (see [6]) a flat ontology $\mathcal{O}'$ and a flat unification problem Γ' such that Γ has a (hybrid or classical) unifier w.r.t. $\mathcal{O}$ iff Γ' has a (hybrid or classical) unifier w.r.t. $\mathcal{O}'$. For this reason, we will assume in the following that all unification problems are flat.

Local Unifiers. The main reason why $\mathcal{EL}$-unification without background ontologies is in NP is that any unification problem that has a unifier also has a local unifier. For classical unification w.r.t. background ontologies this is only true if the background ontology is cycle-restricted.

Given a flat unification problem Γ w.r.t. an ontology $\mathcal{O}$, we denote by At the set of atoms occurring as sub-descriptions in GCIs in Γ or $\mathcal{O}$. The set of *non-variable atoms* is defined by $\mathrm{At}_{\mathsf{nv}} := \mathrm{At} \setminus N_{def}$. Though the elements of $\mathrm{At}_{\mathsf{nv}}$ cannot be defined concepts, they may contain defined concepts if they are of the form $\exists r.X$ for some role r and a concept name $X \in N_{def}$.

In order to define local unifiers, we consider assignments ζ of subsets ζ_X of $\mathrm{At}_{\mathsf{nv}}$ to defined concepts $X \in N_{def}$. Such an assignment induces a TBox

$$T_\zeta := \{X \equiv \prod_{D \in \zeta_X} D \mid X \in N_{def}\}.$$

We call such a TBox *local*. The (hybrid or classical) unifier $\mathcal{T}$ of Γ w.r.t. $\mathcal{O}$ is called *local unifier* if $\mathcal{T}$ is local, i.e., there is an assignment ζ such that $\mathcal{T} = T_\zeta$.

As shown in [3], there are unification problems that have a classical unifier, but no local classical unifier.

Example 7. Let $\mathcal{O} = \{B \sqsubseteq \exists s.D,\ D \sqsubseteq B\}$ and consider the unification problem

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

where $A_1, A_2, B \in N_{prim}$ and $X, Y_1, Y_2 \in N_{def}$. This problem has the classical unifier $\mathcal{T} := \{Y_1 \equiv A_1 \sqcap B, Y_2 \equiv A_2 \sqcap B, X \equiv \exists s.B\}$, which is not local since it uses the atom $\exists s.B$. As shown in [3], Γ actually does not have a local classical unifier w.r.t. $\mathcal{O}$. However, it is easy to see that $\mathcal{T} := \{Y_1 \equiv A_1 \sqcap B, Y_2 \equiv A_2 \sqcap B, X \equiv \exists s.X\}$ is a local hybrid unifier of $\mathcal{T}$. In fact, gfp-semantics applied to $\mathcal{T}$ ensures that X consists of exactly those domain elements that are the origin of an infinite s-chain, and $\mathcal{O}$ ensures that any element of B (and thus also of $\exists s.B$) is the origin of an infinite s-chain.

To overcome the problem of missing local unifiers, the notion of a cycle-restricted ontology was introduced in [3]: the $\mathcal{EL}$-ontology $\mathcal{O}$ is called *cycle-restricted* if there is no nonempty sequence $r_1, \ldots, r_n$ of role names and $\mathcal{EL}$-concept description C such that $C \sqsubseteq_{\mathcal{O}} \exists r_1. \cdots \exists r_n.C$. Note that the ontology $\mathcal{O}$ of Example 7 is not cycle-restricted since $B \sqsubseteq_{\mathcal{O}} \exists s.B$.

The main technical result shown in [3] is that any $\mathcal{EL}$-unification problem Γ that has a classical unifier w.r.t. the cycle-restricted ontology $\mathcal{O}$ also has a local classical unifier. This yields the following brute-force algorithm for classical $\mathcal{EL}$-unification w.r.t. cycle-restricted ontologies: first guess an acyclic local TBox $\mathcal{T}$, and then check whether $\mathcal{T}$ is indeed a unifier of Γ w.r.t. $\mathcal{O}$. As shown in [3], this algorithm runs in nondeterministic polynomial time. NP-hardness follows from the fact that already classical unification in $\mathcal{EL}$ w.r.t. the empty ontology is NP-hard [7].

4 Hybrid $\mathcal{EL}$-Unification is NP-Complete

The fact that hybrid $\mathcal{EL}$-unification w.r.t. arbitrary $\mathcal{EL}$-ontologies is in NP is an easy consequence of the following proposition.

Proposition 8. *Consider a flat $\mathcal{EL}$-unification problem Γ w.r.t. an $\mathcal{EL}$-ontology $\mathcal{O}$. If Γ has a hybrid unifier w.r.t. $\mathcal{O}$ then it has a* local *hybrid unifier w.r.t. $\mathcal{O}$.*

In fact, the NP-algorithm simply guesses a local TBox and then checks (using the polynomial-time algorithm for hybrid subsumption) whether it is a hybrid unifier.

To prove the proposition, we assume that $\mathcal{T}$ is a hybrid unifier of Γ w.r.t. $\mathcal{O}$. We use this unifier to define an assignment $\zeta^{\mathcal{T}}$ as follows:

$$\zeta^{\mathcal{T}}_X := \{D \in \mathrm{At}_{\mathsf{nv}} \mid X \sqsubseteq_{gfp,\mathcal{O},\mathcal{T}} D\}.$$

Let $\mathcal{T}'$ be the TBox induced by this assignment. To show that $\mathcal{T}'$ is indeed a hybrid unifier of Γ w.r.t. $\mathcal{O}$, we consider the set of GCIs

$$\Delta := \{C_1 \sqcap \ldots \sqcap C_m \sqsubseteq D \mid C_1, \ldots, C_m, D \in \mathrm{At}\},$$

and prove that, for any GCI $C_1 \sqcap \ldots \sqcap C_m \sqsubseteq D \in \Delta$, derivability of $C_1 \sqcap \ldots \sqcap C_m \sqsubseteq_\infty D$ in $\mathrm{HC}(\mathcal{O}, \mathcal{T}, \Delta)$ implies derivability of $C_1 \sqcap \ldots \sqcap C_m \sqsubseteq_\infty D$ also in $\mathrm{HC}(\mathcal{O}, \mathcal{T}', \Delta)$. Soundness and completeness of HC, together with the facts that $\Gamma \subseteq \Delta$ and $\mathcal{T}$ is a hybrid unifier of Γ w.r.t. $\mathcal{O}$, then imply that $\mathcal{T}'$ is also a hybrid unifier of Γ w.r.t. $\mathcal{O}$. Thus, to complete the proof of Proposition 8, it is enough to prove the following lemma.

Lemma 9. *Let $C_1 \sqcap \ldots \sqcap C_m \sqsubseteq D \in \Delta$. If $C_1 \sqcap \ldots \sqcap C_m \sqsubseteq_\infty D$ is derivable in $\mathrm{HC}(\mathcal{O}, \mathcal{T}, \Delta)$, then $C_1 \sqcap \ldots \sqcap C_m \sqsubseteq_n D$ is derivable in $\mathrm{HC}(\mathcal{O}, \mathcal{T}', \Delta)$ for all $n \geq 0$.*

Proof. We prove derivability of $C_1 \sqcap \ldots \sqcap C_m \sqsubseteq_n D$ in $\mathrm{HC}(\mathcal{O}, \mathcal{T}', \Delta)$ by induction on n. The *base case* is trivial due to the rule (Start).

Induction Step: We assume that the statement of the lemma holds for $n-1$, and show that it then also holds for n. Let ℓ be such that $\mathcal{D}_\ell(\mathcal{O}, \mathcal{T}, \Delta) = \mathcal{D}_\infty(\mathcal{O}, \mathcal{T}, \Delta)$. We know that there exists a proof tree $\mathcal{P}$ for $C_1 \sqcap \ldots \sqcap C_m \sqsubseteq_\ell D$ in $\mathrm{HC}(\mathcal{O}, \mathcal{T}, \Delta)$. Consider the subtree of $\mathcal{P}$ that is obtained from it by cutting branches at the nodes obtained by an application of one of the rules (DefL) or (DefR). The tree obtained this way contains only sequents with index ℓ and has as its leaves

- instances of the rules (Refl), (Top), or (Start),
- consequences $E_1 \sqsubseteq_\ell E_2$ of instances of the rules (DefL) or (DefR).

In order to show that $C_1 \sqcap \ldots \sqcap C_m \sqsubseteq_n D$ is derivable in $\mathrm{HC}(\mathcal{O}, \mathcal{T}', \Delta)$, it is sufficient to show that, for leaves $E_1 \sqsubseteq_\ell E_2$ of the second kind, $E_1 \sqsubseteq_n E_2$ is derivable in $\mathrm{HC}(\mathcal{O}, \mathcal{T}', \Delta)$ (see [6] for details).

First, assume that $E_1 \sqsubseteq_\ell E_2$ was obtained by an application of (DefR). Then $E_2 \in N_{def}$. Assume that $\zeta^{\mathcal{T}}_{E_2} = \{F_1, \ldots, F_q\}$. By the definition of $\zeta^{\mathcal{T}}$, we have $E_2 \sqsubseteq_{gfp,\mathcal{O},\mathcal{T}} F_i$ for all $i, 1 \leq i \leq q$. In addition, by our choice of ℓ, derivability of $E_1 \sqsubseteq_\ell E_2$ in $\mathrm{HC}(\mathcal{O}, \mathcal{T}, \Delta)$ (using the subtree of $\mathcal{P}$ with this node as root) yields $E_1 \sqsubseteq_{gfp,\mathcal{O},\mathcal{T}} E_2$, and thus $E_1 \sqsubseteq_{gfp,\mathcal{O},\mathcal{T}} F_i$ for all $i, 1 \leq i \leq q$. Consequently, $E_1 \sqsubseteq_\infty F_i$ is derivable in $\mathrm{HC}(\mathcal{O}, \mathcal{T}, \Delta)$ for all $i, 1 \leq i \leq q$. Since E_1 is a conjunction of elements of At and $F_1, \ldots, F_q \in \mathrm{At}$, induction yields that $E_1 \sqsubseteq_{n-1} F_i$ is derivable in $\mathrm{HC}(\mathcal{O}, \mathcal{T}', \Delta)$ for all $i, 1 \leq i \leq q$. Performing $q-1$ applications of (AndR) thus allows us to derive $E_1 \sqsubseteq_{n-1} F_1 \sqcap \ldots \sqcap F_q$ in $\mathrm{HC}(\mathcal{O}, \mathcal{T}', \Delta)$. Since $\mathcal{T}'$ contains the definition $E_2 \equiv F_1 \sqcap \ldots \sqcap F_q$, an application of (DefR) shows that $E_1 \sqsubseteq_n E_2$ is derivable in $\mathrm{HC}(\mathcal{O}, \mathcal{T}', \Delta)$.

Second, assume that $E_1 \sqsubseteq_\ell E_2$ was obtained by an application of (DefL). Then $E_1 \in N_{def}$ and $E_2 = F_1 \sqcap \ldots \sqcap F_m$ for elements $F_1, \ldots, F_m$ of At. By our choice of ℓ we have $E_1 \sqsubseteq_{gfp,\mathcal{O},\mathcal{T}} E_2$, and thus $E_1 \sqsubseteq_{gfp,\mathcal{O},\mathcal{T}} F_i$ for all $i, 1 \leq i \leq q$. It is sufficient to show, for all $i, 1 \leq i \leq q$, that $E_1 \sqsubseteq_n F_i$ is derivable in $\mathrm{HC}(\mathcal{O}, \mathcal{T}', \Delta)$ since $q-1$ applications of (AndR) then yield derivability of $E_1 \sqsubseteq_n E_2$ in $\mathrm{HC}(\mathcal{O}, \mathcal{T}', \Delta)$.

If F_i does not belong to N_{def}, then it is an element of $\mathrm{At}_{\mathsf{nv}}$. The definition of $\zeta^{\mathcal{T}}$ thus yields $F_i \in \zeta^{\mathcal{T}}_{E_1}$. Consequently, F_i occurs as a conjunct on the right-hand side of the definition of E_1 in $\mathcal{T}'$. This implies $E_1 \sqsubseteq_{gfp,\mathcal{O},\mathcal{T}'} F_i$, and thus $E_1 \sqsubseteq_n F_i$ is derivable in $\mathrm{HC}(\mathcal{O}, \mathcal{T}', \Delta)$.

If $F_i \in N_{def}$, then $E_1 \sqsubseteq_{gfp,\mathcal{O},\mathcal{T}} F_i$ implies that $\zeta^{\mathcal{T}}_{F_i} \subseteq \zeta^{\mathcal{T}}_{E_1}$. Consequently, every conjunct on the right-hand side of the definition of F_i in $\mathcal{T}'$ is also a conjunct on the right-hand side of the definition of E_1 in $\mathcal{T}'$. This implies $E_1 \sqsubseteq_{gfp,\mathcal{O},\mathcal{T}'} F_i$, and thus $E_1 \sqsubseteq_n F_i$ is derivable in $\mathrm{HC}(\mathcal{O}, \mathcal{T}', \Delta)$. □

This finishes the proof of Proposition 8, and thus shows that hybrid $\mathcal{EL}$-unification w.r.t. arbitrary $\mathcal{EL}$-ontologies is in NP. NP-hardness does *not* follow directly from NP-hardness of classical $\mathcal{EL}$-unification. In fact, as we have seen in Example 6, an $\mathcal{EL}$-unification problem that does not have a classical unifier may well have a hybrid unifier. Instead, we reduce $\mathcal{EL}$-matching modulo equivalence to hybrid $\mathcal{EL}$-unification.

Using the notions introduced in this paper, $\mathcal{EL}$-matching modulo equivalence can be defined as follows. An *$\mathcal{EL}$-matching problem modulo equivalence* is an $\mathcal{EL}$-unification problem of the form $\{C \sqsubseteq D, D \sqsubseteq C\}$ such that D does not contain elements of N_{def}. A *matcher* of such a problem is a classical unifier of it. As shown in [12], testing whether a matching problem modulo equivalence has a matcher or not is an NP-complete problem. Thus, NP-hardness of hybrid $\mathcal{EL}$-unification w.r.t. $\mathcal{EL}$-ontologies is an immediate consequence of the following lemma, whose (non-trivial) proof can be found in [6].

Lemma 10. *If an $\mathcal{EL}$-matching problem modulo equivalence has a hybrid unifier w.r.t. the empty ontology, then it also has a matcher.*

To sum up, we have thus determine the exact worst-case complexity of hybrid $\mathcal{EL}$-unification.

Theorem 11. *The problem of testing whether an $\mathcal{EL}$-unification problem w.r.t. an arbitrary $\mathcal{EL}$-ontology has a hybrid unifier or not is* NP*-complete.*

5 A Goal-Oriented Algorithm for Hybrid $\mathcal{EL}$-Unification

The brute-force algorithm is not practical since it blindly guesses a local TBox and only afterwards checks whether the guessed TBox is a hybrid unifier. We now introduce a more goal-oriented unification algorithm, in which nondeterministic decisions are only made if they are triggered by "unsolved parts" of the unification problem. In addition, failure due to wrong guesses can be detected early. Any non-failing run of the algorithm produces a hybrid unifier, i.e., there is no need for checking whether the TBox computed by this run really is a hybrid unifier. This goal-oriented algorithm is based on ideas similar to the ones used in the algorithm for classical unification in $\mathcal{EL}$ w.r.t. cycle-restricted ontologies in [4]. However, it differs from the previous algorithm in several respects.

First, it is based on the proof calculus HC rather than on a structural characterization of subsumption, as employed in [4]. Basically, to solve the unification problem Γ w.r.t. the ontology $\mathcal{O}$, the rules of the algorithm try to build, for each GCI $C \sqsubseteq D \in \Gamma$, a proof tree for the sequent $C \sqsubseteq_\ell D$ while simultaneously generating the hybrid unifier $\mathcal{T}$ by adding non-variable atoms to an assignment

ζ inducing $\mathcal{T}$. The index ℓ of the sequent is chosen *large enough*, i.e., such that derivability of $C \sqsubseteq_\ell D$ implies derivability of $C \sqsubseteq_\infty D$. In [6] it is shown how an appropriate number ℓ of polynomial size can be computed from the size of the input Γ and $\mathcal{O}$.

Second, to avoid nonterminating runs of the algorithm, a *blocking mechanism* needs to be employed. This mechanism prevents cyclic dependencies between sequents where the derivability of one sequents depends on the derivability of another sequent and vice versa. This problem did not occur in the algorithm for classical unification in [4] due to the fact that, for classical unification, the generation of a cyclic assignment causes the run to fail. For hybrid unification, cyclic assignments may lead to valid hybrid unifiers. In order to realize blocking, we need to keep track of dependencies between sequents. For this reason, we work with *p-sequents* rather than sequents.

We assume without loss of generality that the input unification problem Γ w.r.t. the input ontology $\mathcal{O}$ is flat. Given $\mathcal{O}$ and Γ, the sets At and $\mathrm{At}_{\mathsf{nv}}$ are defined as above.

Definition 12. *A* flat sequent for Γ and $\mathcal{O}$ *is of the form* $C_1 \sqcap \ldots \sqcap C_m \sqsubseteq_n D$ *where* $C_1, \ldots, C_m \in \mathrm{At}, D \in \mathrm{At} \cup \{\top\}$, $m \geq 0$, *and* $0 \leq n \leq \ell$. *This sequent is called* ground *if no element of* N_{def} *occurs in it. A* p-sequents for Γ and $\mathcal{O}$ *is a pair* $(C \sqsubseteq_n D, P)$ *such that* $\{C \sqsubseteq_n D\} \cup P$ *is a finite set of flat sequents for* Γ *and* $\mathcal{O}$.

Intuitively, the p-sequent $(C \sqsubseteq_n D, P)$ says that we need to find a proof tree for $C \sqsubseteq_n D$, and that the proof trees for all the elements of P must contain this proof tree, i.e., the derivations of the elements of P depend on the derivation of $C \sqsubseteq_n D$.

Starting with the initial set of p-sequents

$$\Gamma_p^{(0)} := \{(C \sqsubseteq_\ell D, \emptyset) \mid C \sqsubseteq D \in \Gamma\}$$

the algorithm maintains a current set of p-sequents Γ_p and a current assignment ζ, which initially assigns the empty set to all $X \in N_{def}$. In addition, for each p-sequent in Γ_p it maintains the information on whether it is *solved* or not. Initially, all p-sequents are unsolved, except those with a defined concept on the right-hand side of its first component.[2] Rules are applied only to unsolved p-sequents. A (non-failing) rule application does the following:

- it solves exactly one unsolved p-sequent,
- it may extend the current assignment ζ, and
- it may add new p-sequents to Γ_p, which are marked unsolved unless their first component has a defined concept on the right-hand side.

Adding a new p-sequent is realized through the blocking procedure. This procedures checks whether the new sequent introduces cyclic derivability obligations

[2] Such p-sequents are dealt with by expansion rather than by applying a rule (see below).

Eager Axiom Solving:

Condition: This rule applies to $(\mathfrak{s}, P)$, if $\mathfrak{s}$ is of the form $C_1 \sqcap \ldots \sqcap C_m \sqsubseteq_0 D$ or $C_1 \sqcap \ldots \sqcap C_m \sqsubseteq_n \top$.
Action: Its application marks $(\mathfrak{s}, P)$ as *solved*.

Eager Ground Solving:

Condition: This rule applies to $(\mathfrak{s}, P)$ with $\mathfrak{s} = C_1 \sqcap \ldots \sqcap C_m \sqsubseteq_n D$, if $\mathfrak{s}$ is ground.
Action: If $C_1 \sqcap \ldots \sqcap C_m \sqsubseteq_{\mathcal{T}} D$ does not hold, the rule application fails. Otherwise, $(\mathfrak{s}, P)$ is marked as *solved*.

Eager Solving:

Condition: This rule applies to $(\mathfrak{s}, P)$ with $\mathfrak{s} = C_1 \sqcap \ldots \sqcap C_m \sqsubseteq_n D$, if there is an index $i \in \{1, \ldots, m\}$ such that $C_i = D$ or $C_i = X \in N_{def}$ and $D \in \zeta_X$.
Action: The application marks $(\mathfrak{s}, P)$ as *solved*.

Fig. 2. The eager rules of hybrid unification

(in which case it fails) and whether the sequent to be added already exists (in which case it re-uses the existing sequent, but updates the dependency information). Only if these two cases do not apply does it add the new sequent. To be more precise, given a set of p-sequents Γ_p and a p-sequents $(C \sqsubseteq_n D, P)$, the procedure *blocking* applied to this input does the following:

B1: If the sequent $C \sqsubseteq_n D$ belongs to P, then blocking *fails*.
B2: Otherwise, if there is a p-sequent of the form $(C \sqsubseteq_n D, P')$ in Γ_p, then do the following:
- Extend the second component of this sequent to $P' \cup P$.
- For each p-sequent $(\_, P'')$ in Γ_p such that $C \sqsubseteq_n D$ is in P'', extend the second component to $P'' \cup P$,

B3: Otherwise, add $(C \sqsubseteq_n D, P)$ to Γ_p.

Each rule application that extends ζ_X additionally *expands* Γ_p *w.r.t.* X as follows: every p-sequent of the form $(C_1 \sqcap \cdots \sqcap C_n \sqsubseteq_n X, P)$ is *expanded* by applying blocking to $(C_1 \sqcap \cdots \sqcap C_n \sqsubseteq_{n-1} D, \emptyset)$ and Γ_p for every $D \in \zeta_X$. Since the second components of the p-sequents provided as inputs for blocking are empty, blocking cannot fail during expansion. Note that expansion basically corresponds to an application of the rule (DefR) of HC together with an appropriate number of applications of (AndR).

If a p-sequent $\mathfrak{p}$ is marked as solved, this does not mean that a proof tree for its first component $\mathfrak{s}$ has already been constructed (w.r.t. $\mathcal{O}$ and the TBox induced by the current assignment). It may be the case that the task of constructing the proof tree for $\mathfrak{s}$ was deferred to constructing a proof tree for the first component $\mathfrak{s}'$ of a "smaller" p-sequent. The proof tree for $\mathfrak{s}'$ is then part of the proof tree for $\mathfrak{s}$, and thus $\mathfrak{s}$ needs to be added to the second component of $\mathfrak{p}'$.

The rules of the algorithm consist of three *eager* rules, which are deterministic (see Figure 2), and three *nondeterministic* rules (see Figure 3). Eager rules are applied with higher priority than nondeterministic rules. Among the eager rules,

Decomposition:

Condition: This rule applies to $(\mathfrak{s}, P)$ with $\mathfrak{s} = C_1 \sqcap \ldots \sqcap C_m \sqsubseteq_n \exists s.D'$, if there is a $C_i = \exists s.C'$ such that *blocking* does not fail if applied to $(C' \sqsubseteq_n D', P \cup \{\mathfrak{s}\})$ and Γ_p. **Action:** Its application chooses such an index i and applies *blocking* to $(C' \sqsubseteq_n D', P \cup \{\mathfrak{s}\})$ and Γ_p. Once *blocking* was applied, it expands Γ_p w.r.t. D' if $D' \in N_{def}$, and marks $(\mathfrak{s}, P)$ as *solved*.

Extension:

Condition: This rule applies to $(\mathfrak{s}, P)$ with $\mathfrak{s} = C_1 \sqcap \ldots \sqcap C_m \sqsubseteq_n D$ if there is at least one $i \in \{1, \ldots, m\}$ with $C_i \in N_{def}$. **Action:** Its application chooses such an index i and adds D to ζ_{C_i}. Γ_p is expanded w.r.t. C_i and $(\mathfrak{s}, P)$ is marked as *solved*.

Mutation:

Condition: This rule applies to $(\mathfrak{s}, P)$ with $\mathfrak{s} = C_1 \sqcap \ldots \sqcap C_m \sqsubseteq_n D$, if there is a GCI $E_1 \sqcap \ldots \sqcap E_k \sqsubseteq F$ in $\mathcal{O}$ and a set $S \subseteq \{1, \ldots, m\}$ such that *blocking* does not fail if applied to Γ_p and each of the p-sequents $(\bigsqcap_{j \in S} C_j \sqsubseteq_n E_1, P \cup \{\mathfrak{s}\}), \ldots, (\bigsqcap_{j \in S} C_j \sqsubseteq_n E_k, P \cup \{\mathfrak{s}\})$, and $(F \sqsubseteq_n D, P \cup \{\mathfrak{s}\})$. **Action:** Its application chooses such a GCI $E_1 \sqcap \ldots \sqcap E_k \sqsubseteq F$ and a set $S \subseteq \{1, \ldots, m\}$. It applies *blocking* to Γ_p and each of the p-sequents $(\bigsqcap_{j \in S} C_j \sqsubseteq_n E_1, P \cup \{\mathfrak{s}\}), \ldots, (\bigsqcap_{j \in S} C_j \sqsubseteq_n E_k, P \cup \{\mathfrak{s}\})$, and $(F \sqsubseteq_n D, P \cup \{\mathfrak{s}\})$. Once *blocking* was applied, $(\mathfrak{s}, P)$ is marked as *solved*.

Fig. 3. The nondeterministic rules of hybrid unification

Eager Axiom Solving has the highest priority, then comes Eager Ground Solving, and then Eager Solving.

Algorithm 13. Let Γ w.r.t. $\mathcal{O}$ be a flat $\mathcal{EL}$-unification problem. We set $\Gamma_p := \Gamma_p^{(0)}$ and $\zeta_X := \emptyset$ for all $X \in N_{def}$. While Γ_p contains an unsolved p-sequent, apply the steps (1) and (2).

(1) **Eager rule application:** If some eager rules apply to an unsolved p-sequent $\mathfrak{p}$ in Γ_p, apply one of highest priority. If the rule application fails, then return "no hybrid unifier".
(2) **Nondeterministic rule application:** If no eager rule is applicable, let $\mathfrak{p}$ be an unsolved p-sequent in Γ_p. If one of the nondeterministic rules applies to $\mathfrak{p}$, nondeterministically choose one of these rules and apply it. If none of these rules apply to $\mathfrak{p}$, then return "no hybrid unifier".

Once all p-sequents are solved, return the TBox $\mathcal{T}$ induced by the current assignment.

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

The *eager rules* are mainly there for optimization purposes, i.e., to avoid nondeterministic choices if a deterministic decision can easily be made. For example,

given a ground sequent $C \sqsubseteq_n D$, as considered in the *Eager Ground Solving* rule, the GCI $C \sqsubseteq D$ either follows from the ontology $\mathcal{O}$, in which case any TBox is a hybrid unifier of it, or it does not, in which case there is no hybrid unifier. This condition can be checked in polynomial time since subsumption w.r.t. hybrid $\mathcal{EL}$-ontologies is polynomial [11,14,13]. In the case considered in the *Eager Solving* rule, the TBox induced by the current assignment obviously already implies the GCI $C_1 \sqcap \ldots \sqcap C_m \sqsubseteq D$. The *Eager Axiom Solving* rule corresponds to the rules (Top) and (Start) of HC. Note that the rule (Refl) of HC is covered by *Eager Solving*.

The *nondeterministic rules* only come into play if no eager rules can be applied. In order to solve an unsolved p-sequent $(\mathfrak{s}, P)$, one considers which rule of HC could have been applied to obtain $\mathfrak{s}$. The rules *Extension* and *Decomposition* respectively correspond to applications of rules (DefL) and (Ex) of HC, together with an appropriate number of applications of the rules (AndLi). The *Mutation* rule corresponds to an application of the (GCI) rule from HC, again together with an appropriate number of applications of the rules (AndLi).

Due to the space restrictions, we cannot give details on how to prove that the algorithm is correct. Complete proofs of soundness, completeness and termination can be found in [6].

Theorem 14. *Algorithm 13 is an* NP*-decision procedure for hybrid $\mathcal{EL}$-unifiability w.r.t. arbitrary $\mathcal{EL}$-ontologies.*

6 Conclusions

In this paper, we have first proved that hybrid $\mathcal{EL}$-unification w.r.t. arbitrary $\mathcal{EL}$-ontologies is NP-complete, and then developed a goal-oriented NP-algorithm for hybrid $\mathcal{EL}$-unification that is better than the brute-force "guess and then test" algorithm used to show the "in NP" result. As illustrated by Example 6, computing hybrid unifiers rather than classical ones may be appropriate in some situations. Nevertheless, the decidability and complexity of classical $\mathcal{EL}$-unification w.r.t. arbitrary $\mathcal{EL}$-ontologies is an important topic for future research. We hope that hybrid unification may also be helpful in this context. Basically, given a hybrid unifier $\mathcal{T}$ of Γ w.r.t. $\mathcal{O}$, we can obtain a classical unifier of Γ w.r.t. $\mathcal{O}$ by finding an acyclic TBox $\mathcal{S}$ such that $\mathcal{O} \cup \mathcal{S}$ entails all the GCIs that $(\mathcal{O}, \mathcal{T})$ entails w.r.t. hybrid semantics, i.e. $C \sqsubseteq_{gfp,\mathcal{O},\mathcal{T}} D$ implies $C \sqsubseteq_{\mathcal{O}\cup\mathcal{S}} D$ for all (relevant) concept descriptions C, D.

References

1. Baader, F.: Terminological cycles in a description logic with existential restrictions. In: Gottlob, G., Walsh, T. (eds.) Proc. of the 18th Int. Joint Conf. on Artificial Intelligence (IJCAI 2003), pp. 325–330. Morgan Kaufmann, Los Altos (2003)
2. Baader, F., Borgwardt, S., Morawska, B.: Unification in the description logic $\mathcal{EL}$ w.r.t. cycle-restricted TBoxes. LTCS-Report 11-05, Chair for Automata Theory, Institute for Theoretical Computer Science, Technische Universität Dresden, Dresden, Germany (2011), `http://lat.inf.tu-dresden.de/research/reports.html`

3. Baader, F., Borgwardt, S., Morawska, B.: Extending unification in $\mathcal{EL}$ towards general TBoxes. In: Proc. of the 13th Int. Conf. on Principles of Knowledge Representation and Reasoning (KR 2012), pp. 568–572. AAAI/MIT Press (2012)
4. Baader, F., Borgwardt, S., Morawska, B.: A goal-oriented algorithm for unification in $\mathcal{ELH}_{R+}$ w.r.t. Cycle-restricted ontologies. In: Thielscher, M., Zhang, D. (eds.) AI 2012. LNCS, vol. 7691, pp. 493–504. Springer, Heidelberg (2012)
5. Baader, F., Calvanese, D., McGuinness, D., Nardi, D., Patel-Schneider, P.F. (eds.): The Description Logic Handbook: Theory, Implementation, and Applications. Cambridge University Press (2003)
6. Baader, F., Fernández Gil, O., Morawska, B.: Hybrid unification in the description logic $\mathcal{EL}$. LTCS-Report 13-07, Chair for Automata Theory, Institute for Theoretical Computer Science, Technische Universität Dresden, Dresden, Germany (2013), `http://lat.inf.tu-dresden.de/research/reports.html`
7. Baader, F., Morawska, B.: Unification in the description logic $\mathcal{EL}$. In: Treinen, R. (ed.) RTA 2009. LNCS, vol. 5595, pp. 350–364. Springer, Heidelberg (2009)
8. Baader, F., Morawska, B.: Unification in the description logic $\mathcal{EL}$. Logical Methods in Computer Science 6(3) (2010)
9. Baader, F., Narendran, P.: Unification of concept terms in description logics. J. of Symbolic Computation 31(3), 277–305 (2001)
10. Brandt, S.: Polynomial time reasoning in a description logic with existential restrictions, GCI axioms, and—what else? In: de Mántaras, R.L., Saitta, L. (eds.) Proc. of the 16th Eur. Conf. on Artificial Intelligence (ECAI 2004), pp. 298–302 (2004)
11. Brandt, S., Model, J.: Subsumption in $\mathcal{EL}$ w.r.t. hybrid tboxes. In: Furbach, U. (ed.) KI 2005. LNCS (LNAI), vol. 3698, pp. 34–48. Springer, Heidelberg (2005)
12. Küsters, R.: Non-Standard Inferences in Description Logics. LNCS (LNAI), vol. 2100. Springer, Heidelberg (2001)
13. Novaković, N.: Proof-theoretic Approach to Deciding Subsumption and Computing Least Common Subsumer in EL w.r.t. Hybrid TBoxes. Master's thesis, Chair for Automata Theory, Institute for Theoretical Computer Science, Technische Universität Dresden, Germany (2007), `http://lat.inf.tu-dresden.de/research/mas/#Nov-Mas-07`
14. Novaković, N.: A proof-theoretic approach to deciding subsumption and computing least common subsumer in $\mathcal{EL}$ w.r.t. hybrid TBoxes. In: Hölldobler, S., Lutz, C., Wansing, H. (eds.) JELIA 2008. LNCS (LNAI), vol. 5293, pp. 311–323. Springer, Heidelberg (2008)