LTCS–Report

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

Franz Baader Oliver Fernández Gil Barbara Morawska

LTCS-Report

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

Franz Baader Oliver Fernández Gil
Barbara Morawska[*]
Theoretical Computer Science, TU Dresden, Germany

July 25, 2013

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.

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

Contents

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 [11]. 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 $\mathsf{Head\_injury}$ and Severe, and the role names $\mathsf{finding}$ and status, we can describe the concept of a *patient with severe head injury* as

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

In a DL ontology, one can use *concept definitions* to introduce abbreviations for concept descriptions. For example, we could use the definition $\mathsf{Head\_injury} \equiv \mathsf{Injury} \sqcap \exists \mathsf{finding\_site}.\mathsf{Head}$ 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,

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

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 $\exists \mathsf{finding}.\exists \mathsf{status}.\mathsf{Severe}$. With respect to the GCI (2), it is thus also subsumed by $\exists \mathsf{status}.\mathsf{Emergency}$, i.e., in all models of this GCI, patients with severe head injury have an emergency status.

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

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

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

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

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

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 [6], 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 [6], 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 [7]. In [7] 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 [6] 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 $\exists\mathsf{child.Human} \sqsubseteq \mathsf{Human}$ satisfies this restriction, whereas the cyclic GCI $\mathsf{Human} \sqsubseteq \exists\mathsf{parent.Human}$ 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 [12, 16]. 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.

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.

2.1 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:

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

2.2 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 [11], 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.

2.3 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 $\mathcal{T}$.

2.4 Subsumption w.r.t. hybrid $\mathcal{EL}$-ontologies

Definition 3. 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 [12, 16], 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 [16] 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

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

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

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

$$\frac{C \sqsubseteq_n D}{X \sqsubseteq_n D} \quad \text{(DefL)} \qquad \frac{D \sqsubseteq_n C}{D \sqsubseteq_{n+1} X} \quad \text{(DefR)} \qquad \frac{C \sqsubseteq_n E \quad F \sqsubseteq_n D}{C \sqsubseteq_n D} \quad \text{(GCI)}$$

$$\text{for } X \equiv C \in \mathcal{T} \qquad\qquad \text{for } X \equiv C \in \mathcal{T} \qquad\qquad \text{for } E \sqsubseteq F \in \mathcal{O}$$

Figure 1: The calculus 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 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 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 [16] for more details)

Definition 4. 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 HC$(\mathcal{O}, \mathcal{T}, \Delta)$ if all sequents $C \sqsubseteq_n D$ for $n \geq 0$ can be derived using the rules of 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 5 (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 [16], 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 [16] 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 $n \geq 0$ [16]. 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 [16] 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 6. 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 7. 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 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. This unification problem does not have a classical unifier.

Assume to the contrary, that an acyclic TBox $\mathcal{T}$ is a classical unifier of Γ w.r.t. $\mathcal{O}$ and let $\sigma_{\mathcal{T}}$ be the corresponding substitution. We know that $\sigma_{\mathcal{T}}$ solves every subsumption in Γ, i.e. $\mathsf{Human} \sqsubseteq_{\mathcal{O}} \sigma_{\mathcal{T}}(X)$, $\mathsf{Horse} \sqsubseteq_{\mathcal{O}} \sigma_{\mathcal{T}}(X)$ and $\sigma_{\mathcal{T}}(X) \sqsubseteq_{\mathcal{O}} \exists \mathsf{parent}.\sigma_{\mathcal{T}}(X)$ must hold. We also can assume without loss of generality that $\sigma_{\mathcal{T}}$ is a ground substitution.

In the argument below, we will use the fact that the ground subsumptions can be easily decided with existing procedures [11].

One can easily see that $\sigma_{\mathcal{T}}(X)$ cannot be $\top$ since $\top \not\sqsubseteq_{\mathcal{O}} \exists \mathsf{parent}.\top$. Thus, let $\sigma_{\mathcal{T}}(X)$ be a ground concept description C (i.e. it does not contain concepts from N_{def}). Hence $\mathsf{Human} \sqsubseteq_{\mathcal{O}} C$, $\mathsf{Horse} \sqsubseteq_{\mathcal{O}} C$ and $C \sqsubseteq_{\mathcal{O}} \exists \mathsf{parent}.C$.

To show the contradiction, we prove that such C cannot exist. For that we use the characterization of subsumption in the presence of GCIs given in [3] and proceed by induction on the role depth of C, $rd(C)$.

Base case is when $rd(C) = 0$. Then C is a conjunction of concept names. But we can check that no concept name A can satisfy $\mathsf{Human} \sqsubseteq_{\mathcal{O}} A$ and $\mathsf{Horse} \sqsubseteq_{\mathcal{O}} A$ at the same time.

Assume now that $rd(C) = n$ and that no concept description C' of the smaller role depth satisfies both subsumptions at the same time: $\mathsf{Human} \sqsubseteq_{\mathcal{O}} C'$, $\mathsf{Horse} \sqsubseteq_{\mathcal{O}} C'$.

In general C may be a conjunction of concept names and existential restrictions $C_1 \sqcap \ldots, \sqcap C_n$. Obviously for each C_i both subsumptions: $\mathsf{Human} \sqsubseteq_{\mathcal{O}} C_i$, $\mathsf{Horse} \sqsubseteq_{\mathcal{O}} C_i$ must be satisfied. By the base case, $rd(C_i) > 0$ for each C_i.

Since and $rd(\mathsf{Human}) = rd(\mathsf{Horse}) = 0$ and $rd(C_i) > 0$ neither of the pairs of the above subsumptions are *structural* [3]. Therefore there must be concept names or existential restrictions $A^i_1, \ldots, A^i_n, B^i$ in $\mathcal{O}$ such that:

$$\mathsf{Human} \sqsubseteq_{\mathcal{O}} A^i_1, \ldots, \mathsf{Human} \sqsubseteq_{\mathcal{O}} A^i_n, B^i \sqsubseteq_{\mathcal{O}} C_i$$

where all these subsumptions are structural and also $A^i_1 \sqcap \cdots \sqcap A^i_n \sqsubseteq_{\mathcal{O}} B^i$ holds.

In general B^i may be a concept name or existential restriction from $\mathcal{O}$, but since $rd(C_i) > 0$, B^i must be an existential restriction, $B^i = \exists \mathsf{parent}.B^i_1$. Obviously since $rd(C_i) > 0$, C_i has to be an existential restriction $\exists \mathsf{parent}.C'_i$.

By the definition of structural subsumption, $B^1_1 \sqcap \cdots \sqcap B^n_1 \sqsubseteq_{\mathcal{O}} C'_1 \sqcap \cdots \sqcap C'_n$. Notice that if $C'_1 \sqcap \cdots \sqcap C'_n = \top$, then $\sigma_{\mathcal{T}}(X) = \exists \mathsf{parent}.\top$, but this is impossible, since we can easily check that $\exists \mathsf{parent}.\top \not\sqsubseteq_{\mathcal{O}} \exists \mathsf{parent} \exists \mathsf{parent}.\top$.

Now each B^i_1 is either Human or Horse.

If any B^i_1 is Horse, then $B^i = \exists \mathsf{parent.Horse}$, which leads to contradition, since then $\mathsf{Human} \sqsubseteq_{\mathcal{O}} \exists \mathsf{parent.Horse}$ which does not hold.

If each B^i_1 is Human, then $\mathsf{Human} \sqsubseteq_{\mathcal{O}} C'_1 \sqcap \cdots \sqcap C'_n$. But since the role depth of $C'_1 \sqcap \cdots \sqcap C'_n$ is smaller than $rd(C)$, hence by induction we have that $\mathsf{Horse} \not\sqsubseteq_{\mathcal{O}} C'_1 \sqcap \cdots \sqcap C'_n$.

Now since the subsumption $\mathsf{Horse} \sqsubseteq_{\mathcal{O}} C$ must also hold, because of role depth difference between Horse and C, we must again have concept names or existential restrictions $A'^i_1, \ldots, A'^i_n, B'^i$ in $\mathcal{O}$ for each C_i such that:

$$\mathsf{Horse} \sqsubseteq_{\mathcal{O}} A'^i_1, \ldots, \mathsf{Horse} \sqsubseteq_{\mathcal{O}} A'^i_m, B'^i \sqsubseteq_{\mathcal{O}} C_i$$

where all these subsumptions are structural and also $A'^i_1 \sqcap \cdots \sqcap A'^i_m \sqsubseteq_{\mathcal{O}} B'^i$ holds.

For the same reason as above B'^i must be an existential restriction from $\mathcal{O}$, $B'^i = \exists \mathsf{parent}.B'^i_1$. B'^i_1 is either Human or Horse.

If any B'^i_1 is Human, then we have a contradition, because then $\mathsf{Horse} \sqsubseteq_{\mathcal{O}} \exists \mathsf{parent.Human}$ should hold, but it does not.

Hence each B'^i_1 is Horse. But this leads also to a contradiction because it implies that $\mathsf{Horse} \sqsubseteq_{\mathcal{O}} C'_1 \sqcap \cdots \sqcap C'_n$.

3.1 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.

$$C_1 \sqcap \exists r.\widehat{D} \sqcap C_2 \, \rho \, E \longrightarrow \{A \equiv \widehat{D}, C_1 \sqcap \exists r.A \sqcap C_2 \, \rho \, E\} \tag{R1}$$

$$E \, \rho \, C_1 \sqcap \exists r.\widehat{D} \sqcap C_2 \longrightarrow \{E \, \rho \, C_1 \sqcap \exists r.A \sqcap C_2, A \equiv \widehat{D}\} \tag{R2}$$

$$E \equiv B_1 \sqcap \cdots \sqcap B_n \longrightarrow \{E \sqsubseteq B_1, \ldots, E \sqsubseteq B_n, B_1 \sqcap \cdots \sqcap B_n \sqsubseteq E\} \tag{R3}$$

$$E \equiv \exists r.B \longrightarrow \{E \sqsubseteq \exists r.B, \exists r.B \sqsubseteq E\} \tag{R4}$$

$$E \sqsubseteq B_1 \sqcap \cdots \sqcap B_n \longrightarrow \{E \sqsubseteq B_1, \ldots, E \sqsubseteq B_n\} \tag{R5}$$

Figure 2: Rules used to normalize a general TBox.

Flattening of an ontology. To transform a given ontology $\mathcal{O}$ into a *flat* ontology, we use a slightly modified normalization procedure proposed in [10] that consists of the exhaustive application of rules $(R1) - (R5)$ shown in Figure 2. In these rules C_1, C_2, E stand for possibly empty conjunctions of concept descriptions, $\widehat{D}$ is a concept description that is neither a concept name nor $\top$, A is always a new concept name not occurring in $\mathcal{O}$ or Γ, $r \in N_R$, $\rho \in \{\sqsubseteq, \equiv\}$ and $B, B_1, \ldots, B_n$ represent concept names.

First, rules $(R1), (R2)$ are exhaustively applied to obtain a new ontology that consists of GCIs constructed from conjunctions of flat atoms and additional flat concept definitions. Second, the application of rules $(R3), (R4)$ transforms those remaining concept definitions into subsumptions, $(R5)$ transforms these subsumptions into the required form.

It is clear that the number of applications of rules $(R1), (R2)$ is limited linearly in the size of the original ontology and applying these rules increases the size of ontology only polynomially. Afterwards, the number of $(R3)$ and $(R4)$ applications is linear in the number of equivalences and subsumptions in the modified ontology and they increase the size polynomially. The same is again true about the applications of $(R5)$.

Now we have to see that Γ has a (hybrid or classical) unifier w.r.t. $\mathcal{O}$ iff Γ has a (hybrid or classical) unifier w.r.t. $\mathcal{O}'$.

Since the above normalization rules preserve equivalence in the descriptive semantics, we have that for any concept descriptions C and D build over the signature of $\mathcal{O}$, $C \sqsubseteq_{\mathcal{O}} D$ iff $C \sqsubseteq_{\mathcal{O}'} D$. Now we prove a similar fact for the hybrid semantics.

Lemma 8. *Let $\mathcal{O}_2$ be obtained from $\mathcal{O}_1$ by normalization and let C, D be any concept descriptions constructed in the signature of $\mathcal{O}_1$, and $\mathcal{T}$ be any TBox.*

Then

$$C \sqsubseteq_{gfp,\mathcal{O}_1,\mathcal{T}} D \text{ iff } C \sqsubseteq_{gfp,\mathcal{O}_2,\mathcal{T}} D$$

Proof. ($\Rightarrow$) Assume that $C \sqsubseteq_{gfp,\mathcal{O}_1,\mathcal{T}} D$ holds. We have to show that for each hybrid-model I of $(\mathcal{O}_2, \mathcal{T})$ for any $\mathcal{T}$, $C^I \subseteq D^I$ holds.

For each GCI $E \sqsubseteq F$ in $\mathcal{O}_1$ one can see that:

- E and F are concept descriptions defined over $sig(\mathcal{O}_1)$.
- Obviously, $E \sqsubseteq_{\mathcal{O}_1} F$ holds.
- Hence $E \sqsubseteq_{\mathcal{O}_2} F$ holds as well.

Now, consider any hybrid-model $\mathcal{I}$ of $(\mathcal{O}_2, \mathcal{T})$ and let J be the primitive interpretation that $\mathcal{I}$ is based on. By a definition of a hybrid model (Definition 2), J must be a model of $\mathcal{O}_2$ and hence $E^{\mathcal{J}} \subseteq F^{\mathcal{J}}$ holds for all GCI $E \sqsubseteq F$ in $\mathcal{O}_1$. Thus, $\mathcal{J}$ is a model of $\mathcal{O}_1$ and consequently $\mathcal{I}$ is a hybrid-model of $(\mathcal{O}_1, \mathcal{T})$.

Finally, by the definition of hybrid subsumption (Definition 3) we obtain that $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$. Thus, $C \sqsubseteq_{gfp,\mathcal{O}_2,\mathcal{T}} D$ holds.

($\Leftarrow$) Assume that $C \sqsubseteq_{gfp,\mathcal{O}_2,\mathcal{T}} D$ holds, and consider an arbitrary hybrid-model $\mathcal{I}$ of $(\mathcal{O}_1, \mathcal{T})$. It is not difficult to see that $\mathcal{I}$ can be extended to a hybrid-model $\mathcal{I}'$ of $(\mathcal{O}_2, \mathcal{T})$, by assigning values to the new primitive concepts introduced in $\mathcal{O}_2$ during the normalization. Therefore, $C^{\mathcal{I}'} \subseteq D^{\mathcal{I}'}$ holds.

Now, let $\mathcal{I}'|_{sig(\mathcal{O}\cup\mathcal{T})}$ be the restriction of $\mathcal{I}'$ to $sig(\mathcal{O} \cup \mathcal{T})$. Since C and D are defined over $sig(\mathcal{O}\cup\mathcal{T})$, it follows that $C^{\mathcal{I}'}|_{sig(\mathcal{O}\cup\mathcal{T})} \subseteq D^{\mathcal{I}'}|_{sig(\mathcal{O}\cup\mathcal{T})}$ holds. Obviously, $\mathcal{I} = \mathcal{I}'|_{sig(\mathcal{O}\cup\mathcal{T})}$ and consequently $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$.

Thus, $C \sqsubseteq_{gfp,\mathcal{O}_1,\mathcal{T}} D$ holds. □

Flattening of a unification problem Γ. To transform a given set of goal equivalences into a set of flat subsumptions, we use the same procedure as for flattening an ontology, with one exception: the new concept names used for flattening (A in ($R1$) and ($R2$)) are defined as new defined concepts i.e. they are added to the set N_{def}.

Lemma 9. *Let Γ' be obtained from Γ by normalization, then:*

- *if $\mathcal{T}$ is a hybrid unifier of Γ' w.r.t. $\mathcal{O}$, then it is also a hybrid unifier of Γ w.r.t. $\mathcal{O}$,*
- *if $\mathcal{T}'$ is a hybrid unifier of Γ w.r.t. $\mathcal{O}$, then $\mathcal{T}'$ can be extended to $\mathcal{T}$ such that $\mathcal{T}$ is a unifier of Γ'.*

Proof. In order to prove the first statement of the lemma, we define an auxiliary TBox in the following way.

$\mathcal{T}_{aux} := \{A \equiv \widehat{D} \mid A \equiv \widehat{D}$ *was produced by rules* $(R1), (R2)$ *after the first stage in the normalization of* $\Gamma\}$

Since $\mathcal{T}_{aux}$ is an acyclic TBox, we know that it induces a substitution $\sigma_{\mathcal{T}_{aux}}$. It is also clear that for each $C \sqsubseteq D \in \Gamma$, there are subsumptions $C' \sqsubseteq D_1, \ldots, C' \sqsubseteq D_k \in \Gamma'$ such that $\sigma_{\mathcal{T}_{aux}}(C') = C$ and $\sigma_{\mathcal{T}_{aux}}(D_1 \sqcap \cdots \sqcap D_k) = D$. Now, we know that $C' \sqsubseteq_{gfp,\mathcal{O},\mathcal{T}} D_1, \ldots, C' \sqsubseteq_{gfp,\mathcal{O},\mathcal{T}} D_k$, but then also $\sigma_{\mathcal{T}_{aux}}(C') \sqsubseteq_{gfp,\mathcal{O},\mathcal{T}} \sigma_{\mathcal{T}_{aux}}(D_1 \sqcap \cdots \sqcap D_k)$ and hence $C \sqsubseteq_{gfp,\mathcal{O},\mathcal{T}} D$ as required.

For the second statement of the lemma, we assume that $\mathcal{T}'$ is a hybrid unifier of Γ w.r.t. $\mathcal{O}$. It is easy to see that a TBox $\mathcal{T} := \mathcal{T}' \cup \mathcal{T}_{aux}$ is a hybrid unifier of Γ' w.r.t. $\mathcal{O}$.

If $C \sqsubseteq D \in \Gamma'$ then either $\sigma_{\mathcal{T}_{aux}}(C) \sqsubseteq \sigma_{\mathcal{T}_{aux}}(D) \sqcap D'$ is in Γ (D' is a conjunction of some atoms in Γ) or $\sigma_{\mathcal{T}_{aux}}(C) \sqsubseteq \sigma_{\mathcal{T}_{aux}}(D)$ is a subsumption of the form $E_1 \sqcap \cdots \sqcap E_n \sqsubseteq E_i$ for $0 < i \leq n$, which is trivially satisfied. Hence $\sigma_{\mathcal{T}_{aux}}(C) \sqsubseteq_{gfp,\mathcal{O},\mathcal{T}'} \sigma_{\mathcal{T}_{aux}}(D)$ and thus $C \sqsubseteq_{gfp,\mathcal{O},\mathcal{T}' \cup \mathcal{T}_{aux}} D$ as required.

$\square$

In the following we will assume that all unification problems are flat.

3.2 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

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

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

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

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 10 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 [6].

4 Some properties of proof trees I

In this section we show some properties of proof trees in $\mathrm{HC}(\mathcal{O}, \mathcal{T}, \Delta)$, which will be used as auxiliary lemmas in the next section. The reader is advised to skip this section and return to it when needed.

Lemma 11. *Let C, D be sub-descriptions of concept descriptions occurring in $\mathcal{O}$, $\mathcal{T}$, and Δ such that C is ground and $\mathcal{O}$ is also ground. Then, for all $n \geq 0$ and any proof tree $\mathcal{P}$ for $C \sqsubseteq_n D$ in $\mathrm{HC}(\mathcal{O}, \mathcal{T}, \Delta)$, it is true that every sequent at a node in $\mathcal{P}$ is left-hand side ground.*

Proof. This is a straight-forward proof. It goes by induction on the structure of proof trees. First, because C is ground, one can see that the only rule from $\mathrm{HC}(\mathcal{O}, \mathcal{T}, \Delta)$ that cannot be used to obtain $C \sqsubseteq_n D$ in $\mathcal{P}$ is the rule (DefL).

Second, if $C \sqsubseteq_n D$ is an instance of one of the rules (Refl), (Top) or (Start), we have that $\mathcal{P}$ is a one-element proof tree and the left-hand side ground condition is implicit.

Finally, it can be seen that the left-hand side of the premise (premises) of any other instance of a rule that could have been applied to obtain $C \sqsubseteq_n D$, is either C, a sub-description of C, or an atom from a GCI in $\mathcal{O}$ which is also ground. Then, applying induction to the sub-proof tree (trees) of $\mathcal{P}$ that has this premise (premises) as its root, we obtain that every sequent in $\mathcal{P}$ is left-hand side ground. □

Now, we define the notion of *maximal* sub-proof tree w.r.t. a set of rules from $\mathrm{HC}(\mathcal{O}, \mathcal{T}, \Delta)$.

Definition 12. Let $\mathcal{R} = \{R_1, \ldots, R_m\}$ be a subset of rules from $\mathrm{HC}(\mathcal{O}, \mathcal{T}, \Delta)$ and $\mathcal{P}$ a proof tree for the sequent $C \sqsubseteq_n D$ in $\mathrm{HC}(\mathcal{O}, \mathcal{T}, \Delta)$. A *maximal* sub-proof tree of $\mathcal{P}$ w.r.t. $\mathcal{R}$ is the subtree $\mathcal{P}_{\mathcal{R}}$ of $\mathcal{P}$ with the same root as $\mathcal{P}$, that satisfies the following conditions:

1. Each sequent at an internal node in $\mathcal{P}_{\mathcal{R}}$ is the consequence of an instance of a rule from $\mathcal{R}$.
2. Each sequent at a leaf in $\mathcal{P}_{\mathcal{R}}$ is either an instance of a rule in {(Refl), (Top), (Start)} or it is obtain as the consequence of an instance of a rule that is not in $\mathcal{R}$.

Based on this definition, we prove the next two propositions w.r.t. the sets of rules $\mathcal{R}_1 = \{(\mathit{AndL}1), (\mathit{AndL}2), (\mathit{AndR})\}$ and $\mathcal{R}_2 = \{(\mathit{AndL}1), (\mathit{AndL}2), (\mathit{AndR}), (\mathit{Ex}), (\mathit{GCI})\}$.

Lemma 13. *Let $\mathcal{P}$ be a proof tree for the sequent $C \sqsubseteq_n D$ in $\mathrm{HC}(\mathcal{O}, \mathcal{T}, \Delta)$ and B a top-level atom of D. Consider the maximal sub-proof tree $\mathcal{P}_{\mathcal{R}}$ of $\mathcal{P}$ w.r.t. $\mathcal{R} = \{(\mathit{AndL}1), (\mathit{AndL}2), (\mathit{AndR})\}$. The following two statements are true:*

1. *There exists a leaf $E \sqsubseteq_n F$ in $\mathcal{P}_{\mathcal{R}}$ such that B is a top-level atom of F.*
2. *For every leaf $E \sqsubseteq_n F$ in $\mathcal{P}_{\mathcal{R}}$, the concept description E is a sub-description of C.*

Proof. Again, we use induction on the structure of proof trees. First, we consider the case when $C \sqsubseteq_n D$ is obtained in $\mathcal{P}$ by using an instance of a rule that is not in $\mathcal{R}$. This means, that $\mathcal{P}_{\mathcal{R}}$ has only one leaf whose sequent is $C \sqsubseteq_n D$ and thus, (1) and (2) are trivially satisfied.

Second, we analyze the case where one of the rules from $\mathcal{R}$ is used to obtain $C \sqsubseteq_n D$ in $\mathcal{P}$. An instance of such a rule has the form:

$$\frac{C' \sqsubseteq_n D}{C \sqsubseteq_n D}\ (\text{AndLi}) \quad \text{or} \quad \frac{C \sqsubseteq_n D_1 \quad C \sqsubseteq_n D_2}{C \sqsubseteq_n D}\ (\text{AndR})$$

where C' and D_1, D_2 are sub-descriptions of C and D respectively.

Let $\mathcal{P}', \mathcal{P}_1$ and $\mathcal{P}_2$ be the corresponding sub-proof trees for the premises of the instances mentioned above. Applying induction to these sub-trees we have that (1) and (2) hold for the leaves in their corresponding maximal sub-proof trees w.r.t. $\mathcal{R}$.

Finally, it can be seen that each leaf in $\mathcal{P}_{\mathcal{R}}$ is a leaf in $\mathcal{P}'$ in the first case, or a leaf in either $\mathcal{P}_1$ or $\mathcal{P}_2$ for the second case. Then, it follows immediately that (1) and (2) are also satisfied for $\mathcal{P}_{\mathcal{R}}$. □

Lemma 14. *Let $\mathcal{T}'$ be a TBox and $C \sqsubseteq_n D$ be a sequent. If we have that:*

1. $\mathcal{R} = \{(AndL1), (AndL2), (AndR), (Ex), (GCI)\}$
2. *There is a proof tree $\mathcal{P}$ for $C \sqsubseteq_n D$ in* $\mathrm{HC}(\mathcal{O}, \mathcal{T}, \Delta)$.
3. *For each sequent $E_1 \sqsubseteq_n E_2$ at a leaf in the maximal sub-proof tree of $\mathcal{P}$ w.r.t. $\mathcal{R}$, it is the case that $E_1 \sqsubseteq_k E_2$ is derivable in* $\mathrm{HC}(\mathcal{O}, \mathcal{T}', \Delta)$ *for some $k \geq 0$.*

then, there exists a proof tree $\mathcal{P}'$ for $C \sqsubseteq_k D$ in $\mathrm{HC}(\mathcal{O}, \mathcal{T}', \Delta)$.

Proof. The proof is by induction on the structure of proof trees. Assume that (1), (2) and (3) hold, we make a two cases distinction w.r.t. the rule used to obtain $C \sqsubseteq_n D$ in $\mathcal{P}$:

1. $C \sqsubseteq_n D$ is the consequence of an instance of a rule not in $\mathcal{R}$. By Definition 12, $\mathcal{P}_{\mathcal{R}}$ is a one-element tree with the root $C \sqsubseteq_n D$ which means that $C \sqsubseteq_n D$ is also a leaf in $\mathcal{P}_{\mathcal{R}}$. Then, $C \sqsubseteq_k D$ is derivable in $\mathrm{HC}(\mathcal{O}, \mathcal{T}', \Delta)$ for some k and thus, there exists a proof tree $\mathcal{P}'$ for $C \sqsubseteq_k D$ in $\mathrm{HC}(\mathcal{O}, \mathcal{T}', \Delta)$.

2. $C \sqsubseteq_n D$ is the consequence of an instance of a rule in $\mathcal{R}$. We show the case where $C \sqsubseteq_n D$ is obtained by an application of the (GCI) rule, the other four cases can be shown in a similar way.

   There is a GCI $E \sqsubseteq F$ in $\mathcal{O}$ such that $C \sqsubseteq_n E$ and $F \sqsubseteq_n D$ are the premises of the (GCI)-instance used to obtain $C \sqsubseteq_n D$ in $\mathcal{P}$. By definition of a proof tree, it can be seen that the subtrees $\mathcal{P}_1$ and $\mathcal{P}_2$ of $\mathcal{P}$ with roots $C \sqsubseteq_n E$ and $F \sqsubseteq_n D$, are proof trees for $C \sqsubseteq_n E$ and $F \sqsubseteq_n D$ in $\mathrm{HC}(\mathcal{O}, \mathcal{T}, \Delta)$.

Moreover, it is not difficult to see that the leaves in the maximal sub-proof trees of $\mathcal{P}_1$ and $\mathcal{P}_2$ w.r.t. $\mathcal{R}$ are also leaves in $\mathcal{P}_{\mathcal{R}}$. Then, by induction we obtain that there exist proof trees for $C \sqsubseteq_k E$ and $F \sqsubseteq_k D$ in $\mathrm{HC}(\mathcal{O}, \mathcal{T}', \Delta)$. Thus, a further application of the GCI rule yields a proof tree for $C \sqsubseteq_k D$ in $\mathrm{HC}(\mathcal{O}, \mathcal{T}', \Delta)$.

□

5 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 15. *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 15, it is enough to prove the following lemma.

Lemma 16. *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 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 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 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 HC$(\mathcal{O}, \mathcal{T}', \Delta)$. One can see that such a tree is a maximal sub-proof tree of $\mathcal{P}$ w.r.t. to the set of rules $\mathcal{R} = \{(AndL1), (AndL2), (AndR), (Ex), (GCI)\}$ and therefore the application of Lemma 14 will complete the proof.

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 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 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 $F_1, \ldots, F_q \in$ At, induction yields that $E_1 \sqsubseteq_{n-1} F_i$ is derivable in 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 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 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 HC$(\mathcal{O}, \mathcal{T}', \Delta)$ since $q-1$ applications of (AndR) then yield derivability of $E_1 \sqsubseteq_n E_2$ in HC$(\mathcal{O}, \mathcal{T}', \Delta)$.

If F_i does not belong to N_{def}, then it is an element of At_{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 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 HC$(\mathcal{O}, \mathcal{T}', \Delta)$. $\square$

This finishes the proof of Proposition 15, 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 7, 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 [13], 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.

Lemma 17. *If an $\mathcal{EL}$-matching problem modulo equivalence has a hybrid unifier w.r.t. the empty ontology, then it also has a matcher.*

For the proof of this theorem we will show that if an $\mathcal{EL}$-matching problem modulo equivalence has a hybrid unifier w.r.t. the empty ontology, it must have a hybrid unifier which is an acyclic TBox. As mentioned above, acyclic hybrid unifier is a classical unifier i.e. a matcher.

Before proving the lemma, we have to refer to another property of cyclic TBoxes, which comes handy in this place.

Namely, it has been shown in [14] that in the presence of greatest fixpoint semantics a TBox $\mathcal{T}$ containing component cycles can be transformed into a TBox $\mathcal{T}'$ that is free of component cycles, where component cycles are defined as follows.

Definition 18. Let $\mathcal{T}$ be a TBox and A_0, A_n defined concepts in $\mathcal{T}$.

A_0 uses A_n as a *component* in its definition iff there is a sequence of defined concepts $A_0, \ldots, A_n (n > 0)$ in $\mathcal{T}$ such that: for each $i, 0 \leq i < n$, $A_i \equiv C \in \mathcal{T}$ and A_{i+1} occurs in C, and, A_{i+1} is a top-level atom in the definition of A_i for all $i > 0$, i.e., A_{i+1} appears outside the scope of any existential restriction in the definition of A_i. If, in addition, $A_0 = A_n$ then $A_0, \ldots, A_n$ is called a *component-cycle* in $\mathcal{T}$.

Then, we say that a cyclic-defined concept A in $\mathcal{T}$ is *component-cyclic-defined* if it uses itself as a component, i.e., there is a component-cycle in $\mathcal{T}$ that contains A. Otherwise, we call it *restricted-cyclic-defined.*

The following lemma is proved in [14].

Lemma 19. *Let $\mathcal{T}$ be a TBox that contains component cycles. Then, there exists a TBox $\mathcal{T}'$ that does not contain component cycles such that:*

$$I \textit{ is a gfp-model of } \mathcal{T} \textit{ iff } I \textit{ is a gfp-model of } \mathcal{T}'$$

Assume that C is a ground concept description. We will show that a subsumption $C \sqsubseteq_\infty D$ cannot be proved in HC w.r.t. empty ontology and a cyclic TBox when a cyclic-defined variable occurs in D. The next lemma is used to identify a sequent in a proof tree for $C \sqsubseteq_\infty D$, which cannot have a proof in HC.

Lemma 20. *Let C and D be two concept descriptions such that C is ground and at least one variable occurs in D.*

For all $n > 0$ and any proof tree $\mathcal{P}$ for $C \sqsubseteq_n D$ w.r.t. a hybrid TBox $(\emptyset, \mathcal{T})$: if B is a non-ground top-level atom of D then there exists a node in $\mathcal{P}$ with a sequent of the form $G \sqsubseteq_n B$, where G is a concept description.

Proof. Let $\mathcal{P}$ be a proof tree for $C \sqsubseteq_n D$ for an arbitrary $n > 0$. There are two observations that can be done about $\mathcal{P}$. First, since C is ground, Lemma 11 says that every sequent at a node in $\mathcal{P}$ is left-hand side ground and therefore, the rule (DefL) is never used to build $\mathcal{P}$. Second, since $\mathcal{P}$ is built w.r.t. the hybrid TBox $(\emptyset, \mathcal{T})$ then, it is clear that no instance of the rule (GCI) is used to build $\mathcal{P}$.

Now, consider the set of rules $\mathcal{R} = \{(AndL1), (AndL2), (AndR)\}$ and the *maximal* sub-proof tree $\mathcal{P}_\mathcal{R}$ of $\mathcal{P}$ w.r.t. $\mathcal{R}$. Applying Lemma 13 (1) to $\mathcal{P}_\mathcal{R}$ we have that if B is a top-level atom of D then, there exists a leaf in $\mathcal{P}_\mathcal{R}$ with the sequent $G \sqsubseteq_n E$ where E is of the form $\ldots \sqcap B \sqcap \ldots$.

Since G is ground and E is not ground, $G \sqsubseteq_n E$ is neither a consequence of an instance of (Refl) nor of an instance of (Top). In addition, $n > 0$ implies that it is not an instance of (Start) as well. Hence, since (DefL) and (GCI) are not used to build $\mathcal{P}$, by Definition 12 $G \sqsubseteq_n E$ must be the consequence of an instance either of rule (Ex) or rule (DefR). Looking at the structure of these two rules, there are two possible cases for the form of E:

1. $E = X$ for some variable X or,
2. $E = \exists s.E'$ for some role name s and a concept description E'.

We can conclude that E contains only one top-level atom and thus, since B is a top-level atom of E it follows directly that $E = B$ and $G \sqsubseteq_n B$ is the sequent of a node in $\mathcal{P}$. □

In the next lemma we will show that for an empty ontology and a cyclic TBox, the number n of a sequentf $C \sqsubseteq_n D$ provable in HC is restricted by the role depth of C, which is ground. This is basically because before applying a definition from a cyclic TBox requires application of the rule (Ex). In order to prove the next lemma, we assume without loss of generality that our cyclic TBox does not contain *component cycles*.

Lemma 21. *Let C and D be two concept descriptions, $\mathcal{T}$ be a cyclic TBox such that C is ground and at least one cyclic-defined variable occurs in D and r be the role depth of C. Then there is no proof tree for $C \sqsubseteq_{r+2} D$ in* HC *w.r.t. empty ontology.*

Proof. We show that in a proof tree $C \sqsubseteq_{r+2} D$ there has to be a node with a sequent of the form $A \sqsubseteq_l \exists r.E$, where A is a primitive concept name and $l > 0$. This is a contradiction, because such sequent cannot be obtained by any rule in HC.

Hence it is enough to prove the following claim:

If $\mathcal{P}$ is a proof tree for $C \sqsubseteq_{r+2} D$, then there is a node in $\mathcal{P}$ with a sequent of the form: $A \sqsubseteq_l \exists r.E$, where A is a primitive concept name and $l > 0$.

We proceed by induction on the role depth r of C.

Base Case: $r = 0$. By assumption $C \sqsubseteq_2 D$ holds and C is of the form $A_1 \sqcap \ldots \sqcap A_k$ where A_i is a primitive concept name for all $i, 1 \leq i \leq k$. Let X be a cyclic-defined variable in $\mathcal{T}$ and B a top level atom of D where X occurs. By Lemma 20, there is a sequent of the form $G \sqsubseteq_2 B$ at a node in $\mathcal{P}$.

Since $G \sqsubseteq_2 B$ is a leaf in $\mathcal{P}_{\mathcal{R}}$ as described in Lemma 20, then by Lemma 13 (2) we have that G is a sub-description of C and consequently it is also a conjunction of primitive concept names. We can assume that G is of the form $A_i \sqcap \ldots \sqcap A_j$ for $1 \leq i, j \leq k$. Next, we make a two cases distinction with respect to the structure of B:

1. $B = \exists s.E$. Since G is ground and a conjunction of primitive concept names, the sequent $G \sqsubseteq_2 B$ can only be derived using successive applications of rules (AndL1) and (AndL2), which are rules that preserve the right-hand side of a sequent. Hence, there must exist a node in $\mathcal{P}$ with a sequent of the form $A_q \sqsubseteq_2 \exists s.E$ where $i \leq q \leq j$.

2. $B = X$. In this case, we can use the rules (AndL1), (AndL2) and (DefR) in order to obtain a sequent of the form $G \sqsubseteq_2 X$. Actually, it is not only that rule (DefR) can be used but, it has to be used:

   Suppose that $G \sqsubseteq_n X$ is obtained by only applying rules (AndL1) and (AndL2). As shown in the previous case, there is a node in $\mathcal{P}$ with a sequent of the form $A_q \sqsubseteq_2 X$ where A_q is a primitive concept name. Obviously, this sequent is not proved yet in HC, and the only rule that could have been used to obtain it, is the rule (DefR).

Hence, we can assume that $\mathcal{P}$ has a node with a sequent of the form $G' \sqsubseteq_2 X$ that is obtained as a consequence of an instance of rule (DefR), where G' is a sub-description of G. The premise of such an instance is also a sequent at a node in $\mathcal{P}$, i.e., $G' \sqsubseteq_1 D_1 \sqcap \ldots \sqcap D_m$ where $X \equiv D_1 \sqcap \ldots \sqcap D_m$ is a concept definition in $\mathcal{T}$.

Since X is cyclic-defined in $\mathcal{T}$ then for some i, D_i is of the form $\exists s.E'$ where E' is not ground and it contains an occurrence of a cyclic-defined variable in $\mathcal{T}$. A second application of Lemma 20 w.r.t. $G' \sqsubseteq_1 D_1 \sqcap \ldots \sqcap D_m$ and $D_i = \exists s.E'$, yields case 1 w.r.t. $\sqsubseteq_1$.

This completes the proof of the claim for $r = 0$, since one case is proved w.r.t. $\sqsubseteq_2$ and the other one w.r.t. $\sqsubseteq_1$.

Induction Step: Assume that the claim holds whenever the role depth of C is less than r and let us see that it holds for r. Using the same reasoning as before one can see that there is a sequent in $\mathcal{P}$ of the form $G \sqsubseteq_{r+2} B$ where B is a non-ground top level atom in D. There are two cases w.r.t. the role depth of G:

1. The role depth of G is less than r. Then, induction hypothesis can be applied to show the claim.
2. The role depth of G is r. If $B = \exists s.E$, $G \sqsubseteq_{r+2} B$ can be obtained using rules (AndL1), (AndL2) or (Ex). A similar reasoning as in the base case for the existence of a (DefR) application, yields that the rule (Ex) must be applied. Then, there is a sequent $G' \sqsubseteq_{r+2} E$ in $\mathcal{P}$ to which the rule (Ex) is applied and it is clear that the role depth of G' is less than r.

   The other possibility is the case when $B = X$, but using the same reasoning as for the base case the existential case is obtained w.r.t. $\sqsubseteq_{r+1}$, and induction can also be applied.

Thus, the claim is proved. Notice that the proof implicitely says that the result not only holds for $C \sqsubseteq_{r+2} D$ but for $C \sqsubseteq_{>r+2} D$ as well. □

Proof of Lemma 17 Assume that Γ has a hybrid unifier $\mathcal{T}$ w.r.t. empty ontology i.e. $C \sqsubseteq_{gfp,\emptyset,\mathcal{T}} D$ holds.

If D does not contain any occurrence of a cycle-defined variable in $\mathcal{T}$, then the definitions of cyclic-defined variables can be removed from $\mathcal{T}$ to obtain an acyclic TBox that is still a hybrid unifier of Γ w.r.t. the empty ontology.

Otherwise, if D contains a cyclic-defined variable, since by assumption $C \sqsubseteq_{gfp,\emptyset,\mathcal{T}} D$, we have that $C \sqsubseteq_n D$ for each $n \geq 0$ and in particular $C \sqsubseteq_{r+2} D$, where r is a role depth of C. But Lemma 21 says that $C \sqsubseteq_{r+2} D$ cannot have a proof tree in HC w.r.t. $(\emptyset, \mathcal{T})$ which is a contradiction. Therefore D does not contain a cyclic-defined variable, and there is an acyclic hybrid unifier $\mathcal{T}$ of Γ w.r.t. the empty ontology. Acyclicity of $\mathcal{T}$ implies the equivalence between greatest fixpoint semantic and descriptive semantics. Hence $\mathcal{T}$ is a classical unifier and a matcher.

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

Theorem 22. *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.*

6 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 $C \sqsubseteq_\infty D$.

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 23. 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-sequent 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 (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-sequent $(C \sqsubseteq_n D, P)$, 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 $P'' \cup P$,

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 $C_1 \sqcap \ldots \sqcap C_m \sqsubseteq_{\mathcal{O}} D$ 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.*

Figure 3: The eager rules of hybrid unification.

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 $\mathfrak{p}'$. 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 3), and three *nondeterministic* rules (see Figure 4). Eager rules are applied with higher priority than nondeterministic rules. Among the eager rules, Eager Axiom Solving has the highest priority, then comes Eager Ground Solving, and then Eager Solving.

Algorithm 24. Let Γ w.r.t. $\mathcal{O}$ be a flat $\mathcal{EL}$-unification problem. We compute a "large enough" index ℓ, and 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]Such p-sequents are dealt with by expansion rather than by applying a rule (see below).

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 $\mathcal{O}$ such that *blocking* does not fail if applied to Γ_p and each of the p-sequents $(C_1 \sqcap \ldots \sqcap C_m \sqsubseteq_n E_1, P \cup \{\mathfrak{s}\}), \ldots, (C_1 \sqcap \ldots \sqcap C_m \sqsubseteq_n E_k, P \cup \{\mathfrak{s}\})$, and $(F \sqsubseteq_n D, P \cup \{\mathfrak{s}\})$.
Action: Its application chooses such a GCI $E_1 \sqcap \ldots \sqcap E_k \sqsubseteq F$. It applies *blocking* to Γ_p and each of the p-sequents $(C_1 \sqcap \ldots \sqcap C_m \sqsubseteq_n E_1, P \cup \{\mathfrak{s}\}), \ldots, (C_1 \sqcap \ldots \sqcap C_m \sqsubseteq_n E_k, P \cup \{\mathfrak{s}\})$, and $(F \sqsubseteq_n D, P \cup \{\mathfrak{s}\})$. Once *blocking* was applied, $(\mathfrak{s}, P)$ is marked as *solved.*

Figure 4: The nondeterministic rules of hybrid unification.

(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.

For a given input Γ and the ontology $\mathcal{O}$, the index ℓ is computed as:

$$\ell := ((\#(\Gamma) + |At|) \times |N_{def}|) + 1$$

where $\#(\Gamma)$ is the number of subsumptions in Γ. In the proof of the soundness of the algorithm, we will see that this number is "large enough" for the algorithm to yield a unifier.

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 [12, 16]. 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).

6.1 Soundness

In this section, we show that if the Algorithm 24 returns a TBox $\mathcal{T}$ given as its input the unification problem Γ, then $C_i \sqsubseteq_\infty D_i$ can be derived in HC$(\mathcal{O}, \mathcal{T}, \Gamma)$ for each $C_i \sqsubseteq D_i \in \Gamma$. Thus, the computed TBox $\mathcal{T}$ is a hybrid-unifier of Γ w.r.t. the ontology $\mathcal{O}$.

Assume that Algorithm 24 has a non-failing run on input Γ, and let ζ be the final assignment computed by this run and $\mathcal{T}$ the induced (cyclic) TBox from ζ. In addition, we denote the final set of p-sequents computed by this run as $\widehat{\Gamma}$.

We prove that for each p-sequent $(C \sqsubseteq_n D, P)$ in $\widehat{\Gamma}$ there is a proof tree for $C \sqsubseteq_n D$ in HC$(\mathcal{O}, \mathcal{T}, \Gamma)$. To show that, we use well founded-induction [9] on the following well-founded $\succ$ order on $\widehat{\Gamma}$.

Definition 25. Let $p = (C \sqsubseteq_n D, P)$ be a p-sequent in $\widehat{\Gamma}$.

- We define $m(p) := (m_1(p), m_2(p))$, where
  - $m_1(p) := n$, where n is from $\sqsubseteq_n$.
  - $m_2(p) := |P|$.
- The strict partial order $\succ$ is the lexicographic order, where the first component is compared w.r.t. the normal order $>$ on natural numbers and the second component is compared with the opposite order $<$ on the finite set of natural numbers $\{0, \ldots, k\}$, where k is at most $|\widehat{\Gamma}|$.
- We extend $\succ$ to $\widehat{\Gamma}$ as: $p_1 \succ p_2$ iff $m(p_1) \succ m(p_2)$.

Notice that the set $\widehat{\Gamma}$ is finite and since every sequent in P corresponds to a p-sequent in $\widehat{\Gamma}$, then P has a finite number of elements. Consequently, the value of k can be properly selected for $\widehat{\Gamma}$.

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

Lemma 26. *If Algorithm 24 outputs a TBox $\mathcal{T}$, then for each p-sequent $(C \sqsubseteq_n D, P)$ in $\widehat{\Gamma}$ there is a proof tree for $C \sqsubseteq_n D$ in* $\mathrm{HC}(\mathcal{O}, \mathcal{T}, \Gamma)$.

Proof. Let $p = (\mathfrak{s}, P) \in \widehat{\Gamma}$ and assume that for all $p' = (\mathfrak{s}', P') \in \widehat{\Gamma}$ with $p \succ p'$, there is a proof tree for $\mathfrak{s}'$ in $\mathrm{HC}(\mathcal{O}, \mathcal{T}, \Gamma)$. We make a two cases distinction w.r.t. $\mathfrak{s}$.

- If $\mathfrak{s}$ has *a non-variable atom on the right-hand side*, then it was initially marked as unsolved and must be solved by a non-failing rule application. Let us see the rules that could have been applied:
  - Eager Axiom Solving: This case is trivially satisfied since $\mathfrak{s}$ is the consequence of an instance of one of the rules (Refl) or (Top) in HC.
  - Eager Ground Solving: $\mathfrak{s}$ is of the form $C_1 \sqcap \ldots \sqcap C_m \sqsubseteq_n D$ and ground. Since there are no defined concepts (variables) occcurring in $C_1, \ldots, C_m, D$, then $C_1 \sqcap \ldots \sqcap C_m \sqsubseteq_{gfp,\mathcal{O},\mathcal{T}} D$ holds for any TBox $\mathcal{T}$. Application of Theorem 5 yields that there is a proof tree for $C_1 \sqcap \ldots \sqcap C_m \sqsubseteq_n D$ in $\mathrm{HC}(\mathcal{O}, \mathcal{T}, \Gamma)$ for all $n \geq 0$.
  - Eager Solving: $\mathfrak{s}$ is of the form $C_1 \sqcap \ldots \sqcap C_m \sqsubseteq_n D$ with $C_i = D$ for some $i \in \{1, \ldots, m\}$. Starting with the sequent $D \sqsubseteq_n D$, several applications of (AndLi) rules yield a proof tree for $\mathfrak{s}$ in $\mathrm{HC}(\mathcal{O}, \mathcal{T}, \Gamma)$. Otherwise, if it is the case that $C_i = X$ and $D \in \zeta_X$, a proof for $\mathfrak{s}$ can be obtained in the following way:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{D \sqsubseteq_n D}\,(\mathrm{Refl})}{\vdots}\,(\mathrm{AndLi})}{\bigsqcap_{E \in \zeta_X} E \sqsubseteq_n D}\,(\mathrm{AndLi})}{X \sqsubseteq_n D}\,(\mathrm{DefL})}{\vdots}\,(\mathrm{AndLi})}{C_1 \sqcap \ldots \sqcap C_m \sqsubseteq_n D}\,(\mathrm{AndLi})$$

  - Decomposition: $\mathfrak{s}$ is of the form $C_1 \sqcap \ldots \sqcap C_m \sqsubseteq_n \exists s.D'$ with $C_i = \exists s.C'$ for some $i \in \{1, \ldots, m\}$. Since *blocking* did not fail when the rule was applied, then there is a p-sequent $p' = (C' \sqsubseteq_n D', P')$ in $\widehat{\Gamma}$ such that

$P \cup \{\mathfrak{s}\} \subseteq P'$. We have that $m_1(p) = m_1(p')$ and that $m_2(p) < m_2(p')$, therefore, $p \succ p'$.

Applying induction we obtain that there is a proof tree for $C' \sqsubseteq_n D'$ in $\mathrm{HC}(\mathcal{O}, \mathcal{T}, \Gamma)$. Thus, an application of the (Ex) rule in HC yields that there is a proof tree for $\exists s.C' \sqsubseteq_n \exists s.D'$ and furthermore, $m-1$ applications of (AndLi) rules give a proof tree for $\mathfrak{s}$ in $\mathrm{HC}(\mathcal{O}, \mathcal{T}, \Gamma)$.

– Extension: $\mathfrak{s}$ is of the form $C_1 \sqcap \ldots \sqcap X \sqcap \ldots \sqcap C_m \sqsubseteq_n D$ for a defined concept X and the definition of X in $\mathcal{T}$ is of the form $X \equiv D_1 \sqcap \ldots \sqcap D \sqcap \ldots \sqcap D_q$. Then, starting with $D \sqsubseteq_n D$ and q applications of (AndLi) rules, we can obtain a proof tree for $D_1 \sqcap \ldots \sqcap D \sqcap \ldots \sqcap D_q \sqsubseteq_n D$. Finally, an application of the rule (DefL) yields a proof tree for $X \sqsubseteq_n D$ and again, subsequent applications of (AndLi) rules give a proof tree for $\mathfrak{s}$ in $\mathrm{HC}(\mathcal{O}, \mathcal{T}, \Delta)$ (see the following diagram).

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{D \sqsubseteq_n D}{\vdots}\,\text{(AndL1)}}{D_1 \sqcap \ldots \sqcap D \sqcap \ldots \sqcap D_q \sqsubseteq_n D}\,\text{(AndL1)}}{X \sqsubseteq_n D}\,\text{(DefL)}}{\vdots}\,\text{(AndL1)}}{C_1 \sqcap \ldots \sqcap X \sqcap \ldots \sqcap C_m \sqsubseteq_n D}\,\text{(AndL1)}$$

– Mutation: $\mathfrak{s}$ is of the form $C_1 \sqcap \ldots \sqcap C_m \sqsubseteq_n D$. Again, since *blocking* did not fail there exists a GCI $E_1 \sqcap \cdots \sqcap E_k \sqsubseteq F$ in $\mathcal{O}$ such that, the p-sequents $p_1 = (C_1 \sqcap \ldots \sqcap C_m \sqsubseteq_n E_1, P_1), \ldots, p_k = (C_1 \sqcap \ldots \sqcap C_m \sqsubseteq_n E_k, P_k)$ and $p_{k+1} = (F \sqsubseteq_n D, P_{k+1})$ are in $\widehat{\Gamma}$ where $P \cup \{\mathfrak{s}\} \subseteq P_1, \ldots, P \cup \{\mathfrak{s}\} \subseteq P_k$ and $P \cup \{\mathfrak{s}\} \subseteq P_{k+1}$. Then, we have that $m_1(p) = m_1(p_1) = \cdots = m_1(p_k) = m_1(p_{k+1})$, $m_2(p) < m_2(p_1), \ldots, m_2(p) < m_2(p_k)$ and $m_2(p) < m_2(p_{k+1})$, therefore $p \succ p_1, \ldots, p \succ p_k$ and $p \succ p_{k+1}$.

We apply induction hypothesis to obtain that there are proof trees $Q_{E_1}, \ldots Q_{E_k}$ and Q_F for $C_1 \sqcap \ldots \sqcap C_m \sqsubseteq_n E_1, \ldots, C_1 \sqcap \ldots \sqcap C_m \sqsubseteq_n E_k$ and $F \sqsubseteq_n D$ respectively. Thus, several applications of (AndR) rule and an application of the (GCI) rule yield a proof tree for $C_1 \sqcap \ldots \sqcap C_m \sqsubseteq_n D$ in $\mathrm{HC}(\mathcal{O}, \mathcal{T}, \Gamma)$ (see the following diagram).

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{Q_{E_1}}{C_1 \sqcap \ldots \sqcap C_m \sqsubseteq_n E_1} \quad \dfrac{Q_{E_2}}{C_1 \sqcap \ldots \sqcap C_m \sqsubseteq_n E_2}}{C_1 \sqcap \ldots \sqcap C_m \sqsubseteq_n E_1 \sqcap E_2}\,\text{(AndR)} \quad \ddots}{\ddots}}{C_1 \sqcap \ldots \sqcap C_m \sqsubseteq_n E_1 \sqcap \cdots \sqcap E_{k-1}}\,\text{(AndR)} \quad \dfrac{Q_{E_k}}{C_1 \sqcap \ldots \sqcap C_m \sqsubseteq_n E_k}}{C_1 \sqcap \ldots \sqcap C_m \sqsubseteq_n E_1 \sqcap \cdots \sqcap E_{k-1} \sqcap E_k}\,\text{(AndR)} \quad \dfrac{Q_F}{F \sqsubseteq_n D}}{C_1 \sqcap \ldots \sqcap C_m \sqsubseteq_n D}\,\text{(GCI)}$$

- If $\mathfrak{s}$ has *a variable as its right-hand side*, then it is of the form $C_1 \sqcap \ldots \sqcap C_m \sqsubseteq_n X$. If ζ_X is empty, then $X \equiv \top$ is the definition of X in $\mathcal{T}$. Obviously, there is a proof tree for $C_1 \sqcap \ldots \sqcap C_m \sqsubseteq_{n-1} \top$ and the rest follows applying rule (DefR) in HC.

  Otherwise, the definition of X in $\mathcal{T}$ is of the form $X \equiv D_1 \sqcap \ldots \sqcap D_q$ and for every D_i there is a p-sequent of the form $p_i = (C_1 \sqcap \ldots \sqcap C_m \sqsubseteq_{n-1} D_i, P)$ in $\widehat{\Gamma}$, because $\widehat{\Gamma}$ is *expanded* w.r.t. X.

  Clearly, $m_1(p) > m_1(p_i)$ and therefore, $m(p) \succ m(p_i)$ for all $i, 1 \leq i \leq q$.

  Now, by induction there is a proof tree for each sequent $C_1 \sqcap \ldots \sqcap C_m \sqsubseteq_{n-1} D_i$. A series of applications of rule (AndR) from HC yield a proof tree Q for $C_1 \sqcap \ldots \sqcap C_m \sqsubseteq_{n-1} D_1 \sqcap \ldots \sqcap D_q$. Then, a proof tree for $C_1 \sqcap \ldots \sqcap C_m \sqsubseteq_n X$ is obtained as follows

$$\dfrac{\dfrac{Q}{C_1 \sqcap \ldots \sqcap C_m \sqsubseteq_{n-1} D_1 \sqcap \ldots \sqcap D_q}}{C_1 \sqcap \ldots \sqcap C_m \sqsubseteq_n X}\ \text{(DefR)}$$

□

The proof of Lemma 26 shows how to construct a proof for a sequent $C \sqsubseteq_n D$ in $\text{HC}(\mathcal{O}, \mathcal{T}, \Gamma)$, if $C \sqsubseteq_n D$ is the first component of a p-sequent in $\widehat{\Gamma}$ and provided Algorithm 24 terminates successfully. Since the initialization of Algorithm 24 adds a p-sequent of the form $(C_i \sqsubseteq_\ell D_i, \emptyset)$ to $\Gamma_p^{(0)}$ for each subsumption $C_i \sqsubseteq D_i$ in Γ and p-sequents with $C_i \sqsubseteq_\ell D_i$ as the first components are in $\widehat{\Gamma}$, this means that we know that $C_i \sqsubseteq_\ell D_i$ is derivable in $\text{HC}(\mathcal{O}, \mathcal{T}, \Gamma)$.

We also know that there is a number ℓ such that if a sequent $C \sqsubseteq_\ell D$ is derivable in $\text{HC}(\mathcal{O}, \mathcal{T}, \Gamma)$, $C \sqsubseteq_n D$ is derivable in $\text{HC}(\mathcal{O}, \mathcal{T}, \Gamma)$ for all $n \geq 0$ and thus $C \sqsubseteq_\infty D$ holds in $\text{HC}(\mathcal{O}, \mathcal{T}, \Gamma)$. Hence if we know such number ℓ which has this property, we can run Algorithm 24 with this ℓ as index for the p-sequents in $\Gamma_p^{(0)}$ and use Theorem 5 to infer that $C_i \sqsubseteq_{gfp,\mathcal{O},\mathcal{T}} D_i$ for each subsumption in Γ. This implies that $\mathcal{T}$ is a unifier of Γ with respect to $\mathcal{O}$.

We will now estimate the proper value of ℓ. The numerical index n in a sequent $C \sqsubseteq_n D$ is meant to control the number of allowed applications of the rule (DefR). This rule is increasing such indices and thus allows some paths in a proof tree to reach the root from the leaves with index 0, i.e. the leaves that are instances of (Start).

Notice that the rule (DefR) has the following form:

$$\frac{D \sqsubseteq_{n-1} C_1 \sqcap \cdots \sqcap C_k}{D \sqsubseteq_n X}$$

where $X \equiv C_1 \sqcap \cdots \sqcap C_k$ is in $\mathcal{T}$.

Now we want to estimate a maximal number of subsumptions of the form $D \sqsubseteq X$ that can appear with an index n in the sequent $D \sqsubseteq_n X$ in p-sequents $\widehat{\Gamma}$.

In order to do it, we notice that there are only $|N_{def}|$ possibilities for the right hand side of such a subsumption.

Now we consider the possible concepts occurring as the left hand sides in the first components of the p-sequents of the form $D \sqsubseteq_n X$ in $\widehat{\Gamma}$. By inspecting the inference rules of Algorithm 24, we can see that such D can be only one of the following.

1. D is the left hand side of a subsumption in Γ or
2. D is an atom of Γ (Decomposition) or
3. D is an atom of $\mathcal{O}$ (Mutation).

Hence the number of the subsumptions of the form $D \sqsubseteq X$ that our algorithm can use is less than

$$\ell = ((\#(\Gamma) + |At|) \times |N_{def}|) + 1$$

where $\#(\Gamma)$ is the number of subsumptions in Γ.

Before we come to proving soundness of our algorithm we need one more definition and easy observation.

In the next definition we describe a method that can be used to obtain a proof tree for a sequent $C \sqsubseteq_n D$ starting from a proof tree for $C \sqsubseteq_m D$, provided that $n < m$.

Definition 27. Let $\mathcal{P}$ be a proof tree for a sequent $C \sqsubseteq_m D$ in HC$(\mathcal{O}, \mathcal{T}, \Delta)$.

The following construction is called *trimming of proof tree* $\mathcal{P}$.

1. Let $n' := m - n$.
   Subtract n' from all indices in the sequents occurring in $\mathcal{P}$.
2. Delete all successors of nodes with indices equal to 0.

Example 28. This example illustrates the above described method of *trimming a proof tree.*

Let $\mathcal{O} := \{\exists r.B \sqsubseteq A\}$ and let $\mathcal{T} := \{X \equiv \exists r.X \sqcap A\}$.

We have the following proof tree for $\exists r.B \sqsubseteq_2 X$ in HC$(\mathcal{O}, \mathcal{T}, \Delta)$, where $\exists r.B \sqsubseteq_2 X \in \Delta$:

$$\dfrac{\dfrac{\dfrac{\dfrac{}{B \sqsubseteq_0 \exists r.X \sqcap A}\,\text{(Start)}}{B \sqsubseteq_1 X}\,\text{(DefR)}}{\exists r.B \sqsubseteq_1 \exists r.X}\,\text{(Ex)} \qquad \dfrac{\dfrac{}{\exists r.B \sqsubseteq_1 \exists r.B}\,\text{(Refl)} \quad \dfrac{}{A \sqsubseteq_1 A}\,\text{(Refl)}}{\exists r.B \sqsubseteq_1 A}\,\text{(GCI)}}{\dfrac{\exists r.B \sqsubseteq_1 \exists r.X \sqcap A}{\exists r.B \sqsubseteq_2 X}\,\text{(DefR)}}\,\text{(AndR)}$$

In order to obtain the proof tree for $\exists r.B \sqsubseteq_1 X$ from the above proof tree first we substract 1 from all the indices in the sequents in the proof:

$$\dfrac{\dfrac{\dfrac{\dfrac{}{B \sqsubseteq_{-1} \exists r.X \sqcap A}\,\text{(Start)}}{B \sqsubseteq_0 X}\,\text{(DefR)}}{\exists r.B \sqsubseteq_0 \exists r.X}\,\text{(Ex)} \qquad \dfrac{\dfrac{}{\exists r.B \sqsubseteq_0 \exists r.B}\,\text{(Refl)} \quad \dfrac{}{A \sqsubseteq_0 A}\,\text{(Refl)}}{\exists r.B \sqsubseteq_0 A}\,\text{(GCI)}}{\dfrac{\exists r.B \sqsubseteq_0 \exists r.X \sqcap A}{\exists r.B \sqsubseteq_1 X}\,\text{(DefR)}}\,\text{(AndR)}$$

Now in order to obtain a *trimmed* proof tree we delete all successors of sequents with index 0:

$$\dfrac{\dfrac{}{\exists r.B \sqsubseteq_0 \exists r.X \sqcap A}\,\text{(Start)}}{\exists r.B \sqsubseteq_1 X}\,\text{(DefR)}$$

In the soundness lemma below we show that the number ℓ is big enough for our algorithm to yield the correct unifier.

Lemma 29. *Let $\mathcal{O}$ be a flat general TBox and $\Gamma = \{C_1 \sqsubseteq D_1, \ldots, C_m \sqsubseteq D_m\}$ be a flat unification problem. If Algorithm 24 with the choice of ℓ as above outputs a TBox $\mathcal{T}$ on input Γ, then $\mathcal{T}$ is a hybrid-unifier of Γ w.r.t. $\mathcal{O}$.*

Proof. Assuming that Algorithm 24 with ℓ computed as above has a successful run on the unification problem Γ, by Lemma 26 we know that for each subsumption $C_i \sqsubseteq D_i \in \Gamma$ we have a proof tree $\mathcal{P}$ of $C_i \sqsubseteq_\ell D_i$ in $\mathrm{HC}(\mathcal{O}, \mathcal{T}, \Gamma)$ constructed as in the proof of Lemma 26. We will call such a proof tree, a proof tree induced by Algorithm 24. Now we will show that having such a proof tree, we can construct a proof tree for $C_i \sqsubseteq_n D_i$ for any $n \geq 0$.

First of all, it is clear that having the proof tree $\mathcal{P}$ we can construct a proof tree for $C_i \sqsubseteq_n D_i$ for any $n \leq \ell$, by *trimming* the proof tree $\mathcal{P}$ (Definition 27).

Now we show how to construct a proof tree for $n > \ell$. We have to look at the structure of the proof $\mathcal{P}$. The construction is by induction on n', where $n = \ell + n'$.

The assumption that we make about the proof tree $\mathcal{P}$ is that the sequents of the form $D \sqsubseteq_k X$ that appear in the proof tree are also in the p-sequents in $\widehat{\Gamma}$. This is satisfied in the base case, i.e. if $n' = 1$, because the proof for $C_i \sqsubseteq_\ell D_i$ is induced by Algorithm 24.

Now consider the leaves of $\mathcal{P}$, i.e. instances of (Refl), (Top), (Start). If there are no (Start) instances among the leaves of $\mathcal{P}$, then we obtain a proof tree for $C_i \sqsubseteq_n D_i$, where $n = \ell + n'$ and $n' > 0$ by adding n' to all indices in the sequents of the proof tree $\mathcal{P}$. It is easy to see that we obtain a valid proof tree in this way. The proof tree satisfies our assumption about the sequents of the form $D \sqsubseteq_k X$ too.

Example 30. Let $\mathcal{O} := \{\exists r.B \sqsubseteq A\}$ and $\mathcal{T} := \{X \equiv \exists r.B\}$. Let the proof tree for $X \sqsubseteq_1 A$ be the following:

$$\dfrac{\dfrac{\dfrac{}{\exists r.B \sqsubseteq_1 \exists r.B}\,\text{(Refl)}}{X \sqsubseteq_1 \exists r.B}\,\text{(DefL)} \qquad \dfrac{}{A \sqsubseteq_1 A}\,\text{(Refl)}}{X \sqsubseteq_1 A}\,\text{(GCI)}$$

The proof tree has no (Start) instances in the leaves, hence it can be easily reused for the proof tree of the same sequent with a bigger index, e.g. :

$$\dfrac{\dfrac{\dfrac{}{\exists r.B \sqsubseteq_4 \exists r.B}\,\text{(Refl)}}{X \sqsubseteq_4 \exists r.B}\,\text{(DefL)} \qquad \dfrac{}{A \sqsubseteq_4 A}\,\text{(Refl)}}{X \sqsubseteq_4 A}\,\text{(GCI)}$$

Now we consider the case where there are leaves in $\mathcal{P}$ which are instances of (Start).

Consider a path from such a leaf to the root $C_i \sqsubseteq_\ell D_i$. There are ℓ applications of (DefR) on the path. Hence there are ℓ consequences of these applications of the form $D \sqsubseteq_k X$. It is easy to see that all consequences of (DefR) in the proofs induced by Algorithm 24 are in $\widehat{\Gamma}$. Since ℓ is bigger than the number of possible subsumptions of the form $D \sqsubseteq X$ explored by the algorithm, there are at least two nodes with $D \sqsubseteq_i X$ and $D \sqsubseteq_j X$ as conclusions of (DefR) with $i > j$ on this path.

We choose such pair of repeated nodes for each path. Let the node with a smaller index (e.g. $D \sqsubseteq_j X$) be called *pumping point* and the node with the bigger index be called a *node associated* with the pumping point.

We choose such pumping points for each path starting with a (Start) instance, one for each path. Notice that if there are multiple (Start) instances in the proof tree, several paths may share such point provided the node is shared by these paths.

We then perform two actions.

1. We increase the indices in the sequents below pumping points and on the paths that do not lead to (Start) instances by 1.

2. Replace subtree rooted in a pumping point on each path by the sub-tree rooted at the node associated with this pumping point trimmed to the proof tree for $D \sqsubseteq_{j+1} X$.

Now, since $i > j$, $i \geq j + 1$, hence the proof tree for $D \sqsubseteq_i X$ can always be trimmed to the correct proof tree for $D \sqsubseteq_{j+1} X$.

In this way we can obtain a proof tree for $C_i \sqsubseteq_{\ell+1} D_i$ from the proof tree induced by Algorithm 24 and hence this new proof tree satisfies the assumption about the sequents of the form $D \sqsubseteq_k X$.

Example 31. Let $\mathcal{O} = \{B \sqsubseteq \exists s.B\}$ and $\mathcal{T} = \{X \equiv \exists s.X\}$. We show a proof tree for $\exists s.B \sqsubseteq_2 \exists s.X$ and show how to obtain a proof tree for $\exists s.B \sqsubseteq_3 \exists s.X$ using the ideas of the above construction.

$$\dfrac{\dfrac{\dfrac{\dfrac{}{B \sqsubseteq_1 B}\,(\text{Refl}) \qquad \dfrac{\dfrac{\dfrac{}{B \sqsubseteq_0 \exists s.X}\,(\text{Start})}{B \sqsubseteq_1 X}\,(\text{DefR})}{\exists s.B \sqsubseteq_1 \exists s.X}\,(\text{Ex})}{B \sqsubseteq_1 \exists s.X}\,(\text{GCI})}{B \sqsubseteq_2 X}\,(\text{DefR})}{\exists s.B \sqsubseteq_2 \exists s.X}\,(\text{Ex})$$

This proof tree has a (Start) instance as a leaf. We choose $B \sqsubseteq_1 X$ as a pumping point and $B \sqsubseteq_2 X$ as the associated node.

First we increase all indices in the sequents that are below the pumping point or are not on the path to (Start).

$$\dfrac{\dfrac{\dfrac{\dfrac{}{B \sqsubseteq_2 B}\,(\text{Refl}) \qquad \dfrac{\dfrac{\dfrac{}{B \sqsubseteq_0 \exists s.X}\,(\text{Start})}{B \sqsubseteq_1 X}\,(\text{DefR})}{\exists s.B \sqsubseteq_2 \exists s.X}\,(\text{Ex})}{B \sqsubseteq_2 \exists s.X}\,(\text{GCI})}{B \sqsubseteq_3 X}\,(\text{DefR})}{\exists s.B \sqsubseteq_3 \exists s.X}\,(\text{Ex})$$

We want to reuse the proof tree rooted in $B \sqsubseteq_2 X$ for $B \sqsubseteq_1 X$ with the index increased by 1. It happens in this example that $i = j + 1$, i.e. we can reuse the proof for $B \sqsubseteq_2 X$ without *trimming* (this may be different if the pumping point and its associated node are seperated by some other applications of (DefR) in other proof trees).

The new proof tree has thus the following form.

$$
\begin{prooftree}
\AxiomC{}
\RightLabel{(Refl)}
\UnaryInfC{$B \sqsubseteq_2 B$}
\AxiomC{}
\RightLabel{(Refl)}
\UnaryInfC{$B \sqsubseteq_1 B$}
\AxiomC{}
\RightLabel{(Start)}
\UnaryInfC{$B \sqsubseteq_0 \exists s.X$}
\RightLabel{(DefR)}
\UnaryInfC{$B \sqsubseteq_1 X$}
\RightLabel{(Ex)}
\UnaryInfC{$\exists s.B \sqsubseteq_1 \exists s.X$}
\RightLabel{(GCI)}
\BinaryInfC{$B \sqsubseteq_1 \exists s.X$}
\RightLabel{(DefR)}
\UnaryInfC{$B \sqsubseteq_2 X$}
\RightLabel{(Ex)}
\UnaryInfC{$\exists s.B \sqsubseteq_2 \exists s.X$}
\RightLabel{(GCI)}
\BinaryInfC{$B \sqsubseteq_2 \exists s.X$}
\RightLabel{(DefR)}
\UnaryInfC{$B \sqsubseteq_3 X$}
\RightLabel{(Ex)}
\UnaryInfC{$\exists s.B \sqsubseteq_3 \exists s.X$}
\end{prooftree}
$$

Now, let us assume that we have already constructed a proof tree $\mathcal{P}'$ for $C_i \sqsubseteq_n D_i$ with $n > \ell$ such that every sequent of the form $D \sqsubseteq_k X$ that appears in $\mathcal{P}'$ appears also in a p-sequent in $\widehat{\Gamma}$. We show how to construct a proof tree for $C_i \sqsubseteq_{n+1} D_i$ with the same property.

To do this, it is enough to find all nodes $D \sqsubseteq_\ell C$ in $\mathcal{P}'$, such that there is no another node $D' \sqsubseteq_\ell C'$ in $\mathcal{P}'$ which is a predecessor of $D \sqsubseteq_\ell C$ (i.e. the nodes with index ℓ which are closest to the root). Note that the sub-trees rooted at such nodes satisfy the property about the sequents of the form $D \sqsubseteq_k X$, and thus we can apply to them the base case construction.

Lets call such nodes *pumping points* for $\mathcal{P}'$. In order to obtain a proof tree for $C_i \sqsubseteq_{n+1} D_i$ we again perform two actions.

1. We increase the indices in the sequents below the pumping points and on the paths that do not lead to pumping points by 1.

2. Replace subtrees rooted in each pumping point $D \sqsubseteq_\ell C$ by a proof tree for $D \sqsubseteq_{\ell+1} C$ obtained as in the construction in the base case.

Notice that since the proofs that we have plugged into the pumping points satisfy the property about $D \sqsubseteq_k X$ nodes, the new proof tree satisfies this property too.

Now the statement of the lemma follows immediately. Since for each $C_i \sqsubseteq D_i \in \Gamma$ and each $n \geq 0$ we can construct a proof tree for $C_i \sqsubseteq_n D_i$ in $\mathrm{HC}(\mathcal{O}, \mathcal{T}, \Gamma)$, hence $C_i \sqsubseteq_\infty D_i$ holds in $\mathrm{HC}(\mathcal{O}, \mathcal{T}, \Gamma)$. By Theorem 5, we know that $C_i \sqsubseteq_{gfp,\mathcal{O},\mathcal{T}} D_i$ for each $C_i \sqsubseteq D_i \in \Gamma$ and thus $\mathcal{T}$ is a hybrid unifier of Γ w.r.t. $\mathcal{O}$.

□

6.2 Some properties of proof trees II

In this section we show some properties of proof trees in $\mathrm{HC}(\mathcal{O}, \mathcal{T}, \Gamma)$ which will be used as auxiliary lemmas when proving completeness of Algorithm 24 in the next section. The reader is advised to skip this section and to return to it when needed.

In Lemma 3.1.2 from [15], the following two statements were shown:

1. $C \sqsubseteq_n D_1 \sqcap D_2$ iff $C \sqsubseteq_n D_1$ and $C \sqsubseteq_n D_2$.
2. $C \sqsubseteq_{n+1} X$ iff $C \sqsubseteq_n D_X$, where $X \equiv D_X \in \mathcal{T}$.

In particular, these two statements say that whenever there exists a proof tree for $C \sqsubseteq_n D_1 \sqcap D_2$ ($C \sqsubseteq_{n+1} X$) there is also a proof tree for $C \sqsubseteq_n D_i$ ($C \sqsubseteq_n D_X$), $1 \leq i \leq 2$. In the following proposition we show that, in addition, for the first case the number of (GCI) rule applications on those new trees do not need to increase (for the second case, the same proposition can also be shown in a similar way).

Lemma 32. *Suppose that $n > 0$. If $\mathcal{P}$ is a proof tree for $C \sqsubseteq_n D_1 \sqcap D_2$ and p is the number of (GCI) rule applications occurring in $\mathcal{P}$. Then, there is a proof tree $\mathcal{P}_i (1 \leq i \leq 2)$ for $C \sqsubseteq_n D_i$ with number of (GCI) rule applications $p_i \leq p$.*

Proof. We use the same inductive argumentation as the "only if" direction in Lemma 3.1.2(1) from [15]. Consider the cases representing the last rule used to derive $C \sqsubseteq_n D_1 \sqcap D_2$:

- Last rule used: (Refl). Then, C is of the form $D_1 \sqcap D_2$ and $\mathcal{P}$:

$$\frac{}{D_1 \sqcap D_2 \sqsubseteq_n D_1 \sqcap D_2}\text{(Refl)}$$

  Then, using (Refl) and (AndLi) we obtain the proof tree:

$$\frac{\frac{}{D_i \sqsubseteq_n D_i}\text{(Refl)}}{D_1 \sqcap D_2 \sqsubseteq_n D_i}\text{(AndLi)}$$

  Since no (GCI) rule is used, the claim trivially holds.

- Last rule used: (AndR), (AndLi) or (DefL). The case for the rule (AndR) is trivial because the proof trees already exist as subtrees in $\mathcal{P}$. For the other two cases, since the new tree is completed not using an application of the (GCI) rule after applying induction, then the claim also holds.

- Last rule used: (GCI). Then, $\mathcal{P}$ is of the following form:

$$\frac{\frac{\vdots}{C \sqsubseteq_n E} \quad \frac{\vdots}{F \sqsubseteq_n D_1 \sqcap D_2}}{C \sqsubseteq_n D_1 \sqcap D_2}\text{(GCI) with } E \sqsubseteq F \in \mathcal{O}$$

Let Q_1 be the subtree rooted at $C \sqsubseteq_n E$, Q_2 the one rooted at $F \sqsubseteq_n D_1 \sqcap D_2$ and $q_j (1 \leq j \leq 2)$ be the number of (GCI) rule applications in Q_j. Since Q_2 is a proof tree for the sequent $F \sqsubseteq_n D_1 \sqcap D_2$ then, induction hypothesis yields that there is a proof tree $Q_{2i} (1 \leq i \leq 2)$ with number of (GCI) rule applications $q_{2i} \leq q_2$ for the sequent $F \sqsubseteq_n D_i$. We build a proof tree $\mathcal{P}_i$ for $C \sqsubseteq_n D_i$ as follows:

$$\dfrac{\dfrac{Q_1}{C \sqsubseteq_n E} \quad \dfrac{Q_{2i}}{F \sqsubseteq_n D_i}}{C \sqsubseteq_n D_i} \text{(GCI) with } E \sqsubseteq F \in \mathcal{O}$$

Since $p = q_1 + q_2 + 1$, $p_i = q_1 + q_{2i} + 1$ and $q_{2i} \leq q_2$ then $p_i \leq p$.

$\square$

Next, we define the notion of *good* derivation and show that derivable sequents in $\text{HC}(\mathcal{O}, \mathcal{T}, \Delta)$ always have *good* derivations in $\text{HC}(\mathcal{O}, \mathcal{T}, \Delta)$.

Definition 33. Let $\mathcal{P}$ be a proof tree in $\text{HC}(\mathcal{O}, \mathcal{T}, \Delta)$ and $\mathfrak{s}$ a sequent occurring in $\mathcal{P}$. We say that $\mathfrak{s}$ is *bad* for $\mathcal{P}$ if it is not an instance of one of the rules (Refl), (Start) or (Top) and one of the following conditions hold:

1. $\mathfrak{s}$ is of the form $C \sqsubseteq_n D_1 \sqcap D_2$ and it is derived in $\mathcal{P}$ in the following way:

$$\dfrac{\dfrac{\vdots}{E \sqsubseteq_n F}}{C \sqsubseteq_n D_1 \sqcap D_2} (R \neq \text{AndR})$$

   i.e.: (AndR) is not the last rule used to derive $C \sqsubseteq_n D_1 \sqcap D_2$.

2. $\mathfrak{s}$ is of the form $C_1 \sqcap C_2 \sqsubseteq_n D$ and it is derived in $\mathcal{P}$ in the following way:

$$\dfrac{\dfrac{\dfrac{\vdots}{C_i \sqsubseteq_n E} \quad \dfrac{\vdots}{F \sqsubseteq_n D}}{C_i \sqsubseteq_n D} \text{(GCI) with } E \sqsubseteq F \in \mathcal{O}}{C_1 \sqcap C_2 \sqsubseteq_n D} \text{(AndLi)}$$

Otherwise, the sequent is called *good* for $\mathcal{P}$. In addition, $\mathcal{P}$ is called a *good* proof tree if there is no *bad* sequent occurring on it. Finally, we say that a sequent $C \sqsubseteq_n D$ has a *good* derivation in $\text{HC}(\mathcal{O}, \mathcal{T}, \Delta)$ if there is a *good* proof tree for $C \sqsubseteq_n D$ in $\text{HC}(\mathcal{O}, \mathcal{T}, \Delta)$.

We show that for each proof tree $\mathcal{P}$ in $\mathrm{HC}(\mathcal{O}, \mathcal{T}, \Delta)$, rooted at a sequent $C \sqsubseteq_n D$, there is a *good* proof tree $\mathcal{P}'$ for $C \sqsubseteq_n D$. We use well founded-induction [9] on the following well-founded $\succ$ order on the set of proof trees in $\mathrm{HC}(\mathcal{O}, \mathcal{T}, \Delta)$.

Definition 34. Let $\mathcal{P}$ be a proof tree for a sequent $C \sqsubseteq_n D$ in $\mathrm{HC}(\mathcal{O}, \mathcal{T}, \Delta)$.

- We define $m(\mathcal{P}) := (m_1(\mathcal{P}), m_2(\mathcal{P}), m_3(\mathcal{P}), m_4(\mathcal{P}))$, where
  - $m_1(\mathcal{P}) := n$, where n is from $\sqsubseteq_n$.
  - $m_2(\mathcal{P}) := p$, where p is the number of (GCI) rule applications occurring in $\mathcal{P}$.
  - $m_3(\mathcal{P}) :=\mid D \mid$, the size of the concept description D.
  - $m_4(\mathcal{P}) :=\mid \mathcal{P} \mid$, the size of the proof tree $\mathcal{P}$.
- The strict partial order $\succ$ is the lexicographic order, where all components in $m(\mathcal{P})$ are compared w.r.t. the normal order $>$ on natural numbers.
- We extend $\succ$ to the set of proof trees in $\mathrm{HC}(\mathcal{O}, \mathcal{T}, \Delta)$ as: $\mathcal{P}_1 \succ \mathcal{P}_2$ iff $m(\mathcal{P}_1) \succ m(\mathcal{P}_2)$.

Since the lexicographic product of well-founded strict partial orders is again well-founded [9], then $\succ$ is a well-founded strict partial order on the set of proof trees in $\mathrm{HC}(\mathcal{O}, \mathcal{T}, \Delta)$.

Lemma 35. *For each proof tree in* $\mathrm{HC}(\mathcal{O}, \mathcal{T}, \Delta)$*, rooted at a sequent* $\mathfrak{s}$*, there exists a* good *proof tree in* $\mathrm{HC}(\mathcal{O}, \mathcal{T}, \Delta)$ *for the same sequent.*

Proof. Let $\mathcal{P}$ be a proof tree in $\mathrm{HC}(\mathcal{O}, \mathcal{T}, \Delta)$ for some sequent $\mathfrak{s}$, $m(\mathcal{P}) = (m_1, m_2, m_3, m_4)$ and assume that for all proof trees $\mathcal{P}'$ with $\mathcal{P} \succ \mathcal{P}'$ the claim holds. First, we can assume that $\mathcal{P}$ is not a one-element proof tree, otherwise it would be already a *good* proof tree. Second, we do a case distinction on the form of the sequent $\mathfrak{s}$:

- $\mathfrak{s}$ is a sequent that can be derived directly by an application of one of the rules (Refl), (Top) or (Start). It is clear that there is a *good* proof tree for $\mathfrak{s}$.
- $\mathfrak{s}$ is of the form $C \sqsubseteq_n D_1 \sqcap D_2$ where C, D_1, D_2 are concept descriptions. Application of Lemma 32 yields that there exists a proof tree $\mathcal{P}_i (1 \leq i \leq 2)$ for $C \sqsubseteq_n D_i$ such that the number of (GCI) rule applications p_i in $\mathcal{P}_i$ is at most the one in $\mathcal{P}$, i.e., $p_i \leq m_2$. In addition, since $\mid D_i \mid < \mid D_1 \sqcap D_2 \mid$ one can see that $m(\mathcal{P}_i)$ is of the form $(m_1, m_{2i}, m_{3i}, \_)$ with $m_{2i} \leq m_2$ and $m_{3i} < m_3$. Therefore, $m(\mathcal{P}) \succ m(\mathcal{P}_i)$, $\mathcal{P} \succ \mathcal{P}_i$ and thus, induction can be applied to $\mathcal{P}_i$ to obtain a *good* proof tree Q_i in $\mathrm{HC}(\mathcal{O}, \mathcal{T}, \Delta)$ for $C \sqsubseteq_n D_i$.

  Then, we can obtain a new proof tree for $C \sqsubseteq_n D_1 \sqcap D_2$ in the following way:

$$\dfrac{\dfrac{Q_1}{C \sqsubseteq_n D_1} \qquad \dfrac{Q_2}{C \sqsubseteq_n D_1}}{C \sqsubseteq_n D_1 \sqcap D_2}\ \text{(AndR)}$$

and, since Q_i is a *good* proof tree for $C \sqsubseteq_n D_i$ and (AndR) is used to derive $C \sqsubseteq_n D_1 \sqcap D_2$, then the obtained proof tree is *good*.

- $\mathfrak{s}$ is of the form $C \sqsubseteq_{n+1} X$ where C is a concept description and $X \in N_{def}$. Application of Lemma 3.1.2 (2) from [15] yields that there exists a proof tree $\mathcal{P}'$ for $C \sqsubseteq_n D_X$ where $X \equiv D_X \in \mathcal{T}$. In addition, since $n < n+1$ one can see that $m(\mathcal{P}')$ is of the form $(m_{1'}, \_, \_, \_)$ with $m_{1'} < m_1$. Therefore, $m(\mathcal{P}) \succ m(\mathcal{P}')$, $\mathcal{P} \succ \mathcal{P}'$ and thus, induction can be applied to $\mathcal{P}'$ to obtain a *good* proof tree Q in HC$(\mathcal{O}, \mathcal{T}, \Delta)$ for $C \sqsubseteq_n D_X$.

  Then, we can obtain a new proof tree for $C \sqsubseteq_{n+1} X$ in the following way:

  $$\dfrac{\dfrac{Q}{C \sqsubseteq_n D_X}}{C \sqsubseteq_{n+1} X}\ \text{(DefR) with } X \equiv D_X \in \mathcal{T}$$

  and, since Q is a *good* proof tree for $C \sqsubseteq_n D_X$ and (DefR) is the last rule used to derive $C \sqsubseteq_{n+1} X$, then the obtained proof tree is *good*.

- $\mathfrak{s}$ is of the form $C \sqsubseteq_n D$ where C is a concept description and D is a *non-variable* atom. We do a case distinction on the last rule used to derive $C \sqsubseteq_n D$ in $\mathcal{P}$:

  – Last rule used: (Ex). Then, $\mathfrak{s}$ is of the form $\exists r.C' \sqsubseteq_n \exists r.D'$ and $\mathcal{P}$:

  $$\dfrac{\dfrac{\vdots}{C' \sqsubseteq_n D'}}{\exists r.C' \sqsubseteq_n \exists r.D'}\ \text{(Ex)}$$

  It is clear that the proof tree rooted at $C' \sqsubseteq_n D'$ is smaller than $\mathcal{P}$ since $\mid D' \mid < \mid \exists r.D' \mid$. Then, application of induction yields a *good* proof tree for $C' \sqsubseteq_n D'$ and a further application of the (Ex) rule gives a *good* proof tree for $\exists r.C' \sqsubseteq_n \exists r.D'$.

  – Last rule used: (DefL). Then, $\mathfrak{s}$ is of the form $X \sqsubseteq_n D$ and $\mathcal{P}$:

  $$\dfrac{\dfrac{\vdots}{D_X \sqsubseteq_n D}}{X \sqsubseteq_n D}\ \text{(DefL) with } X \equiv D_X \in \mathcal{T}$$

  Since the size of the proof tree rooted at $D_X \sqsubseteq_n D$ is smaller than the size of $\mathcal{P}$, a similar argument as for the (Ex) rule yields a *good* proof tree for $X \sqsubseteq_n D$.

– Last rule used: (GCI). Then, $\mathcal{P}$ is of the form:

$$\dfrac{\dfrac{\vdots}{C \sqsubseteq_n E} \qquad \dfrac{\vdots}{F \sqsubseteq_n D}}{C \sqsubseteq_n D}\ \text{(GCI) with } E \sqsubseteq F \in \mathcal{O}$$

Obviously the proof trees rooted at $C \sqsubseteq_n E$ and $F \sqsubseteq_n D$ contain strictly less number of (GCI) rule applications than $\mathcal{P}$ and hence, they are smaller than $\mathcal{P}$. The application of induction to them and a further (GCI) rule application give a *good* proof tree for $C \sqsubseteq_n D$.

– Last rule used: (AndLi). Then, $\mathcal{P}$ is of the form:

$$\dfrac{\dfrac{\vdots}{C_i \sqsubseteq_n D}}{C_1 \sqcap C_2 \sqsubseteq_n D}\ \text{(AndLi)}$$

In this case it is not possible to use the same argument as for the previous cases. Note that, although the proof tree rooted at $C_i \sqsubseteq_n D$ is smaller than $\mathcal{P}$ and the application of induction yields a *good* proof tree for $C_i \sqsubseteq_n D$, the last rule used in such a tree might be (GCI) and then (AndLi) cannot be used to complete a *good* proof tree for $C_1 \sqcap C_2 \sqsubseteq_n D$ (see Definition 33 (3)). We look at the last rule used in $\mathcal{P}$ that is not (AndLi), i.e.:

$$\dfrac{\dfrac{\dfrac{\dfrac{\vdots}{E \sqsubseteq_n F}}{C_i \sqcap \ldots \sqcap C_j \sqsubseteq_n D}\ (R \neq \text{AndLi})}{\vdots}\ \text{(AndLi)}}{C_1 \sqcap \ldots \sqcap C_m \sqsubseteq_n D}\ \text{(AndLi)}$$

If R is (Refl) then $\mathcal{P}$ is already a *good* proof tree. The other three possible rules are (Ex), (DefL) and (GCI). For the rules (Ex) and (DefL), similar as before one can see that the proof tree rooted at the sequent $E \sqsubseteq_n F$ is smaller than $\mathcal{P}$. The application of induction yields a *good* proof tree for $E \sqsubseteq_n F$ which can replace the one in $\mathcal{P}$ to obtain a *good* proof tree for $C_1 \sqcap \ldots \sqcap C_m \sqsubseteq_n D$ since $R \neq$ AndLi. For the remaining case, $\mathcal{P}$ has the following form:

$$\dfrac{\dfrac{\dfrac{\dfrac{\vdots}{C_i \sqcap \ldots \sqcap C_j \sqsubseteq_n E} \qquad \dfrac{\vdots}{F \sqsubseteq_n D}}{C_i \sqcap \ldots \sqcap C_j \sqsubseteq_n D}\ \text{(GCI) with } E \sqsubseteq F \in \mathcal{O}}{\vdots}\ \text{(AndLi)}}{C_1 \sqcap \ldots \sqcap C_m \sqsubseteq_n D}\ \text{(AndLi)}$$

We can transform that proof tree to obtain a new proof tree for $C_1 \sqcap \ldots \sqcap C_m \sqsubseteq_n D$ that has the following form:

$$\dfrac{\dfrac{\dfrac{\dfrac{\vdots}{C_i \sqcap \ldots \sqcap C_j \sqsubseteq_n E}}{\vdots}\,\text{(AndLi)}}{C_1 \sqcap \ldots \sqcap C_m \sqsubseteq_n E}\,\text{(AndLi)} \qquad \dfrac{\vdots}{F \sqsubseteq_n D}}{C_1 \sqcap \ldots \sqcap C_m \sqsubseteq_n D}\,\text{(GCI) with } E \sqsubseteq F \in \mathcal{O}$$

It is important to clarify that in this new proof tree the proof trees rooted at $C_i \sqcap \ldots \sqcap C_j \sqsubseteq_n E$ and $F \sqsubseteq_n D$ are the same as in the previous proof tree and therefore, one can see that they contain smaller number of (GCI) rule applications than $\mathcal{P}$. In addition, since $C_1 \sqcap \ldots \sqcap C_m \sqsubseteq_n E$ is derived by using only (AndLi) rule applications from the sequent $C_i \sqcap \ldots \sqcap C_j \sqsubseteq_n E$, then the proof tree rooted at it contains smaller number of (GCI) rule applications than $\mathcal{P}$ as well.

Thus, the application of induction (to the proof trees rooted at $C_1 \sqcap \ldots \sqcap C_m \sqsubseteq_n E$ and $F \sqsubseteq_n D$) and a further application of the (GCI) rule give a *good* proof tree for $C_1 \sqcap \ldots \sqcap C_m \sqsubseteq_n D$ in $\mathrm{HC}(\mathcal{O}, \mathcal{T}, \Delta)$.

$\square$

The following corollary is an immediate consequence of the previous lemma.

Corollary 36. *Each derivable sequent* $\mathfrak{s}$ *in* $\mathrm{HC}(\mathcal{O}, \mathcal{T}, \Delta)$ *has a* good *derivation in* $\mathrm{HC}(\mathcal{O}, \mathcal{T}, \Delta)$.

Lemma 37. *Let* C, D *be sub-descriptions of concept descriptions occurring in* $\mathcal{O}$, $\mathcal{T}$ *or* Γ *such that* $C \sqsubseteq_\infty D$ *is derivable in* $\mathrm{HC}(\mathcal{O}, \mathcal{T}, \Delta)$.

Then, for all $n \geq 0$ *there exists a proof tree* $\mathcal{P}$ *for* $C \sqsubseteq_n D$ *in* $\mathrm{HC}(\mathcal{O}, \mathcal{T}, \Delta)$ *such that it satisfies the following property:*

$$\textit{for every sequent } E \sqsubseteq_q F \textit{ in } \mathcal{P} \textit{ we have } E \sqsubseteq_\infty F \textit{ in } \mathrm{HC}(\mathcal{O}, \mathcal{T}, \Delta) \qquad (\mathcal{Z})$$

Proof. From Section 2 we know that there exists a value m that depends on $\mathrm{HC}(\mathcal{O}, \mathcal{T}, \Delta)$ such that $\sqsubseteq_0 \supseteq \sqsubseteq_1 \supseteq \ldots \supseteq \sqsubseteq_{m-1} \supseteq \sqsubseteq_m = \sqsubseteq_{m+1} = \sqsubseteq_{m+2} = \ldots$. Let us consider an arbitrary number $n \geq 0$ and a proof tree $\mathcal{P}$ of $C \sqsubseteq_{n+m} D$ in $\mathrm{HC}(\mathcal{O}, \mathcal{T}, \Delta)$ (it exists because $C \sqsubseteq_\infty D$ holds). As a consequence of the selection of m, one can observe that for every sequent $E \sqsubseteq_q F$ occurring in $\mathcal{P}$ with $q \geq m$ holds that $E \sqsubseteq_l F$ is derivable in $\mathrm{HC}(\mathcal{O}, \mathcal{T}, \Delta)$ for all $l \geq 0$. Therefore, $E \sqsubseteq_\infty F$ is derivable in $\mathrm{HC}(\mathcal{O}, \mathcal{T}, \Delta)$.

Now, a proof tree for $C \sqsubseteq_n D$ can be obtained by the application of *trimming* to $\mathcal{P}$. It can be seen (Definition 27) that each sequent occurring in the new proof tree corresponds to a sequent $E \sqsubseteq_q F$ from $\mathcal{P}$ such that $q \geq m$. Then, the *trimmed* tree is a proof tree for $C \sqsubseteq_n D$ in $\mathrm{HC}(\mathcal{O}, \mathcal{T}, \Delta)$ that satisfies property $\mathcal{Z}$. Thus, since n was arbitrarily selected the proposition holds for all $n \geq 0$. □

Corollary 38. *If a sequent $C \sqsubseteq_\infty D$ is derivable in* $\mathrm{HC}(\mathcal{O}, \mathcal{T}, \Delta)$*, then for all $n \geq 0$ there exists a* good *proof tree for $C \sqsubseteq_n D$ in* $\mathrm{HC}(\mathcal{O}, \mathcal{T}, \Delta)$ *satisfying Property $\mathcal{Z}$.*

Proof. From Corollary 36 we know that there is a *good* proof tree Q for the sequent $C \sqsubseteq_{n+m} D$ in $\mathrm{HC}(\mathcal{O}, \mathcal{T}, \Delta)$. If $\mathcal{P}$ in Lemma 37, is selected as Q then the application of *trimming* to $\mathcal{P}$ will give a *good* proof tree for $C \sqsubseteq_n D$ in $\mathrm{HC}(\mathcal{O}, \mathcal{T}, \Delta)$ that satisfies property $\mathcal{Z}$. □

Now, we introduce a disambiguation criterion on proof trees.

Definition 39. A proof tree Q in $\mathrm{HC}(\mathcal{O}, \mathcal{T}, \Delta)$ is *unambiguous* iff whenever there exist two or more occurrences of a sequent $E \sqsubseteq_q F$ in Q, the subtrees rooted at those occurrences are identical.

In addition, we say that a set of proof trees $\mathcal{Q}$ is *unambiguous* iff for each pair of proof trees $Q_1, Q_2 \in \mathcal{Q}$, every occurrence of a sequent $E \sqsubseteq_q F$ in Q_1 or Q_2 is the root of the same subtree.

Next, we show that Property $\mathcal{Z}$ and *goodness* can be preserved under *disambiguation* of a set of proof trees.

Lemma 40. *Let $\mathcal{Q} = \{Q_1, \ldots, Q_n\}$ be any unambiguous set of proof trees and Q be a proof tree for a sequent $C \sqsubseteq_n D$ in* $\mathrm{HC}(\mathcal{O}, \mathcal{T}, \Delta)$*, such that $Q_1, \ldots, Q_n$ and Q are* good *and satisfy the Property $\mathcal{Z}$ from Lemma 37.*

Then, there exists a good *proof tree Q' for $C \sqsubseteq_n D$ in* $\mathrm{HC}(\mathcal{O}, \mathcal{T}, \Delta)$ *such that $\mathcal{Z}$ is preserved in Q' and the set $\mathcal{Q}' = \mathcal{Q} \cup \{Q'\}$ is unambiguous.*

Proof. The proof is by induction on the structure of Q. By assumption $C \sqsubseteq_n D$ is the root of Q. If $C \sqsubseteq_n D$ is the root of some subtree Q_s of some $Q_i \in \mathcal{Q}$ then $\mathcal{Q}' = \mathcal{Q} \cup \{Q_s\}$ fulfills our claim, otherwise we can assume that $C \sqsubseteq_n D$ does not occur in any proof tree from $\mathcal{Q}$. There are three possible cases for the rule application that is used to obtain $C \sqsubseteq_n D$ in Q:

- $C \sqsubseteq_n D$ is obtained by applying one of the rules (Refl), (Top) or (Start). Then, Q is *unambiguous* because it is a one-element proof tree, it satisfies $\mathcal{Z}$ by assumption and it is *good* by definition. Hence, it can be safely added to $\mathcal{Q}$.

- $C \sqsubseteq_n D$ is obtained using a rule of the form $\frac{R}{S}$, then $S = C \sqsubseteq_n D$ and let Q_1 be the proof tree for R. Obviously, since Q_1 is a subtree of Q, it is *good* and satisfies $\mathcal{Z}$. The application of induction yields an unambiguous set $\mathcal{Q} \cup \{Q_1'\}$ satisfying $\mathcal{Z}$, where Q_1' is a *good* proof tree for R.

  Now, applying the same rule one can obtain from Q_1' a proof tree Q' for $C \sqsubseteq_n D$ satisfying $\mathcal{Z}$. If $\mathcal{Q}' = \mathcal{Q} \cup \{Q'\}$ is still *ambiguous*, this is because there is another sequent of the form $C \sqsubseteq_n D$ in $\mathcal{Q}'$, but such a sequent is the root of a *good* proof tree Q_s for $C \sqsubseteq_n D$ in $\mathcal{Q} \cup \{Q_1'\}$ that satisfies $\mathcal{Z}$. Therefore, Q_s represents the proof tree that we are looking for and thus, $\mathcal{Q}' = \mathcal{Q} \cup \{Q_s\}$. Otherwise, Q' is *unambiguous* w.r.t. $\mathcal{Q}$, it also satisfies property $\mathcal{Z}$, but it does not need to be *good* since Q_1' is obtained by induction and it may be the case that the last rule used to derive R is the (GCI) rule. In such a case, one can see that Q' has the following form:

$$\dfrac{\dfrac{\dfrac{\vdots}{C_i \sqsubseteq_n E} \qquad \dfrac{\vdots}{F \sqsubseteq_n D}}{C_i \sqsubseteq_n D}\ \text{(GCI) with } E \sqsubseteq F \in \mathcal{O}}{C_1 \sqcap C_2 \sqsubseteq_n D}\ \text{(AndLi)}$$

Figure 5: Proof tree Q'

We need to show that in such case a *good* proof tree actually exists. In order to do that we prove the following claim:

Claim: Let $\mathcal{Q} = \{Q_1, \ldots, Q_n\}$ be any unambiguous set of proof trees as above and Q be a proof tree for a sequent $C_1 \sqcap C_2 \sqsubseteq_n D$ in HC$(\mathcal{O}, \mathcal{T}, \Delta)$ *with the same form as in Figure 5, such that Q is unambiguous w.r.t. $\mathcal{Q}$, it satisfies property $\mathcal{Z}$ and all its sequents are* good *but the one at its root.*

Then, there exists a proof tree Q' for $C_1 \sqcap C_2 \sqsubseteq_n D$ as required above.

Proof. We show the claim by induction on the number p of (GCI) rule applications occurring in Q.

Induction Base. $p = 1$. We transform the proof tree Q into a new proof tree Q' for $C_1 \sqcap C_2 \sqsubseteq_n D$ that has the following form:

$$\dfrac{\dfrac{\dfrac{\vdots}{C_i \sqsubseteq_n E}}{C_1 \sqcap C_2 \sqsubseteq_n E}\ \text{(AndLi)} \qquad \dfrac{\vdots}{F \sqsubseteq_n D}}{C_1 \sqcap C_2 \sqsubseteq_n D}\ \text{(GCI) with } E \sqsubseteq F \in \mathcal{O}$$

First, note that the new tree contains only one (GCI) rule application and it happens to derive the sequent at the root, hence it is a *good* proof tree. Second, ambiguity of Q w.r.t. $\mathcal{Q}$ could only be caused if there is a sub-proof tree Q_s for $C_1 \sqcap C_2 \sqsubseteq_n E$ occurring in some proof tree $Q_i \in \mathcal{Q}$, but then the proof tree rooted at $C_1 \sqcap C_2 \sqsubseteq_n E$ can be replaced by Q_s to make Q' unambiguous w.r.t. $\mathcal{Q}$. Last, since $C_i \sqsubseteq_n E$ occurs in Q then $C_i \sqsubseteq_\infty E$ is derivable in HC($\mathcal{O}, \mathcal{T}, \Delta$) and obviously $C_1 \sqcap C_2 \sqsubseteq_\infty E$ as well. Then, Q' satisfies property $\mathcal{Z}$ and thus, it is a proof tree for $C_1 \sqcap C_2 \sqsubseteq_n D$ that satisfies the claim.

Induction Step. Assume that the claim holds for all natural numbers less than p and we show that it holds also for p. Then, Q has the same form as before but it contains p applications of the (GCI) rule. Doing the same transformation as for the base case, one can do a two case distinction for the proof tree rooted at the sequent $C_1 \sqcap C_2 \sqsubseteq_n E$:

– it is a *good* proof tree. Then, by the same reasons as for the base case a proof tree Q' exists satisfying the claim.

– it is not *good*. This can only happen when the sequent $C_i \sqsubseteq_n E$ is derived using an application of the (GCI) rule. However, since the proof tree rooted at $C_i \sqsubseteq_n E$ is a *good* proof tree, satisfies property $\mathcal{Z}$, is unambiguous w.r.t. $\mathcal{Q}$ and has less number of (GCI) rule applications than Q, then the proof tree rooted at $C_1 \sqcap C_2 \sqsubseteq_n E$ satisifies the induction hypothesis. Thus, the application of induction yields a proof tree Q_s for $C_1 \sqcap C_2 \sqsubseteq_n E$. Using Q_s to replace the proof tree rooted at $C_1 \sqcap C_2 \sqsubseteq_n E$ in Q' gives a proof tree for $C_1 \sqcap C_2 \sqsubseteq_n D$ that satisfies the claim.

This completes the proof for the *claim* and therefore, also the proof for the general case of a rule application of the form $\frac{R}{S}$.

- $C \sqsubseteq_n D$ is obtained using a rule of the form $\frac{R_1 \quad R_2}{S}$, then $S = C \sqsubseteq_n D$ and let Q_1 and Q_2 be the proof trees for R_1 and R_2 respectively. As before, Q_1 and Q_2 are *good*, satisfy property $\mathcal{Z}$ and applying induction twice we obtain the set of unambiguous proof trees $\mathcal{Q} \cup \{Q'_1, Q'_2\}$, where Q'_1 and Q'_2 are *good* proof trees for R_1 and R_2 respectively.

  Now, applying the same rule one can obtain from Q'_1 and Q'_2 a proof tree Q' for $C \sqsubseteq_n D$ satisfying $\mathcal{Z}$. Note that since the possible rules that can be used are (GCI) and (AndR), the proof tree Q' is *good*. Using the same reasoning as before, there is *good* proof tree Q_s for $C \sqsubseteq_n D$ that is either Q' or a subtree occurring in Q'_1 or Q'_2. Therefore, $\mathcal{Q}' = \mathcal{Q} \cup \{Q_s\}$ is an unambiguous set of proof trees satisfying property $\mathcal{Z}$ and the *goodness* condition.

□

Notice that since the empty set trivially satisfies the premises of Lemma 40, then the following corollary is a particular case and follows immediately.

Corollary 41. *Let Q be a* good *proof tree for the sequent $C \sqsubseteq_n D$, such that Q satisfies the Property $\mathcal{Z}$.*

Then, there exists an unambiguous good *proof tree Q' for $C \sqsubseteq_n D$, whereas $\mathcal{Z}$ is preserved in Q'.*

6.3 Completeness

Assume that Γ is hybrid-unifiable w.r.t. $\mathcal{O}$ and let $\mathcal{T}$ be a hybrid-unifier of Γ w.r.t. $\mathcal{O}$. Like in [2], we can use this unifier to guide the application of the nondeterministic rules such that Algorithm 24 does not fail. More precisely, we use a certain set of proof trees in $\mathrm{HC}(\mathcal{O}, \mathcal{T}, \Gamma)$ for the subsumptions in Γ to guide a non-failing run of the algorithm. To that purpose, we use the definitions and the properties shown in Section 6.2. One important issue that has to be guaranteed while guiding a non-failing run of the algorithm, is that whenever it is needed *blocking* will not fail. To help us in that matter we use the disambiguation criterion introduced in Definition 39.

Let Δ be the set of all sequents $C \sqsubseteq_n D$ such that: $0 \leq n \leq \ell$[3], C is a sub-description of a concept description occurring in $\mathcal{O}$, $\mathcal{T}$ or Γ, $D \in \mathrm{At}$ and $C \sqsubseteq_\infty D$ is derivable in $\mathrm{HC}(\mathcal{O}, \mathcal{T}, \Gamma)$. Since $C \sqsubseteq_\infty D$ is derivable in $\mathrm{HC}(\mathcal{O}, \mathcal{T}, \Gamma)$ then, by Corollaries 38 and 41 and Lemma 40 we can assume without loss of generality that there is an *unambiguous* set of proof trees $\mathcal{Q}$ such that: each sequent $C \sqsubseteq_n D \in \Delta$ has a *good* proof tree in $\mathcal{Q}$ that satisfies Property $\mathcal{Z}$. Note, that since $\mathcal{T}$ is a hybrid-unifier of Γ w.r.t. $\mathcal{O}$ we know that $C_i \sqsubseteq_\infty D_i$ is derivable in $\mathrm{HC}(\mathcal{O}, \mathcal{T}, \Gamma)$ for each subsumption $C_i \sqsubseteq D_i \in \Gamma$. Obviously, C_i is a sub-description of a concept description occurring in Γ and $D_i \in At$ because Γ is flat, then we know that there is a proof tree in $\mathcal{Q}$ for $C_i \sqsubseteq_\ell D_i$.

Now, we are ready to show how to guide a non-failing run of Algorithm 24. The following invariants for the current set of *p-sequents* Γ_p and the current assignment ζ will be maintained w.r.t. $\mathcal{Q}$:

(i) for each p-sequent $(\mathfrak{s}, P)$ in Γ_p, $\mathfrak{s}$ occurs in some Q_i from $\mathcal{Q}$.

(ii) For all $D \in \zeta_X$ we have $X \sqsubseteq_\infty D$ derivable in $\mathrm{HC}(\mathcal{O}, \mathcal{T}, \Gamma)$.

(iii) for each p-sequent $(\mathfrak{s}, P)$ in Γ_p, if $E \sqsubseteq_n F \in P$ then, $\mathfrak{s}$ occurs in any proof tree for $E \sqsubseteq_n F$ that is a sub-tree in $\mathcal{Q}$.

Since ζ_X is initialized to $\emptyset$ for all variables $X \in N_v$, there is a proof tree for each $C_i \sqsubseteq_\ell D_i$ in $\mathcal{Q}$ as described above and Γ_p is initialized to $\Gamma_p^{(0)}$, then these invariants are satisfied after the initialization of the algorithm.

[3]l is the value computed during the initialization of Algorithm 24.

We first show that after applying expansion to Γ_p, the invariants are mantained.

Lemma 42. *The invariants are mantained by the operation of expanding* Γ_p.

Proof. The application of expansion is performed w.r.t. a *defined concept* X. For every p-sequent of the form $(C \sqsubseteq_n X, \_) \in \Gamma_p$ *blocking* is applied to $(C \sqsubseteq_{n-1} D, \emptyset)$ and Γ_p for every $D \in \zeta_X$.

As explained before, since the second components of the p-sequents provided as inputs for *blocking* are empty, blocking cannot fail during expansion. Due to the same reason, if the rule **B2** from *blocking* is applicable, it does not change the second component of any p-sequent in Γ_p and therefore invariant (iii) is satisfied.One can also see than since expansion does not change the current assignment ζ, invariant (ii) is trivially maintained. We show that invariant (i) is also satisfied.

If a p-sequent $(C \sqsubseteq_{n-1} D, \emptyset)$ is added to Γ_p by expansion then, we know that $D \in \zeta_X$ and invariant (ii) yields that $X \sqsubseteq_\infty D$ is derivable in HC$(\mathcal{O}, \mathcal{T}, \Gamma)$. Since, $(C \sqsubseteq_n X, \_) \in \Gamma_p$ then by invariant (i) it occurs in some proof tree $Q_i \in \mathcal{Q}$ and consequently by Property $\mathcal{Z}$, $C \sqsubseteq_\infty X$ is derivable in HC$(\mathcal{O}, \mathcal{T}, \Gamma)$. Hence, transitivity of $\sqsubseteq_\infty$ yields derivability of $C \sqsubseteq_\infty D$ in HC$(\mathcal{O}, \mathcal{T}, \Gamma)$.

In addition, one can see that C is a sub-description of a concept description occurring in $\mathcal{O}$, $\mathcal{T}$ or Γ and since $D \in \zeta_X$ implies $D \in At_{nv}$ then, by construction of $\mathcal{Q}$ there is a proof tree $Q_i \in \mathcal{Q}$ containing $C \sqsubseteq_{n-1} D$. Therefore, invariant (i) is satisfied. □

There is only one *eager* rule that could produce failures, the following lemma shows that this will never be the case.

Lemma 43. *The application of an eager rule never fails and mantains the invariants.*

Proof. We do not need to consider applications of *Eager Axiom Solving* and *Eager Solving*, since they cannot fail nor do they add new *p-sequents* to Γ_p.

Consider an application of *Eager Ground Solving* to an unsolved p-sequent $p = (C \sqsubseteq_n D, \_)$ with $C \sqsubseteq_n D$ ground. By invariant (i), $C \sqsubseteq_n D$ occurs on a proof tree Q_i that satisfies the Property $\mathcal{Z}$ and thus, $C \sqsubseteq_\infty D$ is derivable in HC$(\mathcal{O}, \mathcal{T}, \Gamma)$. Applying Theorem 5 we obtain that $C \sqsubseteq_{gfp,\mathcal{O},\mathcal{T}} D$ holds, and this implies that the rule application does not fail. The invariants are mantained since neither Γ_p nor ζ are modified. □

Now, we need to show that if no *eager* rule is applicable to *p-sequents* in Γ_p and there is still an *unsolved* p-sequent in Γ_p then, there is a nondeterministic rule that can be applied while keeping the invariants.

Lemma 44. *Let $p = (\mathfrak{s}, P)$ be an unsolved p-sequent in Γ_p to which no eager rule applies. Then, there is a nondeterministic rule that can be applied to p while maintaining the invariants.*

Proof. First, $\mathfrak{s}$ must be of the form $C_1 \sqcap \ldots \sqcap C_m \sqsubseteq_n D$ where $D \in At_{nv}$ and $n > 0$. Note that, D must be a *non-variable* atom since the input unification problem is flat and every p-sequent added to Γ_p during a run of Algorithm 24 has a first component which right-hand side is either a *defined concept* (marked as *solved* and no rule applies to it) or a *non-variable* atom.

By invariant (i), there exists a set of *unambiguous* proof trees $\mathcal{Q}$ such that $\mathfrak{s}$ occurs in some $Q_i \in \mathcal{Q}$, Q_i is *good* and satisfies Property $\mathcal{Z}$. Let $\mathcal{P}_{\mathfrak{s}}$ be the subtree rooted at $\mathfrak{s}$ and let $C_i \sqcap \ldots \sqcap C_j \sqsubseteq_n D$ be the leaf of its maximal sub-proof tree w.r.t. $\{AndL1, AndL2\}$. We distinguish between the possible rules from HC$(\mathcal{O}, \mathcal{T}, \Gamma)$ that could have been used to obtain $C_i \sqcap \ldots \sqcap C_j \sqsubseteq_n D$ in $\mathcal{P}_{\mathfrak{s}}$.

- One of the rules (Refl), (Top) or (Start) is used. In this case, either *Eager Axiom Solving* or *Eager Solving* would have been used successfully.

- The rule (DefL) is used. Then, $i = j$ and $C_i = X$ for some defined concept X. Since Q_i satisfies Property $\mathcal{Z}$ and $X \sqsubseteq_n D$ occurs in Q_i then, $X \sqsubseteq_\infty D$ is derivable in HC$(\mathcal{O}, \mathcal{T}, \Gamma)$ and since $D \in At_{nv}$ the addition of D to ζ_X does not affect invariant (ii). Invariants (i) and (iii) are trivially satisfied since Γ_p is not modified. Thus, we can apply the rule *Extension* while maintaining the invariants.

- The rule (Ex) is used. Then, $i = j$, C_i is of the form $\exists s.C'$ and D is of the form $\exists s.D'$. The sequent $C' \sqsubseteq_n D'$ occurs in Q_i and this means that, if the p-sequent $p = (C' \sqsubseteq_n D', P \cup \{\mathfrak{s}\})$ is added to Γ_p, the invariant (i) is maintained. In addition, invariant (ii) is trivially satisfied since the assignment ζ is not modified. It will be shown below why invariant (iii) is preserved and also why *blocking* does not fail for p. Thus, the rule *Decomposition* can be successfully applied while maintaining the invariants.

- The rule (GCI) is used. Then, there exists a GCI $E_1 \sqcap \ldots \sqcap E_k \sqsubseteq F \in \mathcal{O}$ such that $C_i \sqcap \ldots \sqcap C_j \sqsubseteq_n E_1 \sqcap \ldots \sqcap E_k$ and $F \sqsubseteq_n D$ are sequents occurring in Q_i. Since we know that Q_i is a *good* proof tree, then the derivation of $\mathfrak{s}$ in Q_i has the following form:

$$\dfrac{\dfrac{\vdots}{C_1 \sqcap \ldots \sqcap C_m \sqsubseteq_n E_1 \sqcap \ldots \sqcap E_k} \qquad \dfrac{\vdots}{F \sqsubseteq_n D}}{C_1 \sqcap \ldots \sqcap C_m \sqsubseteq_n D}\ \text{(GCI) with } E_1 \sqcap \ldots \sqcap E_k \sqsubseteq F \in \mathcal{O}$$

  In addition, the same *goodness* property in Q_i implies that the derivation of $C_1 \sqcap \ldots \sqcap C_m \sqsubseteq_n E_1 \sqcap \ldots \sqcap E_k$ in Q_i must be of the following form:

$$\dfrac{\dfrac{Q_{E_1}}{C_1 \sqcap \ldots \sqcap C_m \sqsubseteq_n E_1} \quad \dfrac{Q_{E_2}}{C_1 \sqcap \ldots \sqcap C_m \sqsubseteq_n E_2}}{C_1 \sqcap \ldots \sqcap C_m \sqsubseteq_n E_1 \sqcap E_2}\,\text{(AndR)} \qquad \ddots$$

$$\dfrac{\dfrac{\ddots}{C_1 \sqcap \ldots \sqcap C_m \sqsubseteq_n E_1 \sqcap \cdots \sqcap E_{k-1}}\,\text{(AndR)} \qquad \dfrac{Q_{E_k}}{C_1 \sqcap \ldots \sqcap C_m \sqsubseteq_n E_k}}{C_1 \sqcap \ldots \sqcap C_m \sqsubseteq_n E_1 \sqcap \cdots \sqcap E_k}\,\text{(AndR)}$$

Summing up, the sequents $C_1 \sqcap \ldots \sqcap C_m \sqsubseteq_n E_i, \ldots, C_1 \sqcap \ldots \sqcap C_m \sqsubseteq_n E_k$, and $F \sqsubseteq_n D$ occur in Q_i. Thus, the addition of the *p-sequents* $p_1 = (C_1 \sqcap \ldots \sqcap C_m \sqsubseteq_n E_1, P \cup \{\mathfrak{s}\}), \ldots, p_k = (C_1 \sqcap \ldots \sqcap C_m \sqsubseteq_n E_k, P \cup \{\mathfrak{s}\}), q = (F \sqsubseteq_n D, P \cup \{\mathfrak{s}\})$ to Γ_p will maintain invariant (i). As in the case of *Decomposition* rule, invariant (ii) is satisfied and it will be shown below why invariant (iii) is satisfied and *blocking* does not fail for $p_1, \ldots, p_k$ and q. Thus, *Mutation* can be successfully applied while maintaining the invariants.

Now let us see why *blocking* does not fail in the above cases of *Decomposition* and *Mutation* applications. Recall that $p = (\mathfrak{s}, P)$ and let $p' = (\mathfrak{s}', P \cup \{\mathfrak{s}\})$ be one of the p-sequents that *blocking* is applied to during an application of rules *Decomposition* or *Mutation*. Assume that *blocking* fails meaning that $\mathfrak{s}' \in P$.

By invariant (iii), we have that there is a proof tree $Q' \in \mathcal{Q}$ such that $\mathfrak{s}'$ occurs in Q' and $\mathfrak{s}$ occurs in the subtree rooted at $\mathfrak{s}'$. In addition, $\mathfrak{s}'$ occurs in the subtree rooted at $\mathfrak{s}$ in Q_i. Since Q_i and Q' are mutually *unambiguous* then, the subtrees rooted at $\mathfrak{s}$ in Q_i and Q' must be identical. This implies that the subtree rooted at $\mathfrak{s}'$ in Q' contains an additional occurrence of $\mathfrak{s}$ and this, obviously contradicts the assumption that Q' is *unambiguous*. Thus, $\mathfrak{s}'$ cannot be in P and *blocking* does not fail.

Finally, we have to show that invariant (iii) is maintained afterwards. Consider again the p-sequent $(\mathfrak{s}', P \cup \{\mathfrak{s}\})$. Since *blocking* does not fail, two cases are possible:

- Rule **B3** was applied while doing *blocking*. Then, $(\mathfrak{s}', P \cup \{\mathfrak{s}\})$ is added to Γ_p. Since $(\mathfrak{s}, P)$ is in Γ_p then, invariant (iii) implies that $\mathfrak{s}$ occurs in any proof tree for $E \sqsubseteq_n F \in P$ that is a subtree in $\mathcal{Q}$. Now, $\mathfrak{s}'$ occurs in the subtree rooted at $\mathfrak{s}$ in Q_i and thus, the *unambiguous* condition of $\mathcal{Q}$ yields that $\mathfrak{s}'$ occurs in any proof tree for $E \sqsubseteq_n F \in P \cup \{\mathfrak{s}\}$.

- Rule **B2** was applied. Then, there is a p-sequent of the form $(\mathfrak{s}', P')$ in Γ_p. From the previous case we know that $\mathfrak{s}'$ occurs in any proof tree for $E \sqsubseteq_n F \in P \cup \{\mathfrak{s}\}$. Therefore, updating P' as $P' \cup P \cup \{\mathfrak{s}\}$ preserves invariant (iii). In addition, for each p-sequent $(\mathfrak{s}'', P'')$ in Γ_p such that $\mathfrak{s}' \in P''$ invariant (iii) also yields that $\mathfrak{s}''$ occurs in any proof tree for $\mathfrak{s}'$ in $\mathcal{Q}$. Hence, the *unambiguity* of the set $\mathcal{Q}$ guarantees that invariant (iii) is satisfied after the update of P'' as $P'' \cup P \cup \{\mathfrak{s}\}$.

Finally, we can conclude that there is a nondeterministic rule that can be applied to p while maintaining the invariants. □

These lemmas imply that for any hybrid-unifiable input problem Γ there is a non-failing run of Algorithm 24 during which invariants (i), (ii) and (iii) are satisfied. Assuming that any run of the algorithm terminates (see next section), this shows completeness, i.e., whenever Γ has a hybrid-unifier $\mathcal{S}$ w.r.t. $\mathcal{T}$, the algorithm computes one.

6.4 Termination and complexity

Consider a run of Algorithm 24. We will show that at any time any p-sequent $(C \sqsubseteq_n D, P)$ encountered during this run satisfies the following three conditions:

1. if $(C \sqsubseteq_n D, P) \in \Gamma_p$ there is no p-sequent $(C \sqsubseteq_n D, P') \in \Gamma_p$ with $P \neq P'$
2. C is either conjunction on the left hand side of a subsumption from $\Gamma_p^{(0)}$ or an atom in At and D belongs to At
3. $n \leq l$, where l is the value computed during the initialization of the algorithm.

Lemma 45. *If the p-sequents in* Γ_p *satisfy conditions* 1, 2 *and* 3, *then these conditions are satisfied after one rule application.*

Proof. First, since Γ_p is initialized as $\Gamma_p^{(0)}$ then, the conditions are obviously satisfied after the initialization of the algorithm. Now, let us consider the cases when a new p-sequent is added into Γ_p.

- A p-sequent is created by expansion of Γ_p. This is of the form $(C_1 \sqcap \ldots \sqcap C_m \sqsubseteq_{n-1} D, \emptyset)$ for a p-sequent $(C_1 \sqcap \ldots \sqcap C_m \sqsubseteq_n X, P) \in \Gamma_p$ with $D \in At_{nv}$. Due to blocking condition 1 is satisfied. Since $n \leq l$, condition 3 is satisfied. In addition, the set At_{nv} only contains non-variable atoms from $\Gamma \cup \mathcal{T}$, therefore condition 2 is satisfied as well.
- A p-sequent is created by a rule application. These are the rules *Decomposition* and *Mutation*. In both cases blocking ensures condition 1. In the case of Decomposition, from a p-sequent $(C \sqsubseteq_n D, P) \in \Gamma_p$ a new p-sequent is created of the form $(C' \sqsubseteq_n D', P')$, where C' and D' are atoms in At. In the case of Mutation, new p-sequents have the form $(C \sqsubseteq_n D', P')$, where C is of the required form by assumption and D' is an atom in At. Hence condition 2 is satisfied. Condition 3 is satisfied by assumption.

All the other rules from Algorithm 24 maintain conditions 1, 2 and 3 since Γ_p is not modified. $\square$

Based on this lemma, we conclude with the following.

Lemma 46. *Every run of the Algorithm 24 on input Γ terminates in time polynomial in the size of $\Gamma \cup \mathcal{O}$.*

Proof. Since every p-sequent has the form $(C \sqsubseteq_n D, P)$, due to condition 2, there are only at most $\#\Gamma + |\mathrm{At}|$ concept descriptions that can be C, where $\#\Gamma$ is the number of subsumptions in Γ, and only at most $|\mathrm{At}|$ concept descriptions that can be D. In addition, due to condition 3, there are only at most $\ell \times (\#\Gamma + |\mathrm{At}|) \times |\mathrm{At}|$ subsumptions of the form $C \sqsubseteq_n D$ used in the p-sequents during a run of the algorithm. Since ℓ is a polynomial in the size of Γ and $\mathcal{O}$, there are only at most polynomially many first components for p-sequents produced in one run of the algorithm.

Due to condition 1, there cannot be two p-sequents $(C \sqsubseteq_n D, P)$ and $(C \sqsubseteq_n D, P')$ with $P \neq P'$ in Γ_p at the same time. Since, in addition, once a p-sequent is marked as *solved* by the algorithm it will never become *unsolved* again, then a run of the algorithm never solves two p-sequents with the same left-hand side and different right-hand side. Therefore, since every rule application solves one p-sequent, then the algorithm can apply at most polynomially many rules.

Now, verifying whether a rule is applicable is done polynomially often and executing some of the following operations:

- Checking a subsumption between ground sub-descriptions of $\Gamma \cup \mathcal{O}$.
- Checking whether $\mathfrak{s}$ in $(\mathfrak{s}, P)$ is of a specific form, e.g., ground, the *right-hand* side of $\mathfrak{s}$ does not consist of a single top-level atom or $\mathfrak{s}$ is the consequence of an axiom rule from **HC**.
- Guessing a GCI from $\mathcal{O}$ or guessing a conjunct C_i in Eager Solving, Decomposition or Extension.
- Execute rule **B1** from *blocking* on candidates *p-sequents* to be added into Γ_p.

The last operation has to check whether a sequent $C \sqsubseteq_n D$ is contained in the set P. Since P contains only sequents $\mathfrak{s}$ such that $(\mathfrak{s}, \_) \in \Gamma_p$ and the size of Γ_p is polynomial on the size of $\Gamma \cup \mathcal{O}$, then rule **B1** executes in time polynomial in the size of $\Gamma \cup \mathcal{O}$.

If a rule is applicable, its application can execute the following polynomial operations:

- Adding polynomially many *p-sequents* to Γ_p.
- Adding polynomially many atoms to the current assignment.
- Execute one of the rules **B2** or **B3** from *blocking*.

Again, we clarify the case concerning the application of rules from *blocking*. The case for **B3** is clear since its application only adds one p-sequent to Γ_p. The application of rule **B2** searches for each $(\mathfrak{s}, P) \in \Gamma_p$ the elements in P. As explained before, Γ_p and P are of size polynomial on the size of $\Gamma \cup \mathcal{T}$ which implies that **B2** can be applied in time polynomial. $\square$

This lemma with the soundness and completeness shown before (29, 42, 43 and 44), yield the main result of this section.

Theorem 47. *Algorithm 24 is an* NP*-decision procedure for hybrid $\mathcal{EL}$-unifiability w.r.t. arbitrary $\mathcal{EL}$-ontologies.*

7 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 7, 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, Acapulco, Mexico (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), see 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 Press/The 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.) Pro. of 25th Australasian Joint Conf. on Artificial Intelligence (AI'12). Lecture Notes in Artificial Intelligence, vol. 7691, pp. 493–504. Springer-Verlag (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., Morawska, B.: Unification in the description logic $\mathcal{EL}$. In: Treinen, R. (ed.) Proc. of the 20th Int. Conf. on Rewriting Techniques and Applications (RTA 2009). Lecture Notes in Computer Science, vol. 5595, pp. 350–364. Springer-Verlag (2009)

[7] Baader, F., Morawska, B.: Unification in the description logic $\mathcal{EL}$. Logical Methods in Computer Science 6(3) (2010)

[8] Baader, F., Narendran, P.: Unification of concept terms in description logics. J. 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] Brandt, S.: Subsumption and Instance Problem in $\mathcal{ELH}$ w.r.t. General TBoxes. LTCS-Report 04-04, Chair for Automata Theory, Institute for Theoretical Computer Science, Technische Universität Dresden, Dresden, Germany (2004), see http://lat.inf.tu-dresden.de/research/reports.html.

[11] 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)

[12] Brandt, S., Model, J.: Subsumption in $\mathcal{EL}$ w.r.t. hybrid tboxes. In: Proc. of the 28th German Annual Conf. on Artificial Intelligence (KI'05). pp. 34–48. Lecture Notes in Artificial Intelligence, Springer-Verlag (2005)

[13] Küsters, R.: Non-standard Inferences in Description Logics, Lecture Notes in Artificial Intelligence, vol. 2100. Springer-Verlag (2001)

[14] B. Nebel. Terminological cycles: Semantics and computational properties. In J. F. Sowa, editor, Principles of Semantic Networks: Explorations in the Representation of Knowledge, pages 331-361. Morgan Kaufmann Publishers, San Mateo (CA), USA, 1991.

[15] Novakovic, N.: A proof-theoretic approach to deciding subsumption and computing least common subsumer in EL w.r.t. hybrid TBoxes. Master Thesis. Chair of Automata Theory, Institute for Theoretical Computer Science, TU-Dresden, 2007, see http://lat.inf.tu-dresden.de/research/mas/#Nov-Mas-07.

[16] Novakovic, N.: A proof-theoretic approach to deciding subsumption and computing least common subsumer in EL w.r.t. hybrid TBoxes. In: Hölldobler, S., Lutz, C., Wansing, H. (eds.) Proc. of the 11th Eur. Conf. on Logics in Artificial Intelligence (JELIA'2004). Lecture Notes in Computer Science, vol. 5293, pp. 311–323. Springer-Verlag (2008)