Technische Universität Dresden

Fakultät Informatik

EMCL Master's Thesis on

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

by

Oliver Fernández Gil

born on June 11^{th} 1982, in Pinar del Río, Cuba

Supervisor : **Prof.Dr.-Ing. Franz Baader**

Advisor : **Dr. Barbara Morawska**

Dresden, October 2012

Technische Universität Dresden

Author : **Oliver Fernández Gil**
Matrikel-Nr. : **3734913**
Title : **Hybrid Unification in the Description Logic $\mathcal{EL}$**
Degree : **Master of Science**
Date of Submission :

Declaration

Hereby I certify that the thesis has been written by me. Any help that I have received in my research work has been acknowledged. Additionally, I certify that I have not used any auxiliary sources and literature except those cited in the thesis.

Oliver Fernández Gil

Abstract

In recent years, the Description Logic $\mathcal{EL}$ has received a significant interest. It has been shown to be useful as a knowledge representation formalism, to define large biomedical ontologies. Moreover, important inference problems like subsumption have been shown to be decidable in polynomial time in $\mathcal{EL}$. In particular, unification in $\mathcal{EL}$ can be used to detect redundancies in big ontologies. The unification problem in $\mathcal{EL}$ has been shown to be an NP-complete problem even in the presence of background knowledge in the form of an acyclic TBox. Recently, the result was also extended to the case of cycle-restricted TBoxes. Unification in $\mathcal{EL}$ w.r.t. general TBoxes is still an open problem. Here we define a more general notion of unification problem called hybrid unification. In contrast to the usual unification problem, hybrid unification allows solutions that contain cyclic definitions of concept names. We show that hybrid unification in $\mathcal{EL}$ is NP-complete. Furthermore, we provide a goal-oriented NP-decision procedure for the hybrid unification problem.

Acknowledgements

First of all, I would like to thank Barbara Morawska for her constant support, encouragement and infinite patience during the last year. In addition, I am grateful to Professor Franz Baader for his support and for giving me the possibility to do research in his group.

Many thanks go to Professor Steffen Hölldobler for his valuable advice and constant support during these two years. Also, I am grateful for the financial support of the Joint Commission through the European Masters Programme in Computational Logic.

Most importantly, many thanks to my friends and family for everything, wherever they are. Specially, I am deeply grateful to my parents, my brothers and my sister.

Contents

Chapter 1

Introduction

In this thesis we define a new problem called *hybrid unification* and solve it for the Description Logic $\mathcal{EL}$.

Description Logics (DLs)[23] have been studied as a logic-based knowledge representation formalism that can be used to represent concept definitions in a structured and formal way. Knowledge in DLs is meant to be represented by *concept descriptions*. Concept descriptions are built from *concept names* and *role names* using *concept constructors*. In order to give a meaning to concept descriptions, interpretations like in *first-order* logic are used. A nonempty domain of individuals is assumed and concept names are interpreted as subsets of the domain while role names as binary relations over the domain.

In particular, the small DL $\mathcal{EL}$ provides the concept constructors conjunction ($\sqcap$), existential restriction ($\exists r.C$) and the top concept ($\top$). Although $\mathcal{EL}$ is quite inexpressive it turns out to be of great interest since, on the one hand, several large biomedical ontologies are defined within the expressive power of $\mathcal{EL}$[1]. On the other hand, important inference problems like subsumption have been shown to be decidable in polynomial time in $\mathcal{EL}$ [2, 3, 4]. Informally, a concept description C is subsumed by another concept description D if for all interpretations C is interpreted as a subset of D.

Unification in DLs has been proposed as a new and important inference problem, that for instance, can be used to detect redundancies in ontologies [10]. As an intuitive example, assume that one developer of an ontology defines the concept of *grandmother* as

$$Human \sqcap Female \sqcap \exists child.Parent \quad (1)$$

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

while another one represents it as

$$Woman \sqcap \exists child.(Human \sqcap \exists child.Human) \quad (2)$$

These two concept descriptions are meant to represent the same concept, but they are not formally equivalent, because there are interpretations that interpret them as different sets of individuals. However, they can be made equivalent if the concept names $Woman$ and $Parent$ are treated as variables that can be replaced by concept descriptions. For instance, if $Woman$ is substituted by $Human \sqcap Female$ and $Parent$ by $Human \sqcap \exists child.Human$, then these descriptions are equivalent. In this case, we say that the descriptions are unifiable and we can represent the solution as a substitution or equivalently as a set of definitions.

$$Woman \equiv Human \sqcap Female$$
$$Parent \equiv Human \sqcap \exists child.Human$$

In practical applications, often unification of two concept descriptions is considered w.r.t. background knowledge formulated as a big set of definitions or subsumption facts [13]. This knowledge can enable discovery of possible solutions for the unification problem. For example, suppose that the second developer uses the description

$$\exists family.Big \sqcap Woman \sqcap \exists child.(Human \sqcap \exists child.Human) \quad (3)$$

instead of (2). Now, the descriptions (1) and (3) are not unifiable, but they would be with the same unifier as above, if the following subsumption

$$Woman \sqcap \exists child.(\exists child.Human) \sqsubseteq \exists family.Big$$

is considered as a fact in the background ontology.

The unification problem in $\mathcal{EL}$ has been recently investigated [11, 12, 14]. First, it was shown to be NP-Complete in the absence of a background knowledge base [11]. The main idea underlying this "in NP" result, relies in the existence of a *local unifier*. In [13], it was shown that when the unification problem is considered w.r.t. a non-empty general TBox, the same notion of locality does not work. A first solution was proposed to fix this problem: restrict the general TBox, representing the background ontology, to a certain type of general TBoxes. A TBox of such type is characterized by a restriction on the structure of the consequences that are implied from the TBox. However, this solution obviously does not apply to arbitrary general TBoxes.

This thesis, presents an alternative solution to the problem. Instead of considering only ground substitutions as the possible solutions of the unification problem, we propose to allow concept definitions that may contain

cyclic dependencies to be solutions of the unification problem. The new problem is called *hybrid unification*, motivated by the notion of a *hybrid* TBox introduced in [4].

In order to do that, we need to use an appropriate semantics since now cycles may occur in a solution of the problem. The properties of different semantics applied to the problem of interest were investigated in [18], where three types of semantics were defined: *descriptive semantics*, *greatest fixpoint semantics* and *least fixpoint semantics*. In particular, greatest fixpoint semantics was shown to be suitable to interpret cyclic TBoxes.

The combination of a general TBox and a possibly cyclic set of concept definitions, was introduced in [6] under the name of *hybrid* TBoxes. A combination of descriptive semantics with greatest fixpoint semantics was chosen to interpret those TBoxes and the subsumption problem was shown to be decidable in polynomial time. Later in [7], the same polynomial time result was obtained, but using a Gentzen-style calculus as a characterization of the subsumption problem.

The main goal of this thesis is to show that the *hybrid unification* problem is NP-complete. We use the Gentzen-style calculus proposed in [7] and show the "in NP" part of the result based on a similar notion of locality as the one used to show the NP-membership of unification in $\mathcal{EL}$. The NP-hardness is shown by a reduction from the $\mathcal{EL}$-matching problem modulo equivalence which is known to be NP-complete [17].

We start, in Chapter 2, by introducing formal basic definitions concerning the $\mathcal{EL}$-description Logic and the greatest fixpoint semantics. Then, Chapter 3 presents more details about the results that have already been obtained for the $\mathcal{EL}$-unification problem and explains the need to find a new alternative approach. Once we have this necessary introduction, hybrid TBoxes will be presented in Chapter 4 as well as, the subsumption problem w.r.t. hybrid TBoxes, the proof calculus from [7] and finally the formal definition of the hybrid unification problem.

The main results of this thesis are shown in Chapters 5 and 6. In Chapter 5 we show that *hybrid unification* is NP-complete obtaining an immediate NP-decision procedure. Then, in Chapter 6 we provide a more goal-oriented algorithm that is a correct NP-decision procedure for the hybrid unification problem.

Finally, in the conclusions we sum up the results and discuss significance and possible future research development on hybrid unification in $\mathcal{EL}$.

Chapter 2

The Description Logic $\mathcal{EL}$

In this chapter, we introduce formal definitions and concepts that are important for the next chapters. We start by presenting the Description Logic $\mathcal{EL}$, its syntax and semantics. In addition, we introduce the notion of terminological axioms and discuss some related aspects, e.g., cyclic definitions and a normalization procedure used to simplify a TBox. Finally, two types of semantics are presented and some of their properties are shown.

2.1 Syntax and Semantics

Starting from a finite set N_C of concept names and a finite set N_R of role names, $\mathcal{EL}$-concept descriptions are built using the concept constructors top-concept ($\top$), conjunction ($\sqcap$) and existential restriction ($\exists r.C$). For instance, using the set of concept names $\{Baseball, Human\}$ and the role name *plays*, the concept of all *baseball players* can be represented by the concept description:

$$Human \sqcap \exists plays.Baseball$$

The set of all $\mathcal{EL}$-concept descriptions can be formally defined in the following way:

Definition 2.1.1. Let N_C and N_R be disjoint sets of concept names and role names respectively. The set of $\mathcal{EL}$-concept descriptions is the smallest set satisfying the following conditions:

- Every concept name $A \in N_C$ is in the set.
- If C, D are in the set and $r \in N_R$ is a role name, then the top-concept $\top$, the conjunction $C \sqcap D$ and the existential restriction $\exists r.C$ are in the set.

To interpret these concept descriptions the usual semantics as in *first-order-logic* is used. An interpretation $\mathcal{I} = (\mathcal{D}_\mathcal{I}, .^\mathcal{I})$ consists of a non-empty domain $\mathcal{D}_\mathcal{I}$ and an interpretation function $.^\mathcal{I}$ that assigns a subset of $\mathcal{D}_\mathcal{I}$ to each concept name and a binary relation over $\mathcal{D}_\mathcal{I}$ to each role name. The extension of $.^\mathcal{I}$ to arbitrary concept descriptions can be done as shown in the semantics column of Table 2.1.

Name	Syntax	Semantics
concept name	A	$A^\mathcal{I} \subseteq \mathcal{D}_\mathcal{I}$
role name	r	$r^\mathcal{I} \subseteq \mathcal{D}_\mathcal{I} \times \mathcal{D}_\mathcal{I}$
top concept	$\top$	$\top^\mathcal{I} = \mathcal{D}_\mathcal{I}$
conjunction	$C \sqcap D$	$(C \sqcap D)^\mathcal{I} = C^\mathcal{I} \cap D^\mathcal{I}$
existential restriction	$\exists r.C$	$(\exists r.C)^\mathcal{I} = \{x \mid \exists y : (x, y) \in r^\mathcal{I} \wedge y \in C^\mathcal{I}\}$

Table 2.1: Syntax and semantics of $\mathcal{EL}$

Definition 2.1.2. A concept description is called an *atom* iff it is a concept name or an existential restriction. The set $At(C)$ of all atoms of a concept description C is inductively defined as follows:

$$At(C) = \begin{cases} \emptyset, & \text{if } C = \top \\ \{C\}, & \text{if } C \in N_C \\ \{C\} \cup At(D), & \text{if } C = \exists r.D \\ At(C_1) \cup At(C_2), & \text{if } C = C_1 \sqcap C_2 \end{cases}$$

If C is a concept name or an existential restriction $\exists r.D$ where D is a concept name or $\top$, then we say that C is a flat atom.

One can see that every concept description C is defined as a conjunction of atoms $C_1, \ldots, C_n$ which are called the *top-level* atoms of C. If each of these atoms are flat, then C is a flat concept description.

Definition 2.1.3. Let C be a concept description and let $C_1, \ldots, C_n$ be atoms. Then, the set of all sub-descriptions of the concept description C is defined in the following way:

$$Sub(C) = \begin{cases} \{C\}, & \text{if } C = \top \text{ or } C \in N_C \\ \{C_i \sqcap \ldots \sqcap C_j \mid 1 \leq i < j \leq n\} \\ \cup \{Sub(C_k) \mid 1 \leq k \leq n\}, & \text{if } C = C_1 \sqcap \ldots \sqcap C_m \\ \{C\} \cup Sub(D) & \text{if } C = \exists r.D \end{cases}$$

Definition 2.1.4. The *role depth* $rd(C)$ of a concept description C is defined in the following way:

$$rd(C) = \begin{cases} 0, & \text{if } C = \top \text{ or } C \in N_C \\ rd(D) + 1, & \text{if } C = \exists r.D \\ max\{rd(C_1), rd(C_2)\}, & \text{if } C = C_1 \sqcap C_2 \end{cases}$$

2.2 Terminological Axioms

As in any other DL, intensional knowledge about the domain of interest can be represented in $\mathcal{EL}$ using terminological axioms, i.e., definitions and general concept inclusions.

Definition 2.2.1. A concept definition is of the form $A \equiv C$ for a concept name A and a concept description C. A general concept inclusion (GCI) is of the form $C \sqsubseteq D$ for concept descriptions C, D.

An interpretation $\mathcal{I}$ satisfies a concept definition $A \equiv C$ iff $A^{\mathcal{I}} = C^{\mathcal{I}}$. Analogously, $\mathcal{I}$ satisfies a GCI $C \sqsubseteq D$ iff $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$.

Both, concept definitions and GCIs, are called terminological axioms.

A TBox $\mathcal{S}$ is a finite set of concept definitions such that no concept name occurs more than once on the left-hand side of a definition in $\mathcal{S}$. A finite set of GCIs is called a *general* TBox.

Concept names occurring on the left hand side of a definition are called *defined concepts* while all other concept names occurring in the TBox are called *primitive concepts*. The set of defined concepts is denoted as N_{def} and the set of primitive concepts is denoted as N_{prim}. In addition, for a general TBox $\mathcal{T}$ we denote by $sig(\mathcal{T}) \subseteq N_C \cup N_R$ the set of concept names and role names occurring in $\mathcal{T}$.

An interpretation $\mathcal{I}$ is a model of a general TBox $\mathcal{T}$ iff $\mathcal{I}$ satisfies all the axioms in $\mathcal{T}$ (denoted as $\mathcal{I} \models \mathcal{T}$). Note that a TBox is a particular case of a general TBox since a concept definition $A \equiv C$ can be expressed using two GCIs: $A \sqsubseteq C$ and $C \sqsubseteq A$.

Often, it is convenient to transform a general TBox $\mathcal{T}$ into a normal form in order to simplify its structure. We accept the same definition as in [13]:

Definition 2.2.2. Let $\mathcal{T}$ be a general TBox defined over N_{prim} and N_R. $\mathcal{T}$ is in normal form iff $\mathcal{T}$ contains only GCIs and every GCI is of the form $A \sqcap B \sqsubseteq C$ where A, B are flat atoms or $\top$ and C is a flat atom. A normalized general TBox is also called a *flat* general TBox.

$$\widehat{C} \sqcap D \,\rho\, E \longrightarrow \{A \equiv \widehat{C}, A \sqcap D \,\rho\, E\} \quad \text{(R1)}$$
$$C \,\rho\, D \sqcap \widehat{E} \longrightarrow \{C \,\rho\, D \sqcap A, A \equiv \widehat{E}\} \quad \text{(R2)}$$
$$\exists r.\widehat{C} \,\rho\, D \longrightarrow \{A \equiv \widehat{C}, \exists r.A \,\rho\, D\} \quad \text{(R3)}$$
$$C \,\rho\, \exists r.\widehat{D} \longrightarrow \{C \,\rho\, \exists r.A, A \equiv \widehat{D}\} \quad \text{(R4)}$$

$$B \equiv B_1 \sqcap B_2 \longrightarrow \{B \sqsubseteq B_1, B \sqsubseteq B_2, B_1 \sqcap B_2 \sqsubseteq B\} \quad \text{(R5)}$$
$$B \equiv \exists r.B' \longrightarrow \{B \sqsubseteq \exists r.B', \exists r.B' \sqsubseteq B\} \quad \text{(R6)}$$

Figure 2.1: Rules used to normalize a general TBox.

To transform a given general TBox $\mathcal{T}$ into a flat general TBox, we use the normalization procedure proposed in [4] that consists of the application of rules $(R1) - (R6)$ shown in Figure 2.1. In these rules C, D, E stand for arbitrary concept descriptions, $\widehat{C}, \widehat{D}, \widehat{E}$ are concept descriptions that are not concept names, A is always a new concept name not ocurring in $\mathcal{T}$, $r \in N_R$, $\rho \in \{\sqsubseteq, \equiv\}$ and B, B', B_1, B_2 represent concept names.

First, rules $(R1) - (R4)$ are exhaustively applied to obtain a new TBox that consists of flat GCIs and additional flat concept definitions. Second, the application of rules $(R5) - (R6)$ transforms those remaining concept definitions into GCIs. The result is a general TBox $\mathcal{T}'$ whose axioms are of one of the following forms:

$$A \sqsubseteq B$$
$$A_1 \sqcap A_2 \sqsubseteq B$$
$$A \sqsubseteq \exists r.B$$
$$\exists r.A \sqsubseteq B$$

where A_1, A_2, A, B are concept names. One can see that all the atoms occurring in the result of such a normalization procedure are flat atoms.

A TBox $\mathcal{S}$ is called acyclic if there are no cyclic dependencies between its concept definitions. The following example, taken from [18], illustrates the notion of a cyclic dependency:

Example 2.2.3.

$$Car \equiv Vehicle \sqcap \exists has\_engine.Car\_engine$$
$$Car\_engine \equiv Engine \sqcap \exists is\_engine\_of.Car$$

One can see that the definition of Car depends on Car_engine, but at the

same time the definition of Car_engine depends on Car. Thus, Car is defined in terms of itself. Formally,

Definition 2.2.4. Let $\mathcal{S}$ be a TBox. We define $\rightarrow$ as a relation over the set N_{def} that represents *direct dependency* between defined concepts in the following way.

A defined concept A *directly depends on* a defined concept B (denoted as $A \rightarrow B$) iff $A \equiv C \in \mathcal{S}$ and B occurs in C, i.e., there is a top-level atom in C that contains an occurrence of B.

Let $\rightarrow^+$ be the transitive closure of $\rightarrow$. The TBox $\mathcal{S}$ contains a terminological cycle iff there is a defined concept A in $\mathcal{S}$ that depends on itself, i.e., $A \rightarrow^+ A$. We say that A is *cyclic-defined* in $\mathcal{S}$. $\mathcal{S}$ is called cyclic if it contains a terminological cycle, otherwise it is called acyclic.

Cyclic defined concepts can be classified according to the structure of their cyclic definitions. We distinguish two types of cyclic-defined concepts according to [18] in the following way:

Definition 2.2.5. Let $\mathcal{S}$ be a TBox and A_0, A_n defined concepts in $\mathcal{S}$.

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 $\mathcal{S}$ such that:

$$A_i \rightarrow A_{i+1} \text{ for all } i, 0 \leq i < n$$

and, A_i is a top-level atom in the definition of A_{i-1} for all $i > 0$, i.e., A_i appears outside the scope of any existential restriction in the definition of A_{i-1}. If, in addition, $A_0 = A_n$ then $A_0, \ldots, A_n$ is called a *component-cycle* in $\mathcal{S}$.

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

On the side of acyclic definitions, it can be seen that acyclicity in a TBox $\mathcal{T}$ has the advantage that for every model of $\mathcal{T}$, the meaning of the defined concepts follows directly from the meaning of the primitive concepts occurring in their definitions. The following proposition, shown in [18], expresses formally this property:

Proposition 2.2.6. *Given a TBox $\mathcal{S}$ without terminological cycles, any interpretation $\mathcal{J}$ of the primitive concepts occurring in $\mathcal{S}$ can be uniquely extended to a model of $\mathcal{S}$.*

2.3 Greatest Fixpoint Semantics

The semantics that we have defined for TBoxes was introduced by Nebel [18] as *descriptive semantics*. Under this semantics, the interpretation of defined concepts can be completely derived from the interpretation of primitive concepts and role names in the presence of acyclic TBoxes, since there is only one way to extend an interpretation of the primitive concepts to a model of a TBox.

In descriptive semantics, all the possible models of a TBox are considered as *admissible models*. However, for a cyclic defined TBox an interpretation of the primitive concepts could be extended in more than one way to a model of the TBox. The following example from [3], shows a case where not every extension represents correctly the intended meaning of a defined concept.

Example 2.3.1. Graphs can be represented by interpretations where nodes are elements of the primitive concept name $Node$ and edges are represented by the role name *edge*. Suppose that the concept of a node that is in an infinite path is defined in the following way:

$$INode \equiv Node \sqcap \exists edge.INode$$

where $INode$ is a cyclic defined concept. Then, consider the following interpretation $\mathcal{J}$ that assigns values to the primitive concepts and roles:

$$\begin{aligned} D_{\mathcal{J}} &= \{m_1, m_2, \ldots\} \cup \{n_1\} \\ Node^{\mathcal{J}} &= D_{\mathcal{J}} \\ edge^{\mathcal{J}} &= \{(m_i, m_{i+1}) \mid i \geq 1\} \cup \{(n_1, n_1)\} \end{aligned}$$

Now let $\mathcal{S}$ be the TBox consisting of the concept definition for $INode$. The cyclic dependency in the definition of $INode$ implies that there is more than one way to extend the interpretation $\mathcal{J}$ to an interpretation that is a model of $\mathcal{S}$. More precisley, $INode$ can be interpreted as $\{m_1, m_2, \ldots\} \cup \{n_1\}$, $\{m_1, m_2, \ldots\}$, $\{n_1\}$ or $\emptyset$.

All these models are valid w.r.t. descriptive semantics, but only the first of them expresses the intensional meaning of $INode$ correctly. Therefore, there are situations in which it is suitable to consider only one particular model extending a given initial interpretation of the primitive concepts. One of the alternatives is to consider the *greatest fixpoint model* of $\mathcal{S}$.

Definition 2.3.2. Let $\mathcal{S}$ be a TBox containing the role names, primitive concept names and defined concept names in N_R, N_{prim} and N_{def} respectively, where $N_{prim} = \{P_1, \ldots, P_n\}$ and $N_{def} = \{A_1, \ldots, A_m\}$.

A primitive interpretation $\mathcal{J}$ for $\mathcal{S}$ consists of a domain $D_{\mathcal{J}}$, an interpretation of the role names in N_R and an interpretation of the primitive concept names in N_{prim} by subsets of $D_{\mathcal{J}}$. An interpretation $\mathcal{I}$ is based on the primitive interpretation $\mathcal{J}$ iff the following conditions hold:

- The domain of $\mathcal{I}$ is $D_{\mathcal{J}}$, i.e., $D_{\mathcal{I}} = D_{\mathcal{J}}$.
- $P_i^{\mathcal{I}} = P_i^{\mathcal{J}}$ for all the primitive concepts in N_{prim}.
- $r^{\mathcal{I}} = r^{\mathcal{J}}$ for all the role names in N_R.

Given a primitive interpretation $\mathcal{J}$ there could be many interpretations extending $\mathcal{J}$ to the defined concepts in $\mathcal{S}$. The set of all interpretations based on $\mathcal{J}$ is denoted as $Int(\mathcal{J})$ and it can be seen that they are uniquely identified by their assignments to the defined concepts in $\mathcal{S}$. Based on this, the following order can be used to compare interpretations in $Int(\mathcal{J})$:

Let $\mathcal{I}_1, \mathcal{I}_2 \in Int(\mathcal{J})$ then,

$$\mathcal{I}_1 \preceq_{\mathcal{J}} \mathcal{I}_2 \text{ iff } A_i^{\mathcal{I}_1} \subseteq A_i^{\mathcal{I}_2} \text{ for all } i, 1 \leq i \leq m$$

It is not difficult to see that the pair $(Int(\mathcal{J}), \preceq_{\mathcal{J}})$ is a partially ordered set. Moreover, for any subset $\mathcal{P}$ of $Int(\mathcal{J})$ there are interpretations $\mathcal{I}_{sup}$ and $\mathcal{I}_{inf}$ in $Int(\mathcal{J})$ such that $\mathcal{I}_{sup}$ is the *least upper bound* and $\mathcal{I}_{inf}$ is the *greatest lower bound* of $\mathcal{P}$ w.r.t. $\preceq_{\mathcal{J}}$.

Hence, one can see that $(Int(\mathcal{J}), \preceq_{\mathcal{J}})$ is a *complete lattice* on $Int(\mathcal{J})$ and thus, Tarski's fixpoint theorem [19] applies to all monotonic functions from $Int(\mathcal{J})$ to $Int(\mathcal{J})$.

The application of Tarki's theorem w.r.t. $(Int(\mathcal{J}), \preceq_{\mathcal{J}})$ can be understood in the following way: if $f : Int(\mathcal{J}) \rightarrow Int(\mathcal{J})$ is a function such that $\mathcal{I}_1 \preceq_{\mathcal{J}} \mathcal{I}_2$ implies $f(\mathcal{I}_1) \preceq_{\mathcal{J}} f(\mathcal{I}_2)$ (f is increasing), then there is an interpretation $\mathcal{I} \in Int(\mathcal{J})$ such that $f(\mathcal{I}) = \mathcal{I}$, also called a fixpoint of f. In addition, the theorem also says that f has a *least fixpoint* and a *greatest fixpoint*.

It was shown in [3] that every TBox $\mathcal{S}$ and primitive interpretation $\mathcal{J}$ of $\mathcal{S}$ induce an increasing function $O_{\mathcal{S},\mathcal{J}} : Int(\mathcal{J}) \rightarrow Int(\mathcal{J})$ such that $O_{\mathcal{S},\mathcal{J}}(\mathcal{I}) = \mathcal{I}$ iff $\mathcal{I}$ is a model of $\mathcal{S}$. Thus, any primitive interpretation $\mathcal{J}$ can be extended to a model of $\mathcal{S}$ and there is always a greatest and a least model of $\mathcal{S}$ extending $\mathcal{J}$.

Definition 2.3.3. Let $\mathcal{S}$ be a TBox. A model $\mathcal{I}$ of $\mathcal{S}$ is called a greatest fixpoint model (*gfp-model*) of $\mathcal{S}$ iff there is a primitive interpretation $\mathcal{J}$ of $\mathcal{S}$ such that $\mathcal{I} \in Int(\mathcal{J})$ and $\mathcal{I}$ is the greatest fixpoint of $O_{\mathcal{S},\mathcal{J}}$.

In other words, for every model $\mathcal{I}'$ of $\mathcal{S}$ such that $\mathcal{I}' \in Int(\mathcal{J})$ holds that $\mathcal{I}' \preceq_{\mathcal{J}} \mathcal{I}$. *Greatest fixpoint semantics* considers only gfp-models as admissible models.

Note that there is always only one gfp-model of $\mathcal{S}$ extending a primitive interpretation $\mathcal{J}$. Furthermore, greatest fixpoint semantics is equivalent to descriptive semantics w.r.t. acyclic TBoxes.

Lemma 2.3.4. *Let $\mathcal{S}$ be an acyclic TBox. Then, an interpretation $\mathcal{I}$ is a model of $\mathcal{S}$ iff it is a gfp-model of $\mathcal{S}$.*

Proof. ($\Leftarrow$) The *right to left* direction is immediate since every gfp-model of $\mathcal{S}$ is a model of $\mathcal{S}$.

($\Rightarrow$) Let $\mathcal{I}$ be a model of $\mathcal{S}$ based on the primitive interpretation $\mathcal{J}$. Since $\mathcal{S}$ is acyclic, by Proposition 2.2.6, $\mathcal{I}$ is the unique model of $\mathcal{S}$ extending $\mathcal{J}$. Therefore, $\mathcal{I}$ is a fixpoint of $O_{\mathcal{S},\mathcal{J}}$ and it is the only one which implies that it must be the greatest fixpoint of $O_{\mathcal{S},\mathcal{J}}$.

Thus, $\mathcal{I}$ is a gfp-model of $\mathcal{S}$. □

In addition, it has been shown in [18] that in the presence of greatest fixpoint semantics a TBox $\mathcal{S}$ containing component cycles can be transformed into a TBox $\mathcal{S}'$ that is free of component cycles:

Lemma 2.3.5. *Let $\mathcal{S}$ be a TBox that contains component cycles. Then, there exists a TBox $\mathcal{S}'$ that does not contain component cycles such that:*

$$\mathcal{I} \text{ is a gfp-model of } \mathcal{S} \text{ iff } \mathcal{I} \text{ is a gfp-model of } \mathcal{S}'$$

Proof. The idea of the proof described in [18] is the following. Let $\mathcal{S}$ be a TBox and A be *component-cyclic-defined* in $\mathcal{S}$. The set C_c is defined as the *largest* set of concepts that use A as a component (see Definition 2.2.5) and are used by A as components.

Now, let A_D be a concept description identical to A except that all occurrences of concepts from C_c that do not appear in existential restrictions are replaced by $\top$. It can be seen that all concepts B in C_c have the same interpretation under any gfp-model of $\mathcal{S}$ and they can be equivalently expressed as:

$$B \equiv_{gfp,\mathcal{S}} \bigsqcap_{B' \in C_c} B'_D \qquad (1)$$

Note that this new definition for B implies that B is not *component-cyclic-defined* anymore. Using these equivalences the TBox $\mathcal{S}$ can be transformed in an equivalent TBox $\mathcal{S}'$ free of component cycles.

Finally, one can see that the size of the definition of any concept in $\mathcal{S}$ is polynomial on the size of $\mathcal{S}$ and C_c contains polynomially many elements. Then, definitions like (1) are of size polynomial on the size of $\mathcal{S}$. Thus, the size of the obtained TBox $\mathcal{S}'$ is also polynomial on the size of $\mathcal{S}$. □

Chapter 3

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

The purpose of this chapter is to present the $\mathcal{EL}$-unification problem w.r.t. a general TBox formally, and to review the NP-results that has been obtained so far concerning this problem. As important elements required to define the unification problem, we introduce the subsumption problem in $\mathcal{EL}$ and the notion of substitution. The concept of *local unifier* is defined and its role as the main idea underlying the NP-results is stressed. The restriction of these results to a proper subclass of arbitrary general TBoxes, leads to our proposal of a new approach that solves the unification problem w.r.t. arbitrary general TBoxes.

3.1 Subsumption in $\mathcal{EL}$

Definition 3.1.1. Let $\mathcal{T}$ be a general TBox and C, D two concept descriptions. C is subsumed by D w.r.t. $\mathcal{T}$ iff $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ for all models $\mathcal{I}$ of $\mathcal{T}$.

In addition, it makes sense to define the subsumption problem under greatest fixpoint semantics when a (possibly cyclic) TBox is considered:

Let $\mathcal{S}$ be a TBox and A, B two defined concepts, then A is subsumed by B w.r.t. $\mathcal{S}$ under greatest fixpoint semantics iff $A^{\mathcal{I}} \subseteq B^{\mathcal{I}}$ for all the gfp-models $\mathcal{I}$ of $\mathcal{S}$.

The subsumption relation w.r.t. $\mathcal{T}$ is denoted as $C \sqsubseteq_{\mathcal{T}} D$ or alternatively as $\mathcal{T} \models C \sqsubseteq D$ (read as $C \sqsubseteq D$ *follows from* $\mathcal{T}$). Conversely, we denote by $A \sqsubseteq_{gfp,\mathcal{S}} B$ ($\mathcal{S} \models_{gfp} A \sqsubseteq B$) the subsumption relation w.r.t. $\mathcal{S}$ under greatest fixpoint semantics.

Note that the subsumption problem for a TBox $\mathcal{S}$ is restricted to defined concepts occurring in $\mathcal{S}$. This is without loss of generality since deciding $C \sqsubseteq^?_{gfp,\mathcal{S}} D$ can be reduced to decide $A \sqsubseteq^?_{gfp,\mathcal{S}} B$, where $A \equiv C$ and $B \equiv D$ are added to $\mathcal{S}$.

The subsumption problem in $\mathcal{EL}$ has been extensively studied, see [1, 3, 5]. It was shown in [3] that the subsumption problem is decidable in polynomial time for cyclic TBoxes under greatest fixpoint and descriptive semantics. This result was later generalized in [5] for the case of general TBoxes under descriptive semantics.

To conclude this section we present an important relation that exists between a general TBox $\mathcal{T}$ and the general TBox $\mathcal{T}'$, that results from the application of the normalization procedure described in Section 2.2, w.r.t. subsumption reasoning.

In [16] the notion of *inseparability* was introduced as a relation between TBoxes that, for example, can be used to determine whether two TBoxes imply the same subsumption relations. We present such a notion as it was defined in [13]:

Definition 3.1.2. Let $\Sigma \subseteq N_C \cup N_R$ be a signature. Two general TBoxes $\mathcal{T}_1$, $\mathcal{T}_2$ are Σ-*inseparable* if for any concept descriptions C and D built over the signature Σ we have $C \sqsubseteq_{\mathcal{T}_1} D$ iff $C \sqsubseteq_{\mathcal{T}_2} D$.

In particular, from [13], we have that $\mathcal{T}'$ is $sig(\mathcal{T})$-inseparable from $\mathcal{T}$ if $\mathcal{T}'$ is the result of applying the normalization procedure presented in Section 2.2 to a general TBox $\mathcal{T}$. Hence, the following proposition is immediate:

Proposition 3.1.3. *Let $\mathcal{T}$ be a general TBox and $\mathcal{T}'$ be the result of applying the normalization procedure described in Section 2.2. Then, for any concept descriptions C and D built over $sig(\mathcal{T})$ it is true that:*

$$C \sqsubseteq_{\mathcal{T}} D \text{ iff } C \sqsubseteq_{\mathcal{T}'} D$$

3.2 Unification in $\mathcal{EL}$

Before defining the unification problem in $\mathcal{EL}$, we need to introduce the notion of substitution on concept descriptions. In order to do this, the set N_C of concept names is partitioned into a set N_v of *concept variables* and a set N_c of *concept constants*.

Definition 3.2.1. A substitution σ is a mapping from the set of concept variables to the set of $\mathcal{EL}$-concept descriptions over N_C. It can be extended to arbitrary concept descriptions in the usual way:

- $\sigma(A) := A$ for all $A \in N_c \cup \{\top\}$

- $\sigma(C \sqcap D) := \sigma(C) \sqcap \sigma(D)$
- $\sigma(\exists r.C) := \exists r.\sigma(C)$

A concept description C is called *ground* if it does not contain any occurrence of a variable from N_v and a substitution σ is called a *ground substitution* if for any variable X in N_v the concept description $\sigma(X)$ is ground. Finally, a general TBox $\mathcal{T}$ is ground if no variable from N_v occurs in a GCI of $\mathcal{T}$.

Definition 3.2.2. Let $\mathcal{T}$ be a ground general TBox. An $\mathcal{EL}$-unification problem w.r.t. $\mathcal{T}$ is a finite set of subsumptions $\Gamma = \{C_1 \sqsubseteq^? D_1, \ldots, C_n \sqsubseteq^? D_n\}$. Γ has a solution w.r.t. $\mathcal{T}$ if there exists a substitution σ such that all the subsumptions in Γ are solved w.r.t. $\mathcal{T}$ by applying σ to its concept descriptions, i.e.:

$$\sigma(C_1) \sqsubseteq_{\mathcal{T}} \sigma(D_1), \ldots, \sigma(C_n) \sqsubseteq_{\mathcal{T}} \sigma(D_n)$$

If such a substitution σ exists then σ is called an *$\mathcal{EL}$-unifier* of Γ w.r.t. $\mathcal{T}$ and Γ is called *$\mathcal{EL}$-unifiable* w.r.t. $\mathcal{T}$.

Regarding this definition, one can see that the general TBox representing the background knowledge is assumed to be ground. This is not without loss of generality, but it is nevertheless appropriate for the domain of interest. More details concerning this observation can be found in [13].

In [13], it was also emphasized that in order to decide unifiability of Γ it is sufficient to consider ground substitutions defined over the concept and role names occurring in Γ or $\mathcal{T}$. In addition, the relation that exists between ground substitutions and acyclic TBoxes was considered as a different view of what the unification problem is trying to compute.

Any ground substitution σ induces a TBox $\mathcal{S}_\sigma$ that can be built in the following way:

$$\mathcal{S}_\sigma := \{X \equiv \sigma(X) \mid X \in N_v\}$$

Since σ is ground then it is clear that $\mathcal{S}_\sigma$ is acyclic. In addition, one can see that the defined concepts in $\mathcal{S}_\sigma$ are the variables in σ and then, the set N_c of concept constants can be seen as the set of primitive concepts N_{prim} and the set N_v of concept variables as the set of defined concepts N_{def}.

Now, for any concept description C if we consider $C^{\mathcal{S}_\sigma}$ as the expansion of C w.r.t. $\mathcal{S}_\sigma$ where any defined concept occurring in C is substituted by its definition in $\mathcal{S}_\sigma$, then we have that $\sigma(C) = C^{\mathcal{S}_\sigma}$ and the following proposition is immediate:

Proposition 3.2.3. *Let $\mathcal{T}$ be a ground general TBox, σ be a ground substitution and C, D be two concept descriptions. Then,*

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

At the same time, any acyclic TBox $\mathcal{S}$ induces a ground substitution $\sigma_{\mathcal{S}}$, whose variables are the defined concepts in $\mathcal{S}$. The acyclicity of $\mathcal{S}$ implies that the expansion $X^{\mathcal{S}}$ of each defined concept X in $\mathcal{S}$ is ground and $\sigma_{\mathcal{S}}$ can be expressed as:

$$\sigma_{\mathcal{S}}(X) := X^{\mathcal{S}}$$

Similar as above, for any concept description C its expansion w.r.t. $\mathcal{S}$ is equivalent to the application of the substitution $\sigma_{\mathcal{S}}$ to C, i.e., $C^{\mathcal{S}} = \sigma_{\mathcal{S}}(C)$. Thus, the converse of Proposition 3.2.3 also holds,

Proposition 3.2.4. *Let $\mathcal{T}$ be a general TBox, $\mathcal{S}$ be an acyclic TBox and C, D be two concept descriptions. Then,*

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

The combination of these two propositions implies that finding an $\mathcal{EL}$-unifier for Γ w.r.t. $\mathcal{T}$ is equivalent to find an acyclic TBox $\mathcal{S}$ such that the subsumptions in Γ follow from $(\mathcal{T} \cup \mathcal{S})$.

Lemma 3.2.5. *Let $\mathcal{T}$ be a ground general TBox and $\Gamma = \{C_1 \sqsubseteq^? D_1, \ldots, C_n \sqsubseteq^? D_n\}$ be an $\mathcal{EL}$-unification problem w.r.t. $\mathcal{T}$. Then, Γ has a unifier w.r.t. $\mathcal{T}$ iff there exists an acyclic TBox $\mathcal{S}$ whose defined concepts are the variables in N_v and $C_i \sqsubseteq_{\mathcal{T} \cup \mathcal{S}} D_i$ for all $C_i \sqsubseteq^? D_i$ in Γ.*

Proof. ($\Rightarrow$) Assume that there exists a unifier σ for Γ w.r.t. $\mathcal{T}$. Then, by Definition 3.2.2 $\sigma(C_i) \sqsubseteq_{\mathcal{T}} \sigma(D_i)$ for all $C_i \sqsubseteq^? D_i \in \Gamma$. Applying Proposition 3.2.3 we have that there exists an acyclic TBox, namely $\mathcal{S}_\sigma$, such that $C_i \sqsubseteq_{\mathcal{T} \cup \mathcal{S}_\sigma} D_i$ for all $C_i \sqsubseteq^? D_i \in \Gamma$.

($\Leftarrow$) The converse direction can be shown in a similar way applying Proposition 3.2.4. □

This lemma gives a different way to understand the meaning of the unification problem. Given a finite set of GCIs (represented by Γ), where some concept names have been marked as variables, and a knowledge base (represented by $\mathcal{T}$): intuitively, the unification problem is trying to compute concept definitions for those variables (the computed acyclic TBox $\mathcal{S}$), such that every GCI in Γ is a consequence of the new knowledge base $(\mathcal{T} \cup \mathcal{S})$.

3.3 NP-results for Unification in $\mathcal{EL}$

The unification problem in $\mathcal{EL}$ was first studied in [11, 12] w.r.t. the empty TBox and shown to be NP-Complete. It was shown to be in NP based on the idea that any $\mathcal{EL}$-unification problem that is unifiable w.r.t. the empty TBox has a *local unifier*.

Let Γ be a unification problem where the set of atoms of Γ is denoted as At and the set of *non-variable* atoms as $At_{nv} := At \setminus N_v$. The notion of local unifier is defined in the following way:

Definition 3.3.1. Let Γ be a unification problem and ζ be an assignment of subsets of At_{nv} to each variable $X \in N_v$. The assignment ζ induces the following TBox:

$$\mathcal{S} := \{X \equiv \bigsqcap_{D \in \zeta_X} D \mid X \in N_v\}$$

where $\bigsqcap_{D \in \zeta_X} D$ denotes the conjunction of all elements in ζ_X.

The assignment ζ is called acyclic if $\mathcal{S}$ is an acyclic TBox. Then, if $\mathcal{S}$ is acyclic, the TBox $\mathcal{S}$ induces a unique substitution $\sigma_\mathcal{S}$ (called *local substitution*) in the following way:

$$\sigma_\mathcal{S}(X) = \bigsqcap_{D \in \zeta_X} \sigma_\mathcal{S}(D)$$

Finally, if $\sigma_\mathcal{S}$ is a unifier of Γ then it is called a local unifier.

It can be seen that the number of non-variable atoms is polynomial in the size of Γ as well as the number of variables and thus, one can guess an acyclic assignment ζ in polynomial time. Moreover, since subsumption is decidable in polynomial time, one can check whether the induced local substitution $\sigma_\mathcal{S}$ is a unifier of Γ in polynomial time. This yields an NP-procedure to decide the $\mathcal{EL}$-unification problem w.r.t. the empty TBox that searches for the existence of a local unifier.

In [12], it was also shown that the same ideas and results apply for the case when the unification problem is considered w.r.t. a *non-empty* acyclic TBox. Nevertheless, when extending these results for arbitrary general TBoxes, it turns out that the notion of locality does not work. To fix this problem a first approach has been recently proposed in [14] where the unification problem is considered w.r.t. *cycle-restricted* TBoxes.

Definition 3.3.2. A general TBox $\mathcal{T}$ is called *cycle-restricted* iff there is no nonempty word $w \in N_R^+$ and $\mathcal{EL}$-concept description C such that $C \sqsubseteq_\mathcal{T} \exists w.C$.

The $sig(\mathcal{T})$-inseparability that exists between a general TBox $\mathcal{T}$ and its corresponding flat general TBox $\mathcal{T}'$ (see Definition 3.1.2 and Proposition 3.1.3) allows us to assume without loss of generality that unification is considered w.r.t. flat general TBoxes. Likewise, we can assume that the unification problem Γ is flat in the sense that it contains only subsumptions of the form $C_1 \sqcap \ldots \sqcap C_n \sqsubseteq^? D$ where $C_1, \ldots, C_n$ and D are flat atoms, see [13].

Combining this normal form together with an appropriate characterization of the subsumption problem, the following theorem was shown in [13].

Theorem 3.3.3. *Let $\mathcal{T}$ be a flat cycle-restricted TBox and Γ be a flat unification problem. If Γ has a unifier w.r.t. $\mathcal{T}$ then it has a local unifier w.r.t. $\mathcal{T}$.*

This theorem shows that unification in $\mathcal{EL}$ w.r.t. cycle-restricted TBoxes remains in NP, however there is still no solution to the problem if the *cycle-restricted* constraint is lifted. On the one hand, it was also shown in [13] that cycle-restricted TBoxes do not reach the full expressivity of arbitrary general TBoxes. On the other hand, an example is provided that shows a unification problem that is unifiable w.r.t. a not cycle-restricted TBox, but does not have any local unifier.

As a consequence, the solution that has been proposed in [13] is not complete for arbitrary general TBoxes. In order to repair this, we propose to extend what has been considered as a unifier until now, to a TBox that may contain cyclic definitions.

We will show that a similar locality result holds in this case for arbitrary general TBoxes, but first we need to introduce a suitable semantics to deal with possibly cyclic defined solutions of the unification problem, in an appropriate way.

Chapter 4

Hybrid Semantics

In [6], an appropriate semantics was introduced to interpret the so-called *hybrid* TBoxes. In this chapter, we present such a semantics, and describe the proof-system proposed in [7] to characterize and obtain a tractable decision procedure for the subsumption problem w.r.t. hybrid TBoxes.

Based on this semantics and the corresponding subsumption problem, we define the *hybrid* $\mathcal{EL}$-unification problem w.r.t. general TBoxes as the extension of the $\mathcal{EL}$-unification problem proposed at the end of Chapter 3. Providing a solution to this problem is the main result of this thesis, it will be addressed in the next two chapters.

4.1 Hybrid $\mathcal{EL}$-TBoxes

Definition 4.1.1. Let N_R be a set of role names, N_{prim} a set of primitive concept names and N_{def} a set of defined concept names.

A *hybrid* $\mathcal{EL}$-TBox is a pair $(\mathcal{T}, \mathcal{S})$ where $\mathcal{T}$ is a general $\mathcal{EL}$-TBox over N_{prim} and N_R, and $\mathcal{S}$ is an $\mathcal{EL}$-TBox over N_{prim}, N_{def} and N_R such that for each concept definition $A \equiv C$ in $\mathcal{S}$, the concept name A is in N_{def}.

A suitable semantics that combines descriptive semantics and gfp-semantics has been proposed in [6] to interpret hybrid TBoxes. It is called *hybrid-semantics* and it is defined in the following way:

Definition 4.1.2. Let $(\mathcal{T}, \mathcal{S})$ be a hybrid $\mathcal{EL}$-TBox defined over N_R, N_{prim}, N_{def} and $\mathcal{I}$ be an interpretation based on a primitive interpretation $\mathcal{J}$:

$$\mathcal{I} \text{ is a } \textit{hybrid-model} \text{ of } (\mathcal{T}, \mathcal{S})$$

iff

$$\mathcal{J} \models \mathcal{T} \text{ and } \mathcal{I} \text{ is a gfp-model of } \mathcal{S}.$$

In order to define the unification problem w.r.t. hybrid $\mathcal{EL}$-TBoxes first we need to define the corresponding subsumption problem. We introduce such a problem based on the semantics for hybrid $\mathcal{EL}$-TBoxes defined above.

Definition 4.1.3. Let $(\mathcal{T}, \mathcal{S})$ be a hybrid $\mathcal{EL}$-TBox and A, B defined concepts in $\mathcal{S}$. A is subsumed by B w.r.t. $(\mathcal{T}, \mathcal{S})$ iff $A^{\mathcal{I}} \subseteq B^{\mathcal{I}}$ holds for all hybrid-models $\mathcal{I}$ of $(\mathcal{T}, \mathcal{S})$.

We denote such a relation as $A \sqsubseteq_{gfp,\mathcal{T}\cup\mathcal{S}} B$.

Like in Definition 3.1.2, we define the notion of inseparability for hybrid TBoxes:

Definition 4.1.4. Let $\Sigma \subseteq N_C \cup N_R$ be a signature. Two hybrid TBoxes $(\mathcal{T}_1, \mathcal{S}_1)$ and $(\mathcal{T}_2, \mathcal{S}_2)$ are Σ-*hybrid-inseparable* if for all concept descriptions C and D built over the signature Σ we have $C \sqsubseteq_{gfp,\mathcal{T}_1\cup\mathcal{S}_1} D$ iff $C \sqsubseteq_{gfp,\mathcal{T}_2\cup\mathcal{S}_2} D$.

Having this definition, the following proposition shows that hybrid-inseparability holds after applying the normalization procedure described in Section 2.2.

Proposition 4.1.5. *Let $(\mathcal{T}, \mathcal{S})$ be a hybrid TBox and $\mathcal{T}'$ the result of applying the normalization procedure described in Section 2.2 to $\mathcal{T}$. Then, the hybrid TBox $(\mathcal{T}', \mathcal{S})$ is $sig(\mathcal{T} \cup \mathcal{S})$-hybrid-inseparable from $(\mathcal{T}, \mathcal{S})$.*

Proof. We show that for any concept descriptions C and D defined over $sig(\mathcal{T} \cup \mathcal{S})$, $C \sqsubseteq_{gfp,\mathcal{T}\cup\mathcal{S}} D$ iff $C \sqsubseteq_{gfp,\mathcal{T}'\cup\mathcal{S}} D$.

($\Rightarrow$) Assume that $C \sqsubseteq_{gfp,\mathcal{T}\cup\mathcal{S}} D$ holds. We have to show that for each hybrid-model $\mathcal{I}$ of $(\mathcal{T}', \mathcal{S})$, $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ holds.

Since $\mathcal{T}'$ is $sig(\mathcal{T})$-inseparable from $\mathcal{T}$ then, for each GCI $E \sqsubseteq F$ in $\mathcal{T}$ one can see that:

- E and F are concept descriptions defined over $sig(\mathcal{T})$.
- Obviously, $E \sqsubseteq_{\mathcal{T}} F$ holds.
- The application of Proposition 3.1.3 yields that $E \sqsubseteq_{\mathcal{T}'} F$ holds as well.

Now, consider any hybrid-model $\mathcal{I}$ of $(\mathcal{T}', \mathcal{S})$ and let $\mathcal{J}$ be the primitive interpretation that $\mathcal{I}$ is based on. By Definition 4.1.2, $\mathcal{J}$ must be a model of $\mathcal{T}'$ and hence $E^{\mathcal{J}} \subseteq F^{\mathcal{J}}$ holds for all GCI $E \sqsubseteq F$ in $\mathcal{T}$. Thus, $\mathcal{J}$ is a model of $\mathcal{T}$ and consequently $\mathcal{I}$ is a hybrid-model of $(\mathcal{T}, \mathcal{S})$.

Finally, by Definition 4.1.3 we obtain that $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$. Thus, $C \sqsubseteq_{gfp,\mathcal{T}'\cup\mathcal{S}} D$ holds.

($\Leftarrow$) Assume that $C \sqsubseteq_{gfp,\mathcal{T}'\cup\mathcal{S}} D$ holds, and consider an arbitrary hybrid-model $\mathcal{I}$ of $(\mathcal{T}, \mathcal{S})$. It is not difficult to see that $\mathcal{I}$ can be extended to a

hybrid-model $\mathcal{I}'$ of $(\mathcal{T}', \mathcal{S})$, by assigning values to the new primitive concepts introduced in $\mathcal{T}'$ during the normalization. Therefore, $C^{\mathcal{I}'} \subseteq D^{\mathcal{I}'}$ holds.

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

Thus, $C \sqsubseteq_{gfp, \mathcal{T} \cup \mathcal{S}} D$ holds. □

The subsumption problem in $\mathcal{EL}$ w.r.t. hybrid $\mathcal{EL}$-TBoxes was considered first in [6] and later in [7, 8]. Two different approaches were proposed to solve the problem and in both cases a polynomial time decision procedure was provided.

In particular, a Gentzen-style proof calculus was introduced in [1] to decide subsumption in $\mathcal{EL}$ w.r.t. cyclic TBoxes interpreted under greatest fixpoint semantics and w.r.t. general TBoxes under descriptive semantics as well. It was later extended, in [8], to characterize subsumption in $\mathcal{EL}$ for hybrid TBoxes.

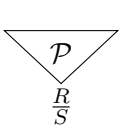
In the following, we introduce the proof calculus **HC** (Hybrid $\mathcal{EL}$ TBox Calculus) from [8] and the corresponding notion of proof-tree.

Definition 4.1.6. Let $(\mathcal{T}, \mathcal{S})$ be a hybrid $\mathcal{EL}$-TBox.

A sequent for $(\mathcal{T}, \mathcal{S})$ is of the form $C \sqsubseteq_n D$ where C and D are sub-descriptions of concept descriptions occurring in $(\mathcal{T}, \mathcal{S})$ and $n \geq 0$. The sub-descriptions C and D are sometimes called the *left* and *right-hand* side of the sequent, respectively.

The proof calculus **HC** is based on the set of rules shown in Figure 4.1. Using these rules, a proof tree in **HC** w.r.t. $(\mathcal{T}, \mathcal{S})$ is constructed as follows (where every node is represented by a sequent):

1. Every sequent that is a conclusion of an instance of the rules (Ax), (Top) and (Start) is a root of a one-element proof tree.
2. Let $\frac{R}{S}$ be an instance of one of the rules (AndL1), (AndL2), (Ex), (DefL) or (DefR). If there is a proof tree $\mathcal{P}$ with the root R, then



   is a proof tree with the root S with R as its only child.
3. Let $\frac{R_1 \quad R_2}{S}$ be an instance of one of the rules (AndR) or (GCI). If there are two proof trees $\mathcal{P}_1$ and $\mathcal{P}_2$ with roots R_1 and R_2, respectively, then

$$\frac{}{C \sqsubseteq_n C}\ (\text{Ax}) \qquad \frac{}{C \sqsubseteq_n \top}\ (\text{Top}) \qquad \frac{}{C \sqsubseteq_0 D}\ (\text{Start})$$

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

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

$$\frac{C \sqsubseteq_n D}{A \sqsubseteq_n D}\ (\text{DefL}) \quad \frac{D \sqsubseteq_n C}{D \sqsubseteq_{n+1} A}\ (\text{DefR}) \qquad \text{for } A \equiv C \in \mathcal{S}$$

$$\frac{C \sqsubseteq_n E \qquad F \sqsubseteq_n D}{C \sqsubseteq_n D}\ (\text{GCI}) \qquad \text{for } E \sqsubseteq F \in \mathcal{T}$$

Figure 4.1: Rule system **HC**

$$\frac{\begin{array}{cc}\mathcal{P}_1 & \mathcal{P}_2 \\ R_1 & R_2\end{array}}{S}$$

is a proof tree with the root S where R_1 and R_2 are the children of S.

For any instance $\frac{}{S}$, $\frac{R}{S}$ or $\frac{R_1 \quad R_2}{S}$ of a rule in **HC**, we say that R (R_1, R_2) is the premise (are the premises) and S is the consequence of the instance. One can see that for any proof tree in **HC**, its nodes are sequents and its leaves are consequences of an instance of some of the rules (Ax), (Top) or (Start).

The calculus **HC** induces a set of binary relations $\sqsubseteq_n$ for all $n \geq 0$ over the set of sub-descriptions of concept descriptions occurring in $(\mathcal{T}, \mathcal{S})$, where the membership relation in $\sqsubseteq_n$ is characterized by the existence of a proof tree in **HC** for two sub-descriptions C and D.

Definition 4.1.7. Let $(\mathcal{T}, \mathcal{S})$ be a hybrid TBox and C, D be sub-descriptions of concept descriptions occurring in $(\mathcal{T}, \mathcal{S})$.

$C \sqsubseteq_n D$ holds iff there exists a proof tree $\mathcal{P}$ in **HC** with the root $C \sqsubseteq_n D$. We say that $\mathcal{P}$ is a proof tree for $C \sqsubseteq_n D$ w.r.t. $(\mathcal{T}, \mathcal{S})$.

In addition, the relation $\sqsubseteq_\infty$ is defined such that $C \sqsubseteq_\infty D$ iff $C \sqsubseteq_n D$ holds for all $n \geq 0$.

The calculus **HC** has been shown to be a sound and complete calculus that characterizes subsumption w.r.t. hybrid $\mathcal{EL}$-TBoxes in the sense of membership into the relation $\sqsubseteq_\infty$. This means that given a hybrid TBox

$(\mathcal{T},\mathcal{S})$, if $A \sqsubseteq_\infty B$ is provable in **HC** w.r.t. $(\mathcal{T},\mathcal{S})$ then $A \sqsubseteq_{gfp,\mathcal{T}\cup\mathcal{S}} B$ holds. Conversely, if $A \sqsubseteq_{gfp,\mathcal{T}\cup\mathcal{S}} B$ holds there has to be a proof for $A \sqsubseteq_\infty B$ in **HC** w.r.t. $(\mathcal{T},\mathcal{S})$.

Theorem 4.1.8. *Let $(\mathcal{T},\mathcal{S})$ be a hybrid $\mathcal{EL}$-TBox and A, B be two defined concepts occurring in $\mathcal{S}$.*

$$A \sqsubseteq_{gfp,\mathcal{T}\cup\mathcal{S}} B \text{ iff } A \sqsubseteq_\infty B \text{ is provable in } \boldsymbol{HC}.$$

Notice, that this theorem implies transitivity of $\sqsubseteq_\infty$. If we have $A \sqsubseteq_\infty B$ and $B \sqsubseteq_\infty C$ then, we have $A \sqsubseteq_{gfp,\mathcal{T}\cup\mathcal{S}} B$ and $B \sqsubseteq_{gfp,\mathcal{T}\cup\mathcal{S}} C$ as well. That implies, that for any hybrid-model $\mathcal{I}$ of $(\mathcal{T},\mathcal{S})$ we have $A^\mathcal{I} \subseteq B^\mathcal{I}$ and $B^\mathcal{I} \subseteq C^\mathcal{I}$. Thus, $A^\mathcal{I} \subseteq C^\mathcal{I}$ and the application of the theorem yields $A \sqsubseteq_\infty C$.

As an important property of **HC** and the set of relations $\sqsubseteq_n$, one can observe that $\sqsubseteq_{n+1} \subseteq \sqsubseteq_n$ holds for all $n \geq 0$. This observation combined with the fact that the number of all possible sub-descriptions is polynomial on the size of a given hybrid TBox, yields a polynomial time fixed-point iteration procedure that decides subsumption for hybrid $\mathcal{EL}$-TBoxes under hybrid-semantics.

The idea of the procedure is that one can start with $\sqsubseteq_0$ and compute $\sqsubseteq_1$, $\sqsubseteq_2 \ldots$ until $\sqsubseteq_m = \sqsubseteq_{m+1}$ for some m.

Corollary 4.1.9. *Let $(\mathcal{T},\mathcal{S})$ be a hybrid $\mathcal{EL}$-TBox and A, B be two defined concepts in $\mathcal{S}$. There is a procedure that decides whether $A \sqsubseteq_{gfp,\mathcal{T}\cup\mathcal{S}} B$ holds for $(\mathcal{T},\mathcal{S})$, in time polynomial on the size of $(\mathcal{T},\mathcal{S})$.*

Complete details for the proofs of Theorem 4.1.8 and Corollary 4.1.9 as well as the description of the polynomial time decision procedure, can be found in [7, 8].

4.2 The Hybrid Unification Problem in $\mathcal{EL}$

Now, we are ready to extend the unification problem in $\mathcal{EL}$ to accept possibly cyclic solutions as we have proposed at the end of Chapter 3. We define such an extension based on the semantics presented above, and we call it the *hybrid* $\mathcal{EL}$-unification problem.

Definition 4.2.1. Let $\mathcal{T}$ be a ground general $\mathcal{EL}$-TBox. A *hybrid* $\mathcal{EL}$-unification problem w.r.t. $\mathcal{T}$ is a finite set of subsumptions $\Gamma = \{C_1 \sqsubseteq^? D_1, \ldots, C_n \sqsubseteq^? D_n\}$ between $\mathcal{EL}$-concept descriptions.

An $\mathcal{EL}$-TBox $\mathcal{S}$ is a *hybrid-unifier* of Γ w.r.t. $\mathcal{T}$ iff all the subsumptions in Γ follow from the hybrid TBox $(\mathcal{T},\mathcal{S})$ under the hybrid-semantics defined above, i.e.:

$$C_i \sqsubseteq_{gfp,\mathcal{T}\cup\mathcal{S}} D_i \text{ for all } C_i \sqsubseteq^? D_i \in \Gamma.$$

We say that Γ is *hybrid-unifiable* w.r.t. $\mathcal{T}$ if it has a *hybrid-unifier*.

The normalization procedure described in Section 2.2 provides significant simplifications that help to prove the existence of a local unifier, whenever an $\mathcal{EL}$-unification problem is unifiable w.r.t. a cycle-restricted TBox. In the next chapter we take advantage of the same normalization criterion to obtain a similar locality result for the hybrid case.

The following lemma shows that despite the differences between both problems, e.g., hybrid semantics versus descriptive semantics and a hybrid-unifier versus a usual unifier, unifiability of a hybrid $\mathcal{EL}$-unification problem is also not influenced by flattening the corresponding general TBox.

Lemma 4.2.2. *Let $\mathcal{T}$ be a ground general $\mathcal{EL}$-TBox, $\Gamma = \{C_1 \sqsubseteq^? D_1, \ldots, C_n \sqsubseteq^? D_n\}$ be a unification problem, and $\mathcal{T}'$ be the result of applying the normalization procedure described in Section 2.2 to $\mathcal{T}$. Then, Γ is hybrid-unifiable w.r.t. $\mathcal{T}$ iff it is hybrid-unifiable w.r.t. $\mathcal{T}'$.*

Proof. ($\Rightarrow$) Assume that Γ is hybrid-unifiable w.r.t. $\mathcal{T}$. Then, there exists a TBox $\mathcal{S}$ such that it is a hybrid-unifier of Γ w.r.t. $\mathcal{T}$, i.e.:

$$C_i \sqsubseteq_{gfp,\mathcal{T}\cup\mathcal{S}} D_i \text{ for all } C_i \sqsubseteq^? D_i \in \Gamma$$

Applying Proposition 4.1.5 we have that:

$$C_i \sqsubseteq_{gfp,\mathcal{T}'\cup\mathcal{S}} D_i \text{ for all } C_i \sqsubseteq^? D_i \in \Gamma$$

Thus, $\mathcal{S}$ is a hybrid-unifier of Γ w.r.t. $\mathcal{T}'$.

($\Leftarrow$) Assume that Γ is hybrid-unifiable w.r.t. $\mathcal{T}'$ and let $\mathcal{S}$ be a hybrid-unifier of Γ. In principle we cannot use $\mathcal{S}$ as a hybrid-unifier for Γ w.r.t. $\mathcal{T}$. It may contain concept names that were introduced during the normalization of $\mathcal{T}$ and that do not appear in $\mathcal{T}$.

However, looking at the rules of the normalization procedure one can see that, for each introduced concept name A there is a concept description C_A occurring in $\mathcal{T}$ such that $A \equiv_{\mathcal{T}} C_A$ holds. We obtain a new TBox $\mathcal{S}'$ from $\mathcal{S}$ as a result of the replacement of such concepts A ocurring in $\mathcal{S}$ by its corresponding concept descriptions C_A.

Obviously, $\mathcal{S}'$ is still a hybrid-unifier of Γ w.r.t. $\mathcal{T}'$ and it is defined over $sig(\mathcal{T})$. Then, similar as above, we apply Proposition 4.1.5 to obtain that $\mathcal{S}'$ is a hybrid-unifier of Γ w.r.t. $\mathcal{T}$. □

For the rest of this thesis we assume, without loss of generality, that every unification problem is flat and for every hybrid TBox $(\mathcal{T}, \mathcal{S})$, $\mathcal{T}$ is ground and flat.

4.3 Some Properties of Proof Trees

This section is dedicated to show some properties of proof trees in **HC**. They will be used as auxiliary propositions, later on, when proving the results in Chapter 5.

Proposition 4.3.1. *Let C, D be two concept descriptions such that C is ground and let $(\mathcal{T}, \mathcal{S})$ be a hybrid TBox such that $\mathcal{T}$ is ground. Then, for all $n \geq 0$ and any proof tree $\mathcal{P}$ for $C \sqsubseteq_n D$, 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 **HC** 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 (Ax), (Top) or (Start), we have that $\mathcal{P}$ is a one-element proof tree and the left-hand side gound 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 $\mathcal{T}$ 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 **HC**. We will use this particular sub-trees in the rest of this section.

Definition 4.3.2. Let $\mathcal{R} = \{R_1, \ldots, R_m\}$ be a subset of rules from **HC** and $\mathcal{P}$ a proof tree for the sequent $C \sqsubseteq_n D$. 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 {(Ax), (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 $\mathcal{R}_1 = \{(AndL1), (AndL2), (AndR)\}$ and $\mathcal{R}_2 = \{(AndL1), (AndL2), (AndR), (Ex), (GCI)\}$.

Proposition 4.3.3. *Let $\mathcal{P}$ be a proof tree for the sequent $C \sqsubseteq_n D$ and B a top-level atom of D. Consider the maximal sub-proof tree $\mathcal{P}_\mathcal{R}$ of $\mathcal{P}$ w.r.t. $\mathcal{R} = \{(AndL1), (AndL2), (AndR)\}$. 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 $\mathcal{P}_\mathcal{R}$. □

Proposition 4.3.4. *Let $(\mathcal{T}, \mathcal{S})$ be a hybrid TBox, $\mathcal{S}'$ 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 $C \sqsubseteq_n D$ w.r.t. $(\mathcal{T}, \mathcal{S})$.*
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 provable w.r.t. $(\mathcal{T}, \mathcal{S}')$ for some $k \geq 0$.*

then, there exists a proof tree $\mathcal{P}'$ for $C \sqsubseteq_k D$ w.r.t. $(\mathcal{T}, \mathcal{S}')$.

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 4.3.2, $\mathcal{P}_\mathcal{R}$ 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 provable w.r.t. $(\mathcal{T}, \mathcal{S}')$ for some k and thus, there exists a proof tree $\mathcal{P}'$ for $C \sqsubseteq_k D$.

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 $\mathcal{T}$ 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 4.1.6, 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 $F \sqsubseteq_n D$ w.r.t. $(\mathcal{T}, \mathcal{S})$.

   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 $F \sqsubseteq_k D$ w.r.t. $(\mathcal{T}, \mathcal{S}')$. Thus, a further application of the GCI rule yields a proof tree for $C \sqsubseteq_k D$ w.r.t. $(\mathcal{T}, \mathcal{S}')$.

□

4.4 Extendible Trees and Unambiguous Forests

In the last section of this chapter, we show some properties that will be used to obtain the results of Chapter 6.

Proposition 4.4.1. *Let $(\mathcal{T}, \mathcal{S})$ be a hybrid TBox and C, D be two concept descriptions such that $C \sqsubseteq_\infty D$ w.r.t. $(\mathcal{T}, \mathcal{S})$.*

Then, for all $n \geq 0$ there exists a proof tree $\mathcal{P}$ for $C \sqsubseteq_n D$ w.r.t. $(\mathcal{T}, \mathcal{S})$ such that it satisfies the following property:

$$\textit{for every sequent } E \sqsubseteq_q F \textit{ in } \mathcal{P} \textit{ we have } E \sqsubseteq_\infty F \textit{ w.r.t. } (\mathcal{T}, \mathcal{S}) \qquad (\mathcal{Z})$$

Proof. From Section 4.1 we know that there exists a value m that depends on $(\mathcal{T}, \mathcal{S})$, C and D 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 $C \sqsubseteq_{n+m} D$ w.r.t. $(\mathcal{T}, \mathcal{S})$ (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 provable in **HC** for all $l \geq 0$ w.r.t. $(\mathcal{T}, \mathcal{S})$. Therefore, $E \sqsubseteq_\infty F$ is valid w.r.t. $(\mathcal{T}, \mathcal{S})$.

Now, let $\mathcal{P}'$ be the largest subtree of $\mathcal{P}$ such that all its sequents are of the form $E \sqsubseteq_q F$ with $q \geq m$. It is not difficult to see that replacing every sequent $E \sqsubseteq_q F$ occurring in $\mathcal{P}'$ by $E \sqsubseteq_{q-m} F$, a proof tree for $C \sqsubseteq_n D$ w.r.t. $(\mathcal{T}, \mathcal{S})$ is obtained. This shows how to obtain a proof tree for $C \sqsubseteq_n D$ that satisfies property $\mathcal{Z}$. Thus, since n was arbitrarily selected the proposition holds for all $n \geq 0$. □

Now, we introduce a disambiguation criterion on proof trees.

Definition 4.4.2. A proof tree Q in **HC** 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 $\mathcal{Z}$ can be preserved under *disambiguation* of set of proof trees.

Proposition 4.4.3. *Let $(\mathcal{T}, \mathcal{S})$ be a hybrid TBox, $\mathcal{Q} = \{Q_1, \ldots, Q_n\}$ be any unambiguous set of proof trees and Q be a proof tree for a sequent $C \sqsubseteq_n D$, such that $Q_1, \ldots, Q_n$ and Q satisfy the Property $\mathcal{Z}$ from Proposition 4.4.1.*

Then, there exists a proof tree Q' for $C \sqsubseteq_n D$, 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 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 (Ax), (Top) or (Start). Then, Q is *unambiguous* because it is a one-element proof tree and it satisfies $\mathcal{Z}$ by assumption. 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 satisfies $\mathcal{Z}$ and the application of induction yields an unambiguous set $\mathcal{Q} \cup \{Q_1'\}$ satisfying $\mathcal{Z}$, where Q_1' is a 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 Q'. But such a sequent is the root of a proof tree Q_s for $C \sqsubseteq_n D$ in $\mathcal{Q} \cup \{Q'_1\}$ satisfying $\mathcal{Z}$. Therefore, Q_s represents the proof tree that we are looking for and thus, $\mathcal{Q}' = \mathcal{Q} \cup \{Q_s\}$.

- $C \sqsubseteq_n D$ 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 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 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 $\mathcal{Z}$. Using the same reasoning as before, but with respect to Q'_1 and Q'_2, a set of unambiguous proof trees $\mathcal{Q} \cup \{Q'_1, Q'_2, Q_s\}$ satisfying property $\mathcal{Z}$ exists, where Q_s is a proof tree for $C \sqsubseteq_n D$. Thus, since removing elements from such a set does not introduce any ambiguity, then $\mathcal{Q}' = \mathcal{Q} \cup \{Q_s\}$ is an unambiguous set of proof trees satisfying property $\mathcal{Z}$.

□

Notice that since the empty set trivially satisfies the premises of Proposition 4.4.3, then the following proposition is a particular case and follows immediately.

Proposition 4.4.4. *Let $(\mathcal{T}, \mathcal{S})$ be a hybrid TBox and Q be a proof tree for the sequent $C \sqsubseteq_n D$, such that Q satisfies the Property $\mathcal{Z}$.*

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

Chapter 5

A Brute-Force NP-Algorithm

As said before, deciding unifiability of $\mathcal{EL}$-unification problems w.r.t. cycle-restricted TBoxes is NP-Complete [13]. In this chapter, we extend this result to the case when unification is considered w.r.t. arbitrary general TBoxes under the hybrid-semantics defined in the previous chapter. More precisley, we show that hybrid $\mathcal{EL}$-unification is NP-Complete.

First, we generalize the notion of a local unifier to the case of hybrid unification, and we show that a similar locality result holds in this case. Based on this, we obtain that hybrid-unifiability is still in NP.

Second, we show the NP-hardness of the problem by a reduction from the $\mathcal{EL}$-matching problem modulo equivalence.

5.1 Local hybrid-unifiers

By Lemma 4.2.2, we can assume without loss of generality that the unification problem Γ and the general TBox $\mathcal{T}$ are flat. The sets At and At_{nv} are defined as in Section 3.3, with the addition that At also contains the atoms from $\mathcal{T}$.

From now on, a cyclic TBox may be a solution of a hybrid unification problem, the acyclic restriction for the assignment ζ required in Definition 3.3.1 is not longer needed. This, for example, allows the possibility to have non-variable atoms of the form $\exists r.X$ assigned to a variable X.

Definition 5.1.1. Let $\mathcal{T}$ be a flat general TBox and Γ a flat unification problem. A TBox $\mathcal{S}$ is a *local hybrid-unifier* of Γ w.r.t. $\mathcal{T}$ if:

1. $\mathcal{S}$ is a hybrid-unifier of Γ w.r.t. $\mathcal{T}$, and
2. There exists an assignment ζ of subsets of At_{nv} to each variable X in

N_v such that: $\mathcal{S}$ is the induced TBox from ζ in the sense of Definition 3.3.1.

Hence, if hybrid-unifiability is bounded to the existence of a *local hybrid-unifier*, as it is for unification w.r.t. cycle-restricted TBoxes and local unifiers, we would have decidability of hybrid unification within NP.

The following theorem represents one of the main results of this chapter, it directly implies that hybrid unification in $\mathcal{EL}$ w.r.t. general TBoxes is decidable within NP.

Theorem 5.1.2. *Let $\mathcal{T}$ be a ground flat general TBox and Γ be a flat unification problem. Then, Γ is hybrid-unifiable w.r.t. $\mathcal{T}$ iff Γ has a local hybrid-unifier w.r.t. $\mathcal{T}$.*

The proof of this theorem is presented in the next section.

5.2 Existence of a *local hybrid-unifier*

We prove Theorem 5.1.2 following the same idea used in [13, 14] for the cycle-restricted case, but using a different characterization for the corresponding subsumption problem. Since hybrid-unifiers may be cyclic TBoxes, we use the relation $\sqsubseteq_\infty$ defined in Chapter 4 as a characterization for subsumption w.r.t hybrid TBoxes.

Assume that $\mathcal{S}$ is a hybrid-unifier of Γ w.r.t. $\mathcal{T}$. We define the assignment $\zeta^{\mathcal{S}}$ as:

$$\zeta^{\mathcal{S}}_X = \{D \in At_{nv} \mid X \sqsubseteq_{gfp,\mathcal{T}\cup\mathcal{S}} D\}$$

Let $\mathcal{S}'$ be the induced TBox from $\zeta^{\mathcal{S}}$. Showing that $\mathcal{S}'$ is a hybrid-unifier of Γ w.r.t. $\mathcal{T}$ would imply the existence of a local hybrid-unifier and complete the proof for Theorem 5.1.2. The following lemma implies that $\mathcal{S}'$ is certainly a hybrid-unifier of Γ w.r.t. $\mathcal{T}$.

Lemma 5.2.1. *Let $\mathcal{T}$ be a flat general TBox, Γ be a flat unification problem and $C_1, \ldots, C_m, D \in At$. Let $\mathcal{S}$ and $\mathcal{S}'$ as defined above. Then:*

$$C_1 \sqcap \ldots \sqcap C_m \sqsubseteq_{gfp,\mathcal{T}\cup\mathcal{S}} D \text{ implies } C_1 \sqcap \ldots \sqcap C_m \sqsubseteq_{gfp,\mathcal{T}\cup\mathcal{S}'} D.$$

Proof. Assume that $C_1 \sqcap \ldots \sqcap C_m \sqsubseteq_{gfp,\mathcal{T}\cup\mathcal{S}} D$ holds. The application of Theorem 4.1.8 yields that $C_1 \sqcap \ldots \sqcap C_m \sqsubseteq_\infty D$ is provable in **HC** w.r.t. the hybrid TBox $(\mathcal{T}, \mathcal{S})$. We prove by induction on n the following claim:

If $C_1 \sqcap \ldots \sqcap C_m \sqsubseteq_\infty D$ *is provable in* **HC** *w.r.t.* $(\mathcal{T}, \mathcal{S})$ *then* $C_1 \sqcap \ldots \sqcap C_m \sqsubseteq_n D$ *is provable w.r.t.* $(\mathcal{T}, \mathcal{S}')$ *for all* $n \geq 0$.

Base Case: For $n = 0$ it is clear since we have the rule (Start) in **HC**.

Induction Step: We assume that the claim holds for $n-1$ and we show that it also holds for n.

By Definition 4.1.7 we know that there exists a proof tree for $C_1 \sqcap \ldots \sqcap C_m \sqsubseteq_n D$ in **HC** for all $n \geq 0$ w.r.t. $(\mathcal{T}, \mathcal{S})$. In particular, consider a proof tree $\mathcal{P}$ of $C_1 \sqcap \ldots \sqcap C_m \sqsubseteq_l D$ where l is selected *large enough*[1]. Furthermore, consider the set of rules $\mathcal{R} = \{(AndL1), (AndL2), (AndR), (Ex), (GCI)\}$ and the maximal sub-proof tree $\mathcal{P}_\mathcal{R}$ of $\mathcal{P}$ w.r.t. $\mathcal{R}$.

We show that for every sequent $E_1 \sqsubseteq_l E_2$ at a leaf in $\mathcal{P}_\mathcal{R}$, there is a proof tree for $E_1 \sqsubseteq_n E_2$ w.r.t. $(\mathcal{T}, \mathcal{S}')$. Two observations are in order w.r.t. the leaves in $\mathcal{P}_\mathcal{R}$:

- Since l is selected large enough, then $E_1 \sqsubseteq_\infty E_2$ is provable w.r.t. $(\mathcal{T}, \mathcal{S})$ and consequently $E_1 \sqsubseteq_{gfp, \mathcal{T} \cup \mathcal{S}} E_2$.
- By Definition 4.3.2, each sequent $E_1 \sqsubseteq_l E_2$ at a leaf in $\mathcal{P}_\mathcal{R}$ must be either an instance of one of the axiom rules (Ax, Top or Start), the consequence of an instance of the rule (DefR) or one that is the consequence of an instance of the rule (DefL).

Therefore, every such sequent has one of the following three forms:

1. $C \sqsubseteq_l C$, $C \sqsubseteq_l \top$ or $C \sqsubseteq_0 D$. By Definition 4.1.6, these three cases represent a one-element proof tree in **HC** w.r.t. any hybrid TBox and any value of $l \geq 0$.

2. $D \sqsubseteq_l X$, where D is a concept description and X a variable in N_v. Assume that $\zeta_X^\mathcal{S} = \{D_1, \ldots, D_q\}$. By definition of $\zeta_X^\mathcal{S}$, we have that $X \sqsubseteq_{gfp, \mathcal{T} \cup \mathcal{S}} D_i$ for all $i \in \{1, \ldots, q\}$. Then, since $D \sqsubseteq_{gfp, \mathcal{T} \cup \mathcal{S}} X$ as observed above, we also have that $D \sqsubseteq_{gfp, \mathcal{T} \cup \mathcal{S}} D_i$ for all $i \in \{1, \ldots, q\}$.

   Applying Theorem 4.1.8, once more, we obtain that $D \sqsubseteq_\infty D_i$ is provable w.r.t. $(\mathcal{T}, \mathcal{S})$. Since $D, D_1, \ldots, D_q$ are in At, then the application of induction hypothesis and Definition 4.1.7 yield that: $D \sqsubseteq_{n-1} D_i$ has a proof tree in **HC** w.r.t. $(\mathcal{T}, \mathcal{S}')$ for all $i \in \{1, \ldots, q\}$.

   Performing $(q-1)$ applications of rule (AndR), it is possible to obtain a proof tree for $D \sqsubseteq_{n-1} D_1 \sqcap \ldots \sqcap D_q$ w.r.t. $(\mathcal{T}, \mathcal{S}')$. Thus, since $D_1 \sqcap \ldots \sqcap D_q$ is the definition of X in $\mathcal{S}'$, the application of rule (DefR) yields a proof tree for $D \sqsubseteq_n X$ w.r.t. $(\mathcal{T}, \mathcal{S}')$.

[1] l can be selected, for example, as a value such that $\sqsubseteq_l = \sqsubseteq_{l+1}$. In Chapter 4 it is mentioned that such a value always exists.

3. $X \sqsubseteq_l D_1 \sqcap D_2$ or $X \sqsubseteq_l D_1$, where $D_1, D_2 \in At_{nv}$

   Since the choice of l guarantees that $X \sqsubseteq_\infty D_1 \sqcap D_2$, then by Theorem 4.1.8 we have that $X \sqsubseteq_{gfp,\mathcal{T}\cup\mathcal{S}} D_1 \sqcap D_2$ and consequently: $X \sqsubseteq_{gfp,\mathcal{T}\cup\mathcal{S}} D_1$ and $X \sqsubseteq_{gfp,\mathcal{T}\cup\mathcal{S}} D_2$.

   The definition of $\zeta_X^{\mathcal{S}}$ implies that D_1 and D_2 are in $\zeta_X^{\mathcal{S}}$, and hence $X \sqsubseteq_{gfp,\mathcal{T}\cup\mathcal{S}'} D_1 \sqcap D_2$, $X \sqsubseteq_\infty D_1 \sqcap D_2$ and $X \sqsubseteq_n D_1 \sqcap D_2$ are implied in that order w.r.t. $(\mathcal{T}, \mathcal{S}')$. Thus, there is a proof tree for $X \sqsubseteq_n D_1 \sqcap D_2$ ($X \sqsubseteq_n D_1$) w.r.t. $(\mathcal{T}, \mathcal{S}')$.

We have shown that for every sequent $E_1 \sqsubseteq_l E_2$, at a leave in $\mathcal{P}_\mathcal{R}$, there is a proof tree for $E_1 \sqsubseteq_n E_2$ w.r.t. $(\mathcal{T}, \mathcal{S}')$. Then, we can apply Proposition 4.3.4 to obtain that there exists a proof tree $\mathcal{P}'$ for $C_1 \sqcap \ldots \sqcap C_m \sqsubseteq_n D$ w.r.t. $(\mathcal{T}, \mathcal{S}')$ and therefore, the inductive claim is proved.

Thus, we have that $C_1 \sqcap \ldots \sqcap C_m \sqsubseteq_\infty D$ is provable in **HC** w.r.t $(\mathcal{T}, \mathcal{S}')$ and then we can conclude:

$$C_1 \sqcap \ldots \sqcap C_m \sqsubseteq_{gfp,\mathcal{T}\cup\mathcal{S}'} D$$

□

Theorem 5.1.2 follows immediately from the previous lemma. Since the number of elements in At_{nv} is polynomial on the size of Γ and $\mathcal{T}$, one can guess an assignment ζ in polynomial time. To check whether the induced TBox $\mathcal{S}$ is a hybrid-unifier of Γ w.r.t. $\mathcal{T}$, the polynomial time algorithm provided in [7, 8] can be used to check whether $C \sqsubseteq_{gfp,\mathcal{T}\cup\mathcal{S}} D$ holds for all subsumptions $C \sqsubseteq^? D$ in Γ. Thus, we obtain that hybrid unification in $\mathcal{EL}$ is decidable within NP.

Corollary 5.2.2. *Hybrid unification in $\mathcal{EL}$ w.r.t. general TBoxes is in NP.*

5.3 NP-hardness of hybrid $\mathcal{EL}$-unification

In this section, we reduce the $\mathcal{EL}$-matching problem modulo equivalence to the hybrid unification problem in $\mathcal{EL}$.

Definition 5.3.1. An $\mathcal{EL}$-matching problem modulo equivalence is of the form $C \equiv^? D$ where C and D are $\mathcal{EL}$-concept descriptions, C is ground and D is not ground. A solution or matcher of this problem is an $\mathcal{EL}$-substitution σ such that $C \equiv \sigma(D)$. A matching problem is solvable if it has a solution.

We define a translation from a given matching problem to a unification problem in the following way. Let $C \equiv^? D$ be a matching problem the corresponding hybrid $\mathcal{EL}$-unification problem consists of the set of subsumptions

$\Gamma = \{C \sqsubseteq^? D, D \sqsubseteq^? C\}$ and it is considered w.r.t. the empty TBox, i.e., $\mathcal{T} = \emptyset$.

In the following, we prove that a matching problem has a solution *if* and *only if* the corresponding hybrid unification problem has a solution. First, we prove two auxiliary propositions.

Proposition 5.3.2. *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 $(\emptyset, \mathcal{S})$: 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, Proposition 4.3.1 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 $(\emptyset, \mathcal{S})$ 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 Proposition 4.3.3 (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 (Ax) 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 4.3.2 $G \sqsubseteq_n E$ 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 $\mathcal{P}$. □

Before going into the next proposition, one assumption can be made w.r.t. cyclic TBoxes. Let $\mathcal{S}$ by a cyclic TBox, then applying Proposition 2.3.5 it can be assumed without loss of generality that $\mathcal{S}$ does not contain *component cycles*. Thus, we can assume that the definition of any cyclic-defined variable (Definition 2.2.4) X in $\mathcal{S}$ contains a top-level atom of the form $\exists s.E$, such that E contains an occurrence of a cyclic-defined variable in $\mathcal{S}$.

Proposition 5.3.3. *Let C and D be two concept descriptions, $\mathcal{S}$ 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.*

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

Proof. The proof is by induction on the role depth r of C and we consider an arbitrary proof tree $\mathcal{P}$.

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 $\mathcal{S}$ and B a top level atom of D where X occurs. By Proposition 5.3.2, 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 Proposition 5.3.2, then by Proposition 4.3.3 (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 $\mathcal{S}$.

   Since X is cyclic-defined in $\mathcal{S}$ 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 $\mathcal{S}$. A second application of Proposition 5.3.2 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. □

An easy consequence of the previous proposition is that a ground concept description C cannot be subsumed by a non-ground concept description D w.r.t. a hybrid TBox $(\emptyset, \mathcal{S})$ if, S is cyclic and D contains an occurrence of a cyclic-defined variable in $\mathcal{S}$.

Lemma 5.3.4. *Let C and D be two concept descriptions such that C is ground. Then, there is no cyclic TBox $\mathcal{S}$ such that $C \sqsubseteq_{gfp,\emptyset\cup\mathcal{S}} D$ holds and D contains an occurrence of a cyclic-defined variable in $\mathcal{S}$.*

Proof. The proof is by contradiction. Assume that there exists a cyclic TBox $\mathcal{S}$ such that $C \sqsubseteq_{gfp,\emptyset\cup\mathcal{S}} D$ holds and D contains an occurrence of a cyclic-defined variable in $\mathcal{S}$. Then for all $n \geq 0$ there exists a proof tree for $C \sqsubseteq_n D$ w.r.t. $(\emptyset, \mathcal{S})$.

Let r be the role depth of C and $\mathcal{P}$ be a proof tree for $C \sqsubseteq_{r+2} D$ w.r.t. $(\emptyset, \mathcal{S})$. Applying Proposition 5.3.3 there is a sequent at a node in $\mathcal{P}$ of the form $A \sqsubseteq_l \exists s.E$ where A is a primitive concept name and $l > 0$.

It is not difficult to see that since $l > 0$, there is no immediate rule that could be used to derive $A \sqsubseteq_l \exists s.E$. Hence, we have a contradiction with the fact that $\mathcal{P}$ is a proof tree for $C \sqsubseteq_{r+2} D$ since $A \sqsubseteq_l \exists s.E$ is a sequent at a node in $\mathcal{P}$, but it cannot be proved.

This tell us that $C \sqsubseteq_{r+2} D$ cannot be proved w.r.t. $(\emptyset, \mathcal{S})$, contradicting the initial assumption. □

This lemma says, in addition, that any unification problem which contains a subsumption $C \sqsubseteq^? D$ of the form previously discussed, does not have a cyclic hybrid-unifier w.r.t. the empty general TBox. Based on this we show the equivalence between a matching problem and its corresponding hybrid unification problem.

Lemma 5.3.5. *An $\mathcal{EL}$ matching problem $C \equiv^? D$ has a solution iff the hybrid $\mathcal{EL}$-unification problem $\Gamma = \{C \sqsubseteq^? D, D \sqsubseteq^? C\}$ has a hybrid unifier w.r.t. the empty general TBox.*

Proof. ($\Rightarrow$) Assume that $C \equiv^? D$ has a solution. Then, there exists a ground substitution σ such that $C \equiv \sigma(D)$. Since σ is ground then, it induces an acyclic TBox $\mathcal{S}$ such that $C \equiv_{\mathcal{S}} D$ and consequently $C \sqsubseteq_{\mathcal{S}} D$ and $D \sqsubseteq_{\mathcal{S}} C$ hold, see Section 3.2.

The acyclicity of $\mathcal{S}$ and the application of Lemma 2.3.4 yield that greatest fixpoint semantics coincides with descriptive semantics. Hence, it can be seen that $C \sqsubseteq_{gfp,\emptyset\cup\mathcal{S}} D$ and $D \sqsubseteq_{gfp,\emptyset\cup\mathcal{S}} C$ hold. Thus, $\mathcal{S}$ is a hybrid-unifier of Γ w.r.t. the empty TBox.

($\Leftarrow$) Assume that Γ has a hybrid-unifier $\mathcal{S}$ w.r.t. the empty TBox, then $C \sqsubseteq_{gfp,\emptyset\cup\mathcal{S}} D$ holds. First, we can see that $\mathcal{S}$ is an acyclic TBox:

1. If D does not contain any occurrence of a cyclic-defined variable in $\mathcal{S}$, then cyclic-definitions in $\mathcal{S}$ (if there exists any) can be removed to obtain an acyclic TBox that is still a hybrid-unifier of Γ w.r.t. the empty TBox.

2. Otherwise, since C is ground, the application of Lemma 5.3.4 yields that $\mathcal{S}$ has to be an acyclic TBox.

Second, the acyclicity of $\mathcal{S}$ implies the equivalence between greatest fixpoint semantics and descriptive semantics. Therefore, $C \sqsubseteq_{\mathcal{S}} D$ and $D \sqsubseteq_{\mathcal{S}} C$ hold. In addition as it was shown in Section 3.2, $\mathcal{S}$ induces a ground substitution $\sigma_{\mathcal{S}}$ such that $\sigma_{\mathcal{S}}(C) \sqsubseteq \sigma_{\mathcal{S}}(D)$ and $\sigma_{\mathcal{S}}(D) \sqsubseteq \sigma_{\mathcal{S}}(C)$. Thus, $C \equiv \sigma_{\mathcal{S}}(D)$ holds and thus, $\sigma_{\mathcal{S}}$ is a matcher for $C \equiv^? D$. □

Since the $\mathcal{EL}$-matching problem is NP-hard [17], this lemma yields the following theorem.

Theorem 5.3.6. *The problem of deciding hybrid-unifiability in $\mathcal{EL}$ w.r.t. general TBoxes is NP-hard.*

Chapter 6

A Goal Oriented Unification Algorithm

The result obtained in the previous chapter immediately yields an algorithm. However, this algorithm is not practical since it blindly guesses a local assignment and only afterwards checks whether the induced TBox is a hybrid-unifier of the given unification problem.

In this chapter, we introduce a more goal-oriented unification algorithm in which nondeterministic decisions are only made if they are induced by "unsolved parts" of the unification problem. In addition, failure due to wrong guesses can be detected early. Whenever the algorithm runs successfully, it is the case that the computed assignment induces a hybrid-unifier and there is no need to verify it.

This algorithm is designed following similar ideas as the corresponding goal-oriented algorithm proposed in [13] for unification with a cycle-restricted TBox. However, the rules look quite different since now we use the proof calculus **HC** as a characterization of subsumption under hybrid-semantics. We introduce the notion of *p-sequent* as the elements that the algorithm work with and need to introduce a *blocking* procedure to avoid infinite runs of the algorithm.

We will first present an overview of the existent algorithm for the cycle-restricted case. Then, we will show how to transform this algorithm in order to obtain a new algorithm to solve the hybrid unification problem. At the end, we will show that the new algorithm is a correct NP-decision procedure for the hybrid unification problem.

6.1 Unification with a Cycle-Restricted TBox

The goal oriented unification algorithm provided in [13] to solve $\mathcal{EL}$-unification with a cycle-restricted TBox relies on an appropriate characterization of the subsumption problem in $\mathcal{EL}$ w.r.t. general TBoxes, and on the existence of a *local* unifier.

It starts with a set Γ_0 of subsumptions and mantains a current unification problem Γ and a current assignment S, which initially assigns the empty set to all variables. Initially, all subsumptions in Γ_0 are marked as *unsolved* except those with a variable on the right-hand side. Rules are applied only to unsolved subsumptions and a non-failing application of a rule of this algorithm does the following:

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

Whenever a rule application extends S, it expands Γ w.r.t. X in the following sense: every subsumption $s \in \Gamma$ of the form $C_1 \sqcap \ldots \sqcap C_n \sqsubseteq^? X$ is *expanded* by adding the subsumption $C_1 \sqcap \ldots \sqcap C_n \sqsubseteq^? A$ to Γ for every $A \in S_X$.

In general, subsumptions are only added to Γ by a rule application or by expansion if they have not been already added. Every time a new subsumption is added to Γ, it is initially marked as *unsolved*, except if it has a variable on the right-hand side. A subsumption will never be removed from Γ, or become unsolved once it has been marked as *solved*.

The rules of the algorithm consists of three *eager* rules (see Figure 6.1) and several nondeterministic rules (see Figure 6.2). Eager rules are used with the purpose of optimization, they are applied with higher priority than nondeterministic rules and among them, *Eager Ground Solving* has the highest priority, then comes *Eager Solving*, and then *Eager Extension*.

The nondeterministic rules are only used if no eager rule can be applied. In particular there are four *mutation* rules, they allow the algorithm to solve subsumptions by using the consequences implied from the GCIs in $\mathcal{T}$. We only show the rule *Mutation 1* here, but the other *mutation* rules (see [13]) follow the same idea.

As a remark, is worth to mention that this algorithm tries to compute ground substitutions that are *local* unifiers of a given unification problem. In fact, every time the assignment S becomes cyclic by an application of the *(Eager) Extension* rule, the rule application fails.

Eager Ground Solving:

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

Eager Solving:

Condition: This rule applies to $\mathfrak{s} = C_1 \sqcap \ldots \sqcap C_n \sqsubseteq^? D$ if either.
• there is an index $i \in \{1, \ldots, n\}$ such that $C_i = D$ or $C_i = X \in N_v$ and $D \in S_X$, or
• D is ground and $\prod \mathcal{G} \sqsubseteq_{\mathcal{T}} D$ holds, where $\mathcal{G}$ is the set of all ground atoms in $\{C_1, \ldots, C_n\} \cup \bigcup_{X \in \{C_1, \ldots, C_n\} \cap N_v} S_X$
Action: The application marks $\mathfrak{s}$ as *solved*.

Eager Extension:

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

Figure 6.1: The *eager* rules of Algorithm 6.1.1 w.r.t. a cycle-restricted TBox $\mathcal{T}$.

Finally, the unification algorithm w.r.t. cycle-restricted TBoxes (Algorithm 27 in [13]) works as follows:

Algorithm 6.1.1. Let Γ_0 be a flat $\mathcal{EL}$-unification problem. We initialize $\Gamma := \Gamma_0$ and $S_X := \emptyset$ for all variables $X \in N_v$. While Γ contains an unsolved subsumption do the following:

1. **Eager rule application:** If some eager rules apply to an unsolved subsumption $\mathfrak{s}$ in Γ, apply one of highest priority. If the rule application fails, then return "not unifiable".

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

Once all subsumptions in Γ are solved, return the substitution σ that is induced by the current assignment.

The choice of which unsolved subsumption to consider next by nondeterministic rules, is don't care nondeterministic. In contrast, choosing which rule to apply to a chosen subsumption is don't know nondeterministic.

It was shown in [13, 15] that Algorithm 6.1.1 is sound and complete for unification w.r.t. cycle-restricted TBoxes and it is an NP-decision procedure.

Decomposition:

Condition: This rule applies to $\mathfrak{s} = C_1 \sqcap \ldots \sqcap C_n \sqsubseteq^? \exists s.D'$ if there is at least one $i \in \{1, \ldots, n\}$ with $C_i = \exists s.C'$.

Action: Its application chooses such an index i, adds the subsumption $C' \sqsubseteq_n^? D'$ to Γ, expands it w.r.t. D' if D' is a variable, and marks $\mathfrak{s}$ as *solved*.

Extension:

Condition: This rule applies to $\mathfrak{s} = C_1 \sqcap \ldots \sqcap C_n \sqsubseteq^? D$ if there is at least one $i \in \{1, \ldots, n\}$ with $C_i \in N_v$.

Action: Its application chooses such an index i and adds D to S_{C_i}. If this makes S cyclic, the rule application fails. Otherwise, Γ is expanded w.r.t. C_i and $\mathfrak{s}$ is marked as *solved*.

Mutation 1:

Condition: This rule applies to $\mathfrak{s} = C_1 \sqcap \ldots \sqcap C_n \sqsubseteq^? D$ if $n > 1$ and there are atoms $A_1, \ldots, A_k, B$ of $\mathcal{T}$ such that $A_1 \sqcap \ldots \sqcap A_k \sqsubseteq_{\mathcal{T}} B$ holds.

Action: Its application chooses such atoms, marks $\mathfrak{s}$ as *solved*, and generates the following subsumptions:

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

Figure 6.2: The nondeterministic rules of Algorithm 6.1.1 w.r.t. a cycle-restricted TBox $\mathcal{T}$.

We will now use the same idea of this algorithm to obtain a goal oriented decision procedure for hybrid unification.

6.2 Hybrid Unification.

The Algorithm 6.1.1 is sound for hybrid-unifiability. Whenever it returns a substitution σ, the induced acyclic TBox $\mathcal{S}_\sigma$ is a hybrid-unifier of Γ w.r.t. $\mathcal{T}$, since descriptive semantics coincides with gfp-semantics in the presence of acyclic TBoxes. However, it is not complete since there are not cycle-restricted TBoxes for which Γ is hybrid-unifiable, but they yield a failure of the algorithm because the rules do not allow to build cyclic assignments.

To fix this problem, we transform the previous algorithm into a new algorithm that solves the hybrid unification problem. Given $\mathcal{T}$ and a unification problem $\Gamma = \{C_1 \sqsubseteq^? D_1, \ldots, C_m \sqsubseteq^? D_m\}$, our new algorithm will try to compute a *local* TBox $\mathcal{S}$ such that $C_i \sqsubseteq_\infty D_i$ holds in **HC** w.r.t. the hybrid TBox $(\mathcal{T}, \mathcal{S})$, for all the subsumptions in Γ.

The intuitive idea underlying our new algorithm is first to select a value l large enough and then, to try to build proof trees for each sequent $C_i \sqsubseteq_l D_i$ corresponding to a subsumption in Γ while adding the necessary non-variable atoms to the variable definitions in $\mathcal{S}$, such that those proof trees can indeed

be computed w.r.t. $(\mathcal{T}, \mathcal{S})$. The existence of such proof tree and the selection of the value l guarantee that $C_i \sqsubseteq_\infty D_i$ holds.

To this purpose, we do a *bottom-up* search that starts with all those sequents as the goals that have to be proved. In a certain way, the algorithm non-deterministically applies one of the **HC** rules backwards to extend a proof tree, whereas the premises of the selected rule are considered as new goals to be proved. In addition, non-variable atoms are added to the variable definitions in $\mathcal{S}$ whenever they are needed for a (DefL) rule application. If at some point, all the sequents that remain to be proved are instances of the rules (Ax), (Top) or (Start) then, every initial goal have a proof tree w.r.t. $\mathcal{T}$ and the resulting TBox $\mathcal{S}$.

It was pointed out in [8] that the termination of such a procedure, even knowing the TBox $\mathcal{S}$ in advance, is not in principle guaranteed mainly because of the GCI rule. Though it was conjectured that a blocking mechanism could solve the problem, such approach was nevertheless not chosen and instead a *top-down* approach was proposed. However, we cannot use a *top-down* search since we are not trying to find all the possible sequents that have a proof tree w.r.t. a hybrid TBox $(\mathcal{T}, \mathcal{S})$, but to compute a TBox $\mathcal{S}$ to obtain proof trees for a given set of sequents.

Therefore, we need to use a blocking mechanism. This mechanism will work on *p-sequents* which are defined in the following way:

Definition 6.2.1. A *p-sequent* is a pair $(C \sqsubseteq_n D, P)$ where P is a set of sequents.

As a remark, it is important to notice that a sequent $C \sqsubseteq_n D$ is not only identified by the concept descriptions C and D, but also by the value n in $\sqsubseteq_n$. Next, we define the procedure *blocking*:

Definition 6.2.2. Given a set of *p-sequents* Γ_p and a *p-sequent* $(C \sqsubseteq_n D, P)$. The procedure *blocking* consists of the following three rules applied in the given order:

B1: If there is a sequent $C \sqsubseteq_n D$ in P, then *blocking* fails.

B2: If there is a *p-sequent* of the form $(C \sqsubseteq_n D, P')$ in Γ_p, then do the following:

- For each *p-sequent* $(\_, P'')$ in Γ_p such that $C \sqsubseteq_n D$ is in P'', make $P'' := P'' \cup P$,
- Make $P' := P' \cup P$.

B3: If none of the previous rules are applicable, then add $(C \sqsubseteq_n^? D, P)$ to Γ_p.

Using these definitions, the Algorithm 6.1.1 is modified to work on *p-sequents* instead of subsumptions. The unification problem $\Gamma = \{C_1 \sqsubseteq^? D_1, \ldots, C_m \sqsubseteq^? D_m\}$ is transformed into an initial set of *p-sequents* Γ_{p_0} that are of the form $(C_i \sqsubseteq_l^? D_i, \emptyset)$. Like before, starting with Γ_{p_0}, the algorithm maintains a current set of goals Γ_p and a current assignment ζ.

Marking a *p-sequent* $(\mathfrak{s}, P)$ as *solved* does not always mean that there is a proof tree for $\mathfrak{s}$ w.r.t. $\mathcal{T}$ and the TBox $\mathcal{S}$ induced by the current assignment. It may be the case that the task of finding a proof tree for $\mathfrak{s}$, was deferred to solving other *p-sequents* such that, a proof tree for $\mathfrak{s}$ can be built from the proof trees of the subsumptions in those *p-sequents*.

In a very general sense, this algorithm behaves similar to Algorithm 6.1.1, but several changes are introduced. These modifications are motivated by the fact, that now we work with *p-sequents* and the calculus **HC** is used as the characterization to obtain a TBox $\mathcal{S}$ that solves our problem.

First, the notion of expansion is modified in the following way: every *p-sequent* $(\mathfrak{s}, P) \in \Gamma_p$ with $\mathfrak{s}$ of the form $C_1 \sqcap \ldots \sqcap C_m \sqsubseteq_n^? X$ is *expanded* by adding the *p-sequent* $(C_1 \sqcap \ldots \sqcap C_m \sqsubseteq_{n-1}^? D, \emptyset)$ to Γ_p for every $D \in \zeta_X$. This is motivated by *Lemma* 3.1.2 from [7], which says that $C_1 \sqcap \ldots \sqcap C_m \sqsubseteq_n X$ has a proof tree iff $C_1 \sqcap \ldots \sqcap C_m \sqsubseteq_{n-1} D_1 \sqcap \ldots \sqcap D_q$ has a proof tree, where $D_1 \sqcap \ldots \sqcap D_q$ is the definition of X.

Second, our algorithm consists of four *eager* rules (see Figure 6.3), two of them modified rules from Algorithm 6.1.1 and the rest are new rules. For example, the meaning of the *Eager Ground Solving* rule is kept as in Algorithm 6.1.1, because finding a proof tree for a ground sequent only depends on the given general TBox $\mathcal{T}$. The *Eager Axiom Solving* rule is introduced to solve the trivial cases (Top) and (Start) from **HC**, while *Eager Solving* covers a possible application of rule (Ax). Last, the new *Eager AndR* rule represents an application of the rule (AndR) from **HC**. The introduction of this rule as deterministic is based on *Lemma* 3.1.2 from [7], which also shows that $C_1 \sqcap \ldots \sqcap C_m \sqsubseteq_n D_1 \sqcap \ldots \sqcap D_q$ has a proof tree iff $C_1 \sqcap \ldots \sqcap C_m \sqsubseteq_n D_i$ has a proof tree for all $i, 1 \leq i \leq q$. The priority among the *eager* rules is given by their order in Figure 6.3. Note that, although the initial set of *p-sequents* contains only sequents with a single flat atom on their right-hand side, an application of the new *Mutation* rule may introduce a new *p-sequent* representing a sequent that has a conjunction of flat atoms as its right-hand side. That is why the *Eager AndR* rule is considered for $q > 1$ and at the same time has the highest priority among all *eager* rules.

Third, the nondeterministic rules (see Figure 6.4) are defined to simulate applications of rules from **HC**. Rules *Extension* and *Decomposition* can be seen as applications of rules (DefL) and (Ex) in **HC** provided that rule (AndLi) has been previously applied, as needed. The new *Mutation* rule is the equivalent to the four *Mutation* rules from Algorithm 6.1.1, but it

corresponds to an application of the (GCI) rule from **HC**.

Finally, one of the most significant differences is the introduction of the blocking mechanism to avoid nonterminating runs of the algorithm. It is based on the following observation: Assume that a sequent $C \sqsubseteq_n D$ has a proof tree $\mathcal{P}$ w.r.t. some hybrid TBox $(\mathcal{T}, \mathcal{S})$. In addition, suppose that there is a node in $\mathcal{P}$ that is also labeled with the sequent $C \sqsubseteq_n D$. Obviously, such a node also have a proof tree $\mathcal{P}'$ which is smaller than $\mathcal{P}$, but proves the same sequent. Therefore, we can replace $\mathcal{P}$ by $\mathcal{P}'$. The rule **B2** in *blocking* is also based on this idea and represents an optimization for the algorithm. It is important to see that the update steps described in rule **B2** are needed, otherwise the procedure may be incorrect as shown in the following example.

Example 6.2.3. Consider the general TBox $\mathcal{T} = \{D \sqsubseteq B, B \sqsubseteq D, B \sqcap D \sqsubseteq E\}$ and the unification problem

$$\Gamma = \{X \sqsubseteq^? E, X \sqsubseteq^? A, A \sqsubseteq^? X\}$$

It can be seen that Γ is not hybrid-unifiable w.r.t. $\mathcal{T}$. If that were the case, then $X \equiv A$ has to be the definition of X in $\mathcal{S}$, but then $\mathcal{S}$ does not solve $X \sqsubseteq E$ for the given $\mathcal{T}$.

Now, suppose that the update steps described in *blocking* for rule **B2** are not performed. The following is a non-failing sequence of rules applications on the set Γ_p obtained from Γ:

- Apply *Mutation* rule to the *p-sequent* $p_1 = (\mathfrak{s}_1 = X \sqsubseteq_l^? E, \emptyset)$ using the GCI $B \sqcap D \sqsubseteq E$. Then, two new *p-sequents* are added to Γ_p of the form $p_2 = (\mathfrak{s}_2 = X \sqsubseteq_l^? B \sqcap D, \{\mathfrak{s}_1\})$ and $p_3 = (\mathfrak{s}_3 = E \sqsubseteq_l^? E, \{\mathfrak{s}_1\})$.
- p_3 is trivially solved and *Eager AndR* rule is applied to p_2 to obtain the *p-sequents* $p_{21} = (X \sqsubseteq_l^? B, \{\mathfrak{s}_1, \mathfrak{s}_2\})$ and $p_{22} = (X \sqsubseteq_l^? D, \{\mathfrak{s}_1, \mathfrak{s}_2\})$.
- Next, a new application of *Mutation* rule to p_{21} using the GCI $D \sqsubseteq B$ will yield the sequent $X \sqsubseteq_l D$, however since there is already a *p-sequent* in Γ_p representing $X \sqsubseteq_l D$, no new *p-sequent* is added to Γ_p and p_{21} is marked as *solved*.
- Similar as before, apply *Mutation* rule to p_{22} using the GCI $B \sqsubseteq D$ to obtain the sequent $X \sqsubseteq_l B$. Same reasoning implies that no new *p-sequent* is added to Γ_p and p_{22} is marked as *solved*. One can see that without removing the update steps from rule **B2**, *blocking* would have failed in this case.

After this series of rules applications the only remaining *unsolved p-sequent* in Γ_p is $(X \sqsubseteq_l^? A, \emptyset)$ and applying the rule *Extension*, a non-failing run of the

Eager AndR:

Condition: This rule applies to $(\mathfrak{s}, P)$ with $\mathfrak{s} = C_1 \sqcap \ldots \sqcap C_m \sqsubseteq_n^? D_1 \sqcap \ldots \sqcap D_q$, if $q > 1$.
Action: For each D_i, executes *blocking* w.r.t. $(C_1 \sqcap \ldots \sqcap C_m \sqsubseteq_n D_i, P \cup \{\mathfrak{s}\})$. If *blocking* fails for some D_i, then the rule application fails. Otherwise, $(\mathfrak{s}, P)$ is marked as *solved*.

Eager Axiom Solving:

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

Eager Ground Solving:

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

Eager Solving:

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

Figure 6.3: The *eager* rules of hybrid unification.

algorithm can be easily completed. However, the TBox $\{X \equiv A\}$ induced by the final assignment is not a unifier of Γ w.r.t. $\mathcal{T}$.

Now, consider the TBox $\mathcal{S}_0$ consisting of a single concept definition of the form $A_0 \equiv \bigsqcap_{D \in At_{nv}} D$. The reason of defining $\mathcal{S}_0$, is that the number of sub-descriptions of the definition of A_0 in $\mathcal{S}_0$ is an *upper bound* for the number of sub-descriptions of any concept definition $A \equiv C$ in a TBox $\mathcal{S}$ computed by the algorithm.

We define the hybrid unification algorithm in the following way,

Algorithm 6.2.4. Given a flat general TBox $\mathcal{T}$ and a flat unification problem $\Gamma = \{C_1 \sqsubseteq^? D_1, \ldots, C_m \sqsubseteq^? D_m\}$. First, do the following initializations:

1. Compute the value[1] of l as:

$$l := (|Sub(\mathcal{T})| + |Sub(\Gamma)| + |Vars(\Gamma)| * |Sub(\mathcal{S}_0)|)^2$$

2. Initialize the set of *p-sequents* $\Gamma_p := \Gamma_{p_0}$, where Γ_{p_0} is obtained from Γ as:

[1]This value represents the number of all possible subsumptions built from the maximal possible number of sub-descriptions from the known general TBox $\mathcal{T}$, the unification problem Γ and a (cyclic) TBox $\mathcal{S}$ which is to be constructed. Note that the maximal size of $\mathcal{S}$ may be $|Vars(\Gamma)| \times |Sub(\mathcal{S}_0)|$.

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 w.r.t. $(C' \sqsubseteq_n D', P \cup \{\mathfrak{s}\})$.

Action: Its application chooses such an index i and applies *blocking* w.r.t. $(C' \sqsubseteq_n D', P \cup \{\mathfrak{s}\})$. Once *blocking* is applied, it expands Γ w.r.t. D' if D' is a variable, 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 $C_i \in N_v$.

Action: Its application chooses such an index i and adds D to ζ_{C_i}. Γ 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 $E \sqsubseteq F$ in $\mathcal{T}$ and indexes i, j with $1 \leq i \leq j \leq m$, such that *blocking* does not fail w.r.t. $(C_i \sqcap \ldots \sqcap C_j \sqsubseteq_n E, P \cup \{\mathfrak{s}\})$ and $(F \sqsubseteq_n D, P \cup \{\mathfrak{s}\})$.

Action: Its application chooses such a GCI $E \sqsubseteq F$ and indexes i, j. It applies *blocking* w.r.t. $(C_i \sqcap \ldots \sqcap C_j \sqsubseteq_n E, P \cup \{\mathfrak{s}\})$ and $(F \sqsubseteq_n D, P \cup \{\mathfrak{s}\})$. Once *blocking* is applied, $(\mathfrak{s}, P)$ is marked as *solved*.

Figure 6.4: The nondeterministic rules of hybrid unification.

$$\Gamma_{p_0} := \{(C_1 \sqsubseteq_l^? D_1, \emptyset), \ldots, (C_m \sqsubseteq_l^? D_m, \emptyset)\}$$

3. Initialize the assignment ζ with $\zeta_X := \emptyset$ for all variables $X \in N_v$.

Next, while Γ_p contains an unsolved *p-sequent* do the following:

1. **Eager rule application:** If some eager rules apply to an unsolved *p-sequent* in Γ_p, apply one of highest priority. If the rule application fails, then return "failure".

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

Once all *p-sequents* in Γ_p are solved, return the TBox $\mathcal{S}$ that is induced by the current assignment ζ.

6.3 Soundness

In this section, we show that if the Algorithm 6.2.4 returns a TBox $\mathcal{S}$ given as its input the unification problem Γ, then $C_i \sqsubseteq_\infty D_i$ holds w.r.t. $(\mathcal{T}, \mathcal{S})$

for each $C_i \sqsubseteq^? D_i \in \Gamma$ and thus, the computed TBox $\mathcal{S}$ is a hybrid-unifier of Γ w.r.t. $\mathcal{T}$. Assume that Algorithm 6.2.4 has a non-failing run on input Γ, and let ζ be the final assignment computed by this run and $\mathcal{S}$ 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 $C \sqsubseteq_n D$ w.r.t. $(\mathcal{T}, \mathcal{S})$. To show that, we use well founded-induction [22] on the following well-founded $\succ$ order on $\widehat{\Gamma}$.

Definition 6.3.1. 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 $\sqsubseteq_n$.
  - $m_2(p) :=|P|$, i.e., the number of predecessors of 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 [22], then $\succ$ is a well-founded strict partial order on $\widehat{\Gamma}$.

Lemma 6.3.2. *If Algorithm 6.2.4 outputs a TBox $\mathcal{S}$, then for each p-sequent $(C \sqsubseteq_n^? D, P)$ in $\widehat{\Gamma}$ there is a proof tree for $C \sqsubseteq_n D$ w.r.t. $(\mathcal{T}, \mathcal{S})$.*

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 $\mathfrak{s}'$ w.r.t. $(\mathcal{T}, \mathcal{S})$. 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 (Ax) 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 $C_1 \sqcap \ldots \sqcap C_m \sqsubseteq_{gfp,\mathcal{T}\cup\mathcal{S}} D$ holds for any TBox $\mathcal{S}$. Application of Theorem 4.1.8 yields that there is a proof tree for $C_1 \sqcap \ldots \sqcap C_m \sqsubseteq_n D$ w.r.t. $(\mathcal{T}, \mathcal{S})$ 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 $\mathfrak{s}$ w.r.t. $(\mathcal{T}, \mathcal{S})$. 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:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{D \sqsubseteq_n D}\,(\text{Ax})}{\vdots}\,(\text{AndLi})}{S_X \sqsubseteq_n D}\,(\text{AndLi})}{X \sqsubseteq_n D}\,(\text{DefL})}{\vdots}\,(\text{AndLi})}{C_1 \sqcap \ldots \sqcap C_m \sqsubseteq_n D}\,(\text{AndLi})$$

– Eager AndR: $\mathfrak{s}$ is of the form $C_1 \sqcap \ldots \sqcap C_m \sqsubseteq_n^? D_1 \sqcap \ldots \sqcap D_q$ with $q > 1$. Since *blocking* did not fail for any D_i then, there is a *p-sequent* $p_i = (C_1 \sqcap \ldots \sqcap C_m \sqsubseteq_n^? D_i, P_i)$ in $\widehat{\Gamma}$ for each D_i such that $P \cup \{\mathfrak{s}\} \subseteq P_i$. Hence, we have that $m_1(p) = m_1(p_i)$ and $m_2(p) < m_2(p_i)$, therefore $p \succ p_i$ for all $i, 1 \leq i \leq q$.

The application of induction yields that there is a proof tree Q_i for $C_1 \sqcap \ldots \sqcap C_m \sqsubseteq_n D_i$ for each D_i. Thus, several instances of the (AndR) rule in **HC** provides a proof tree for $C_1 \sqcap \ldots \sqcap C_m \sqsubseteq_n D_1 \sqcap \ldots \sqcap D_q$ w.r.t. $(\mathcal{T}, \mathcal{S})$ as it is illustrated in the following diagram

$$\dfrac{\dfrac{\dfrac{\dfrac{Q_1}{C \sqsubseteq_n D_1} \quad \dfrac{Q_2}{C \sqsubseteq_n D_2}\,(\text{AndR})}{C \sqsubseteq_n D_1 \sqcap D_2}}{\vdots} \quad \dfrac{Q_{q-1}}{C \sqsubseteq_n D_{q-1}}\,(\text{AndR})}{C \sqsubseteq_n D_1 \sqcap \ldots \sqcap D_{q-1}} \quad \dfrac{Q_q}{C \sqsubseteq_n D_q}\,(\text{AndR})}{C \sqsubseteq_n D_1 \sqcap \ldots \sqcap D_q}$$

where $C = C_1 \sqcap \ldots \sqcap C_m$.

– Extension: $\mathfrak{s}$ is of the form $X \sqcap C_1 \sqcap \ldots \sqcap C_m \sqsubseteq_n^? D$ for a variable X and the definition of X in $\mathcal{S}$ is of the form $D \sqcap D_1 \sqcap \ldots \sqcap D_q$. Then, starting with $D \sqsubseteq_n D$ and q applications of (AndLi) rules, we can obtain a proof tree for $D \sqcap D_1 \sqcap \ldots \sqcap D_q \sqsubseteq_n D$. 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 $\mathfrak{s}$ w.r.t. $(\mathcal{T}, \mathcal{S})$ (see the following diagram).

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

– 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 *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 $C' \sqsubseteq_n D'$ w.r.t. $(\mathcal{T}, \mathcal{S})$. 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 $\mathfrak{s}$ w.r.t. $(\mathcal{T}, \mathcal{S})$.

– 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 $E \sqsubseteq F$ in $\mathcal{T}$ and indexes i, j such that, the *p-sequents* $p_1 = (C_i \sqcap \ldots \sqcap C_j \sqsubseteq_n^? E, P_1)$ and $p_2 = (F \sqsubseteq_n^? D, P_2)$ are in $\widehat{\Gamma}$ where $P \cup \{\mathfrak{s}\} \subseteq P_1$ and $P \cup \{\mathfrak{s}\} \subseteq P_2$. Then, we have that $m_1(p) = m_1(p_1) = m_1(p_2)$, $m_2(p) < m_2(p_1)$ and $m_2(p) < m_2(p_2)$, therefore $p \succ p_1$ and $p \succ p_2$.

  We can apply induction hypothesis to obtain that there are proof trees Q_E and Q_F for $C_i \sqcap \ldots \sqcap C_j \sqsubseteq_n E$ and $F \sqsubseteq_n D$. Thus, an application of the (GCI) rule following several applications of (AndLi) rules, yields a proof tree for $C_1 \sqcap \ldots \sqcap C_m \sqsubseteq_n D$ w.r.t. $(\mathcal{T}, \mathcal{S})$ (see diagram below).

$$\dfrac{\dfrac{\dfrac{\dfrac{Q_E}{C_i \sqcap \ldots \sqcap C_j \sqsubseteq_n E} \quad \dfrac{Q_F}{F \sqsubseteq_n D}}{C_i \sqcap \ldots \sqcap C_j \sqsubseteq_n D}\,(\text{GCI})}{\vdots}\,(\text{AndLi})}{C_1 \sqcap \ldots \sqcap C_m \sqsubseteq_n D}\,(\text{AndLi})$$

- 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 $\mathcal{S}$. 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 $\mathcal{S}$ is of the form $D_1 \sqcap \ldots \sqcap D_q$ and for every D_i there is a *p-sequent* of the form $p_i = (C_1 \sqcap \ldots \sqcap C_m \sqsubseteq^?_{n-1} D_i, \emptyset)$ in $\widehat{\Gamma}$. Clearly, because of $\sqsubseteq_n$ and $\sqsubseteq_{n-1}$, $m_1(p) > m_1(p_i)$ and therefore, $m(p) \succ m(p_i)$ for all $i, 1 \leq i \leq q$.

Now, we apply induction to obtain that there is a proof tree for each sequent $C_1 \sqcap \ldots \sqcap C_m \sqsubseteq_{n-1} D_i$. Similar as for the *Eager AndR* rule, 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

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

□

Since the initialization of Algorithm 6.2.4 adds a *p-sequent* of the form $(C_i \sqsubseteq^?_l D_i, \emptyset)$ to Γ_{p_0} for each subsumption $C_i \sqsubseteq^? D_i$ in Γ, and we have that $\Gamma_{p_0} \subseteq \widehat{\Gamma}$ then, the following lemma can be obtained as a consequence of Lemma 6.3.2 within the proper selection of the value for l.

Lemma 6.3.3. *Let $\mathcal{T}$ 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 6.2.4 outputs a TBox $\mathcal{S}$ on input Γ, then $\mathcal{S}$ is a hybrid-unifier of Γ w.r.t. $\mathcal{T}$.*

Proof. Since Γ_{p_0} is contained in $\widehat{\Gamma}$, then the application of Lemma 6.3.2 yields that $C_i \sqsubseteq_l D_i$ has a proof tree w.r.t. $(\mathcal{T}, \mathcal{S})$ for each subsumption $C_i \sqsubseteq^? D_i$ in Γ.

We extend the TBox $\mathcal{S}$ in the following way: For each $C_i \sqsubseteq^? D_i$ in Γ two new concept definitions $A_i \equiv C_i$ and $B_i \equiv D_i$ are added into $\mathcal{S}$. The new TBox is denoted as $\mathcal{S}_\Gamma$.

One can see, that $C_i \sqsubseteq_k D_i$ has a proof tree w.r.t. $(\mathcal{T}, \mathcal{S})$ iff $A_i \sqsubseteq_{k+1} B_i$ has a proof tree w.r.t. $(\mathcal{T}, \mathcal{S}_\Gamma)$, for all $k \geq 0$. In particular, there is proof tree for each sequent of the form $A_i \sqsubseteq_{l+1} B_i$ w.r.t. $(\mathcal{T}, \mathcal{S}_\Gamma)$.

Now, consider the sequence of relations $\sqsubseteq_0, \sqsubseteq_1, \ldots, \sqsubseteq_l, \sqsubseteq_{l+1}$ w.r.t. $(\mathcal{T}, \mathcal{S}_\Gamma)$. Based on Section 4.1, the following two observations can be made:

1. $\sqsubseteq_0 \supseteq \sqsubseteq_1 \supseteq \ldots \supseteq \sqsubseteq_l \supseteq \sqsubseteq_{l+1}$.
2. Those relations are defined over sub-descriptions of the concept descriptions occurring in $(\mathcal{T}, \mathcal{S}_\Gamma)$.

Therefore, the number of elements in $\sqsubseteq_0$ is $|Sub((\mathcal{T}, \mathcal{S}_\Gamma))|^2$. Moreover, it can be seen that the selection of the value of l guarantees that $| \sqsubseteq_0 | \leq l$ and this clearly implies that $\sqsubseteq_l = \sqsubseteq_{l+1} = \sqsubseteq_{l+2} \ldots$ w.r.t. $(\mathcal{T}, \mathcal{S}_\Gamma)$. As a consequence we obtain that $C_i \sqsubseteq_n D_i$ holds in $(\mathcal{T}, \mathcal{S})$ for all $n \geq 0$.

Thus, $C_i \sqsubseteq_\infty D_i$ holds w.r.t. $(\mathcal{T}, \mathcal{S})$ for all $C_i \sqsubseteq^? D_i \in \Gamma$ and, $\mathcal{S}$ is a hybrid-unifier of Γ w.r.t. $\mathcal{T}$. □

6.4 Completeness

Assume that Γ is hybrid-unifiable w.r.t. $\mathcal{T}$ and let $\mathcal{S}$ be a hybrid-unifier of Γ w.r.t. $\mathcal{T}$. Like in [13], we can use this unifier to guide the application of the nondeterministic rules such that Algorithm 6.2.4 does not fail. More precisley, we use a certain set of proof trees for the subsumptions in Γ w.r.t. $(\mathcal{T}, \mathcal{S})$ to guide a non-failing run of the algorithm.

To that purpose, we use the definitions and the properties shown in Section 4.4. Since $\mathcal{S}$ is a hybrid-unifier of Γ w.r.t. $\mathcal{T}$ we know that $C_i \sqsubseteq_\infty D_i$ holds for each subsumption $C_i \sqsubseteq^? D_i \in \Gamma$. Then, by Proposition 4.4.1 we can assume without loss of generality that there is a set of proof trees $\mathcal{Q} = \{Q_1, \ldots, Q_m\}$ such that Q_i is a proof tree for $C_i \sqsubseteq_l D_i$[2] that satisfies the Property $\mathcal{Z}$.

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 4.4.2.

By Propositions 4.4.4 and 4.4.3, we can assume that there is an *unambiguos* set of proof trees $\mathcal{Q}_0 = \{Q_1, \ldots, Q_m\}$ such that: Q_i is a proof tree for $C_i \sqsubseteq_l D_i$, for each subsumption $C_i \sqsubseteq^? D_i \in \Gamma$, that satisfies Property $\mathcal{Z}$. Now, we are ready to show how to guide a non-failing run of Algorithm 6.2.4.

The following invariants for the current set of *p-sequents* Γ_p and the current assignment ζ will be maintained:

(i) There is an *unambiguos* set of proof trees $\mathcal{Q} = \{Q_1, \ldots, Q_n\}$ w.r.t. $(\mathcal{T}, \mathcal{S})$ such that:

- each $Q_i \in \mathcal{Q}$ satisfies the Property $\mathcal{Z}$.
- 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 $X \sqsubseteq_\infty D$ w.r.t. $(\mathcal{T}, \mathcal{S})$.

[2] l is the value computed during the initialization of Algorithm 6.2.4.

(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 $X \in N_v$, $\mathcal{Q}_0$ exists as described above and Γ_p is initialized to Γ_{p_0}, these invariants are satisfied after the initialization of the algorithm.

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

Lemma 6.4.1. *The invariants are mantained by the operation of expanding* Γ_p*.*

Proof. First, since expansion does not change the current assignment ζ, invariant (ii) is trivially maintained. In addition, since the new *p-sequent* added to Γ_p is of the form $(\_, \emptyset)$ then, invariant (iii) is also mantained. We show that invariant (i) is also satisfied.

Consider a *p-sequent* $(\mathfrak{s}, \_)$ where $\mathfrak{s}$ is of the form $C_1 \sqcap \ldots \sqcap C_m \sqsubseteq_n^? X$ for which a new *p-sequent* $p' = (C_1 \sqcap \ldots \sqcap C_m \sqsubseteq_{n-1}^? D, \emptyset)$ is created with $D \in \zeta_X$. By the invariants, we have $X \sqsubseteq_\infty D$ w.r.t. $(\mathcal{T}, \mathcal{S})$ and $\mathfrak{s}$ occurs in some Q_i from a set $\mathcal{Q}$ of proof trees satisfying $\mathcal{Z}$. Hence, $C_1 \sqcap \ldots \sqcap C_m \sqsubseteq_\infty X$ w.r.t. $(\mathcal{T}, \mathcal{S})$ and transitivity of $\sqsubseteq_\infty$ (see Section 4.1) yields $C_1 \sqcap \ldots \sqcap C_m \sqsubseteq_\infty D$.

The application of Lemma 4.4.1 yields that there exists a proof tree $\mathcal{P}$ for $C_1 \sqcap \ldots \sqcap C_m \sqsubseteq_{n-1}^? D$ w.r.t. $(\mathcal{T}, \mathcal{S})$ that satisfies Property $\mathcal{Z}$. Thus, application of Proposition 4.4.3 implies that invariant (i) is mantained. □

There are two *eager* rules that could produce failures, the following lemma shows that this will never be the case.

Lemma 6.4.2. *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 Axiom*, 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* $p = (\mathfrak{s}, \_)$ with $\mathfrak{s}$ ground. By invariant (i), $\mathfrak{s}$ occurs on a proof tree Q_i that satisfies the Property $\mathcal{Z}$ and thus, it is valid w.r.t. $\sqsubseteq_\infty$. Applying Theorem 4.1.8 we obtain that $\mathfrak{s}$ also holds w.r.t. $\sqsubseteq_{gfp,\mathcal{T}\cup\mathcal{S}}$, and this implies that the rule application does not fail. The invariants are mantained since neither Γ_p nor ζ are modified.

Finally, consider the case that the *Eager AndR* rule is applied to an unsolved *p-sequent* of the form $(\mathfrak{s} = C_1 \sqcap \ldots \sqcap C_m \sqsubseteq_n^? D_1 \sqcap \ldots \sqcap D_q, P)$ with $q > 1$. By the invariants we have that $\mathfrak{s}$ occurs in a proof tree $Q_i \in \mathcal{Q}$. In addition,

we can assume that $C_1 \sqcap \ldots \sqcap C_m \sqsubseteq_n D_j$ occurs in Q_i for all $j, 1 \leq j \leq q$. Then, adding a *p-sequent* of the form $(C_1 \sqcap \ldots \sqcap C_m \sqsubseteq_n D_j, P \cup \{\mathfrak{s}\})$ for all j to Γ_p will maintain invariant (i).

Now let us see why *blocking* does not fail. Assume that some sequent $\mathfrak{s}_j = C_1 \sqcap \ldots \sqcap C_m \sqsubseteq_n D_j \in P$. By invariant (iii), we have that there is a proof tree $Q_j \in \mathcal{Q}$ such that $\mathfrak{s}_j$ occurs in Q_j and $\mathfrak{s}$ occurs in the subtree rooted at $\mathfrak{s}_j$. In addition, $\mathfrak{s}_j$ occurs in the subtree rooted at $\mathfrak{s}$ in Q_i. Since Q_i and Q_j are mutually *unambiguos* then, the subtrees rooted at $\mathfrak{s}$ in Q_i and Q_j must be identical. This implies that the subtree rooted at $\mathfrak{s}_j$ in Q_j contains an additional occurrence of $\mathfrak{s}_j$ and this, obviously contradicts the assumption that Q_j is *unambiguos*. Thus, $C_1 \sqcap \ldots \sqcap C_m \sqsubseteq_n D_j$ cannot be in P and *blocking* does not fail.

Finally, we have to show that invariant (iii) is maintained afterwards. Consider the *p-sequent* $(\mathfrak{s}_j, P \cup \{\mathfrak{s}\})$ for any $j, 1 \leq j \leq q$. Since *blocking* does not fail, two cases are possible:

- Rule **B3** was applied while doing *blocking*. Then, $(\mathfrak{s}_j, P \cup \{\mathfrak{s}\})$ 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, $\mathfrak{s}_j$ occurs in the subtree rooted at $\mathfrak{s}$ in Q_i and thus, the *unambiguos* condition of $\mathcal{Q}$ yields that $\mathfrak{s}_j$ 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 $(\mathfrak{s}_j, P')$ in Γ_p. From the previous case we know that $\mathfrak{s}_j$ occurs in any proof tree for $E \sqsubseteq_n F \in P \cup \{\mathfrak{s}\}$. Therefore, updating P' as $P' \cup P \cup \{\mathfrak{s}\}$ does not violate invariant (iii). In addition, for each *p-sequent* $(\mathfrak{s}'', P'')$ in Γ_p such that $\mathfrak{s}_j \in P''$ we have that $\mathfrak{s}''$ occurs in any proof tree for $\mathfrak{s}_j$. Hence, updating P'' as $P'' \cup P \cup \{\mathfrak{s}\}$ maintains invariant (iii).

Thus, we conclude that the application of *blocking* maintains invariant (iii). Since ζ is not changed then, invariant (ii) is mantained and *Eager AndR* can be sucessfully applied while satisfying the invariants. □

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 6.4.3. *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 $C_1, \ldots, C_m$ are flat atoms in At, $D \in At_{nv}$ and $n > 0$. Note that, D must consist of a

single top-level atom, otherwise the application of rule *Eager AndR* would have either failed or succeed.

By invariant (i), there exists a set of *unambiguos* proof trees $\mathcal{Q}$ such that $\mathfrak{s}$ occurs in some $Q_i \in \mathcal{Q}$ and Q_i 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** 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 (Ax), (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 variable X. Since Q_i satisfies Property $\mathcal{Z}$ and $X \sqsubseteq_n D$ occurs in Q_i then, $X \sqsubseteq_\infty D$ holds w.r.t. $(\mathcal{T}, \mathcal{S})$ and hence, adding D to ζ_X does not violate invariant (ii). 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. Thus, the rule *Decomposition* can be successfully applied provided that *blocking* does not fail for p.
- The rule (GCI) is used. Then, there exists a GCI $E \sqsubseteq F \in \mathcal{T}$ such that $C_i \sqcap \ldots \sqcap C_j \sqsubseteq_n E$ and $F \sqsubseteq_n D$ are sequents occurring in Q_i. Similar as above, the addition of the *p-sequents* $p_1 = (C_i \sqcap \ldots \sqcap C_j \sqsubseteq^?_n E, P \cup \{\mathfrak{s}\})$ and $p_2 = (F \sqsubseteq^?_n D, P \cup \{\mathfrak{s}\})$ to Γ_p will maintain invariant (i) and rule *Mutation* can be successfully applied provided that *blocking* does not fail for p_1 and p_2.

A similar reasoning as for the *Eager AndR* rule can be used to show that *blocking* does not fail and invariant (iii) is maintained when (Ex) or (GCI) are used. □

These lemmas imply that for any hybrid-unifiable input problem Γ there is a non-failing run of Algorithm 6.2.4 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.

6.5 Termination and Complexity

Consider a run of Algorithm 6.2.4. We will show that any *p-sequent* ($C \sqsubseteq_n D, P$) encountered during this run satisfies the following two conditions:

1. C and D are *sub-descriptions* of concept descriptions in $\Gamma \cup \mathcal{T}$
2. $n \leq l$, where l is the value computed during the initialization of the algorithm.

Lemma 6.5.1. *If the p-sequents in* Γ_p *satisfy conditions* 1 *and* 2*, then these conditions are satisfied after one rule application.*

Proof. First, since Γ_p is initialized as Γ_{p0} 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 $D \in At_{nv}$. Since $n \leq l$, condition 2 is satisfied. In addition, the set At_{nv} only contains non-variable atoms from $\Gamma \cup \mathcal{T}$, therefore condition 1 is satisfied as well.
- A *p-sequent* is created by a rule application. These are the rules *Eager AndR*, *Decomposition* and *Mutation*. One can see that these rules are applied to a *p-sequent* $(C \sqsubseteq^?_n D, P) \in \Gamma_p$ by creating new *p-sequents* of the form $(C' \sqsubseteq^?_n D', P')$, where C' and D' are either sub-descriptions of C and D or sub-descriptions of concept descriptions occurring in $\mathcal{T}$ (see *Mutation* rule). Then, conditions are clearly satisfied after a rule application.

All the other rules from Algorithm 6.2.4 maintain conditions 1 and 2 since Γ_p is not modified. □

Based on this lemma, we conclude with the following.

Lemma 6.5.2. *Every run of the Algorithm 6.2.4 on input* Γ *terminates in time polynomial in the size of* $\Gamma \cup \mathcal{T}$*.*

Proof. There are only polynomially many *p-sequents* satisfying conditions 1 and 2. Let us see why:

- for each *p-sequent* $(C \sqsubseteq^?_n D, P)$, condition 1 is satisfied and $Sub(\Gamma \cup \mathcal{T})$ is of size polynomial on the size of $\Gamma \cup \mathcal{T}$. Hence, there are at most polynomially many pairs (C, D).

- for each *p-sequent* $(C \sqsubseteq_n^? D, P)$, condition 2 is satisfied. Since, $n \leq l$ and the value of l is polynomial on the size of $\Gamma \cup \mathcal{T}$, then there are polynomially many sequents of the form $C \sqsubseteq_n D$.
- By the application of *blocking*, it can be seen that there are no two *p-sequents* in Γ_p of the form $(C \sqsubseteq_n^? D, P_1)$ and $(C \sqsubseteq_n^? D, P_2)$ with $P_1 \neq P_2$.

Every rule application solves one *p-sequent*, then by Lemma 6.5.1 the algorithm can apply at most polynomially many rules. In addition, 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 $\Gamma \cup \mathcal{T}$.
- 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 $\mathcal{T}$.
- 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 $\Gamma \cup \mathcal{T}$, then rule **B1** executes in time polynomial in the size of $\Gamma \cup \mathcal{T}$.

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

- Guessing polynomially many atoms from the left-hand side of $\mathfrak{s}$ for a *p-sequent* $(\mathfrak{s}, P) \in \Gamma_p$.
- 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. □

This lemma within the soundness and completeness shown before (6.3.3, 6.4.1, 6.4.2 and 6.4.3), yield the main result of this chapter.

Theorem 6.5.3. *The Algorithm 6.2.4 is an NP-decision procedure for hybrid-unifiability in $\mathcal{EL}$.*

Chapter 7

Conclusions

In this thesis, we have considered the *hybrid unification* problem in the Description Logic $\mathcal{EL}$. The main objective was to investigate whether it is possible to extend the unification problem in $\mathcal{EL}$ by allowing (cyclic) TBoxes as candidate solutions, while the problem remains in NP.

To that purpose, first, we studied the available results for the $\mathcal{EL}$-unification problem [11, 13]. Second, the notion of *hybrid* TBoxes [6] was selected as a formalism to represent the combination of a general TBox and a (cyclic) TBox. Hybrid TBoxes are interpreted under the hybrid-semantics and, in particular we used a Gentzen-style calculus proposed in [7] as a characterization of the subsumption problem in $\mathcal{EL}$ w.r.t. hybrid TBoxes.

We proved that the hybrid unification problem in $\mathcal{EL}$ is NP-Complete. On the one hand, to show that the problem is in NP, we have used a similar notion of locality as in [13], but without restricting the occurrence of cyclic definitions. Based on this we proposed a pure *brute-force* NP-procedure that guesses a local TBox and verifies whether it is a hybrid-unifier of a given unification problem. On the other hand, we have shown that *hybrid unification* in $\mathcal{EL}$ is NP-hard by reducing the $\mathcal{EL}$-matching problem modulo equivalence to *hybrid unification* in $\mathcal{EL}$.

The initial *brute force* NP-procedure is not practical since it blindly guesses a local assignment and only afterwards checks whether the induced TBox is a hybrid-unifier. To improve this algorithm, we have proposed a more goal-oriented unification algorithm in Chapter 6. The Algorithm 6.2.4 is designed following the ideas of the goal-oriented algorithm for unification with a cycle-restricted TBox proposed in [13], but using the proof calculus **HC** proposed in [7] as a characterization of subsumption w.r.t. hybrid TBoxes. In addition, we have shown that our algorithm is a correct NP-decision procedure for the hybrid unification problem.

As future work, a practical implementation of the goal-oriented algorithm

should be considered. Such implementation would require several optimizations, specially w.r.t. the *blocking* procedure. An interesting question is to see if the algorithm really needs to keep track of two *p-sequents* of the form $(C \sqsubseteq_i D, \_)$ and $(C \sqsubseteq_j D, \_)$ with $i \neq j$. If that were not the case, it may represent a significant improvement, though probably the *blocking* procedure has to be modified in a suitable way. If on the other hand, this was not the case, then it would be interesting to see an example which proves that the conjecture is false.

Furthermore, the NP-complete result shown for hybrid unification can be seen as a motivation to find a polytime reduction of the hybrid unification problem to a propositional satisfiability problem (SAT). Finding such a reduction could represent a direction for future work that will allow the use of the strength of the modern SAT solvers.

Finally we hope that the algorithm will find practical applications e.g. in analyzing big knowledge databases and in dealing with redundancies in the biomedical ontologies.

Bibliography

[1] M. Hofmman. Proof-theoretic approach to description-logic. In Proc. LICS'05, IEEE Press, 2005.

[2] F. Baader, R. Küsters, and R. Molitor. Computing Least Common Subsumers in Description Logics with Existential Restrictions. In T. Dean, editor, Proceedings of the 16th International Joint Conference on Artificial Intelligence (IJCAI'99), pages 96-101. Morgan Kaufmann, 1999.

[3] F. Baader. Terminological cycles in a description logic with existential restrictions. LTCS-Report LTCS-02-02, Chair for Automata Theory, Institute for Theoretical Computer Science, Dresden University of Technology, Germany, 2002. See http://lat.inf.tu-dresden.de/research/reports.html.

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

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

[6] S.Brandt and J.Model. Subsumption in $\mathcal{EL}$ w.r.t. hybrid TBoxes. In Proc. KI'05, Springer LNAI 3698, 2005.

[7] N. Novakovic. Proof-theoretic approach to subsumption and least common subsumer in $\mathcal{EL}$ w.r.t. hybrid TBoxes. Master thesis, Institute for Theoretical Computer Science, TU Dresden, Germany, 2007.

[8] F. Baader, N. Novakovic, B. Boontawee. A Proof-Theoretic Subsumption Reasoner for Hybrid $\mathcal{EL}$ TBoxes. Proceedings of the 2008 International Workshop on Description Logics (DL2008), volume 353 of CEUR-WS, 2008.

[9] F. Baader, W. Snyder. Unification theory. In Robinson, J., and Voronkov, A., eds., Handbook of Automated Reasoning, volume I. Elsevier Science Publishers. 477-533, 2001.

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

[11] F. Baader and B. Morawska. Unification in the Description Logic $\mathcal{EL}$. In Ralf Treinen, editor, Proceedings of the 20th International Conference on Rewriting Techniques and Applications (RTA 2009), volume 5595 of Lecture Notes in Computer Science, pages 350-364. Springer-Verlag.

[12] F. Baader and B. Morawska. Unification in the Description Logic $\mathcal{EL}$. Logical Methods in Computer Science, 6(3), 2010. Special Issue of the 20th International Conference on Rewriting Techniques and Applications (RTA'09).

[13] F. Baader, S. Borgwardt, and B. Morawska. Unification in the Description Logic 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.

[14] F. Baader, S. Borgwardt, B. Morawska. Extending Unification in $\mathcal{EL}$ towards General Tboxes. Extending Unification in EL Towards General TBoxes. In Gerhard Brewka, Thomas Eiter, and Sheila A. McIlraith, editors, Proceedings of the Thirteenth International Conference on Principles of Knowledge Representation and Reasoning (KR'12), pages 568-572. AAAI Press, 2012.

[15] F Baader, S. Borgwardt, B. Morawska. A Goal-Oriented Algorithm for Unification in EL w.r.t. Cycle-Restricted TBoxes. In Proceedings of the 25th International Workshop on Description Logics (DL'12), CEUR Workshop Proceedings, Rome, Italy, 2012.

[16] Carsten Lutz and Frank Wolter. Deciding inseparability and conservative extensions in the description logic EL. Journal of Symbolic Computation, 45(2):194-228, 2010.

[17] R. Küsters. Non-standard Inferences in Description Logics, Springer LNAI 2100, 2001.

[18] 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.

[19] A. Tarski. Lattice-theoretic fixpoint theorem and its applications. Pacific Journal of Mathematics, (5):285-309, 1955.

[20] B. Nebel. Terminological reasoning is inherently intractable. Artificial Intelligence, 43:235-249, 1990

[21] B. A. Davey and H. A. Priestley. Introduction to Lattices and Order. Cambridge University Press, United Kingdom, 2002.

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

[23] Franz Baader, Diego Calvanese, Deborah McGuinness, Deniele Nardi, and Peter F. Patel-Schneider, editors. The Description Logic Handbook: Theory, Implementation, and Applications. Cambridge University Press, 2003.