Matching Concept Descriptions with Existential Restrictions Revisited

Franz Baader and Ralf Küsters
LuFg Theoretical Computer Science, RWTH Aachen
email: {baader,kuesters}@informatik.rwth-aachen.de

LTCS-Report 99-13

An abridged version of this technical report has been submitted to KR 2000.

Abstract

Matching of concepts against patterns is a new inference task in Description Logics, which was originally motivated by applications of the Classic system. Consequently, the work on this problem was until now mostly concerned with sublanguages of the Classic language, which does not allow for existential restrictions.

Motivated by an application in chemical process engineering, which requires a description language with existential restrictions, this paper investigates the matching problem in Description Logics with existential restrictions. It turns out that existential restrictions make matching more complex in two respects. First, whereas matching in sublanguages of Classic is polynomial, deciding the existence of matchers is an NP-complete problem in the presence of existential restrictions. Second, whereas in sublanguages of Classic solvable matching problems have a unique least matcher, this is not the case for languages with existential restrictions. Thus, it is not a priori clear which of the (possibly infinitely many) matchers should be returned by a matching algorithm.

After determining the complexity of the decision problem, the present paper first investigates the question of what are "interesting" sets of matchers, and then describes algorithms for computing these sets for the languages $\mathcal{EL}$ (which allows for conjunction and existential restrictions) and $\mathcal{ALE}$ (which additionally allows for value restrictions, primitive negation, and the bottom concept).

Contents

1 Introduction

Matching concepts against patterns is a relatively new inference problem in Description Logics (DLs), which has originally been introduced in [10, 14] to help filter out the unimportant aspects of large concepts appearing in knowledge bases of the CLASSIC DL system [11, 8]. More recently, matching (as well as the more general problem of unification) has been proposed as a tool for detecting redundancies in knowledge bases [5] and to support the integration of knowledge bases by prompting possible interschema assertions to the integrator [9].

All three applications have in common that one wants to search the knowledge base for concepts having a certain (not completely specified) form. This "form" can be expressed with the help of so-called *concept patterns*, i.e., concept descriptions containing variables (which stand for descriptions). For example, assume that we want to find concepts that are concerned with individuals having a son and a daughter sharing some characteristic. This can be expressed by the pattern $D := \exists \mathsf{has\text{-}child}.(\mathsf{Male} \sqcap X) \sqcap \exists \mathsf{has\text{-}child}.(\mathsf{Female} \sqcap X)$, where X is a variable standing for the common characteristic. The concept description $C := \exists \mathsf{has\text{-}child}.(\mathsf{Tall} \sqcap \mathsf{Male}) \sqcap \exists \mathsf{has\text{-}child}.(\mathsf{Tall} \sqcap \mathsf{Female})$ matches this pattern in the sense that, if we replace the variable X by the description Tall, the pattern becomes *equivalent* to the description. Thus, the substitution $\sigma := \{X \mapsto \mathsf{Tall}\}$ is a *matcher modulo equivalence* of the matching problem $C \equiv^? D$. Note that not only the fact that there is a matcher is of interest, but also the matcher itself, since it tells us what is the common characteristic of the son and the daughter.

Looking for such an exact match (called *matching modulo equivalence* in the following) is not always appropriate, though. In our example, using matching modulo equivalence means that all the additional characteristics of the son and daughter mentioned in the concept must be common to both. Thus, the description $C' := \exists \mathsf{has\text{-}child}.(\mathsf{Tall} \sqcap \mathsf{Male} \sqcap \mathsf{Talkative}) \sqcap \exists \mathsf{has\text{-}child}.(\mathsf{Tall} \sqcap \mathsf{Female} \sqcap \mathsf{Quiet})$ does not match the pattern modulo equivalence. *Matching modulo subsumption* only requires that, after the replacement, the pattern subsumes the description. Thus, the substitution σ from above is a *matcher modulo subsumption* of the matching problem $C' \sqsubseteq^? D$.

Previous results on matching in DLs were mostly concerned with sublanguages of the CLASSIC description language, which does not allow for existential restrictions of the kind used in our example. A polynomial-time algorithm for computing matchers modulo subsumption for a rather expressive DL was introduced in [10]. The main drawback of this algorithm is that it requires the concept patterns to be in structural normal form, and thus it cannot handle arbitrary matching problems. In addition, the algorithm is incomplete, i.e., it does not always find a matcher, even if one exists. For the DL $\mathcal{ALN}$, a polynomial-time algorithm for matching modulo subsumption and equivalence was presented in [1]. This algorithm is complete and it applies to arbitrary patterns.

The main purpose of this work is to investigate matching in DLs allowing for existential restrictions. We will show that existential restrictions make matching more complex in two respects. First, whereas matching in the DLs considered in [10, 1] is polynomial, even deciding the existence of matchers is an NP-complete problem in the presence of existential restrictions. Second, the algorithms described in [10, 1] always compute the least matcher (w.r.t. subsumption of substitutions; see the definition of $\sqsubseteq_s$) of the given matching problem. For languages with existential restrictions, such a unique least matcher need not exist. However, the set of minimal matchers is finite (though possibly exponential in the size of the matching problem), and we will show how to compute this set. It has turned out, however, that the minimal matchers are not necessarily the most interesting ones since they may contain certain redundancies. Thus, one also needs a kind of post-processing step that removes these redundancies. Since giving an answer to the question of what are good sets of matchers is not trivial, we will treat it separately.

This paper is structued as follows: In the two subsequent sections, we introduce the basic notions. In Section 3 we first exploit the problem of matching for the small language $\mathcal{EL}$ (which only allows for concept conjunction and existential restrictions) in order to present the main ideas underlying our results and to avoid the rather involved technical details necessary for the extensions of $\mathcal{EL}$. In particular, we investigate the problem of deciding the solvability of $\mathcal{EL}$-matching problem. We then characterize the sets of "interesting" matchers, which, as already mentioned, becomes in important issue and subsequently show how to compute these sets. Finally, the definitions and results are extended to the more expressive languages $\mathcal{FLE}$ and $\mathcal{ALE}$.

2 Preliminaries

In this section, we introduce the basic notions used in this work and state some fundamental properties for matching in $\mathcal{ALE}$ which have already been presented in [1] for the language $\mathcal{ALN}$.

The Language $\mathcal{ALE}$ and sublanguages

Concept descriptions are inductively defined with the help of a set of *constructors*, starting with a set N_C of *concept names* and a set N_R of *role names.* The constructors determine the expressive power of the DL. In this work, we consider concept descriptions built from the constructors shown in Table 1. In the description logic $\mathcal{EL}$, concept descriptions are formed using the constructors top-concept ($\top$), conjunction ($C \sqcap D$) and existential restriction ($\exists r.C$); $\mathcal{FLE}$ extends $\mathcal{EL}$ by value restrictions ($\forall r.C$); the description logic $\mathcal{ALE}$ allows for all the constructors shown in Table 1. In the following, we refer to concept descriptions in the languages $\mathcal{EL}$, $\mathcal{FLE}$, and $\mathcal{ALE}$ by $\mathcal{EL}$-, $\mathcal{FLE}$-, and $\mathcal{ALE}$-concept descriptions, respectively. The size of a concept description C is denoted by $|C|$ where simply all symbols in C are counted. Later on, we will also refer to the role depth of a concept description.

Construct name	Syntax	Semantics
primitive concept $P \in N_C$	P	$P^I \subseteq \Delta$
top-concept	$\top$	Δ
conjunction	$C \sqcap D$	$C^I \cap D^I$
existential restr. for $r \in N_R$	$\exists r.C$	$\{x \in \Delta \mid \exists y : (x,y) \in r^I \wedge y \in C^I\}$
value restr. for $r \in N_R$	$\forall r.C$	$\{x \in \Delta \mid \forall y : (x,y) \in r^I \rightarrow y \in C^I\}$
primitive negation, $P \in N_C$	$\neg P$	$\Delta \setminus P^I$
bottom-concept	$\bot$	$\emptyset$

Table 1: Syntax and semantics of concept descriptions.

The *role depth* $depth(C)$ of an $\mathcal{ALE}$-concept description C is inductively defined as follows:

- $depth(\top) := depth(\bot) := depth(P) := depth(\neg P) := 0$;
- $depth(C \sqcap D) := max(depth(C), depth(D))$;
- $depth(\forall r.E) := depth(\exists r.E) := 1 + depth(E)$.

The semantics of a concept description is defined in terms of an *interpretation* $I = (\Delta, \cdot^I)$. The domain Δ of I is a non-empty set of individuals and the interpretation function $\cdot^I$ maps each concept name $P \in N_C$ to a set $P^I \subseteq \Delta$ and each role name $r \in N_R$ to a binary relation $r^I \subseteq \Delta \times \Delta$. The extension of $\cdot^I$ to arbitrary concept descriptions is inductively defined, as shown in the third column of Table 1.

Subsumption, equivalence, and least common subsumer

One of the most important traditional inference services provided by DL systems is computing the subsumption hierarchy.

Definition 1 Let C, D be concept descriptions.

- D *subsumes* C (for short $C \sqsubseteq D$) iff $C^I \subseteq D^I$ for all interpretations I.
- C is *equivalent* to D (for short $C \equiv D$) iff $C \sqsubseteq D$ and $D \sqsubseteq C$, i.e., $C^I = D^I$ for all interpretations I.
- D *strictly subsumes* C (for short $C \sqsubset D$) iff $C \sqsubseteq D$ and $C \not\equiv D$.

As shown in [12], deciding subsumption of $\mathcal{ALE}$-concept descriptions is NP-complete. In $\mathcal{EL}$ subsumption can be decided in time polynomial in the size of the concept descriptions [3, 2].

As it turns out, in order to solve a matching problem we need to compute the *least common subsumer* (lcs) of concept descriptions.

Definition 2 Let $C_1, \ldots, C_n$ and C be concept descriptions in a DL $\mathcal{L}$. The concept description C is a *least common subsumer* (lcs) of $C_1, \ldots, C_n$ (for short $C = lcs(C_1, \ldots, C_n)$) iff

1. $C_i \sqsubseteq C$ for all $1 \leq i \leq n$, and
2. C is the least concept description with this property, i.e., if C' is a concept description satisfying $C_i \sqsubseteq C'$ for all $1 \leq i \leq n$, then $C \sqsubseteq C'$.

In [3], the following complexity results regarding the lcs have been shown:[1]

Fact 3 The size of the lcs of a sequence $C_1, \ldots, C_n$ of $\mathcal{ALE}$-concept descriptions may grow exponential in the size of the sequence. The lcs of a sequence of concept descriptions can be computed in time exponential in the size of the sequence.

This fact also carries over to sublanguages of $\mathcal{ALE}$.

Matching

We now formally define matching problems for $\mathcal{ALE}$ and its sublanguages. In particular, we must introduce the notion of a concept pattern and of substitutions operating on patterns. For this purpose, we need an additional set $\mathcal{X}$ of symbols (*concept variables*) disjoint from $N_C \cup N_R$.

Definition 4 The set of all $\mathcal{ALE}$-concept patterns over N_C, N_R, $\mathcal{X}$ is inductively defined as follows:

- Every concept variable $X \in \mathcal{X}$ is a pattern.
- Every $\mathcal{ALE}$-concept description over N_C, N_R is a pattern.
- If C and D are concept patterns, then $C \sqcap D$ is a concept pattern.
- If C is a concept pattern and r is a role name, then $\forall r.C$ and $\exists r.C$ are concept patterns.

Concept patterns for sublanguages of $\mathcal{ALE}$ are defined analogously. Later on, we will need a certain normalform of concept patterns. We therefore have to define equivalence of concept patterns: Two concept patterns C, D are called *equivalent* ($C \equiv D$ for short) if and only if C and D are equivalent concept descriptions where variables are considered to be concept names.

The following notions can be restricted to sublanguages of $\mathcal{ALE}$ as well. A *substitution* σ is a mapping from $\mathcal{X}$ into the set of all $\mathcal{ALE}$-concept descriptions. This mapping is extended to concept patterns in the obvious way, i.e.,

- $\sigma(P) := P$ for all $P \in N_C$,

[1]Actually, only complexity results for the lcs of two concept descriptions are contained in that paper. However, the proofs allow us to simply extend the results to sequences of concept descriptions.

- $\sigma(\top) := \top$ and $\sigma(\bot) := \bot$,
- $\sigma(C \sqcap D) := \sigma(C) \sqcap \sigma(D)$,
- $\sigma(\forall r.C) := \forall r.\sigma(C)$ and $\sigma(\exists r.C) := \exists r.\sigma(C)$, $r \in N_R$.

For example, applying the substitution $\sigma := \{X \mapsto E \sqcap \forall r.E, Y \mapsto F\}$ to the pattern $X \sqcap Y \sqcap \forall r.X$ yields the description $E \sqcap (\forall r.E) \sqcap F \sqcap \forall r.(E \sqcap \forall r.E)$.

Obviously, the result of applying a substitution to an $\mathcal{ALE}$-concept pattern is an $\mathcal{ALE}$-concept description.

For $\mathcal{ALE}$, and more generally, as already mentioned in [1], for any description language in which variables in patterns may only occur in the scope of "monotonic" opertors, one can easily show

Lemma 5 Let D be a $\mathcal{ALE}$-concept pattern and let σ, τ be two substitutions such that $\sigma(X) \sqsubseteq \tau(X)$ for all variables X occurring in D. Then, $\sigma(D) \sqsubseteq \tau(D)$.

Definition 6 An *$\mathcal{ALE}$-matching problem* is of the form $C \equiv^? D$ where C is an $\mathcal{ALE}$-concept description and D is an $\mathcal{ALE}$-concept pattern. A *solution* or *matcher* of this problem is a substitution σ such that $C \equiv \sigma(D)$. A matching problem is said to be *solvable* if there exists a solution.

Instead of a single matching problem, we may also consider a finite system $\{C_1 \equiv^? D_1, \ldots, C_m \equiv^? D_m\}$ of such problems. The substitution σ is a solution of this system if and only if it is a solution of all the matching problems $C_i \equiv^? D_i$ contained in the system. However, as already stated in [1] for the language $\mathcal{ALN}$, solving systems of matching problems can be reduced (in linear time) to solving a single matching problem. The proof of this result can be extended to $\mathcal{ALE}$ where instead of value-restrictions one can also use existential restrictions to simulate a set of matching problems.

Lemma 7 Let $r_1, \ldots, r_m$ be distinct roles names. Then, σ solves the system $\{C_1 \equiv^? D_1, \ldots, C_m \equiv^? D_m\}$ if and only if it solves the single matching problem

$$\forall r_1.C_1 \sqcap \cdots \sqcap \forall r_m.C_m \equiv^? \forall r_1.D_1 \sqcap \cdots \sqcap \forall r_m.D_m.$$

Consequently, we may (without loss of generality) restrict our attention to single matching problems.

In [10, 14, 1] a different type of matching problems has been considered. We will refer to those problems as *matching problems modulo subsumption* in order to distinguish them from the *matching problems modulo equivalence* introduced above.

Definition 8 A *matching problem modulo subsumption* is of the form $C \sqsubseteq^? D$ where C is a concept description and D is a pattern. A solution of this problem is a substitution σ satisfying $C \sqsubseteq \sigma(D)$.

For any description language allowing conjunction of concepts, matching modulo subsumption can be reduced (in linear time) to matching modulo equivalence:

Lemma 9 The substitution σ solves the matching problem $C \sqsubseteq^? D$ if and only if it solves $C \equiv^? C \sqcap D$.

For $\mathcal{ALE}$, and, as mentioned in [1], more generally for any description language in which variables in patterns may only occur in the scope of "monotonic" operators, solvability of matching problems modulo subsumption can be reduced to subsumption:

Lemma 10 Let $C \sqsubseteq^? D$ be a matching problem modulo subsumption in $\mathcal{ALE}$, and let $\sigma_\top$ be the substitution that replaces each variable by $\top$. Then, $C \sqsubseteq^? D$ has a solution if and only if $C \sqsubseteq \sigma_\top(D)$.

3 Matching in $\mathcal{EL}$

In this section, we investigate matching for the language $\mathcal{EL}$. More precisely, there are two algorithmic problems to consider. First, the complexity of deciding the sovability of $\mathcal{EL}$-matching problems is examined. Second, we present algorithms to actually compute matchers. But before, motivated by examples, we formally characterize the set of potentially interesting matchers.

The reason why first exploring matching problems for the small language $\mathcal{EL}$ instead of looking at the more expressive languages $\mathcal{FLE}$ and $\mathcal{ALE}$ right away is twofold: Theoretically, results for matching in one language do not necessarily carry over to sublanguage. The main point, however, are didactical reasons since $\mathcal{FLE}$ and $\mathcal{ALE}$ often require quite involved techniques and proofs.

In the two following subsections, we introduce the basic tools needed throughout this paper.

3.1 Description trees, homomorphisms, and subsumption in $\mathcal{EL}$

$\mathcal{EL}$-description trees are trees where the nodes are labeled with sets of concept names and variables and where the edges are labeled with roles.

Definition 11 An *$\mathcal{EL}$-description tree* is a tree of the form $\mathcal{G} = (V, E, v_0, \ell)$ where

- V is a finite set of *nodes* of $\mathcal{G}$;
- $E \subseteq V \times N_R \times V$ is a finite set of so-called $\exists$-edges labeled with role names $r \in N_R$;
- v_0 is the *root* of $\mathcal{G}$; and
- ℓ is the *labeling function* mapping every node of V to a subset of $N_R \cup \mathcal{X}$.

The empty label corresponds to the top-concept.

For $v, w \in V$ and $r \in N_R$ we write $\exists$-edges as vrw. For the sake of simplicity, we occasionally write $v \in \mathcal{G}$ instead of $v \in V$; $vrw \in \mathcal{G}$ instead of $vrw \in E$; and $\mathcal{G}(v)$ instead of $\ell(v)$.

A sequence $w_0r_1w_1 \cdots r_nw_n$ is a *path* in $\mathcal{G}$ from w_0 to w_n ($w_0r_1w_1 \cdots r_nw_n \in \mathcal{G}$ for short) iff $w_{i-1}r_iw_i \in \mathcal{G}$ for all $i = 1, \dots n$. Such a path is called *rooted* in case w_0 is the root of $\mathcal{G}$.

For $v \in V$, w is a *direct successor* of v in $\mathcal{G}$ if there exists a role $r \in N_R$ with $vrw \in E$; w is a *successor* of v if there exists a path from v to w. Since we allow for empty paths, v is a successor of itself. Analogously, we define *(direct) predecessors.*

A *subtree* $\mathcal{G}'$ of $\mathcal{G}$ is a description tree consisting of a subset of nodes of $\mathcal{G}$. The labels of the nodes in $\mathcal{G}'$ are subsets of the corresponding ones in $\mathcal{G}$; $\mathcal{G}'$ is called *rooted subtree* in case the root of $\mathcal{G}'$ coincides with one for $\mathcal{G}$.

For a node $v \in V$, $\mathcal{G}_v$ denotes the subtree of $\mathcal{G}$ consisting of all successors of v in $\mathcal{G}$. The root of $\mathcal{G}_v$ is v and the labels of the nodes in $\mathcal{G}_v$ coincide with the corresponding ones in $\mathcal{G}$.

With $|\mathcal{G}|$ we denote the size of the description tree $\mathcal{G}$. By *depth*$(\mathcal{G})$ we refer to the maximal length of a rooted path in $\mathcal{G}$.

For $\mathcal{EL}$-description trees $\mathcal{G}$ and $\mathcal{H}$ (with disjoint sets of nodes) and a node $v \in \mathcal{G}$, *instantiating* $\mathcal{G}$ *at node* v *with* $\mathcal{H}$ yields an extension $\mathcal{G}' = (V', E', v_0, \ell')$ of $\mathcal{G} = (V, E, v_0, \ell)$ defined as follows: First, the root of $\mathcal{H}$ is replaced by v, which yields the tree $\mathcal{H}' = (V'', E'', v, \ell'')$. Then,

- $V' := V \cup V''$;
- $E' := E \cup E''$;
- $\ell'(w) := \ell(w)$ for all $w \in V \setminus \{v\}$; $\ell'(w) := \ell''(w)$ for all $w \in V'' \setminus \{v\}$; $\ell'(v) := \ell(v) \cup \ell(v'')$.

An $\mathcal{EL}$-concept description/pattern C can be tranlated into an $\mathcal{EL}$-description tree $\mathcal{G}(C)$ (see [3] for details). Intuitively, $\mathcal{G}(C)$ is the syntax tree for C.

Example 12 The $\mathcal{EL}$-concept description

$$C := P \sqcap \exists r.(\exists r.(P \sqcap Q) \sqcap \exists s.Q) \sqcap \exists r.(P \sqcap \exists s.P)$$

yields the tree $\mathcal{G}(C)$ depicted on the left hand-side of Figure 1.

On the other hand, every $\mathcal{EL}$-description tree $\mathcal{G} = (V, E, v_0, \ell)$ can be translated into an $\mathcal{EL}$-concept description $C_\mathcal{G}$.

Example 13 The $\mathcal{EL}$-description tree $\mathcal{G}$ in Figure 1 yields the $\mathcal{EL}$-concept description

$$C_\mathcal{G} = \exists r.(\exists r.P \sqcap \exists s.Q) \sqcap \exists r.P.$$

The semantics of $\mathcal{EL}$-description trees $\mathcal{G}$ without variables in their labels is defined by the semantics of their corresponding concept descriptions $C_\mathcal{G}$, i.e.,

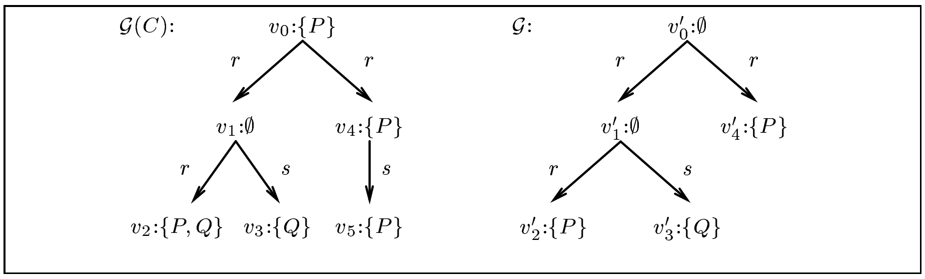

Figure 1: $\mathcal{EL}$-description trees.

$\mathcal{G}^I := C^I_\mathcal{G}$ for an interpretation I. As shown in [3], the translation of concept descriptions and description trees in one another preserves semantics, i.e., $C \equiv C_{\mathcal{G}_C}$. With the formal semantics for description trees, the subsumption relationship can be stated not only between concept descriptions but also between description trees ($\mathcal{G} \sqsubseteq \mathcal{H}$) or between concept descriptions and description trees ($C \sqsubseteq \mathcal{G}$) in the obvious way.

In [3], homomorphisms between description trees have been employed to characterize subsumption between concept descriptions. Moreover, such characterizations have been used to describe the lcs of concept descriptions as product of descriptions trees. We will see that homomorphisms are also crucial for the matching algorithms we propose here.

Definition 14 Let $\mathcal{G} = (V_G, E_G, v_0, \ell_G)$ and $\mathcal{H} = (V_H, E_H, w_0, \ell_H)$ be $\mathcal{EL}$-description trees. A mapping $\varphi : V_H \longrightarrow V_G$ is a *homomorphism* from $\mathcal{H}$ into $\mathcal{G}$ iff the following conditions are satisfied:

1. $\varphi(w_0) = v_0$,
2. $(\ell_H(v) \setminus \mathcal{X}) \subseteq \ell_G(\varphi(v))$ for all $v \in V_H$, and
3. $\varphi(v) r \varphi(w) \in E_G$ for all $vrw \in E_H$.

In order to characterize equivalence of concept descriptions we also need to define isomorphisms between description trees.

Definition 15 Let $\mathcal{G} = (V_G, E_G, v_0, \ell_G)$ and $\mathcal{H} = (V_H, E_H, w_0, \ell_H)$ be two $\mathcal{EL}$-description trees. The mapping φ from V_H onto V_G is called *isomorphism* from $\mathcal{H}$ onto $\mathcal{G}$ iff

- φ is a bijection from V_H onto V_G;
- $\varphi(w_0) = v_0$;
- for all $v, w \in V_H$ and $r \in N_R$: $vrw \in E_H$ iff $\varphi(v) r \varphi(w) \in E_G$;
- for all $v \in V_H$: $\ell_H(v) = \ell_G(\varphi(v))$.

Two description trees $\mathcal{G}$ and $\mathcal{H}$ are called *isomorphic* ($\mathcal{G} \cong \mathcal{H}$ for short) if there exists an isomorphism between them.

Definition 16 Let $\mathcal{G} = (V_G, E_G, v_0, \ell_G)$ and $\mathcal{H} = (V_H, E_H, w_0, \ell_H)$ be $\mathcal{EL}$-description trees, and let φ be an homomorphism from $\mathcal{H}$ into $\mathcal{G}$. Finally, let $\mathcal{H}' = (V', E', v', \ell')$ be a subtree of $\mathcal{H}$. Then, the *homomorphic image* $\varphi(\mathcal{H}') = (V, E, v, \ell)$ of φ w.r.t. $\mathcal{H}'$ is defined as follows:

- $V := \varphi(V') := \{w \mid \text{ there exists a } w' \in V' \text{ with } w = \varphi(w')\}$;
- $E := E_G \cap (V \times N_R \times V)$;
- $v := \varphi(v')$;
- $\ell(w) := \left(\bigcup_{w' \in \varphi^{-1}(w)} \ell'(w) \right)$ for all $w \in V$ where $\varphi^{-1}(w') := \{w' \mid \varphi(w') = w\}$.

It is easy to see that

Lemma 17 1. $\varphi(\mathcal{H}')$ is a subtree of $\mathcal{G}$; and

2. φ is an surjective homomorphism from $\mathcal{H}'$ onto $\varphi(\mathcal{H}')$.

Definition 18 Let $\mathcal{G} = (V_G, E_G, v_0, \ell_G)$ and $\mathcal{H} = (V_H, E_H, w_0, \ell_H)$ be $\mathcal{EL}$-description trees, and let $\psi : V_H \longrightarrow V_G$ be a homomorphism from $\mathcal{H}$ into $\mathcal{G}$. Finally, let $\mathcal{G}' = (V', E', v', \ell')$ be some subtree of $\mathcal{G}$ in case ψ is injective, and a rooted subtree in case ψ is not injective. Then, the inverse image of $\mathcal{G}'$ w.r.t. ψ, $\psi^{-1}(\mathcal{G}') = (V, E, v, \ell)$, is defined as follows:

- $V := \psi^{-1}(V') := \{w \in V_H \mid \psi(w) \in V'\}$; if $V = \emptyset$, then let $\psi^{-1}(\mathcal{G}')$ be a description tree containing only a root with empty label; otherwise
- $E := E_H \cap (V \times N_R \times V)$;
- $v := \psi^{-1}(v')$;
- $\ell(w) := \ell_H(w) \cap \ell'(\psi(w))$ for all $w \in V$.

We summarize some simple properties of the inverse homomorphism in case $V \neq \emptyset$.

Lemma 19 1. $\psi^{-1}(\mathcal{G}')$ is a subtree of $\mathcal{H}$ with root $\psi^{-1}(v')$.

2. $\psi(\psi^{-1}(\mathcal{G}'))$ is a rooted subtree of $\mathcal{G}'$.

3. In case ψ is injective, $\psi(\psi^{-1}(\mathcal{G}'))$ and $\psi^{-1}(\mathcal{G}')$ are isomorphic.

Subsumption between $\mathcal{EL}$-concept descriptions can be characterized in terms of homomorphisms between their corresponding $\mathcal{EL}$-description trees [3]:

Theorem 20 Let C, D be $\mathcal{EL}$-concept descriptions and $\mathcal{G}(C), \mathcal{G}(D)$ the corresponding description trees. Then $C \sqsubseteq D$ iff there exists a homomorphism from $\mathcal{G}(D)$ into $\mathcal{G}(C)$.

Example 21 (Example 12 continued)
Consider the $\mathcal{EL}$-description trees depicted in Figure 1. We have $C \sqsubseteq C_{\mathcal{G}}$, because mapping v_i' onto v_i for $0 \leq i \leq 4$ yields a homomorphism from $\mathcal{G}$ to $\mathcal{G}(C)$.

The following observation is an easy consequence of Theorem 20, where for an $\mathcal{EL}$-concept description C, $\mathit{prim}(C)$ denotes the set of concept names on the top-level of C. We write $\exists r.E \in C$ to say that $\exists r.E$ is an existential restriction on the top-level of C.

Observation 22 Let C, D be two $\mathcal{EL}$-concept descriptions. Then, $C \sqsubseteq D$ iff

1. $\mathit{prim}(D) \subseteq \mathit{prim}(C)$; and
2. for every existential restriction $\exists r.E \in D$, there exists an existential restriction $\exists r.F \in C$ such that $F \sqsubseteq E$.

3.2 Equivalence of $\mathcal{EL}$-concept descriptions

In this section, we will show that for $\mathcal{EL}$-concept descriptions there exist unique minimal representations. In particular, equivalence of $\mathcal{EL}$-concept descriptions can be characterized in terms of isomorphisms between the description trees of the minimal representations. This result is important in different respects: First, minimal representations will be used as basis to specify sets of matchers that do not contain redundancies. Second, we will employ the characterization in order to prove complexity results for deciding the solvability of matching problems. Finally, this result is interesting on its own right in the context of computing minimal rewritings [4].

Intuitively, minimal representations of concepts are those concept descriptions that do not contain redundancies. Formally, such concepts are called reduced. In order to define reduced concept descriptions we need to define subdescriptions.

Definition 23 For an $\mathcal{EL}$-concept description C, the $\mathcal{EL}$-concept description $\widehat{C}$ is a *subdescription* of C ($\widehat{C} \preceq_d C$) iff $\widehat{C}$ is obtained from C by removing some concept names or existential restrictions on the top-level of C, and for all remaining existential restrictions $\exists r.E$ replacing E by a subdescription of E.

Clearly, if everthing is removed from C, then the resulting subdescription is $\top$. On the other hand, if nothing is removed, then $\widehat{C} = C$. The concept description $\widehat{C}$ is a *strict subdescription* of C if $\widehat{C} \neq C$. Now, we are primed to define reduced concept descriptions.

Definition 24 An $\mathcal{EL}$-concept description C is *reduced* iff there exists no strict subdescription of C which is equivalent to C.

In fact, equivalent and reduced $\mathcal{EL}$-concept description are equal up to commutativity and assoziativity of concept conjunction.

Theorem 25 Let C, D be reduced $\mathcal{EL}$-concept descriptions. Then, $C \equiv D$ iff $\mathcal{G}(C) \cong \mathcal{G}(D)$.

For $\mathcal{ALE}$, a more general statement is proved in Section 5. We therefore postpone the proof of Theorem 25.

With the help of the following proposition, we show that every $\mathcal{EL}$-concept description C can be turned into an equivalent reduced concept.

Proposition 26 Let C be an $\mathcal{EL}$-concept description. Then, C is reduced iff

- every concept name on the top-level of C only occurs ones;
- for two distinct existential restrictions $\exists r.E, \exists r.F \in C$ it is $E \not\sqsubseteq F$; and
- for all existential restrictions $\exists r.E \in C$, E is reduced.

Thus, a concept C can be turned in polynomial time into an equivalent reduced concept description, called $C \setminus \top$ in the sequel, by recursively deleting i) multiple occurrences of concept names and ii) redundant existential restrictions using the (polynomial) subsumption algorithm for $\mathcal{EL}$. Obviously, $|C \setminus \top| \leq |C|$. More precisely, using Theorem 25, we can show that $C \setminus \top$ is the minimal representation of concepts equivalent to C (see the proof of Corollary 74 in the more general case of $\mathcal{ALE}$-concept descriptions).

Corollary 27 For every $\mathcal{EL}$-concept description C one can compute (in polynomial time) an equivalent $\mathcal{EL}$-conept description C', namely $C \setminus \top$, with $|C'| \leq |E|$ for all $E \equiv C$.

In the subsequent sections, we need a variant of Theorem 25 where only one concept description is reduced.

Proposition 28 Let C and D be $\mathcal{EL}$-concept descriptions with C reduced. Let $\mathcal{G}(C) = (V, E, v_0, \ell)$ and $\mathcal{G}(D) = (V', E', v_0', \ell')$ be the corresponding $\mathcal{EL}$-description trees. Then, $C \equiv D$ implies that every homomorphism ψ from $\mathcal{G}(C)$ into $\mathcal{G}(D)$ is an injective homomorphism and for all $v \in V$, $\mathcal{G}(C)_v \equiv \mathcal{G}(D)_{\psi(v)}$.

Proof. Let $r \in N_R$, $(v_0, r, v_1) \in E$, and ψ be a homomorphism from $\mathcal{G}(C)$ into $\mathcal{G}(D)$. We know that $\mathcal{G}(C)_{v_1} \sqsupseteq \mathcal{G}(D)_{\psi(v_1)}$. Assume, that the subsumption relation is strict. Since $C \equiv D$, by Observation 22 there exists a node $v_2 \in V$, $(v_0, r, v_2) \in E$ with $\mathcal{G}(D)_{\psi(v_1)} \sqsupseteq \mathcal{G}(C)_{v_2}$. Clearly, $v_1 \neq v_2$. But then, $\mathcal{G}(C)_{v_1} \sqsupset \mathcal{G}(C)_{v_2}$ is a contradiction to the fact that C is reduced. Thus, $\mathcal{G}(C)_{v_1} \equiv \mathcal{G}(D)_{\psi(v_1)}$. By induction this proves that for all $v \in V$, $\mathcal{G}(C)_v \equiv \mathcal{G}(D)_{\psi(v)}$. Note that if C is reduced, then also C_v for all $v \in V$.

It remains to show that ψ is injective. Assume, there exist $v_1, v_2 \in V$, $v_1 \neq v_2$, $(v_0, r, v_1), (v_0, r, v_2) \in E$ with $\psi(v_1) = \psi(v_2)$. Then, $\mathcal{G}(C)_{v_1} \equiv \mathcal{G}(C)_{v_2} \equiv \mathcal{G}(D)_{\psi(v_1)}$. Again, a contradiction to the fact that C is reduced. ∎

3.3 Deciding solvability of matching in $\mathcal{EL}$

Lemma 10 shows that the solvability of an $\mathcal{EL}$-matching problem modulo subsumption can be decided in time polynomial in the size of the matching problem.

The aim of this section, is to show that deciding the solvability of $\mathcal{EL}$-matching problems modulo equivalence is NP-complete.

In order to prove that there is an NP algorithm, it is sufficient to show that every solvable $\mathcal{EL}$-matching problem has a matcher, which is i) polynomially bounded in the size of the matching problem and ii) uses only concept names and role names already contained in the matching problem. An NP algorithm then guesses a matcher σ polynomially bounded in the size of the matching problem and checkes whether C is equivalent to $\sigma(D)$.

Theorem 29 If the $\mathcal{EL}$-matching problem $C \equiv^? D$ is solvable, then there exists a matcher of size polynomial in the size of the matching problem which only uses concept names and role names already contained in the matching problem.

In the following we construct a matcher σ polynomially bounded in the size of the matching problem (using only identifiers in C) given a matcher σ' for $C \equiv^? D$.

Since $C \equiv \sigma'(D)$ there exists a homomorphism ψ from $\mathcal{G}(C)$ into $\mathcal{G}(\sigma'(D))$ according to Theorem 20. By Proposition 28, we know that ψ is an injective homomorphism.

Before constructing σ we need some more notation. We refer to the subset of all nodes in $\mathcal{G}(D)$ with variable X in their label by $V_D(X) := \{w \in \mathcal{G}(D) \mid X \in \mathcal{G}(D)(w)\}$. The description tree $\mathcal{G}(\sigma'(D))$ is obtained by instantiating $\mathcal{G}(D)$ at every node $w \in V_D(X)$ and every variable X in D by (a copy of) $\mathcal{G}(\sigma'(X))$, which we call $\mathcal{G}_{\sigma'(X),w}$ in the sequel. The root of $\mathcal{G}_{\sigma'(X),w}$ is w.

Now σ is defined as follows:

$$\sigma(X) := \bigsqcap_{\substack{w \in V_D(X) \\ \psi^{-1}(w) \text{ is defined}}} C_{\psi^{-1}(\mathcal{G}_{\sigma'(X),w})}$$

for every variable X in D.

Note that by construction, $\sigma(X)$ is build up only from identifiers in C.

Lemma 30 $\sigma \sqsupseteq_s \sigma'$.

Proof. Let w be a node in $V_D(X)$. By Lemma 17, ψ is a homomorphism from $\psi^{-1}(\mathcal{G}_{\sigma'(X),w})$ onto $\psi(\psi^{-1}(\mathcal{G}_{\sigma'(X),w}))$, which, according to Lemma 19, is a rooted subtree of $\mathcal{G}_{\sigma'(X),w}$. This shows that $C_{\psi^{-1}(\mathcal{G}_{\sigma'(X),w})} \sqsupseteq \mathcal{G}_{\sigma'(X),w} \equiv \sigma'(X)$. Hence, $\sigma(X) \sqsupseteq \sigma'(X)$. ∎

By virtue of Lemma 5, we can conclude $\sigma(D) \sqsupseteq \sigma'(D)$.

Lemma 31 $\sigma(D) \sqsubseteq C$.

PROOF. We show that ψ is a homomorphism from $\mathcal{G}(C \setminus \top)$ into $\mathcal{G}(\sigma(D))$.

Let $\mathcal{G}$ be the description tree obtained by instantiating $\mathcal{G}(D)$ as follows: For every variable X in D and node $w \in V_D(X)$ instantiate $\mathcal{G}(D)$ by $\psi(\psi^{-1}(\mathcal{G}_{\sigma'(X),w}))$. It is easy to see that ψ is an homomorphism from $\mathcal{G}(C \setminus \top)$ into $\mathcal{G}$ (Section 6.2 contains a more detailed proof for $\mathcal{FLE}$).

Now observe that for $w \in V_D(X)$, $\psi^{-1}(\mathcal{G}_{\sigma'(X),w})$ is isomorphic to $\psi(\psi^{-1}(\mathcal{G}_{\sigma'(X),w}))$ (see Lemma 19). Therefore, $\mathcal{G}$ is a subtree of $\mathcal{G}(\sigma(D))$. Thus, ψ is also a homomorphism from $\mathcal{G}(C \setminus \top)$ into $\mathcal{G}(\sigma(D))$, which implies $C \sqsupseteq \sigma(D)$. ∎

From Lemma 30 and 31 we can deduce that σ is a matcher of the problem $C \equiv^? D$. It remains to show that the size of σ is polynomially bounded in the size of the matching problem. This fact is a consequence of the following lemma.

Lemma 32 For every variable X in D, the size of $\sigma(X)$ is linearly bounded in the size of C.

PROOF. Let $w \in V_D(X)$ and $\psi^{-1}(w)$ be defined. We know that $\psi^{-1}(\mathcal{G}_{\sigma'(X),w})$ is a subtree of $C\setminus\top$. Furthermore, for different w's one obtains disjoint subtrees. ∎

We now show NP-hardness by reducing SAT [13] to the problem of deciding the solvability of an $\mathcal{EL}$-matching problem modulo equivalence.

Let $\phi = p_1 \wedge \cdots \wedge p_m$ be a propositional formulae in conjunctive normal form and let $\{x_1, \ldots, x_n\}$ be the propositional variables of this problem. For these variables, we introduce the concept variables $\{X_1, \ldots, X_n, \overline{X}_1, \ldots, \overline{X}_n\}$. Furthermore, we need concept names A and B as well as the role names r, r', s, s'. We first show that one can specify a matching problem such that either X_i must be substituted by a concept equivlanet to A (corresponding to *true*) and $\overline{X}_i$ by (a concept equivalent to) B or vice versa. This corresponds to assigning *true* or *false* to x_i. Such a matching problem can be written in terms of the following concept descriptions/patterns:

$$\begin{aligned}
C_0 &:= \top \\
C_{k+1} &:= \exists r'.A \sqcap \exists r'.B \sqcap \exists r.C_k \\
D_0 &:= \top \\
D_{k+1} &:= \exists r'.X_{k+1} \sqcap \exists r'.\overline{X}_{k+1} \sqcap \exists r.D_k
\end{aligned}$$

Now, the matching problem can be stated as $C_n \equiv^? D_n$. It is easy to see that a matcher of this problem must substitute either X_i by (a concept equivalent to) A and $\overline{X}_i$ by (a concept equivalent to) B or vice versa.

In order to decode ϕ we translate a conjunct p_i, $1 \leq i \leq m$, into a concept pattern D_{p_i} as follows. For example, if $p_i = x_1 \vee \overline{x}_2 \vee x_3 \vee \overline{x}_4$, then $D_{p_i} := X_1 \sqcap \overline{X}_2 \sqcap X_3 \sqcap \overline{X}_4 \sqcap B$. We now have to decode the evaluation of a formulae:

$$\begin{array}{rcl} C'_0 & := & \top \\ C'_{k+1} & := & \exists s'.(A \sqcap B) \sqcap \exists s.C'_k \\ D'_0 & := & \top \\ D'_{k+1} & := & \exists s'.D_{p_{k+1}} \sqcap \exists s.D'_k \end{array}$$

The matching problem $C'_m \equiv^? D'_m$ ensures that among the variables in D_{p_i} there must be at least one variable substituted by (a concept equivalent to) A. This corresponds to the fact that within one conjunct p_i there must be at least one propositional variable that evaluates to *true*. Note that we need the concept B in D_{p_i} because otherwise if all variables in D_{p_i} were substituted by A then C'_m and D'_m would not be equivalent.

We combine the two parts of the reduction in the final matching problem as follows:

$$\begin{array}{rcl} C_\phi & := & C_n \sqcap C'_m \\ D_\phi & := & D_n \sqcap D'_m \end{array}$$

It is easy to verify that ϕ is satisfiable if and only if the matching problem $C_\phi \equiv^? D_\phi$ is solvable since a truth assignment can be directly translated into a substitution and vice versa. Together with the upper bound this proves:

Corollary 33 Deciding the solvability of an $\mathcal{EL}$-matching problem modulo equivalence is NP-complete.

3.4 Solutions of $\mathcal{EL}$-matching problems

We will use the $\mathcal{EL}$-concept description C_{ex} and the pattern D_{ex} shown in Figure 2 together with their description trees in order to illustrate and to formally characterize the potentially most interesting matchers of matching problems.

It is easy to see that the substitution $\sigma_\top$, which maps all variables on the top-concept, is a matcher of $C_{\text{ex}} \sqsubseteq^? D_{\text{ex}}$, and thus this matching problem modulo subsumption is indeed solvable. However, the matcher $\sigma_\top$ is obviously not an interesting one. We are interested in matchers that bring us as close as possible to the description C_{ex}. In this sense, the matcher $\sigma_1 := \{X \mapsto \mathsf{W} \sqcap \exists \mathsf{hc}.\mathsf{W}, Y \mapsto \mathsf{W}\}$ is better than $\sigma_\top$, but still not optimal. In fact, $\sigma_2 := \{X \mapsto \mathsf{W} \sqcap \exists \mathsf{hc}.\mathsf{W} \sqcap \exists \mathsf{hc}.(\mathsf{W} \sqcap \mathsf{P}), Y \mapsto \mathsf{W} \sqcap \mathsf{D}\}$ is better than σ_1 since it satisfies $C_{\text{ex}} \equiv \sigma_2(D_{\text{ex}}) \sqsubset \sigma_1(D_{\text{ex}})$.

We formalize this intuition with the help of the following precedence ordering on matchers. For a given $\mathcal{EL}$-matching problem $C \sqsubseteq^? D$ and two matchers σ, τ we define

$$\sigma \sqsubseteq_i \tau \text{ iff } \sigma(D) \sqsubseteq \tau(D).$$

Here "i" stands for "instance". Observe that $\sqsubseteq_i$ is a quasi-ordering, i.e., a reflexiv and transitiv ordering. Thus, $\sqsubseteq_i$ induces an equivalence relation $\equiv_i$:

$C_{\text{ex}} := \mathsf{W} \sqcap \exists \mathsf{hc}.(\mathsf{W} \sqcap \exists \mathsf{hc}.(\mathsf{W} \sqcap \mathsf{D}) \sqcap \exists \mathsf{hc}.(\mathsf{W} \sqcap \mathsf{P})) \sqcap \exists \mathsf{hc}.(\mathsf{W} \sqcap \mathsf{D} \sqcap \exists \mathsf{hc}.(\mathsf{W} \sqcap \mathsf{P}))$
$D_{\text{ex}} := \mathsf{W} \sqcap \exists \mathsf{hc}.(X \sqcap \exists \mathsf{hc}.(\mathsf{W} \sqcap Y)) \sqcap \exists \mathsf{hc}.(X \sqcap Y)$

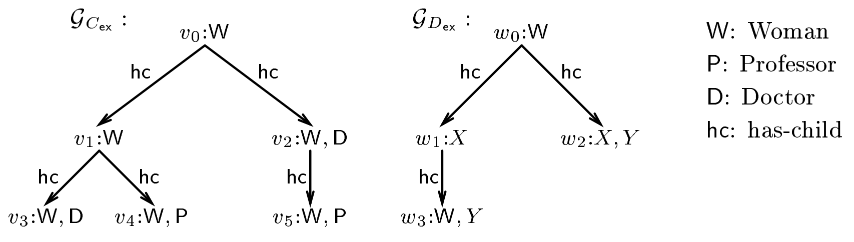

Figure 2: $\mathcal{EL}$-concept description and pattern, and their $\mathcal{EL}$-description trees.

two matchers σ, τ are *i-equivalent* ($\sigma \equiv_i \tau$) iff $\sigma \sqsubseteq_i \tau$ and $\tau \sqsubseteq_i \sigma$. A matcher σ is called *i-minimal* iff, for every matcher τ, $\tau \sqsubseteq_i \sigma$ implies $\tau \equiv_i \sigma$.

We are interested in *computing i-minimal matchers*; more precisely, we want to obtain at least one i-minimal matcher for each of the minimal i-equivalence classes (i.e., i-equivalence classes of i-minimal matchers). Since an i-equivalence class usually contains more than one matcher, the question is which ones to prefer.

In [1], it is shown that a given $\mathcal{ALN}$-matching problem[2] always has a unique minimal i-equivalence class, and that this class is the class of the least matcher w.r.t. the ordering

$$\sigma \sqsubseteq_s \tau \ \text{ iff } \ \sigma(X) \sqsubseteq \tau(X) \text{ for all } X \in \mathcal{X},$$

where "s" stands for "substitution". The matcher σ is a *least matcher w.r.t.* $\sqsubseteq_s$ iff $\sigma \sqsubseteq_s \tau$ for all matchers τ. The notions s-minimal, s-maximal, s-equivalent, etc. are defined in the obvious way.

For $\mathcal{EL}$, things are quite different. As illustrated by the example $\exists r.A \sqcap \exists r.B \sqsubseteq^? \exists r.X$, a given matching problem may have several non-equivalent i-minimal (s-minimal) matchers: the substitutions $\{X \mapsto A\}$ and $\{X \mapsto B\}$ are both i- and s-minimal, and they are obviously neither i- nor s-equivalent. In Section 3.5, we will show that the set of all s-minimal matchers (up to s-equivalence) also contains all i-minimal matchers (up to i-equivalence). However, the s-minimal matchers are usually not the best representatives of their i-equivalence class.

In our running example, σ_2 is a least and therefore i-minimal matcher. Nevertheless, it is not the one we really want to compute since it contains redundancies, i.e., expressions that are not really necessary for obtaining the instance $\sigma_2(D_{\text{ex}})$ (modulo equivalence). In fact, σ_2 contains two different kinds of redundancies. First, the concept description $\sigma_2(X)$ is redundant since removing $\exists\mathsf{hc}.\mathsf{W}$ still yields a concept description equivalent to $\sigma_2(X)$. In other words,

[2]$\mathcal{ALN}$ allows for concept conjunction, primitive negation, value restrictions, and number restrictions.

$\sigma_2(X)$ is not reduced. Second, W in $\sigma_2(Y)$ is redundant in that the substitution obtained by deleting W from $\sigma_2(Y)$ still yields the same instance of D_{ex} (although the substitution obtained this way is no longer s-equivalent to σ_2). In our example, the only i-minimal matcher (modulo associativity and commutativity of concept conjunction) that is free of redundancies in this sense is $\sigma_3 := \{X \mapsto \mathsf{W} \sqcap \exists \mathsf{hc}.(\mathsf{W} \sqcap \mathsf{P}),\ Y \mapsto \mathsf{D}\}$.

We want to *compute i-minimal matchers that are reduced*, i.e., free of redundancies. It remains to formalize the notion "reduced" more rigorously. For this purpose, we extend the ordering $\preceq_d$ on concept descriptions to matchers as follows: For matchers σ, τ we define

$$\sigma \preceq_d \tau \ \text{ iff } \ \sigma(X) \preceq_d \tau(X) \text{ for all } X \in \mathcal{X}.$$

Again, this specifies a quasi-ordering and the notions d-minimal, d-equivalent, etc. are defined in the obvious way. The matcher σ of $C \sqsubseteq^? D$ is called *reduced* iff it is a d-minimal matcher (i.e., minimal w.r.t. $\preceq_d$). Note that, given a reduced matcher, every concept description $\sigma(X)$ is reduced. However, as illustrated in our running example (removal of W in $\sigma_2(Y)$), just replacing the descriptions $\sigma(X)$ by equivalent reduced descriptions does not necessarily yield a reduced matcher.

To sum up, given a matching problem $C \sqsubseteq^? D$, we want to compute matchers that are i-minimal and reduced. It should be noted that a given i-equivalence class of matchers may contain different reduced matchers. Since reduced and s-equivalent matchers are equal up to associativity and commutativity of conjunction (Theorem 25), it is, however, sufficient to compute the reduced matchers up to s-equivalence.

A second example to illustrate the different kinds of matchers, is taken from one of our applications in process engineering [6].

Example 34 The $\mathcal{EL}$-concept description Reactor:

```
CompositeDeviceImplementation ⊓
∃hasPart.(
   ReactingPhase ⊓
   ∃isCoupledTo.(
      CompositeConnectionImplementation ⊓ ∃isCoupledTo.(Film ⊓ ...) ⊓
      ∃isCoupledTo.CoolingPhase
   )
)
```

is matched against the $\mathcal{EL}$-concept pattern NewReactor:

$\mathsf{CompositeDeviceImplementation} \sqcap$
$\exists\mathsf{hasPart}.($
$\quad\exists\mathsf{isCoupledTo}.($
$\quad\quad X_{\mathsf{Wall}} \sqcap$
$\quad\quad\exists\mathsf{isCoupledTo}.\mathsf{CoolingPhase}$
$\quad)$
$).$

where in NewReactor the definition of a Wall is left unspecified, and therefore, is replaced by a variable.

Again, the matching problem $\mathsf{Reactor} \sqsubseteq^? \mathsf{NewReactor}$ has several solutions. In particular, $\sigma_\top$ is one of them. As in the previous example, $\sigma_\top$ is not i-minimal. Mapping X_{Wall} onto

$$\mathsf{CompositeConnectionImplementation} \sqcap \exists\mathsf{isCoupledTo}.(\mathsf{Film} \sqcap \ldots) \sqcap \exists\mathsf{isCoupledTo}.\mathsf{CoolingPhase},$$

however, yields an i-minimal matcher σ_1. It is plain that deleting the subdescription $\exists\mathsf{isCoupledTo}.\mathsf{CoolingPhase}$ of $\sigma_1(X_{\mathsf{Wall}})$ still yields an i-minimal matcher. Thus, σ_1 is not reduced. In fact, the only reduced and i-minimal matcher (up to s-equivalence) is the one mapping X_{Wall} on

$$\mathsf{CompositeConnectionImplementation} \sqcap \exists\mathsf{isCoupledTo}.(\mathsf{Film} \sqcap \ldots).$$

Our approach for computing i-minimal and reduced matchers of $C \sqsubseteq^? D$ proceeds in two steps (which we consider in more detail in the next two sections):

1. Compute the set of all i-minimal matchers of $C \sqsubseteq^? D$ up to i-equivalence (i.e., one matcher for each i-equivalence class).
2. For each i-minimal matcher σ computed in the first step, compute the d-minimal matchers up to s-equivalence of the matching problem $\sigma(D) \equiv^? D$.

Of course, if we are interested in matching modulo equivalence in the first place, we just apply the second step to $C \equiv^? D$.

3.5 Computing i-minimal matchers in $\mathcal{EL}$

In this section, we show how the first step of our approach to compute i-minimal and reduced matchers can be performed, i.e., we present an algorithm that computes the set of all i-minimal matchers up to i-equivalence.

It turns out that this task can be split in two subtasks: First, compute a so-called s-complete set containing (at least) all s-minimal matchers up to s-equivalence. Second, given an s-complete sets, filter out the i-minimal matchers (up to i-equivalence).

Before going into the details of the algorithms, we introduce the notion of complete sets in a more general setting known from unification theory.

Complete sets

I-minimality and s-minimality are defined with respect to the quasi-orderings $\sqsubseteq_i$ and $\sqsubseteq_s$ on matchers. Therefore, when computing sets of i-minimal (and later on also s-maximal) matchers we are conceptually faced with a similar problem as in unification theory (see e.g., [7]). In unification theory the problem is to compute a set of minimal unifiers for a given unification problem. The orderings on unifiers are quasi-orderings as well. It has turned out that so-called minimal complete sets, employed in unification theory, exactly represent the set of minimal/maximal solutions we are interested in.

Following [7], we introduce complete sets for the quasi-ordering $\sqsubseteq_q$ on some set S ("S" for solutions) before coming back to the orderings $\sqsubseteq_i$ and $\sqsubseteq_s$. Let $\equiv_q$ be the equivalence relation induced by $\sqsubseteq_q$, i.e., for all $x, y \in S$, $x \equiv_q y$ if and only if $x \sqsubseteq_q y$ and $y \sqsubseteq_q x$. The strict ordering $\sqsubset_q$ of $\sqsubseteq_q$ is defined as usual: $x \sqsubset y$ if and only if $x \sqsubseteq_q y$ and $x \not\equiv_q y$. Also, *q-minimal* and *q-maximal* element are defined as above.

Now, minimal complete sets of solutions are defined as follows:

Definition 35 A subset $\mathcal{C} \subseteq S$ of S is called *q-(co-)complete* if and only if for all elements $s \in S$ there exists an element $s' \in \mathcal{C}$ such that $s' \sqsubseteq_q s$ ($s' \sqsupseteq_q s$). Furthermore, $\mathcal{C}$ is called *miminal q-(co-)complete* if and only if $\mathcal{C}$ is (co-)complete and any two distinct elements in $\mathcal{C}$ are incomparable, i.e., for all $s, s' \in \mathcal{C}$, $s \sqsubseteq_q s'$ implies $s = s'$.

In the sequel, we will only consider the relationship between minimal elements and complete sets. Analogous properties are true for maximal elements and co-complete sets.

Later on we will need the following lemma in order to compare two quasi-orderings.

Lemma 36 If $\sqsubseteq_q$ and $\sqsubseteq_p$ are two quasi-orderings over S, then $\sqsubseteq_q \subseteq \sqsubseteq_p$ implies that every q-complete set is also p-complete.

PROOF. Let $\mathcal{C}$ be a q-complete set and let $s \in S$. Thus, there exists an element $s' \in \mathcal{C}$ such that $s' \sqsubseteq_q s$. We know that then $s' \sqsubseteq_p s$ which shows that $\mathcal{C}$ is p-complete.[3] ∎

In order to obtain a characterization of minimal q-complete sets, we now consider q-equivalence classes over S. For an element $s \in S$, its q-equivalence class is defined as usual by $[s]_q := \{s' \in S \mid s \equiv_q s'\}$. Then, $\overline{S} := \{[s]_q \mid s \in S\}$ denotes the set of q-equivalence classes of S. The ordering $\sqsubseteq_q$ can be extended to $\overline{S}$ as follows: $[s]_q \leq_q [s']_q$ iff $s \sqsubseteq_q s'$. Now, $\leq_q$ is a partial ordering over $\overline{S}$. The notions $\leq_q$*-minimal*, $\leq_q$-complete, and minimal $\leq_q$-complete are defined analogously to the ordering $\sqsubseteq_q$. As shown in [7], a minimal $\leq_q$-complete set can equivalently be defined as a complete set which is minimal among all complete sets with respect to set inclusion.

[3]We obtain the same result for q-co-complete and p-co-complete sets.

The following lemma describes the connection between minimal $\leq_q$-complete sets and the set of all $\leq_q$-minimal elements in $\overline{S}$ [7].

Lemma 37 Let M be the set of $\leq_q$-minimal elements of $\overline{S}$. Then:

1. If $\mathcal{C} \subseteq \overline{S}$ is a minimal $\leq_q$-complete set, then $\mathcal{C} = M$.
2. If M is $\leq_q$-complete, then it is minimal $\leq_q$-complete.

This means that if an minimal $\leq_q$-complete set exists, then this set is exactly the set of $\leq_q$-minimal elements. And on the other hand, if the set of $\leq_q$-minimal elements is not complete, then there is no minimal $\leq_q$-complete set.

An easy consequence of this lemma is the following theorem [7]:

Theorem 38 Let M be the set of all $\leq_q$-minimal elements of $\overline{S}$. If $\mathcal{C}$ is a minimial q-complete set over S, then $M = \{[s]_q \mid s \in \mathcal{C}\}$. Conversely, if M is $\leq_q$-complete, then any set of representatives of M is a minimal q-complete set over S.

From this theorem one can conclude that there exists a minimal q-complete set over S if and only if M is $\leq_q$-complete. Furthermore, the minimal q-complete sets in S are unique up to q-equivalence. Theorem 38 also holds true when considering the set of $\leq_q$-maximal elements (as opposed to minimal elements) and co-complete sets instead of complete sets.

We now apply the results stated above to matching. If, for a given matching problem, S is the set of matchers of this problem and $\sqsubseteq_q$ is a quasi-ordering on these matchers, then Theorem 38 shows that a minimal q-(co-)complete set, as defined in Definition 35, exactly represents the q-minimal/maximal solutions of the matching problem.

By Lemma 5, we can conclude $\sqsubseteq_s \subseteq \sqsubseteq_i$. Then, Lemma 36 ensures

Lemma 39 Every s-(co-)complete set is also i-(co-)complete.

The converse direction is not true in general: for the matching problem $\exists r.A \sqcap \exists r.B \sqsubseteq^? \exists r.X \sqcap \exists r.Y$ the solutions $\sigma := \{X \mapsto A, Y \mapsto B\}$ and $\tau := \{X \mapsto B, Y \mapsto A\}$ form an i-complete set. However, this set is not s-complete since for the solution $\delta := \{X \mapsto A, Y \mapsto A\}$ we have $\sigma \not\sqsubseteq_s \delta$ and $\tau \not\sqsubseteq_s \delta$.

Lemma 39 guarantees that it is sufficient to compute s-complete sets of matchers. From these sets one can derive minimal i-complete sets which exactly contain for every i-equivalence class of i-minimal matchers one representative.

Computing s-complete sets in $\mathcal{EL}$

The matching algorithm presented in the following for computing s-complete sets is an extension of the one computing s-complete sets for matching modulo subsumption in that it computes the complete sets for matching modulo equivalence. By Lemma 9 this is a more general problem. However, when deleting the last line of the algorithm in Figure 3, one obtains the algorithm for matching modulo subsumption. We shall come back to this point later on.

```
Input: $\mathcal{EL}$-matching problem $C \equiv^? D$
Output: s-complete set $\mathcal{C}$ of matchers for $C \equiv^? D$

Compute $\mathcal{G}(C)$ and $\mathcal{G}(D) = (V, E, w_0, \ell)$
$\mathcal{C} := \emptyset$
For all homomorphisms $\varphi$ from $\mathcal{G}(D)$ into $\mathcal{G}(C)$
   Define $\sigma$ by $\sigma(X) := lcs(C_{(\mathcal{G}(C))_{\varphi(w)}} \mid X \in \ell(w))$
     for all variables $X$ in $D$
  If $C \sqsupseteq \sigma(D)$ then $\mathcal{C} := \mathcal{C} \cup \{\sigma\}$
```

Figure 3: The $\mathcal{EL}$-matching algorithm.

The matching algorithm described in Figure 3 first tries to construct substitutions σ such that $C \sqsubseteq \sigma(D)$, i.e., there is a homomorphism from $\mathcal{G}(\sigma(D))$ into $\mathcal{G}(C)$. In a second step, it checks which of the computed substitutions really solve the matching problem, i.e., also satisfies $C \sqsupseteq \sigma(D)$. (As already mentioned, for a matching problem modulo subsumption, this second step can be dispensed with.) The first step is achieved by first computing all homomorphisms from $\mathcal{G}(D)$ into $\mathcal{G}(C)$. The remaining problem is that a variable X may occur more than once in D. Thus, we cannot simply define $\sigma(X)$ as $C_{(\mathcal{G}(C))_{\varphi(w)}}$ where w is such that X occurs in the label of w. Since there may exist several nodes w with this property, we take the lcs of the corresponding subconcepts of C. The reason for taking the *least* common subsumer is that we want to compute substitutions that are as small as possible w.r.t. $\sqsubseteq_s$. An algorithm for computing the lcs of $\mathcal{EL}$-concepts has been described in [3].

Before proving the soundness of our matching algorithm, we illustrate the algorithm by the example depicted in Figure 2.

There are six homomorphisms from $\mathcal{G}(D_{\text{ex}})$ into $\mathcal{G}(C_{\text{ex}})$. We consider the ones mapping w_i onto v_i for $i = 0, 1, 2$, and w_3 onto v_3 or w_3 onto v_4, which we denote by φ_1 and φ_2, respectively. The homomorphism φ_1 yields the substitution τ_1:

$$\begin{array}{rcccc}\tau_1(X) & := & lcs\{C_{\mathcal{G}(C_{\text{ex}})_{v_1}}, C_{\mathcal{G}(C_{\text{ex}})_{v_2}}\} & \equiv & \mathsf{W} \sqcap \exists \mathsf{hc}.(\mathsf{W} \sqcap \mathsf{P}), \\ \tau_1(Y) & := & lcs\{C_{\mathcal{G}(C_{\text{ex}})_{v_2}}, C_{\mathcal{G}(C_{\text{ex}})_{v_3}}\} & \equiv & \mathsf{W} \sqcap \mathsf{D},\end{array}$$

whereas φ_2 yields the substitution τ_2:

$$\begin{array}{rcccc}\tau_2(X) & := & lcs\{C_{\mathcal{G}(C_{\text{ex}})_{v_1}}, C_{\mathcal{G}(C_{\text{ex}})_{v_2}}\} & \equiv & \mathsf{W} \sqcap \exists \mathsf{hc}.(\mathsf{W} \sqcap \mathsf{P}), \\ \tau_2(Y) & := & lcs\{C_{\mathcal{G}(C_{\text{ex}})_{v_2}}, C_{\mathcal{G}(C_{\text{ex}})_{v_4}}\} & \equiv & \mathsf{W}.\end{array}$$

For τ_1 the test $C \sqsupseteq \tau_1(D)$ is successful, but for τ_2 the test fails. Therefore, only τ_1 belongs to the computed set $\mathcal{C}$. In fact, the last test also fails for the substitution computed for the remaining four homomorphisms.

Soundness of the $\mathcal{EL}$-matching algorithm

For the set $\mathcal{C}$ computed by the matching algorithm (Figure 3) we need to verify two properties. First, we have to show that the substitutions in $\mathcal{C}$ are matchers for the given matching problem. Second, we have to prove that $\mathcal{C}$ is s-complete, i.e., for every matcher σ' of $C \equiv^? D$ there is a matcher $\sigma \in \mathcal{C}$ such that $\sigma \sqsubseteq_s \sigma'$. We will break down the proof in two lemmas. The first lemma is also useful to specify an optimized algorithm for matching problems modulo subsumption. It says that for every substitution σ computed by the algorithm we have $C \sqsubseteq \sigma(D)$.

Lemma 40 Let φ be a homomorphism from $\mathcal{G}(D) = (V, E, r, \ell)$ into $\mathcal{G}(C) = (V', E', r', \ell')$ and let σ be the corresponding substitution as specified by the matching algorithm in Figure 3. Then, $C \sqsubseteq \sigma(D)$.

Proof. Let X be a variable in D and let $V_D(X)$ be defined as in Section 3.3. Then, for every $w \in V_D(X)$ there exists a node $v \in V'$ with $\varphi(w) = v$. By the definition of σ it follows $C_{\mathcal{G}(C)_v} \sqsubseteq \sigma(X)$. According to Theorem 20 there exists a homomorphism from $\mathcal{G}(\sigma(X))$ into $\mathcal{G}(C)_v$.

Obviously, one obtains $\mathcal{G}(\sigma(D))$ by instantiating $\mathcal{G}(D)$ for every variable X and every node $w \in V_D(X)$ by (a new copy of) $\mathcal{G}(\sigma(X))$. With that, it is easy to see that φ can be extended to an homomorphism from $\mathcal{G}(\sigma(D))$ into $\mathcal{G}(C)$. By Theorem 20, this shows $C \sqsubseteq \sigma(D)$. ∎

Now, let σ' be a matcher for $C \equiv^? D$. This implies $C \sqsubseteq \sigma'(D)$. By Theorem 20, there is a homomorphism φ' from $\mathcal{G}(\sigma(D'))$ into $\mathcal{G}(C)$. When deleting the variables in $\mathcal{G}(D)$ then $\mathcal{G}(D)$ is a subtree of $\mathcal{G}(\sigma'(D))$. Thus, restricting φ' to the nodes of $\mathcal{G}(D)$ yields a homomorphism φ from $\mathcal{G}(D)$ into $\mathcal{G}(C)$. For all variables X in D let $\sigma(X) := lcs(C_{\mathcal{G}(C)_{\varphi(w)}} \mid X \in \ell(w))$ be defined as specified by the matching algorithm in Figure 3.

Lemma 41 $\sigma \sqsubseteq_s \sigma'$.

Proof. We have to verify $\sigma(X) \sqsubseteq \sigma'(X)$ for every variable X in D. Let X be a variable in D, $V_D(X)$ be the set of all nodes in $\mathcal{G}(D)$ with X in their labels (see also Section 3.3), $\mathcal{G}(D) = (V, E, r, \ell)$, $w \in V_D(X)$, and $\varphi(w) = v$. Restricting φ' to the description tree $\mathcal{G}(\sigma'(D))_w$ provides us with a homomorphism from $\mathcal{G}(\sigma'(D))_w$ into $\mathcal{G}(C)_v$. Since $X \in \ell(w)$, $\mathcal{G}(\sigma'(D))_w$ contains a subtree corresponding to $\sigma'(X)$. Consequently, there is a homomorphism from $\mathcal{G}(\sigma'(X))$ into $\mathcal{G}(C)_v$, which shows $C_{\mathcal{G}(C)_v} \sqsubseteq \sigma'(X)$. Thus, $\sigma(X) \sqsubseteq \sigma'(X)$. ∎

With these two lemmas at hand the soundness of the matching algorithm can be derived as follows: If $\sigma \in \mathcal{C}$ is a substitution computed by the matching algorithm, then by Lemma 40 we know $C \sqsubseteq \sigma(D)$. But then, the subsumption test $C \sqsupseteq \sigma(D)$ in the matching algorithm ensures $C \equiv \sigma(D)$ which shows that σ is a matcher of the matching problem $C \equiv^? D$.

Now, let σ' be some matcher for $C \equiv^? D$ and let σ be defined as specified above Lemma 41. From Lemma 41 we know $\sigma \sqsubseteq_s \sigma'$. Thus, by Lemma 5 we can conclude $\sigma(D) \sqsubseteq \sigma'(D)$. According to the definition, σ is computed by the matching algorithm in Figure 3. But then, Lemma 40 implies $C \sqsubseteq \sigma(D)$. Thus, we have $\sigma(D) \sqsubseteq \sigma'(D) \equiv C \sqsubseteq \sigma(D)$ which means $C \equiv \sigma(D)$. Hence, σ belongs to the set $\mathcal{C}$ computed by the matching algorithm. Furthmore, the fact $\sigma \sqsubseteq_s \sigma'$ shows that $\mathcal{C}$ is s-complete. To sum up,

Theorem 42 For a matching problem modulo equivalence, the set $\mathcal{C}$ computed by the algorithm depicted in Figure 3 is s-complete.

Furthermore, note that the Lemmas 40 and 41 do not make use of the fact that the algorithm in Figure 3 checks $C \sqsupseteq \sigma(D')$. Also, Lemma 41 only uses the fact that σ' satisfies $C \sqsubseteq \sigma'(D)$. As a result, we obtain

Theorem 43 For a matching problem modulo subsumption, the algorithm depicted in Figure 3 when deleting the last line computes an s-complete set $\mathcal{C}$ of matchers.

Complexity of computing s-complete and i-complete sets

We now investigate both the size of (minimal) s-complete (i-complete) sets as well as the complexity of the matching algorithm in Figure 3.

We start by considering the cardinality of s-complete and i-complete sets.

Example 44 Let C be the $\mathcal{EL}$-concept description

$$\bigsqcap_{i=1}^{n} \exists r.(\bigsqcap_{j=1}^{n} \exists r.(A_i \sqcap B_j)$$

and D be the $\mathcal{EL}$-concept pattern

$$\bigsqcap_{i=1}^{n} \exists r.\exists r.X_i.$$

For the $\mathcal{EL}$-matching problem $C \sqsubseteq^? D$ and a word $w := a_1 \cdots a_n \in \{1, \dots, n\}^n$ of length n over the alphabet $\{1, \dots, n\}$, the substitution $\sigma_w(X_i) := A_i \sqcap B_j$ for $a_i = j$ is obviously an i-minimal and s-minimal matcher. Furthermore, for different words one obtains i-incomparable and s-incomparable matchers. Also, for the problem $C \equiv^? C \sqcap D$ the σ_w's are s-incomparable, s-minimal matchers. Since there are n^n such words, the example shows

Corollary 45
1. For $\mathcal{EL}$-matching problems modulo equivalence the cardinality of a (minimal) s-complete set of matchers might grow exponentially in the size of the matching problem.
2. For $\mathcal{EL}$-matching problems modulo subsumption the cardinality of (minimal) s-complete and i-complete sets of matchers might grow exponentially in the size of the matching problem.

Note that, by definition, a minimal i-complete set for matching problems modulo equivalence contains at most one matcher.

The next example shows that even the size of matchers in s-complete and i-complete sets can grow exponentially in the size of the matching problem.

Example 46 In [3], it has been shown that the size of the lcs of a sequence $C_1, \dots, C_n$ of $\mathcal{EL}$-concept descriptions can grow exponentially in the size of the sequence. Let C be the $\mathcal{EL}$-concept description

$$\exists r_1.C_1 \sqcap \cdots \sqcap \exists r_n.C_n$$

and D be the $\mathcal{EL}$-concept pattern

$$\exists r_1.X \sqcap \cdots \sqcap \exists r_n.X$$

Clearly, for an i-minimal or s-minimal matcher σ of the matching problem $C \sqsubseteq^? D$, $\sigma(X) \equiv lcs(C_1, \dots, C_n)$. This still holds true for s-minimal matchers of the problem $C \equiv^? C \sqcap D$. But then, the result in [3] shows

Corollary 47
1. For $\mathcal{EL}$-matching problems modulo equivalence the size of s-minimal matchers might grow exponentially in the size of the matching problem.
2. For $\mathcal{EL}$-matching problems modulo subsumption the size of s-minimal and i-minimal matchers might grow exponentially in the size of the matching problem.

Using the algorithm in Figure 3, we can prove matching upper bounds for the size of s- and i-complete sets: The number of mappings from a description tree $\mathcal{G}(D)$ into $\mathcal{G}(C)$ is exponential in the size of the description trees. Since the size of these trees is linear in the size of the matching problem $C \equiv^? D$ ($C \sqsubseteq^? D$) we can conclude that the cardinality of an s-complete (i-complete) set of matchers computed by our matching algorithm is at most exponential in the size of the matching problem. Moreover, as shown in [3], the size of the lcs of a sequence of $\mathcal{EL}$-concept descriptions can exponentially be bounded in the size of the sequence. Thus, the size of every substitution computed by the matching algorithm is at most exponential in the size of the matching problem. Finally, as shown in Section 3.4, every s-complete set is also i-complete. To sum up, we obtain the following upper bounds for the size of s- and i-complete sets.

Corollary 48
1. For $\mathcal{EL}$-matching problems modulo equivalence
   (a) the cardinality of a (minimal) s-complete set of matchers can exponentially be bounded in the size of the matching problem; and
   (b) the size of s-minimal matchers can exponentially be bounded in the size of the matching problem.
2. For $\mathcal{EL}$-matching problems modulo subsumption
   (a) the cardinality of (minimal) s-complete and i-complete sets of matchers can exponentially be bounded in the size of the matching problem; and
   (b) the size of s-minimal and i-minimal matchers can exponentially be bounded in the size of the matching problem.

Note that, by Theorem 29, the size of the matcher in a minimal i-complete set for matching problems modulo equivalence can polynomially be bounded in the size of the matching problem.

We now exploit the complexity of the algorithm in Figure 3 (and its variant for matching modulo subsumption) itself. Subsumption of $\mathcal{EL}$-concept descriptions can be decided by a polynomial time algorithm [3]. As already mentioned, the size of a substitution σ computed by our matching algorithm is at most exponential in the size of the matching problem. Thus, $C \sqsupseteq \sigma(D)$ can be decided in time exponential in the size of the matching problem $C \equiv^? D$. Since the lcs of a sequence of $\mathcal{EL}$-concept description can be computed in time exponential in the size of the sequence [3], it is easy to see that the loop body of the algorithm in Figure 3 runs in exponential time. As mentioned above, there exists only an exponential number of mappings from $\mathcal{G}(D)$ into $\mathcal{G}(C)$. For these mappings it can be decided in time polynomial in the size of the matching problem if they are homomorphisms. Consequently, our matching algorithm runs in time exponential in the size of the matchng problem. Moreover, for an s-complete set computed by the algorithm one can obtain a minimal s-complete (i-complete) set in time exponential in the size of the matching problem using the subsumption algorithm for $\mathcal{EL}$-concept descriptions.

Corollary 49 A (minimal) s-complete and i-complete set of matchers for an $\mathcal{EL}$-matching problem (both modulo equivalence and modulo subsumption) can be computed in time exponential in the size of the matching problem.

3.6 Computing d-minimal matchers in $\mathcal{EL}$

In this section, we show how d-complete sets of $\mathcal{EL}$-matching problems modulo equivalence can be computed, where *d-complete sets* are those sets containing (at least) all d-minimal matchers up to s-equivalence. A *minimal d-complete set* contains exactly all d-minimal matchers up to s-equivalence. Note that the notion of a complete set introduced here differs from the one used in Section 3.5 in that d-minimal matchers are not computed modulo d-equivalence. The reason is that d-equivalent matchers syntactically coincide, but clearly we do not want to distinguish matchers that are equal modulo commutativity and assoziativity of concept conjunction. Therefore, s-equivalence is the appropriate ordering to use.

In order to compute d-complete sets, we proceed as follows: First, we show that reduced matchers can be characterized by means of s-maximal matchers. Then, we propose an algorithm for computing s-maximal matchers. Finally, we investigate the complexity of the algorithm.

S-maximal and reduced matchers

We prove the following theorem.

Theorem 50 Let $C \equiv^? D$ be an $\mathcal{EL}$-matching problem. Then, a matcher σ of this problem is reduced if and only if σ is s-maximal and $\sigma(X)$ is reduced for all variables X in D.

The if direction of this theorem is pretty obvious. The main observation to use is that for $\mathcal{EL}$-concept descriptions E, F, $E \preceq_d F$ implies $F \sqsubseteq E$, which is easy to verify by the definition of subdescription. Now, let σ be an s-maximal matcher with the images of the variables reduced. Assume that σ is not reduced. Then, there exists a matcher $\tau \prec_d \sigma$. By the observation, we know $\tau \sqsupseteq_s \sigma$. Furthermore, there exists a variable X such that $\tau(X) \prec_d \sigma(X)$. Since $\sigma(X)$ is reduced, by Theorem 25, we can conclude $\tau(X) \sqsupset \sigma(X)$. Therefore, $\tau \sqsupset_s \sigma$, which is a contradiction to the fact that σ is an s-maximal matcher and completes the proof of the if direction.

The only-if direction is more involved. Let σ be a reduced matcher. Then, by definition of reduced matchers the images of the variables are reduced. It remains to show that σ is s-maximal. We lead the assumption that σ' is a matcher with $\sigma' \sqsupset_s \sigma$ to a contradiction.

The problem is that for $\mathcal{EL}$-concept descriptions E, F, $E \sqsubseteq F$ does not necessarily imply $F \preceq_d E$, e.g., $E := \exists r.(A \sqcap B)$ and $F := \exists r.A \sqcap \exists r.B$.Otherwise one could immediately lead the assume $\sigma' \sqsupset_s \sigma$ to a contradiction.

From $\sigma' \sqsupset_s \sigma$ it follows that there exists a variable X in D with $\sigma'(X) \sqsupset \sigma(X)$. Let $\sigma''(X) := \sigma'(X)$ and $\sigma''(Y) := \sigma(Y)$ for all $Y \neq X$. Then, $C \equiv \sigma(D) \sqsubseteq \sigma''(D) \sqsubseteq \sigma'(D) \equiv C$. Thus, σ'' is a matcher with $\sigma' \sqsupset_s \sigma$ as well. Therefore, we may assume that σ and σ' coincide on all variables but X. The core of the proof is to show that for $\sigma(X)$ and $\sigma'(X)$ the phenomenon shown for E, F cannot occur. For that reason, we modify $\sigma'(X)$ in such a way that eventually the difference of $\sigma(X)$ and $\sigma'(X)$ corresponds to the one for E and F, which then leads to a contradiction.

Because of $\sigma'(X) \sqsupset \sigma(X)$ we know that $prim(\sigma(X)) \subseteq prim(\sigma'(X))$ (see Section 3.2 for the definition of $prim$). We claim $prim(\sigma(X)) = prim(\sigma'(X))$. If a concept name P is in $prim(\sigma(X))$ but not in $prim(\sigma'(X))$, then one could delete P in $\sigma(X)$ which yields a substitution σ'' with $C \equiv \sigma(D) \sqsubseteq \sigma''(D) \sqsubseteq \sigma'(D) \equiv C$, i.e., σ'' is a matcher, and $\sigma''(X) \prec_d \sigma(X)$. A contradiction to the fact that σ is reduced.

Then, since $\sigma'(X) \sqsupset \sigma(X)$ it follows from Observation 22 that there exists an existential restriction $\exists r.F \in \sigma(X)$ with $\sigma'(X) \not\sqsubseteq \exists r.F$. By an argument similar to the one above, we therefore may assume that $\sigma'(X)$ and $\sigma(X)$ coincide on all existential restrictions but the ones for r. In the following, let $\exists r.F_1', \dots, \exists r.F_m'$ be the existential restrictions for r on the top-level of $\sigma'(X)$ and $\exists r.F_1, \dots, \exists r.F_n$ the ones for $\sigma(X)$.

Since $\sigma'(X) \sqsupset \sigma(X)$, by Observation 22 there exists a mapping φ from $\{1, \dots, m\}$ into $\{1, \dots, n\}$ such that for all $i \in \{1, \dots, m\}$, $F_{\varphi(i)} \sqsubseteq F_i'$. We claim that φ is surjective. If φ is not surjective, then there exists a $j \in \{1, \dots, n\}$ with $j \notin image(\varphi)$. Let σ'' be the substitution obtained from σ by deleting the existential restrictions $\exists r.F_j$ in $\sigma(X)$. Then, similar to the previous argument for $prim$, one can conclude that σ'' is still a matcher of $C \equiv^? D$, which is a contradiction to the fact that σ is reduced and $\sigma'' \prec_d \sigma$. Thus, φ must be surjective.

Let k be the element in $\{1, \dots, n\}$ with $F_k = F$. For $i \in \{1, \dots, m\}$ and $\varphi(i) = k$ it is $F_k \sqsubset F'_i$ since $\sigma'(X) \not\sqsubseteq \exists r.F$. Let $\varphi^{-1}(k) =: \{k_1, \dots, k_s\}$ be the set of all elements in $\{1, \dots, m\}$ that are mapped on k by φ. Since φ is surjective this set is not empty. Let σ'' by the substitution that coincides with σ except that in $\sigma''(X)$ the existential restriction $\exists r.F_k$ is replaced by $\exists r.F'_{k_1} \sqcap \cdots \sqcap \exists r.F'_{k_s}$. Then, since $\sigma(X) \sqsubseteq \sigma''(X) \sqsubseteq \sigma'(X)$ it follows that σ'' is a matcher. Furthermore, $\sigma \sqsupset_s \sigma''$. Therefore, we may assume σ' to be σ''. Note that for σ' thus obtained the relation between $\sigma'(X)$ and $\sigma(X)$ exactly corresponds to the one between E and F described above. We will now show that this contradicts to the fact that σ is a matcher.

Since $C \equiv \sigma'(D)$ there is an homomorphism ψ from $\mathcal{G}(C \setminus \top)$ into $\mathcal{G}(\sigma'(D))$. Let $V_D(Y)$ and $\mathcal{G}_{\sigma'(Y),w}$ be defined as in Section 3.3. As usual, $\mathcal{G}(\sigma'(D))$ is considered to be the instantiation of $\mathcal{G}(D)$ by $\mathcal{G}_{\sigma'(Y),w}$ for every variable Y and node $w \in V_D(Y)$.

The following lemma says that none of the existential restrictions $\exists r.F'_{k_j}$ can be deleted from $\sigma'(X)$ in order to guarantee that ψ is a homomorphism from $\mathcal{G}(C \setminus \top)$ into $\mathcal{G}(\sigma'(D))$. For $w \in V_D(X)$ and X defined above, let $w_1, \dots, w_s$ be the direct successors of w in $\mathcal{G}(\sigma'(D))$ with $C_{(\mathcal{G}_{\sigma'(X),w})_{w_j}} = F'_{k_j}$ (modulo commutativity and assoziativity of concept conjunction).

Lemma 51 There exists a node $w \in V_D(X)$, $j \in \{1, \dots, s\}$, and $v' \in \mathcal{G}(C \setminus \top)$ with $\psi(v') = w_j$.

Assume that there does not exist such a w. Let σ'' be the substitution obtained by deleting the existential restrictions $\exists r.F'_{k_i}$ in $\sigma'(X)$. Then, $\sigma(X) \sqsubseteq \sigma''(X)$. Furthermore, ψ is a homomorphism from $\mathcal{G}(C \setminus \top)$ into $\mathcal{G}(\sigma''(D))$, i.e., $C \sqsupseteq \sigma''(D)$. Because of $\sigma'' \sqsupseteq_s \sigma$, we have $C \equiv \sigma(D) \sqsubseteq \sigma''(D)$. Thus, σ'' is a matcher, which is a contradiction to the fact that σ is reduced and $\sigma'' \prec_d \sigma$. This completes the proof of the lemma.

If, for all $w \in V_D(X)$, the subtrees with root w_i in $\mathcal{G}(\sigma'(D))$ are deleted and replaced by one direct r-successor w' with a subtree corresponding to $F = F_k$, then the resulting tree is the one for $\sigma(D)$. Since $F_{k_i} \sqsupset F$, the nodes v' in $\mathcal{G}(C \setminus \top)$ mapped on some w_i can now be mapped on w' and for successors of v' one can modify ψ in such a way that ψ is a homomorphism form $\mathcal{G}(C \setminus \top)_{v'}$ into $\mathcal{G}(\sigma(D))_{w'}$. With that we have homomorphism from $\mathcal{G}(C \setminus \top)$ into $\mathcal{G}(\sigma(D))$. However, for w, j, v' specified in the lemma above and the r-successor w' we know by construction that $\psi(v') = w'$ and $\mathcal{G}(C \setminus \top)_{v'} \sqsupset \mathcal{G}(\sigma(D))_{w'}$. Then, with Proposition 28 we can conclude that $C \not\equiv \sigma(D)$, a contradiction. This completes the proof of Theorem 50.

By Theorem 50, the task of computing the set of reduced matchers up to s-equivalence, can be split into two subtasks: i) Compute the set of all s-maximal matchers up to s-equivalence; ii) reduce the images of the matchers computed in the first step.

As already mentioned, given an $\mathcal{EL}$-concept description E the corresponding reduced concept description $E \setminus \top$ can be computed in polynomial time in the size of E. Thus, reducing the images of the matchers can be done in time polynomial in the size of the set computed by the first step.

Computing s-maximal matchers

According to Section 3.5, computing all s-maximal matchers up to s-equivalence means to compute minimal s-co-complete sets. In this section, we present an algorithm to compute s-co-complete sets. Given such a sets, a minimal s-co-complete set can easily be computed using the subsumption algorithm for $\mathcal{EL}$-concept descriptions.

In Section 3.3, for a given matcher σ' of $C \equiv^? D$ a matcher σ of size polynomial in the size of the matching problem has been defined with $\sigma' \sqsubseteq_s \sigma$. Consequently, the size of s-maximal matcher can polynomially be bounded. This shows that in order to compute all s-maximal matchers (up to s-equivalence) it is sufficient to compute all matchers of size polynomially bounded in the size of the matching problem and filter out the ones that do not solve the problem or that are not s-maximal. This can be done by an exponential time algorithm.

Apparently, such an algorithm is not of practical use. For that reason, we present an algorithm with a better average case complexity. Roughly speaking, this algorithm for computing s-co-complete sets is the dual version of the one in Figure 3 computing s-complete sets.

The duality occurs in different steps of the algorithms.

- The algorithm in Figure 3 considers homomorphisms from $\mathcal{G}(D)$ into $\mathcal{G}(C)$, whereas now we look at (partial) homomorphisms from $\mathcal{G}(C)$ into $\mathcal{G}(D)$;
- for computing s-complete sets, possible matchers σ are constructed based on the lcs of concepts; now, σ is built from conjunctions of concepts;[4]
- the algorithm in Figure 3 ensures that $\sigma(D) \sqsupseteq C$ and needs to check $\sigma(D) \sqsubseteq C$, whereas now the algorithm guarantees $\sigma(D) \sqsubseteq C$ but checks $\sigma(D) \sqsupseteq C$.

In the sequel, we specify the algorithm for computing s-co-complete sets in detail. The idea behind the algorithm is as follows: As mentioned, we consider certain partial homomorphisms φ from $\mathcal{G}(C)$ into $\mathcal{G}(D)$. The parts of $\mathcal{G}(C)$ not mapped by φ are those which are substituted for the variables in D. In this way, φ can be extended to a total homomorhism from $\mathcal{G}(C)$ into $\mathcal{G}(\sigma(D))$ where σ is the possible matcher constructed in this way. Thus, the construction will guarantee $\sigma(D) \sqsubseteq C$. More precisely, φ and σ will be constructed as follows. If a node n in $\mathcal{G}(C)$ is mapped onto a node n' in $\mathcal{G}(D)$ where the label of n' contains a variable X, then some subtree $\mathcal{G}$ of $\mathcal{G}(C)_n$ need not be mapped by φ. This subtree is then part of one conjunct in the substitution for X. Other conjuncts

[4]Note that in a lattice of concept descriptions induced by the subsumption relation the lcs of concepts is the supremum and the conjunction of concepts the infimum of concepts.

may come from multiple occurrences of X in $\mathcal{G}(D)$. If n' contains more than one variable, then only parts of $\mathcal{G}$ are substituted for one variable. One only needs to make sure that the substitutions for the variables in n' "cover" $\mathcal{G}$.

In order to formally define the algorithm, we need to introduce partial homomorphism from $\mathcal{G}(C)$ into $\mathcal{G}(D)$.

Definition 52 Let $\mathcal{G}(C) = (V_C, E_C, r_C, \ell_C)$ and $\mathcal{G}(D) = (V_D, E_D, r_D, \ell_D)$ be the descriptions trees corresponding to the $\mathcal{EL}$-concept description C and the $\mathcal{EL}$-concept pattern D, respectively. Then, φ is a partial homomorphism from $\mathcal{G}(C)$ into $\mathcal{G}(D)$ iff

- φ is a partial mapping from V_C into V_D;
- $r_C \in \mathit{dom}(\varphi)$ and $\varphi(r_C) = r_D$;
- $v \notin \mathit{dom}(\varphi)$ implies that for all successors w of v, $w \notin \mathit{dom}(\varphi)$;
- If $w \in \mathit{dom}(\varphi)$, $v \notin \mathit{dom}(\varphi)$ and $wrv \in E_C$ for some role r, then $\ell_D(\varphi(w))$ contains a variable;
- for all $v, w \in \mathit{dom}(\varphi)$, $vrw \in E_C$ implies $\varphi(v)r\varphi(w) \in E_D$;
- If $v \in \mathit{dom}(\varphi)$ and $v \notin V_\varphi$, then $\ell_C(v) \subseteq \ell_D(\varphi(v))$ where $V_\varphi := \{v \in \mathit{dom}(\varphi) \mid \ell_D(\varphi(v))$ contains a variable$\}$.

For the matching problem $C_{\text{ex}} \equiv^? D_{\text{ex}}$, $\varphi := \{v_0 \mapsto w_0, v_1 \mapsto w_1, v_2 \mapsto w_2, v_3 \mapsto w_3\}$, where $\varphi(v_4)$ and $\varphi(v_5)$ are undefined, is an example of a partial homomorphism from $\mathcal{G}(C)$ into $\mathcal{G}(D)$.

In order to specify the algorithm, we need some more notation. With the notation introduced in the definition above, we define for every $v \in V_\varphi$ the subtree $\mathcal{G}_v = (V_v, E_v, v, \ell_v)$ of $\mathcal{G}(C)$ containing those parts of $\mathcal{G}(C)_v$ that need to be covered by the variables in $\varphi(v)$:

- $V_v := \{w \mid w$ is a successor[5] of v in $\mathcal{G}(C)$ with $w \notin \mathit{dom}(\varphi)\} \cup \{v\}$;
- $E_v := E_C \cap (V_v \times N_R \times V_v)$;
- $\ell_v(w) := \ell_C(w)$ for all $w \in V_v \setminus \{v\}$ and $\ell_v(v) := \ell_C(v) \setminus \ell_D(\varphi(v))$.

Furthermore, for a variable X we need the subset $V_\varphi^X := \{v \in \mathit{dom}(\varphi) \mid X \in \ell_D(\varphi(v))\}$ of V_φ.

We can now specify the algorithm computing s-co-complete sets in Figure 4. Of course, the algorithm must compute every possible guess $\mathcal{G}_v^X$. Thus, in general, for one partial homomorphism several possible matchers σ are computed.

The assumption that C is reduced has been made for two reasons: First, it simplifies the soundness proof of the algorithm. Second, it reduces the number of homomorphisms, and thus, the number of matchers in $\mathcal{C}$.

Before proving the soundness of the algorithm in Figure 4, we illustrate the algorithm by our example $C_{\text{ex}} \equiv^? D_{\text{ex}}$. Let φ be the partial homomorphism

[5]See the definition in Section 3.1.

```
Input: $\mathcal{EL}$-matching problem $C \equiv^? D$ with $C$ reduced
        (otherwise take $C \setminus \top$)
Output: s-co-complete set $\mathcal{C}$ of matchers for $C \equiv^? D$

Compute $\mathcal{G}(C)$ and $\mathcal{G}(D)$
$\mathcal{C} := \emptyset$
For all partial homomorphisms $\varphi$ from $\mathcal{G}(C)$ into $\mathcal{G}(D)$
  For all tuples $(v, X)$, $v \in V_\varphi^X$ and $X$ variable in $D$
    guess subtrees $\mathcal{G}_v^X$ of $\mathcal{G}_v$ such that $\mathcal{G}_v \not\sqsupseteq \bigsqcap_{v \in V_\varphi^X}^{X} C_{\mathcal{G}_v^X}$
  For all variables $X$ define $\sigma(X) := \bigsqcap_{v \in V_\varphi^X} C_{\mathcal{G}_v^X}$
  If $\sigma(D) \sqsupseteq C$, then $\mathcal{C} := \mathcal{C} \cup \{\sigma\}$.
```

Figure 4: An algorithm for computing s-co-complete sets.

already defined. Then, $\mathcal{G}_{v_1} \equiv \mathsf{W} \sqcap \exists \mathsf{hc}.(\mathsf{W} \sqcap \mathsf{P})$, $\mathcal{G}_{v_2} \equiv \mathsf{W} \sqcap \mathsf{D} \sqcap \exists \mathsf{hc}.(\mathsf{W} \sqcap \mathsf{P})$, and $\mathcal{G}_{v_3} \equiv D$. Thus, the algorithm can choose the concept descriptions $C_{\mathcal{G}_v^X}$ as follows: $C_{\mathcal{G}_{v_1}^X} \equiv C_{\mathcal{G}_{v_2}^X} \equiv \mathsf{W} \sqcap \exists \mathsf{hc}.(\mathsf{W} \sqcap \mathsf{P})$, $C_{\mathcal{G}_{v_2}^Y} \equiv D$, and $C_{\mathcal{G}_{v_3}^Y} \equiv D$. It is easy to check that $\mathcal{G}_{v_1} \equiv C_{\mathcal{G}_{v_1}^X}$, $\mathcal{G}_{v_2} \equiv C_{\mathcal{G}_{v_2}^X} \sqcap C_{\mathcal{G}_{v_2}^Y}$, and $\mathcal{G}_{v_3} \equiv C_{\mathcal{G}_{v_3}^Y}$. Finally, the subsitution σ defined as specified in the algorithm satisfies $\sigma(D) \sqsupseteq C$. Thus, σ is a matcher in the computed s-co-complete set.

Soundness of the algorithm. Let $C \equiv^? D$ be an $\mathcal{EL}$-matching problem where C is reduced. We need to show (i) that every substitution $\sigma \in \mathcal{C}$ computed by the algorithm solves $C \equiv^? D$, and (ii) that $\mathcal{C}$ is indeed s-co-complete, i.e., for every matcher σ' of $C \equiv^? D$ there exists a matcher $\sigma \in \mathcal{C}$ with $\sigma \sqsupseteq_s \sigma'$.

In order to show (i), let $\sigma \in \mathcal{C}$. By definition of the algorithm, we know $\sigma(D) \sqsupseteq C$. Assume that σ is constructed with respect to the partial homomorhism φ. Now, the idea is to extend φ to a total homomorphism from $\mathcal{G}(C)$ into a tree $\mathcal{G}$ with $\mathcal{G}(D)$ as subtree such that $\mathcal{G} \sqsupseteq \sigma(D)$. The tree $\mathcal{G}$ is defined as follows: for every X and node $v \in V_\varphi^X$ instantiate $\varphi(v)$ in $\mathcal{G}(D)$ by $\mathcal{G}_v^X$. Then, the condition $\mathcal{G}_v \sqsupseteq \bigsqcap_{v \in V_\varphi^X}^{X} C_{\mathcal{G}_v^X}$ ensures that φ can be extended to $\mathcal{G}_v$ for every $v \in V_\varphi$. But then, φ is a total homomorphism from $\mathcal{G}(C)$ into $\mathcal{G}$. Thus, $\mathcal{G}(C) \sqsupseteq \mathcal{G}$. By construction of σ, $\mathcal{G} \sqsupseteq \sigma(D)$, which yields $C \equiv \mathcal{G}(C) \sqsupseteq \sigma(D)$.

To prove (ii), let σ' be a matcher for $C \equiv^? D$. We prove that there exists a run of the algorithm in Figure 4, such $\mathcal{C}$ contains a substitution σ with $\sigma \sqsupseteq \sigma'$.

Let the set $V_D(X)$ and the subtree $\mathcal{G}_{\sigma'(X),w} = (V_{X,w}, E_{X,w}, w, \ell_{X,w})$ of $\mathcal{G}(\sigma'(D))$ for a variables X in D and $w \in V_D(X)$ be defined as in Section 3.3. Since C is reduced we know that there exists an injective homomorphism ψ from $\mathcal{G}(C) = (V_C, E_C, r_C, \ell_C)$ into $\mathcal{G}(\sigma'(D))$(Theorem 20 and Proposition 28). We define a partial homomorphism φ from $\mathcal{G}(C)$ into $\mathcal{G}(D)$ as follows:

- $dom(\varphi) := V_C \setminus \left(\bigcup_{\substack{X,w \\ w \in V_D(X)}} \psi^{-1}(V_{X,w} \setminus \{w\}) \right)$;
- $\varphi(v) := \psi(v)$ for all $v \in dom(\varphi)$.

It is easy to verify that φ is a partial homomorphism from $\mathcal{G}(C)$ into $\mathcal{G}(D)$. Then, for $v \in V^X_\varphi$ we define $\mathcal{G}^X_v$ to be the description tree $\psi^{-1}(\mathcal{G}_{\sigma'(X),\varphi(v)})$ where we eliminate the atomic concepts from the label of the root v that are in $\ell_D(\varphi(v))$. We claim

$$\mathcal{G}_v \equiv \bigsqcap_{\substack{X \\ v \in V^X_\varphi}} \mathcal{G}^X_v$$

Proof of the claim: By definition of φ, we know that $\mathcal{G}_v$ is obtained by merging the trees $\psi^{-1}(\mathcal{G}_{\sigma'(X),\varphi(v)})$ for every X with $v \in V^X_v$ where again in the label of v the atomic concepts in $\ell_D(\varphi(v))$ are eliminated. Thus, $\mathcal{G}_v$ is obviously equivalent to the conjunction as stated above.

Now, in the algorithm $\sigma(X) := \bigsqcap_{v \in V^X_\varphi} \mathcal{G}^X_v$. Let $\tau(X)$ be the conjunction defined above Lemma 30. By construction, $\sigma(X) \sqsupseteq \tau(X)$. By Lemma 30, $\tau(X) \sqsupseteq \sigma'(X)$, and thus, $\sigma(X) \sqsupseteq \sigma'(X)$. In particular, $\sigma(D) \sqsupseteq \sigma'(D) \equiv C$. This shows that $\sigma \in \mathcal{C}$ and $\sigma \sqsupseteq_s \sigma'$, which completes the proof of the soundness of the algorithm in Figure 4.

Complexity of computing d-complete sets

Apparently, the algorithm depicted in Figure 4 runs, like the naïve one, in exponential time. Summing up the previous results, we obtain

Corollary 53 For an $\mathcal{EL}$-matching problem modulo equivalence a (minimal) d-complete set can be computed by an exponential time algorithm.

Furthermore, the example $\exists r.A_1 \sqcap \cdots \sqcap \exists r.A_n \equiv^? \exists r.X_1 \sqcap \cdots \exists r.X_n$ shows that (minimal) d-complete sets might grow exponentially in the size of the matching problem.

4 Description trees, homomorphisms, and subsumption in $\mathcal{ALE}$

In this section, we extend the characterization of subsumption presented in Section 3.1 to $\mathcal{ALE}$-concept descriptions. All definitions and results for $\mathcal{ALE}$ presented in this section analogously carry over to $\mathcal{FLE}$. Originally the characterization has been shown in [3] in order to compute least common subsumers of $\mathcal{ALE}$-concept descriptions. In our work, the characterization of subsumption will be employed to compute sets of i-minimal matchers. Furthermore, just as for $\mathcal{EL}$, the characterization is also crucial in many proofs presented in this work. In addition to the results already discussed in [3], we introduce the notion of an image resp. inverse image of a homomorphism and prove some simple properties needed in the subsequent sections.

$\mathcal{ALE}$-description trees

The notion of $\mathcal{EL}$-description trees is extended to $\mathcal{ALE}$ in a straightforward manner.

Definition 54 An *$\mathcal{ALE}$-description tree* is a tree of the form $\mathcal{G} = (V, E, v_0, \ell)$ where

- V is a finite set of *nodes* of $\mathcal{G}$;
- $E \subseteq V \times (N_R \cup \forall N_R) \times V$ if a finite set of *edges* labeled with roles names r ($\exists$-edges) or with $\forall r$ ($\forall$-edges); $\forall N_R := \{\forall r \mid r \in N_R\}$;
- v_0 is the *root* of $\mathcal{G}$;
- ℓ is a *labeling function* mapping the nodes in V to finite sets $\{P_1, \ldots, P_n\}$ where each P_i, $1 \leq i \leq n$, is of one of the following forms: $P_i \in N_C \cup \mathcal{X}$, $P_i = \neg P$ for some $P \in N_C$, or $P_i = \bot$.

The empty label corresponds to the top-concept.

For $v, w \in V$ and $r \in N_R$ we write $\exists$-edges from v to w labeled r as vrw and $\forall$-edges as $v\forall rw$. For the sake of simplicity, we occasionally write $v \in \mathcal{G}$ instead of $v \in V$; $vrw \in \mathcal{G}$ ($v\forall w \in \mathcal{G}$) instead of $vrw \in E$ ($v\forall w \in E$); and $\mathcal{G}(v)$ instead of $\ell(v)$.

A sequence $w_0r_1w_1 \cdots r_nw_n$ is a *path* in $\mathcal{G}$ from w_0 to w_n ($w_0r_1w_1 \cdots r_nw_n \in \mathcal{G}$ for short) iff $w_{i-1}r_iw_i \in \mathcal{G}$ or $w_{i-1}\forall r_iw_i \in \mathcal{G}$ for all $i = 1, \ldots n$. Such a path is called *rooted* in case w_0 is the root of $\mathcal{G}$. The path is called *$\exists$-path* (*$\forall$-path*) iff $w_{i-1}r_iw_i \in \mathcal{G}$ ($w_{i-1}\forall r_iw_i \in \mathcal{G}$) for all $i = 1, \ldots, n$.

For $v \in V$, w is a *direct successor* of v in $\mathcal{G}$ if there exists $r \in N_R$ with $vrw \in E$ or $v\forall rw \in E$; w is a *successor* of v if there exists a path from v to w. Since we allow for empty paths, v is a successor of itself. Analogously, we define *(direct) predecessors.*

A *subtree* $\mathcal{G}'$ of $\mathcal{G}$ is a description tree consisting of a subset of nodes of $\mathcal{G}$. The labels of the nodes in $\mathcal{G}'$ are subsets of the corresponding ones in $\mathcal{G}$; $\mathcal{G}'$ is called *rooted subtree* in case the root of $\mathcal{G}'$ coincides with one for $\mathcal{G}$.

For a node $v \in V$, $\mathcal{G}_v$ denotes the subtree of $\mathcal{G}$ consisting of all successors of v in $\mathcal{G}$. The root of $\mathcal{G}_v$ is v and the labels of the nodes in $\mathcal{G}_v$ coincide with the corresponding ones in $\mathcal{G}$.

Similar to $\mathcal{EL}$, $|\mathcal{G}|$ denotes the size of the description tree $\mathcal{G}$. By *depth*$(\mathcal{G})$ we refer to the maximal length of a rooted path in $\mathcal{G}$.

Also, for $\mathcal{ALE}$-description trees $\mathcal{G}$ and $\mathcal{H}$ (with disjoint sets of nodes) and a node $v \in \mathcal{G}$, *instantiating $\mathcal{G}$ at node v with $\mathcal{H}$* yields an extension $\mathcal{G}' = (V', E', v_0, \ell')$ of $\mathcal{G} = (V, E, v_0, \ell)$ defined as follows: First, the root of $\mathcal{H}$ is replaced by v, which yields the tree $\mathcal{H}' = (V'', E'', v, \ell'')$. Then,

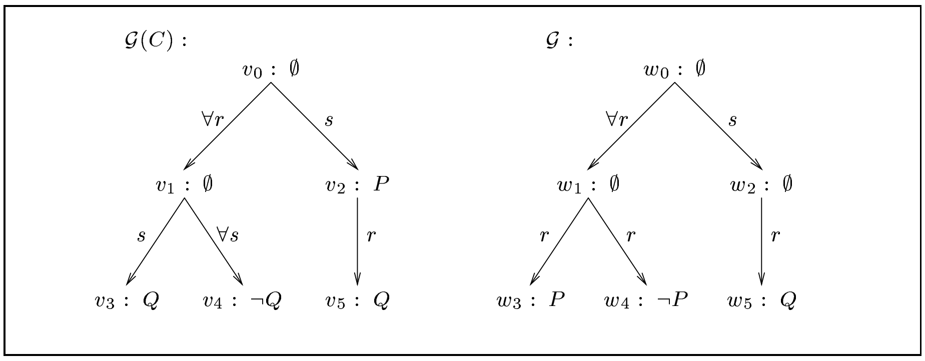

Figure 5: The description trees for $\mathcal{G}(C)$ and $\mathcal{G}$.

- $V' := V \cup V''$;
- $E' := E \cup E''$;
- $\ell'(w) := \ell(w)$ for all $w \in V \setminus \{v\}$; $\ell'(w) := \ell''(w)$ for all $w \in V'' \setminus \{v\}$; $\ell'(v) := \ell(v) \cup \ell(v'')$.

Just as for $\mathcal{EL}$, an $\mathcal{ALE}$-concept description/pattern C can be tranlated into an $\mathcal{ALE}$-description tree $\mathcal{G}(C)$ (see [3] for details). As an example, consider the $\mathcal{ALE}$-concept description

$$C \quad := \quad \forall r.(\exists s.Q \sqcap \forall s.\neg Q) \sqcap \exists s.(P \sqcap \exists r.Q).$$

The corresponding description tree $\mathcal{G}(C)$ is depicted in Figure 5. On the other hand, every $\mathcal{ALE}$-description tree $\mathcal{G}$ without variables in the labels can be translated into an $\mathcal{ALE}$-concept description $C_\mathcal{G}$ ([3] contains a formal translation). The description graph $\mathcal{G}$ in Figure 5 yields the $\mathcal{ALE}$-concept description

$$D \quad := \quad C_\mathcal{G} \quad = \quad \forall r.(\exists r.P \sqcap \exists r.\neg P) \sqcap (\exists s.\exists r.Q).$$

Just as for $\mathcal{EL}$, the semantics of $\mathcal{ALE}$-description trees $\mathcal{G}$ without variables in their labels is defined by the semantics of their corresponding concept descriptions $C_\mathcal{G}$, i.e., $\mathcal{G}^I := C_\mathcal{G}^I$ for an interpretation I. Again, it is easy to see that the translation of concept descriptions and description trees in one another preserves semantics, i.e., $C \equiv C_{\mathcal{G}_C}$. With the formal semantics for description trees, the subsumption relationship can be stated not only between concept description but also between description trees (like $\mathcal{G} \sqsubseteq \mathcal{H}$) or between concept description and description trees (like $C \sqsubseteq \mathcal{G}$) in the obvious way.

Homomorphisms between $\mathcal{ALE}$-description trees

Obviously, a homomorphism between $\mathcal{ALE}$-description trees must take $\forall$-edges into account, which are dealt with like $\exists$-edges. In addition, homomorphisms as introduced for $\mathcal{EL}$-description trees are extended by allowing them to map a node and all its successors onto an inconsistent node, i.e., a node containing $\bot$. Formally, a homomorphism is defined as follows:

Definition 55 A mapping $\varphi : V_H \longrightarrow V_G$ from an $\mathcal{ALE}$-description tree $\mathcal{H} = (V_H, E_H, w_0, \ell_H)$ to an $\mathcal{ALE}$-description tree $\mathcal{G} = (V_G, E_G, v_0, \ell_G)$ is called *homomorphism* if and only if the following conditions are satisfied:

1. $\varphi(w_0) = v_0$,
2. for all $v \in V_H$ we have $(\ell_H(v) \setminus \mathcal{X}) \subseteq \ell_G(\varphi(v))$ or $\bot \in \ell_G(\varphi(v))$,
3. for all $vrw \in E_H$, either $\varphi(v)r\varphi(w) \in E_G$, or $\varphi(v) = \varphi(w)$ and $\bot \in \ell_G(\varphi(v))$, and
4. for all $v\forall rw \in E_H$, either $\varphi(v)\forall r\varphi(w) \in E_G$, or $\varphi(v) = \varphi(w)$ and $\bot \in \ell_G(\varphi(v))$.

An isomorphism between description trees is defined in the obvious way as follows:

Definition 56 Let $\mathcal{G} = (V_G, E_G, v_0, \ell_G)$ and $\mathcal{H}(V_H, E_H, w_0, \ell_H)$ be $\mathcal{ALE}$-description trees. The mapping φ from V_H onto V_G is called *isomorphism* from $\mathcal{H}$ onto $\mathcal{G}$ iff

- φ is a bijection from V_H onto V_G;
- $\varphi(w_0) = v_0$;
- for all $v, w \in V_H$ and $r \in N_R$: $vrw \in E_H$ $(v\forall rw \in E_H)$ iff $\varphi(v)r\varphi(w) \in E_G$ $(\varphi(v)\forall r\varphi(w) \in E_G)$;
- for all $v \in V$: $\ell_H(v) = \ell_G(\varphi(v))$.

Two description trees $\mathcal{G}$ and $\mathcal{H}$ are called *isomorphic* ($\mathcal{G} \cong \mathcal{H}$ for short) if there exists an isomorphism between them.

Similar to $\mathcal{EL}$, the notions of an image and the inverse image of an homomorphism are defined as follows:

Definition 57 Let $\mathcal{G} = (V_G, E_G, v_0, \ell_G)$ and $\mathcal{H} = (V_H, E_H, w_0, \ell_H)$ be $\mathcal{ALE}$-description trees, and let φ be an homomorphism from $\mathcal{H}$ into $\mathcal{G}$. Finally, let $\mathcal{H}' = (V', E', v', \ell')$ be a subtree of $\mathcal{H}$. Then, the *homomorphic image* $\varphi(\mathcal{H}') = (V, E, v, \ell)$ of φ w.r.t. $\mathcal{H}'$ is defined as follows:

- $V := \varphi(V') := \{w \mid \text{ there exists a } w' \in V' \text{ with } w = \varphi(w')\}$;
- $E := E_G \cap (V \times (N_R \cup \forall N_R) \times V)$;
- $v := \varphi(v')$;

- $\ell(w) := \left(\bigcup_{w' \in \varphi^{-1}(w)} \ell'(w') \right) \cap \ell_G(w)$ for all $w \in V$ where $\varphi^{-1}(w) := \{w' \mid \varphi(w') = w\}$.

It is easy to see that

Lemma 58 1. $\varphi(\mathcal{H}')$ is a subtree of $\mathcal{G}$; and

2. φ is an surjective homomorphism from $\mathcal{H}'$ onto $\varphi(\mathcal{H}')$.

Note that in order to guarantee 1. for $\mathcal{EL}$ and $\mathcal{FLE}$, we can dispensed with $\ell_G(w)$ in the definition of $\ell(w)$ (Definition 57). Because of $\perp$ in $\mathcal{ALE}$, however, the first statement in Lemma 58 is only true when restricting the label $\ell(w)$ of w to $\ell_G(w)$.

Definition 59 Let $\mathcal{G} = (V_G, E_G, v_0, \ell_G)$ and $\mathcal{H} = (V_H, E_H, w_0, \ell_H)$ be $\mathcal{ALE}$-description trees, and let $\psi : V_H \longrightarrow V_G$ be a homomorphism from $\mathcal{H}$ into $\mathcal{G}$. Finally, let $\mathcal{G}' = (V', E', v', \ell')$ be some subtree of $\mathcal{G}$ in case ψ is injective, and a rooted subtree in case ψ is not injective. Then, the inverse image of $\mathcal{G}'$ w.r.t. ψ, $\psi^{-1}(\mathcal{G}') = (V, E, v, \ell)$, is defined as follows:

- $V := \psi^{-1}(V')$; if $V = \emptyset$, then let $\psi^{-1}(\mathcal{G}')$ be a description tree containing only the root with empty label; otherwise
- $E := E_H \cap (V \times (N_R \cup \forall N_R) \times V)$;
- $v := \psi^{-1}(v')$;
- $\ell(w) := \ell_H(w) \cap \ell'(\psi(w))$ for all $w \in V$.

We summarize some simple properties of the inverse homomorphism in case $V \neq \emptyset$.

Lemma 60 1. $\psi^{-1}(\mathcal{G}')$ is a subtree of $\mathcal{H}$ with root $\psi^{-1}(v')$.

2. $\psi(\psi^{-1}(\mathcal{G}'))$ is a rooted subtree of $\mathcal{G}'$.

3. In case ψ is injective, $\psi(\psi^{-1}(\mathcal{G}'))$ and $\psi^{-1}(\mathcal{G}')$ are isomorphic.

Characterizing subsumption in $\mathcal{ALE}$

As shown in [3], in order to characterize subsumption of $\mathcal{ALE}$-concept descriptions in terms of homomorphisms between the corresponding description trees, the concept descriptions need to be normalized before translating them into description trees.

Definition 61 Let E, F be two $\mathcal{ALE}$-concept descriptions and $r \in N_R$ a primitive role. The *$\mathcal{ALE}$-normalization rules* are defined as follows

$$\begin{array}{rcl}
\forall r.E \sqcap \forall r.F & \longrightarrow & \forall r.(E \sqcap F) \\
\forall r.E \sqcap \exists r.F & \longrightarrow & \forall r.E \sqcap \exists r.(E \sqcap F) \\
\forall r.\top & \longrightarrow & \top \\
E \sqcap \top & \longrightarrow & E \\
P \sqcap \neg P & \longrightarrow & \bot, \text{ for each } P \in N_C \\
\exists r.\bot & \longrightarrow & \bot \\
E \sqcap \bot & \longrightarrow & \bot
\end{array}$$

A concept descripton C is called *normalized* if none of the above normalization rules can be applied to some subdescription of C. The rules should be read modulo commutativity of conjunction; e.g., $\exists r.E \sqcap \forall r.F$ is also normalized to $\exists r.(E \sqcap F) \sqcap \forall r.F$. An unnormalized concept description C can be normalized by exhaustively applying the normalization rules to subconcepts of C. The resulting (normalized) concept description is called *normal form of* C. Since each normalization rule preserves equivalence, the normal form of C is equivalent to C. We refer to $\mathcal{G}_C$ as the description tree corresponding to the normal form of C, i.e., if C' is the normal form of C then $\mathcal{G}_C := \mathcal{G}(C')$.

If only the rule $\forall r.\top \longrightarrow \top$ is exhaustively applied to a concept description C, then the resulting concept description is called $\top$-*normal form of* C. We refer to $\mathcal{G}_C^\top$ as the description tree corresponding to the $\top$-normal form of C.

Now, subsumption can be characterized in terms of homomorphisms [3]:

Theorem 62 Let C, D be $\mathcal{ALE}$-concept descriptions. Then, $C \sqsubseteq D$ iff there exists a homomorphism from $\mathcal{G}_D^\top$ to $\mathcal{G}_C$.

It should be noted that the theorem stated in [3] requires a homomorphism from $\mathcal{G}_D$ instead of $\mathcal{G}_D^\top$. However, the proof reveals that $\top$-normalization of the subsumer is sufficient. Furthermore, we can conclude from the proof of Theorem 62:

Remark 63 The existence of a homomorphism from a description tree $\mathcal{G}$ into $\mathcal{H}$ implies $\mathcal{H} \sqsubseteq \mathcal{G}$, i.e., for all interpretations I it is $\mathcal{H}^I \subseteq \mathcal{G}^I$. This means that the if-direction of Theorem 62 does not require normalization.

We illustrate the theorem by means of the concept description C and D introduced above. The normal form of C is $\forall r.\bot \sqcap \exists s.(P \sqcap \exists r.Q)$; D is already in $\top$-normal form. A homomorphism from $\mathcal{G}_D^\top$ into $\mathcal{G}_C$ is depicted in Figure 6. By Theorem 62 we can conclude $C \sqsubseteq D$. Observe, however, that there is no homomorphism from $\mathcal{G}(D)$ into $\mathcal{G}(C)$. This shows that the only-if direction of Theorem 62 requires to normalize the concept descriptions before translating them into description trees.

As an easy consequence of Theorem 62, we can derive the following observation. First, we need to introduce the $\forall$-normal form: A concept description is in $\forall$-*normal form*, if the $\forall$-*rule*

$$\forall r.E \sqcap \forall r.F \longrightarrow \forall r.(E \sqcap F)$$

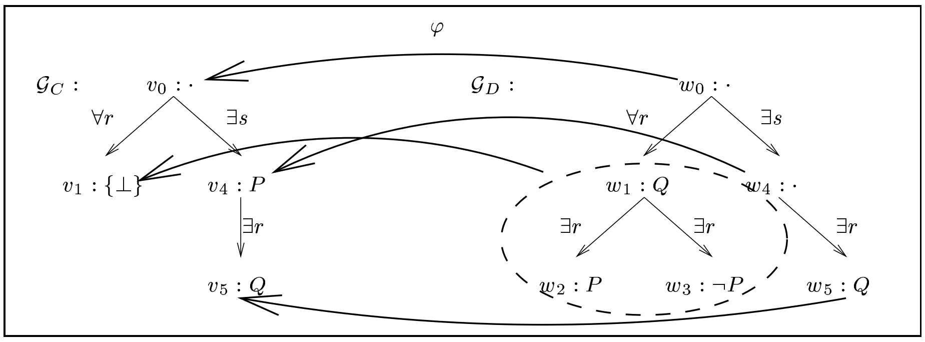

Figure 6: Subsumption for $\mathcal{ALE}$.

cannot be applied anymore. Certainly, every given $\mathcal{ALE}$-concept description can be turned into its $\forall$-normal form in polynomial time. Note that if a concept description C is in $\forall$-normal form, then for every $r \in N_R$ there exists at most one value restriction $\forall r.E$ on top-level of C. We shall refer to E by $C.r$; $C.r := \top$, if C does not contain a value restriction for r. In the sequel, with $\exists r.E \in C$ means that $\exists r.E$ occurs on the top-level of C. Furthermore, let $prim(C)$ be the set of (negated) concept names on the top-level of C; in case $\bot$ occurs on the top-level of C, then $\bot$ also belongs to $prim(C)$.

Observation 64 Let C, D be two $\mathcal{ALE}$-concept descriptions in $\forall$-normal form. Then, $C \sqsubseteq D$ iff i) $C \equiv \bot$ or ii)

1. $prim(C) \subseteq prim(D)$;
2. for every $r \in N_R$, $C.r \sqsubseteq D.r$;
3. for every existential restriction $\exists r.E \in D$, there exists an existential restriction $\exists r.F \in C$ such that $C.r \sqcap F \sqsubseteq D.r \sqcap E$.

5 Equivalence of $\mathcal{ALE}$-concept descriptions

In this section, we generalize the results on reduced $\mathcal{EL}$-concept descriptions (cf. Section 3.2) to $\mathcal{ALE}$. Just as for $\mathcal{EL}$, we will show that for $\mathcal{ALE}$-concept descriptions there exist unique minimal representations. In particular, equivalence of $\mathcal{ALE}$-concept descriptions can be characterized in terms of isomorphisms between the description trees of the minimal representations. As before, this result is important in different respects: First, minimal representations will be used to formalize sets of matchers that do not contain redundancies. Second, we will employ the characterization in order to prove complexity results for deciding the solvability of matching problems. Finally, this result is interesting on its own right in the context of computing minimal rewritings [4]. All results presented in this section, also carry over to $\mathcal{FLE}$ when adjusting the definitions appropriately.

Reduced $\mathcal{ALE}$-concept descriptions

Just as for $\mathcal{EL}$, intuitively, a concept description is called reduced if it does not contain redundancies. We generalize the notion of subdescriptions introduced in Section 3.2 to formally capture this idea.

Definition 65 For an $\mathcal{ALE}$-concept description C, the $\mathcal{ALE}$-concept description $\widehat{C}$ is a *subdescription* of C ($\widehat{C} \preceq_d C$) iff

1. $\widehat{C} = \bot$; or
2. $\widehat{C}$ is obtained from C by removing some (negated) concept names, value restrictions, or existential restrictions on the top-level of C, and for all remaining value/existential restrictions $\forall r.E/\exists r.E$ replacing E by a subdescription of E.

If everthing in C is removed, then the resulting concept description is $\top$. On the other hand, if nothing is removed or replaced by $\bot$, then the resulting concept description is of course C. Clearly, if we are only interested in $\mathcal{FLE}$-concept description, then (1) is not allowed. Given an $\mathcal{FLE}$- or $\mathcal{ALE}$-concept description C, an *$\mathcal{FLE}$-subdescription* of C is a subdescription of C where (1) has not been used to obtain this description.

The concept description $\widehat{C}$ is a *strict ($\mathcal{FLE}$-)subdescription* of C if $\widehat{C} \neq C$. Now we are primed to define reduced concept descriptions analogous to Definition 24.

Definition 66 An $\mathcal{ALE}$-concept description C is *reduced* iff there exists no strict subdescription of C that is equivalent to C.

The following observation gives a first idea of how reduced concepts look like:

1. $\exists r.C \equiv \exists r.C \sqcap \exists r.D$ if $C \sqsubseteq D$;
2. If $\forall r.C \sqcap \exists r.D$ is minimal then D needs to be reduced w.r.t. C.

The first item shows that in reduced concept descriptions there must not be subsumption relationships among existential restrictions. The second item implies that one needs to define reduced concepts with respect to given concepts. For that purpose, we now define what it means for a concept to be E-reduced, i.e., reduced w.r.t. the concept description E. We use the notations $prim(C)$, $C.r$, $\exists r.D \in C$ as introduced in Section 4.

Definition 67 Let F and E be $\mathcal{ALE}$-concept descriptions in $\forall$-normal form. Then, F called *E-reduced* iff the following conditions are satisfied:

1. if $E \sqcap F \equiv \bot$, then $F = \bot$;
2. $prim(E) \cap prim(F) = \emptyset$; every concept name on the top-level of F occurs exactly ones;

3. for all distinct existential restrictions $\exists r.F_1, \exists r.F_2 \in F$: $E.r \sqcap F.r \sqcap F_1 \not\sqsubseteq E.r \sqcap F.r \sqcap F_2$;

4. for all existential restrictions $\exists r.F' \in F$, $\exists r.E' \in E$: $E.r \sqcap F.r \sqcap E' \not\sqsubseteq F'$;

5. for all $r \in N_R$, a) $E.r \not\sqsubseteq F.r$ or b) $F.r = \top$ and there is no value restriction for r on the top-level of F;

6. for all $r \in N_R$, $F.r$ is $E.r$-reduced;

7. for all existential restrictions $\exists r.F' \in F$, F' is $(E.r \sqcap F.r)$-reduced.

In order to show the main theorem of this section, we need two lemmas stating properties of E-reduced concepts. The first lemma, supports the intuition that if a concept description is reduced with respect to some conept, then also with respect to a more general one.

Lemma 68 Let F, E, H be $\mathcal{ALE}$-concept description in $\forall$-normal form. Then, if F is E-reduced and $E \sqsubseteq H$, then F is H-reduced.

Proof. Assume that F is E-reduced and $E \sqsubseteq H$.

i) If $E \sqcap F \equiv \bot$, then $F = \bot$. In this case, $H \sqcap F \equiv \bot$ and F is H-reduced. Now, assume $E \sqcap F \not\equiv \bot$, thus, $H \sqcap F \not\equiv \bot$.

ii) Then we know, $prim(E) \cap prim(F) = \emptyset$. Since $E \equiv H \not\equiv \bot$, we have $prim(H) \subseteq prim(E)$ by Observation 64. Hence, $prim(H) \cap prim(F) = \emptyset$.

iii) Let $r \in N_R$. Then, by Observation 64, $E.r \sqsubseteq H.r$. Assume that there exist distinct existential restrictions $\exists r.F_1, \exists r.F_2 \in F$ with $H.r \sqcap F.r \sqcap F_1 \sqsubseteq H.r \sqcap F.r \sqcap F_2$. Consequently, $E.r \sqcap H.r \sqcap F.r \sqcap F_1 \sqsubseteq E.r \sqcap H.r \sqcap F.r \sqcap F_2$. Now, since $E' \sqsubseteq H'$, we get $E.r \sqcap F.r \sqcap F_1 \sqsubseteq E.r \sqcap F.r \sqcap F_2$ in contradiction to the assumption that F is E-reduced.

iv) Let $r \in N_R$. Assume that there exists a existential restrictions $\exists r.F' \in F$, $\exists r.H' \in H$ with $H.r \sqcap F.r \sqcap H' \sqsubseteq F'$. By Observation 64, we know that there exists $\exists r.E' \in E$ such that $E.r \sqcap E' \sqsubseteq H.r \sqcap H'$. Thus, we obtain $E.r \sqcap F.r \sqcap E' \sqsubseteq F'$ in contradiction to the fact that F is E-reduced.

v) Let $r \in N_R$. Assume $F.r \neq \top$. We know $E.r \not\sqsubseteq F.r$. By Observation 64, it is $E.r \sqsubseteq H.r$. Thus $H.r \not\sqsubseteq F.r$.

Finally, the last two conditions for F to be H-reduced in Definition 66 are satisfied by induction and the fact that $E.r \sqsubseteq H.r$. ∎

Now, we proof that two concepts that are equivalent modulo a given concept and reduced w.r.t. that concept must in fact be equivalent.

Lemma 69 Let C, D, E be $\mathcal{ALE}$-concept descriptions in $\forall$-normal from. Then, if $E \sqcap C \equiv E \sqcap D$ and C, D are E-reduced, then $C \equiv D$.

Proof. If $E \sqcap C \equiv E \sqcap D \equiv \bot$, then we know $C = D = \bot$.

Now, assume $E \sqcap C \equiv E \sqcap D \not\equiv \bot$.

i) We know $prim(C), prim(D)$ are disjoint from $prim(E)$. Then, Observation 64 implies $prim(C) = prim(D)$.

ii) Let $r \in N_R$. We know that $C.r, D.r$ are $E.r$-reduced. Furthermore, by assumption $E.r \sqcap C.r \equiv E.r \sqcap D.r$. By induction, we can conclude $C.r \equiv D.r$.

iii) By Observation 64 we know that for every existential restriction $\exists r.D' \in D$ there a) exists an existential restriction $\exists r.C' \in C$ with $E.r \sqcap D.r \sqcap C' \sqsubseteq E.r \sqcap D.r \sqcap D'$ (recalling $C.r \equiv D.r$) or b) there exists an existential restriction $\exists r.E' \in E$ with $E.r \sqcap D.r \sqcap E' \sqsubseteq E.r \sqcap D.r \sqcap D'$. However, since D is E-reduced, b) cannot occur. Analogously, for $\exists r.C' \in C$ there exists an existential restriction $\exists r.D'' \in D$ such that $E.r \sqcap D.r \sqcap D'' \sqsubseteq E.r \sqcap D.r \sqcap C'$. Since D is E-reduced, we can deduce $D' = D''$. Therefore, $E.r \sqcap D.r \sqcap D' \equiv E.r \sqcap D.r \sqcap C'$. By assumption, D' is $E.r \sqcap D.r$-reduced and C' is $E.r \sqcap C.r$-reduced. By Lemma 68, C' is also $E.r \sqcap D.r$-reduced. By induction, we may conclude $C' \equiv D'$. Analogously, by symmetry, for every existential restriction $\exists r.C' \in C$ there exists an existential restriction $\exists r.D' \in D$ with $D' \equiv C'$.

Now from i), ii), and iii) it follows $C \equiv D$. ∎

Theorem 70 Let C, D, E be $\mathcal{ALE}$-concept descriptions in $\forall$-normal form where C, D are E-reduced. Then, $C \equiv D$ iff $\mathcal{G}(C) \cong \mathcal{G}(D)$.

Proof. The if direction of the statement is obvious. We proceed by proving the only-if direction. If $E \sqcap C \equiv \bot$, then by definition $C = \bot$. As a consequence, $D \equiv \bot$. Thus, $E \sqcap D \equiv \bot$ implies $D = \bot$, which shows $\mathcal{G}(C) \cong \mathcal{G}(D)$.

Now, assume that $E \sqcap C \equiv E \sqcap D \not\equiv \bot$. The proof proceeds by induction on the depths of the quantifiers of C, D and E. We inductively construct an isomorphism from $\mathcal{G}(C)$ onto $\mathcal{G}(D)$.

i) Since $C \equiv D \not\equiv \bot$ we know by Observation 64 that $prim(C) = prim(D)$. Thus, mapping the root of $\mathcal{G}(C)$ to the root of $\mathcal{G}(D)$ is an isomorphism on the roots.

ii) Also, from Observation 64 we can conclude that for every $r \in N_R$, $C.r \equiv D.r$. By definition, $C.r$, $D.r$ are $E.r$-reduced. Thus, the induction step yields $\mathcal{G}(C.r) \cong \mathcal{G}(D.r)$. Consequently, we can extend the isomorphism from i) to the value restrictions.

iii) Again, by Observation 64 we know that for every existential restriction $\exists r.D' \in D$ there exists an existential restriction $\exists r.C' \in C$ such that $D.r \sqcap C' \sqsubseteq D.r \sqcap D'$, recalling $D.r \equiv C.r$. Analogously, for C' there exists an existential restriction $\exists r.D'' \in D$ with $D.r \sqcap D'' \sqsubseteq D.r \sqcap C'$. Hence, $D.r \sqcap D'' \sqsubseteq D.r \sqcap D'$, and adding $E.r$, $E.r \sqcap D.r \sqcap D'' \sqsubseteq E.r \sqcap D.r \sqcap D'$. Since C, D are E-reduced, it is $D' = D''$. Therefore, $E.r \sqcap D.r \sqcap C' \equiv E.r \sqcap D.r \sqcap D'$. By assumption C' is $E.r \sqcap C.r$-reduced. Furthermore, D' is $E.r \sqcap D.r$-reduced. Now, by Lemma 68 D' is $E.r \sqcap C.r$-reduced as well. Employing Lemma 69, $C' \equiv D'$. Again, the induction step yields $\mathcal{G}(C') \cong \mathcal{G}(D')$. Since C is E-reduced, we know that for D' there is exactly one C' equivalent to D'. Analogously, for every existential restriction $\exists r.C'$ there exists exactly one existential restriction $\exists r.D' \in D$ such that $C' \equiv D'$. In particular, there exists a bijection mapping every existential restriction $\exists r.D'$ in D to one restriction $\exists r.C'$ in C with $D' \equiv C'$.

Now, from i), ii), and iii) it follows $\mathcal{G}(C) \cong \mathcal{G}(D)$. ∎

Computing reduced $\mathcal{ALE}$-concept descriptions

In the following we show that every $\mathcal{ALE}$-concept description C can be turned into an equivalent $\top$-reduced concept. Then, using Theorem 70, we can deduce that the two notions $\top$-reduction and reduction in fact mean the same.

Definition 71 Let F, E be $\mathcal{ALE}$-concept descriptions in $\forall$-normal form. Then, $H := F \setminus E$ is the E-reduced concept of F iff
$H = \bot$ in case $E \sqcap F \equiv \bot$; otherwise

1. $prim(H) = prim(F) \setminus prim(E)$; and every concept name on the top-level of H only occurs ones;
2. for all $r \in N_R$, $H.r = F.r \setminus E.r$ and H does not have a value restriction for r on top-level of H if $H.r = \top$;
3. for all $r \in N_R$, let $\exists r.H_1, \ldots, \exists r.H_q \in H$, $\exists r.F_1, \ldots, \exists r.F_n \in F$, and $\exists r.E_1, \ldots, \exists .E_m \in E$ be the all existential restrictions on the top-level of H, F, and E, respectively. Then, there exists a subset $\{C_1, \ldots, C_q\}$ of $\{F_1, \ldots, F_n\}$ such that
   (a) there exist no $j_1, j_2 \in \{1, \ldots, q\}$, $j_1 \neq j_2$ with $E.r \sqcap F.r \sqcap C_{j_1} \sqsubseteq E.r \sqcap F.r \sqcap C_{j_2}$;
   (b) there exists no $i \in \{1, \ldots, m\}$ and $j \in \{1, \ldots, q\}$ with $E.r \sqcap F.r \sqcap E_i \sqsubseteq C_j$;
   (c) for all $i \in \{1, \ldots, n\}$ there exist $j \in \{1, \ldots, q\}$ with $E.r \sqcap F.r \sqcap C_j \sqsubseteq E.r \sqcap F.r \sqcap F_i$; and
   (d) for all $j \in \{1, \ldots, q\}$, $H_j = C_j \setminus (E.r \sqcap F.r)$.

We summarize simple properties of $F \setminus E$.

Lemma 72
1. $F \setminus E$ is a subdescription of F (up to commutativity and assoziativity of concept conjunction).
2. $|F \setminus E| \leq |F|$ where $| \cdot |$ denotes the size of a concept.
3. $F \setminus E$ is E-reduced.
4. $F \setminus E \sqcap E \equiv F \sqcap E$, in particular $F \setminus \top \equiv F$.
5. For every $\mathcal{ALE}$-concept description F, $F \setminus E$ can be computed by a polynomial time algorithm with an oracle for deciding subsumption.

Now, we are primed to show

Corollary 73 Let C be an $\mathcal{ALE}$-concept description in $\forall$-normal form. Then, C is reduced iff C is $\top$-reduced.

Proof. Assume that C is reduced. We know that $C \setminus \top$ is a subdescription of C, $C \setminus \top \equiv C$, and $C \setminus \top$ is $\top$-reduced. But then, since C is reduced, C and $C \setminus top$ must syntactically coincide (up to commutativity and assoziativity of concept conjunction). As a result, C is $\top$-reduced as well.

Assume that C is $\top$-reduced. Then, by Theorem 70 we know that every $\top$-reduced $\mathcal{ALE}$-concept description equivalent to C syntactically coincides with C (up to commutativity and assoziativity of concept conjunction). Therefore, C must be reduced. Otherwise, there would exist a strict and equivalent subdescription of C, which then does not syntactically coincide with C. ∎

Reduced concepts are indeed minimal representations of their equivalence class in the following sense:

Corollary 74 For every $\mathcal{ALE}$-concept description F one can compute (in polynomial time with an oracle for deciding subsumption in $\mathcal{ALE}$) an equivalent $\mathcal{ALE}$-concept description F' with $|F'| \leq |E|$ for all $E \equiv F$.

Proof. Every $\mathcal{ALE}$-concept description can be turned into $\forall$-normal form by a polynomial time algorithm. Furthermore, turning a concept into its $\forall$-normal form decreases the size of the concept. Therefore, we can assume that F and E are in $\forall$-normal form. We show that the statement is true for $F' := F \setminus \top$. So, let $F' := F \setminus \top$ and let E be a concept in $\forall$-normal form equivalent to F. By Corollary 73, we know that F is $\top$-reduced. Furthermore, $E \setminus \top \equiv E$ and $|E \setminus \top| \leq |E|$. Then, since $E \setminus \top$ is $\top$-reduced, Theorem 70 implies $\mathcal{G}(F \setminus \top) \cong \mathcal{G}(E \setminus \top)$, which means $|F| = |E \setminus \top| \leq |E|$. ∎

In the subsequent sections, we will need a variant of Theorem 70 where only one concept description is reduced. In order to formulate the statement we need the following notation. Let C be an $\mathcal{ALE}$-concept description. Then, the $\bot$-*extension* $C_\bot$ of C denotes the $\mathcal{ALE}$-concept description obtained from C by adding $\bot$ at all positions in C such that the resulting concept description is equivalent to C. In other words, in $C_\bot$ inconsistencies are made explicit. Analogously, $\bot$-extensions $\mathcal{G}_\bot$ are defined for description trees $\mathcal{G}$, which are needed later on.

Proposition 75 Let C, D be $\mathcal{ALE}$-concept descriptions in $\forall$-normal form and let C be reduced. Then, there exists an injective homomorphism from $\mathcal{G}(C)$ into $\mathcal{G}(D_\bot) (= \mathcal{G}(D)_\bot)$.

Proof. By Theorem 70, we know that there exists an isomorphism ψ from $\mathcal{G}(C)$ onto $\mathcal{G}(D \setminus \top)$. By Lemma 72, $D \setminus \top$ is a subdescription of D. Then, it is easy to see that $\mathcal{G}(D \setminus \top)$ is a subtree of $\mathcal{G}(D_\bot)$. For that reason, ψ is an injective homomorphism from $\mathcal{G}(C)$ into $\mathcal{G}(D_\bot)$. ∎

Note that for $\mathcal{FLE}$, $D_\bot = D$.

6 Deciding the solvability of matching in $\mathcal{FLE}$

In this section, we investigate the complexity of deciding the solvability of matching in $\mathcal{FLE}$.

As an immediate consequence of Lemma 10 and since subsumption in $\mathcal{FLE}$ is an NP-complete problem [12] we know

Corollary 76 Deciding the solvability of $\mathcal{FLE}$-matching problems modulo subsumption is an NP-complete problem.

Also, this problem is NP-hard for matching modulo equivalence. The main contribution of this section is to show that the solvability of matching modulo equivalence in $\mathcal{FLE}$ can be decided in non-deterministic polynomial time in the size of the matching problem. More precisely, we show that if a matching problem $C \equiv^? D$ is solvable, then there exists a matcher of size polynomially bounded in the size of the matching problem. The reason why considering $\mathcal{FLE}$ separately from $\mathcal{ALE}$ is twofold. First, results for one language not necessarily carry over to its sublanguages. Second, the proof presented here for $\mathcal{FLE}$ differs from the one for $\mathcal{ALE}$ in that in $\mathcal{FLE}$ matchers can always be built from conjunctions of subdescriptions of C. For $\mathcal{ALE}$, on the other hand, we can only proof the polynomial bound without any further information about the structure of the matchers. We conjecture that with the additional information for $\mathcal{FLE}$ we might be able to specify an NP-decision algorithm similar to the one for $\mathcal{EL}$ in Section 3.6.

The proof for $\mathcal{FLE}$ makes use of the minimal representation of concept descriptions presented in Section 5. Therefore, we need to introduce so-called $\forall$-mappings that turn description trees into their $\forall$-normal from.

6.1 The $\forall$-mapping

When representing a concept description/pattern C by its description tree $\mathcal{G} := \mathcal{G}(C) = (V, E, v, \ell)$, then applying the $\forall$-rule means merging certain nodes in $\mathcal{G}$: Let n, n_1, n_2 be nodes in $\mathcal{G}$ and $r \in N_R$ with $n\forall rn_1, n\forall rn_2 \in E$. Now, applying the $\forall$-rule means *merging* n_1 and n_2, i.e. we construct a new node $n_1 \otimes n_2$ with label $\ell(n_1) \cup \ell(n_2)$. In addition, in all edges of $\mathcal{G}$, n_1 and n_2 are replaced by $n_1 \otimes n_2$. Just as for concepts, $\mathcal{G}$ is in $\forall$-normal form if the $\forall$-rule cannot be applied.

The $\forall$-rule takes $\mathcal{G}$ into a new tree $\mathcal{G}'$ where two nodes of $\mathcal{G}$ are merged. This induces a homomorphism from $\mathcal{G}$ into $\mathcal{G}'$ mapping n_1 and n_2 on $n_1 \otimes n_2$ in $\mathcal{G}'$ and mapping all other nodes onto themselves.

Exhaustively applying the $\forall$-rule to $\mathcal{G}$ induces a sequence $\varphi_1, \dots, \varphi_n$ of homomorphisms such that $\varphi := \varphi_n \circ \cdots \circ \varphi_1$ is a homomorphism from $\mathcal{G}$ into the $\forall$-normal form of $\mathcal{G}$; φ is called the $\forall$-*mapping* of $\mathcal{G}$. Properties of φ are summarized in the following lemma.

Lemma 77 1. φ is a homomorphism from $\mathcal{G}$ into the $\forall$-normal form of $\mathcal{G}$.

2. If $\mathcal{H}$ is a subtree of $\mathcal{G}$, then $\varphi(\mathcal{H})$ is the $\forall$-normal form of $\mathcal{H}$, in particular, $\varphi(\mathcal{H}) \equiv \mathcal{H}$.

3. If the subtree $\mathcal{H}$ of $\mathcal{G}$ is $\forall$-normalized, then $\varphi(\mathcal{H})$ is isomorphic to $\mathcal{H}$.

6.2 NP-completeness of matching in $\mathcal{FLE}$

We shall prove the following theorem.

Theorem 78 If the $\mathcal{FLE}$-matching problem $C \equiv^? D$ is solvable, then there exists a matcher of size polynomially bounded in the size of the matching problem which only uses concept names and role names already contained in the matching problem.

Just as for $\mathcal{EL}$, we construct a matcher σ polynomially bounded in the size of the matching problem (using only identifiers in C) given a matcher σ' for $C \equiv^? D$. Without loss of generality, we may assume that $\sigma'(X)$ is in $\forall$-normal form for every variable X in D.

Let φ be the $\forall$-mapping on $\mathcal{G}(\sigma'(D))$. Then, according to Proposition 75, there exists an injective homomorphism ψ from $\mathcal{G}(C \setminus \top)$ into $\varphi(\mathcal{G}(\sigma'(D)))$.

The description tree $\mathcal{G}(D)$ can be viewed as subtree of $\mathcal{G}(\sigma'(D))$. As in Section 3.3, $V_D(X) := \{w \in \mathcal{G}(D) \mid X \in \mathcal{G}(D)(w)\}$ denotes the set of nodes in $\mathcal{G}(D)$ with X in their label. Also, for $w \in V_D(X)$ we refer to the subtree of $\mathcal{G}(\sigma'(D))$ with root w isomorphic to $\mathcal{G}(\sigma'(X))$ by $\mathcal{G}_{\sigma'(X),w}$.

Now σ is defined as follows:

$$\sigma(X) := \bigsqcap_{w \in V_D(X)} C_{\psi^{-1}(\varphi(\mathcal{G}_{\sigma'(X),w}))}$$

for every variable X in D.

By construction, $\sigma(X)$ only contains identifiers used in C. Just as for $\mathcal{EL}$, we first show that σ is more general than σ'.

Lemma 79 $\sigma \sqsupseteq_s \sigma'$.

Proof. Let $w \in V_D(X)$. By Lemma 58, ψ is a homomorphism from $\psi^{-1}(\varphi(\mathcal{G}_{\sigma'(X),w}))$ into $\psi(\psi^{-1}(\varphi(\mathcal{G}_{\sigma'(X),w})))$, which, according to Lemma 60, is a rooted subtree of $\varphi(\mathcal{G}_{\sigma'(X),w})$. Thus, ψ is a homomorhism from $\psi^{-1}(\varphi(\mathcal{G}_{\sigma'(X),w}))$ into $\varphi(\mathcal{G}_{\sigma'(X),w})$, which by Remark 63 implies $\psi^{-1}(\varphi(\mathcal{G}_{\sigma'(X),w})) \sqsupseteq \varphi(\mathcal{G}_{\sigma'(X),w})$. Following Lemma 77, $\varphi(\mathcal{G}_{\sigma'(X),w})$ is equivalent to $\mathcal{G}_{\sigma'(X),w}$, and by construction, $\mathcal{G}_{\sigma'(X),w} \equiv \sigma'(X)$. This shows that $C_{\psi^{-1}(\varphi(\mathcal{G}_{\sigma'(X),w}))} \sqsupseteq \mathcal{G}_{\sigma'(X),w} \equiv \sigma'(X)$, and hence, $\sigma(X) \sqsupseteq \sigma'(X)$. ∎

Consequently, by virtue of Lemma 5, $\sigma(D) \sqsupseteq \sigma'(D)$.

Lemma 80 $\sigma(D) \sqsubseteq C$.

Proof. By Remark 63, it suffices to show that ψ is a homomorphism from $\mathcal{G}(C \setminus \top)$ into a description tree isomorphic to $\varphi(\mathcal{G}(\sigma(D)))$, since $\varphi(\mathcal{G}(\sigma(D))) \equiv \mathcal{G}(\sigma(D)) \equiv \sigma(D)$ (Lemma 77).

Since $\sigma'(X)$ is in $\forall$-normal form, Lemma 77 implies that for every $w \in V_D(X)$, $\varphi(\mathcal{G}_{\sigma'(X),w})$ is isomorphic to $\mathcal{G}_{\sigma'(X),w}$. Furthermore, due to Lemma 60, $\psi(\psi^{-1}(\varphi(\mathcal{G}_{\sigma'(X),w}))$ is a rooted subtree of $\varphi(\mathcal{G}_{\sigma'(X),w})$, and thus, isomorphic to a rooted subtree of $\mathcal{G}_{\sigma'(X),w}$. In the sequel, we will call this subtree $\mathcal{G}_{X,w}$ and refer to the isomorphism from $\mathcal{G}_{X,w}$ onto the corresponding subtree in $\varphi(\mathcal{G}_{\sigma'(X),w})$ by $\psi_{X,w}$. Hence, $\mathcal{G}_{X,w} = \psi_{X,w}^{-1}(\psi(\psi^{-1}(\varphi(\mathcal{G}_{\sigma'(X),w}))))$. Note that for every $v \in G_{X,w}$, $\psi_{X,w}(v) = \varphi(v)$.

Let $\mathcal{G}$ be the description tree obtained by instantiating $\mathcal{G}(D)$ as follows: For every $w \in V_D(X)$ extend $\mathcal{G}(D)$ by $\mathcal{G}_{X,w}$ at the node w.

By construction, $\mathcal{G}$ is a rooted subtree of $\mathcal{G}(\sigma'(D))$. We prove that ψ is a homomorphism from $\mathcal{G}(C \setminus \top)$ into $\varphi(\mathcal{G})$:

Let $v \in \mathcal{G}(C \setminus \top)$. If there exists a node $v' \in \mathcal{G}(D)$ with $\varphi(v') = \psi(v)$, then, as $v' \in \mathcal{G}$, we know $\psi(v) \in \varphi(\mathcal{G})$. In case, there is no node $v' \in \mathcal{G}(D)$ with $\varphi(v') = \psi(v)$, then there exists a variable X in D, and a node $w \in V_D(X)$ as well as $v' \in \mathcal{G}_{\sigma'(X),w}$ with $\varphi(v') = \psi(v)$. Hence, $\psi(\psi^{-1}(\varphi(v')) = \psi(v)$. Therefore, $\psi_{X,w}^{-1}(\psi(v))$ is an element of $\mathcal{G}_{X,w}$, thus an element of $\mathcal{G}$ and $\varphi(\psi_{X,w}^{-1}(\psi(v))) = \psi(v)$ is an element of $\varphi(\mathcal{G})$. This shows that for every $v \in \mathcal{G}(C\setminus\top)$, $\psi(v) \in \varphi(\mathcal{G})$.

Let $v, w \in \mathcal{G}(C \setminus \top)$, $r \in N_R$, and $vrw \in \mathcal{G}(C \setminus \top)$ ($v\forall rw \in \mathcal{G}(C \setminus \top)$). We know that ψ is a homomorphism from $\mathcal{G}(C \setminus \top)$ into $\varphi(\mathcal{G}(\sigma'(D)))$. Therefore, $\psi(v)r\psi(w) \in \varphi(\mathcal{G}(\sigma'(D)))$ ($\psi(v)\forall r\psi(w) \in \varphi(\mathcal{G}(\sigma'(D)))$). As verified above, $\psi(v), \psi(w) \in \varphi(\mathcal{G})$ and $\mathcal{G}$ is a rooted subtree of $\mathcal{G}(\sigma'(D))$. Consequently, $\psi(v)r\psi(w) \in \varphi(\mathcal{G})$ ($\psi(v)\forall r\psi(w) \in \varphi(\mathcal{G})$).

It remains to be shown that for every node $v \in \mathcal{G}(C \setminus \top)$, $\mathcal{G}(C \setminus \top)(v) \subseteq \varphi(\mathcal{G})(\psi(v))$. Let $A \in \mathcal{G}(C \setminus \top)(v)$. We know $\mathcal{G}(C \setminus \top)(v) \subseteq \varphi(\mathcal{G}(\sigma'(D)))(\psi(v))$. By definition of φ, there exists a node $v' \in \mathcal{G}(\sigma'(D))$ with $\varphi(v') = \psi(v)$ and $A \in \mathcal{G}(\sigma'(D))(v')$. If $v' \in \mathcal{G}(D)$, then since $\mathcal{G}$ extends $\mathcal{G}(D)$, it follows $A \in \mathcal{G}(v')$, and therefore, $A \in \varphi(\mathcal{G})(\varphi(v'))$. Otherwise, there exists a variable X in D and $w \in V_D(X)$ as well as $v' \in \mathcal{G}_{\sigma'(X),w}$ with $\varphi(v') = \psi(v)$ and $A \in \mathcal{G}_{\sigma'(X),w}(v')$. As a result, $A \in \varphi(\mathcal{G}_{\sigma'(X),w})(\varphi(v'))$. Together with our assumption $A \in \mathcal{G}(C \setminus \top)(v)$ this implies $A \in \psi^{-1}(\varphi(\mathcal{G}_{\sigma'(X),w}))(v)$. Therefore, $A \in \psi(\psi^{-1}(\varphi(\mathcal{G}_{\sigma'(X),w})))(\psi(v))$. Consequently, $A \in \mathcal{G}_{X,w}(\psi_{X,w}^{-1}(\psi(v))$, which shows that $A \in \varphi(\mathcal{G}_{X,w})(\psi(v))$. Thus, $A \in \varphi(\mathcal{G})(\psi(v))$.

This shows that ψ is a homomorphism from $\mathcal{G}(C \setminus \top)$ into $\varphi(\mathcal{G})$. By Remark 63, this implies that $\mathcal{G}(C \setminus \top) \sqsupseteq \varphi(\mathcal{G})$.

Finally, observe that for $w \in V_D(X)$, $C_{\psi^{-1}(\varphi(\mathcal{G}_{\sigma'(X),w}))}$ is equivalent to $\psi(\psi^{-1}(\varphi(\mathcal{G}_{\sigma'(X),w})))$ (see Lemma 60), which, by construction, is isomorphic to $\mathcal{G}_{X,w}$. Therefore, $\mathcal{G}_{X,w}$ is isomorphic to a subtree of $\mathcal{G}(\sigma(X))$. Hence, $\mathcal{G} \sqsupseteq \mathcal{G}(\sigma(D))$.

To sum up, we have shown that $\mathcal{G}(C \setminus \top) \sqsupseteq \varphi(\mathcal{G}) \equiv \mathcal{G} \sqsupseteq \mathcal{G}(\sigma(D))$. ∎

From Lemma 79 and 80 we can deduce that σ is a matcher of the problem $C \equiv^? D$. It remains to show that the size of σ is polynomially bounded in the size of the matching problem. This fact is a consequence of the following lemma.

Lemma 81 For every variable X in D, the size of $\sigma(X)$ is polynomially bounded in the size of C.

Proof. By Lemma 60, we know that $\psi^{-1}(\varphi(\mathcal{G}_{\sigma'(X),w}))$ is a subtree of $\mathcal{G}(C \setminus \top)$. Thus, $C_{\psi^{-1}(\varphi(\mathcal{G}_{\sigma'(X),w}))}$ is linearly bounded in the size of $C \setminus \top$. This shows that $\sigma(X)$ is bounded by the $|D| \cdot |C|$. ∎

Now, a non-deterministic algorithm can guess a possible matcher σ polynomially bounded in the size of the matching problem and checks $C \equiv \sigma(D)$. For $\mathcal{FLE}$-concept descriptions, equivalence can be tested in non-deterministic polynomial time. Thus, we obtain an NP-algorithm for deciding solvability of $\mathcal{FLE}$-matching problems modulo equivlanece.

Furthermore, deciding the solvability of the matching problem $C \equiv^? D$ in case D is an $\mathcal{FLE}$-concept description corresponds to deciding equivalence of C and D. As equivalence of $\mathcal{FLE}$-concept description is an NP-complete problem, we know that deciding the solvability of matching problems modulo equivalence is NP-hard. To sum up, we obtain the following complexity result:

Corollary 82 Deciding the solvability of $\mathcal{FLE}$-matching problems modulo equivalence is an NP-complete problem.

7 Deciding the solvability of matching in $\mathcal{ALE}$

In this section, we investage the complexity of deciding the solvability of matching problems in $\mathcal{ALE}$. Just as for $\mathcal{FLE}$, for matching modulo subsumption we obtain:

Corollary 83 Deciding the solvability of $\mathcal{ALE}$-matching problems modulo subsumption is an NP-complete problem.

Again, this problem is NP-hard for matching modulo equivalence. The main contribution of this section is to prove that matching modulo equivalence in $\mathcal{ALE}$ is an NP-complete problem. Analogously to $\mathcal{EL}$ and $\mathcal{FLE}$, given some matcher σ', we construct a matcher σ of size polynomially bounded in the size of the matching problem. For $\mathcal{EL}$ and $\mathcal{FLE}$, such a σ has been built as conjunction of subdescriptions of C. For $\mathcal{ALE}$, however, it is not clear whether there always exists such a matcher. Therefore, we rather define $\sigma(X)$ as a certain subdescription of $\sigma'(X)$. Within that construction, an important step is to show that an inconsistent concept contains a "small" inconsistent $\mathcal{FLE}$-subdescription. For that reason, we introduce the notion of "traces".

7.1 Traces and inconsistent concepts

In this section, we shall show that for every inconsistent concept description C, there exists an inconsistent $\mathcal{FLE}$-subdescription C' of C of size polynomially bounded in the role depth *depth*(C) of C.

A first characterization of inconsistent $\mathcal{ALE}$-concept descriptions comes from [12]. In their paper a tableaux algorithm is employed in order to check for inconsistency. One can simulate their approach by applying the following so-called *p-rule* ('p' for propagation) to concept descriptions:

$$\forall r.E \sqcap \exists r.F \longrightarrow \forall r.E \sqcap \exists r.(F \sqcap E).$$

A concept description is in *p-normal form* if the p-rule cannot be applied. On the other hand, a concept description can be turn into p-normal form by exhaustively applying the p-rule. The p-rule can be applied to description trees as well. If $n_0rn_1, n_0\forall rn_2 \in \mathcal{G}$ are edges in $\mathcal{G}$, then applying the p-rule means instantiating $\mathcal{G}$ at n_1 with a copy of $\mathcal{G}_{n_2}$. Now, analogously to concept descriptions, a description tree is in p-normal form if the p-rule cannot by applied. Also, every description tree can be turned into p-normal form by exhaustively applying the p-rule. We refer to $G^p(C)$ as the p-normal form of $G(C)$.

Based on the results in [12], it is easy to see that inconsistency can be described in terms of the existence of certain $\exists$-paths in $\mathcal{G}^p(C)$.

Lemma 84 Let C be an $\mathcal{ALE}$-concept description. Then, $C \equiv \bot$ iff $\mathcal{G}^p(C)$ contains a rooted $\exists$-path p of length less or equal *depth*(C) such that the last node of p has i) $\bot$ in its label or ii) P and $\neg P$.

Such an $\exists$-path corresponds to a role chain in an ABox leading to a contradictory assertion $x : \bot$ resp. $x : P$ and $x : \neg P$.

Recall that we are interested in a "small" inconsistent $\mathcal{FLE}$-subdesription of an inconsistent C. According to Lemma 84, the p-normal form of such a subdescription of C should contain an $\exists$-path leading to an inconsistent label. The question is which parts of C are necessary to obtain such paths. We will now define so-called traces which exactly describe those parts of a concept description that contribute to $\exists$-paths in the p-normal form of a concept description. With $\neg N_C := \{\neg A \mid A \in N_C\}$ we denote the set of negated concept names.

Definition 85 Let $\mathcal{G} = (V, E, v_0, \ell)$ be an $\mathcal{ALE}$-description tree and L be a finite subset of $N_C \cup \neg N_C \cup \{\bot\}$. Then, a *trace* of the description tree $\mathcal{G}$ labeled $r_1^1 \cdots r_{i_1}^1 \cdots r_1^l \cdots r_{i_l}^l L$ is a multi-set

$$\bigcup_{j=1}^{l} \{v_0^j, \ldots, v_{i_j}^j\} \cup \bigcup_{A \in L} \{v_A\}$$

such that

1. $v_0^1 = v_0$;
2. $v_0^j r_1^j \cdots r_{i_j}^j v_{i_j}^j$ is an $\exists$-path in $\mathcal{G}$;
3. for all $j = 2, \ldots, l$ there exits a $k = 1, \ldots, j-1$ and an $m = 0, \ldots, i_k-1$ such that there exists a $\forall$-path in $\mathcal{G}$ from v_m^k to v_0^j labeled $r_{m+1}^k \cdots r_{i_k}^k r_1^{k+1} \cdots r_{i_{j-1}}^{j-1}$; and

4. for all $A \in L$, $A \in \ell(v_A)$ and there exists a $k \in \{1, \dots, l\}$ and an $m \in \{0, \dots, i_k\}$ such that there is a $\forall$-path from v_m^k to v_A in $\mathcal{G}$ labeled $r_{m+1}^k \cdots r_{i_l}^l$ in case $m < i_k$ and $r_1^{k+1} \cdots r_{i_l}^l$ otherwise. Note that if $k = l$ and $m = i_k$, then the $\forall$-path must be labeled with ε.

In order to state the relationship between $\exists$-paths and traces we need the following notations: Let L be a finite subset of $N_C \cup \neg N_C \cup \{\bot\}$. Then, a path in $\mathcal{G}$ labeled $r_1 \cdots r_n L$ stands for a path in $\mathcal{G}$ labeled $r_1 \cdots r_n$ where the label of the last node in the path contains L.

Theorem 86 Let $\mathcal{G}$ be an $\mathcal{ALE}$-description tree and $\mathcal{G}^p$ its p-normal form. Then, $\mathcal{G}^p$ contains a rooted $\exists$-path labeled $r_1 \cdots r_n L$ iff there exists a trace in $\mathcal{G}$ labeled $r_1 \cdots r_n L$.

Proof. The if-direction of the statement is easy to proof by induction on the number of segments l of the trace (see Definition 85 for l). As for the only-if direction, assume that $\mathcal{G}_0 := \mathcal{G}, \mathcal{G}_1, \dots, \mathcal{G}_m := \mathcal{G}^p$ denotes the sequence of graphs obtained by applying the p-rule in order to turn $\mathcal{G}$ into its p-normal form $\mathcal{G}^p$.

We now show that if $\mathcal{G}_{i+1}$ contains a trace labeled $r_1 \cdots r_n L$, then $\mathcal{G}_i$ contains such a trace. By induction, this would imply for every trace in $\mathcal{G}_m$ that there exists a trace in $\mathcal{G}_0$ with the same label. In particular, for a rooted $\exists$-path in $\mathcal{G}^p$ labeled $r_1 \cdots r_n L$ there exists a trace in $\mathcal{G}_0$ with the same label.

Let $\mathcal{G}_{i+1} = (V_{i+1}, E_{i+1}, v_0, \ell_{i+1})$ be obtained from $\mathcal{G}_i = (V_i, E_i, v_0, \ell_i)$ by applying the p-rule at $n_0 \in \mathcal{G}_i$ with $n_0 r n_1, n_0 \forall r n_2 \in G_i$. This means that $\mathcal{G}_{i+1}$ is obtained by instantiating $\mathcal{G}_i$ at n_1 by a copy $\mathcal{G}'$ of $\mathcal{G}_{i,n_2}$. In what follows, we call the nodes in $\mathcal{G}'$, except for its root, *new nodes*. All other nodes in $\mathcal{G}_{i+1}$ are called *old*. The old node of a copy is called *corresponding old node*.

Now, let t be a trace in $\mathcal{G}_{i+1}$ labeled $r_1 \cdots r_n L$. We shall prove that there exists a trace in $\mathcal{G}_i$ with the same label. We distinguish two cases:

1. t only contains old nodes:
    (a) If $n_1 \notin t$, then, obviously, t is a trace in $\mathcal{G}_i$ since old nodes only have old nodes as their predecessors. Furthermore, the labels of the nodes in t coincide for $\mathcal{G}_i$ and $\mathcal{G}_{i+1}$.
    (b) Assume $n_1 \in t$. We define a new trace t' as follows: If for $A \in L$, $A \notin \ell_i(n_1)$ and $v_A = n_1$, then v_A in t is replaced by n_2. It is easy to see that t' is a trace with label $r_1 \cdots r_n L$ in $\mathcal{G}_i$.

2. Assume that t contains new nodes n. W.l.o.g, every node v_A, $A \in L$, is an old node. Otherwise, for every $A \in L$, if v_A is a new node, then v_A can be replaced by its corresponding old node. This still yields a trace labeled $r_1 \cdots r_n L$ in $\mathcal{G}_{i+1}$. There exists $k = 1, \dots, l$ and $m = 0, \dots, i_k$ with $n = v_m^k$. We assume k to be minimal and for that k we assume m to be minimal. Again, two cases are distinguished:
    (a) If n is connected to its predecessor by an $\forall$-edge, then by construction there must exist an $\forall$-path from n_1 to n. But then, it is easy to verify

that replacing the new nodes by their corresponding old ones yields a trace in $\mathcal{G}_i$ labeled $r_1 \cdots r_n L$. Note that there cannot be an $A \in L$ with $v_A = n_1$. This means that the problem in 1.,(b) cannot occur here.

(b) If n is connected to its predecessor by an $\exists$-edge, then because of the minimality of k and m, the predecessor must be n_1. Then, replacing all new nodes by their corresponding old ones and adding n_2 yields the desired trace in $\mathcal{G}_i$.

∎

A trace t of $\mathcal{G}$ induces a rooted subtree $\mathcal{G}_t$ of $\mathcal{G}$ which will yield the $\mathcal{FLE}$-subdesription we are interested in.

Definition 87 Let t be a trace in $\mathcal{G}$ labeled $r_1 \cdots r_n L$. Then, $\mathcal{G}_t$ is defined to be a subtree of $\mathcal{G}$ containing all nodes of t and their predecessors. For a node $n \in \mathcal{G}_t$, its label $\mathcal{G}_t(n)$ is defined as follows:

$$A \in \mathcal{G}_t(n) \text{ iff there exists } A \in L \text{ with } v_A = n \in t.$$

Obviously, the size of $\mathcal{G}_t$ is polynomially bounded in the size of the label $r_1 \cdots r_n L$ of t. As an immediate consequence of Theorem 86 we obtain:

Corollary 88 Let t be a trace in $\mathcal{G}$ labeled $r_1 \cdots r_n L$. Then, $\mathcal{G}_t$ contains an $\exists$-path labeled $r_1 \cdots r_n L$.

Finally, we are primed to prove the main theorem of this subsection.

Theorem 89 Let C be an $\mathcal{ALE}$-concept description. Then, $C \equiv \bot$ iff there exists an $\mathcal{FLE}$-subdesription C' of C of size polynomially bounded in *depth*(C) such that $C' \equiv \bot$.

Proof. The if direction of the proposition is trivial. Let us assume that C is inconsistent. Then, by Lemma 84 $\mathcal{G}^p(C)$ contains a rooted $\exists$-path labeled $r_1 \cdots r_n L$ where $L = \{\bot\}$ or $L = \{P, \neg P\}$ for some atomic concept P. According to Theorem 86, $\mathcal{G}(C)$ contains a trace t labeled $r_1 \cdots r_n L$. By Corollary 88, we know that $\mathcal{G}_t$ is an inconsistent $\mathcal{FLE}$-subdescription of $\mathcal{G}(C)$. We define $C' := C_{\mathcal{G}_t}$.

It remains to show that $\mathcal{G}_t$ (and thus C') is of size polynomially bounded in the size of C. First, note that L is a set of cardinality at most two. Thus, the size of the label $r_1 \cdots r_n L$ is linear in n. Moreover, $n \leq$ *depth*(C) by Lemma 84. Finally, as mentioned above, the size of $\mathcal{G}_t$ is polynomially bound in the size of the label of t, thus, polynomially bounded in *depth*(C). ∎

7.2 NP-completeness of matching in $\mathcal{ALE}$

In what follows we will prove the following theorem:

Theorem 90 If the $\mathcal{ALE}$ matching problem $C \equiv^? D$ is solvable, then there exists a matcher of size polynomially bounded in the size of the matching problem which only uses concept names and role names already contained in the matching problem.

First, we show that one may restrict to matchers where the role depth of the image of every variable is restricted by the role depth of C. For that purpose, we prove a more general statement which will also be used later on.

Proposition 91 For every matcher τ of the $\mathcal{ALE}$-matching problem $C \equiv^? D$, there exists a matcher τ' such that $\tau'(X)$ is a subdescription of $\tau(X)$ with $\mathsf{depth}(\tau'(X)) \leq \mathit{depth}(C)$ for all variables X in D and $\tau'(X)$ only contains identifiers already used in C.

Proof. First, every $\tau(X)$ can be turned into $\top$-normal form, which yields a matcher τ'' of $C \equiv^? D$ with $\tau(X) \equiv \tau''(X)$ and $\tau''(X)$ subdescription of $\tau(X)$ for all variables X in D. Since, $C \equiv \tau''(D)$, by Theorem 62 we know that there exists a homormorphism φ from $\mathcal{G}^\top(\tau''(D))$ into $\mathcal{G}_C$. The tree $\mathcal{G}(\tau''(D))$ is obtained by instantiating $\mathcal{G}(D)$ with a new copy of $\mathcal{G}(\tau''(X))$ for every X and every node in $\mathcal{G}(D)$ with X in its label. Since $\tau''(X)$ is in $\top$-normal form, the copies of $\mathcal{G}(\tau''(X))$ in $\mathcal{G}^\top(\tau''(D))$ are unchanged in case $\tau''(X) \not\equiv \top$. Now, by the definition of a homomorphism, it is easy to see that every node v in (a copy of) $\mathcal{G}(\tau''(X))$ which is reached by a rooted path in $\mathcal{G}^\top(\tau''(D))$ of length greater $\mathit{depth}(\mathcal{G}_C)$ must be mapped by φ on a node n in $\mathcal{G}_C$ with $\bot$ in its label. But then, also the predecessor v' of v must be mapped on n. Consequently, in $\mathcal{G}(\tau''(X))$ the label of v' can be replaced by $\bot$ and one can delete the subtree of v'. The same is true if v' has an outgoing edge labeled by a role name not used in C. Furthermore, if a node v' in $\mathcal{G}(\tau''(X))$ contains a concept name not used in C, then this node must be mapped on a node in $\mathcal{G}_C$ containing $\bot$. Again, the label of v' can be replaced by $\bot$ and one can delete the subtree of v'. Changing $\mathcal{G}(\tau''(X))$ in this way, provides us with the description tree for $\tau'(X)$ of the desired matcher τ'. ∎

Now, let σ' be a matcher of $C \equiv^? D$ with $\mathit{depth}(\sigma'(X)) \leq \mathit{depth}(C)$ such that $\sigma'(X)$ only contains identifiers already used in C. Also, w.l.o.g, we may assume that $\sigma'(X)$ is in $\forall$-normal form. In the following, we will turn σ' into a new matcher σ of size polynomially bounded in the size of the matching problem $C \equiv^? D$.

More precisely, our proof proceeds in two steps: First, we define a rooted subtree $\mathcal{G}$ of $\mathcal{G}(\sigma'(D))$ of size polynomially bounded in the size of the matching problem $C \equiv^? D$ such that $\mathcal{G} \sqsubseteq C$; we obtain $\mathcal{G} \sqsupseteq \mathcal{G}(\sigma'(D))(\equiv C)$ since $\mathcal{G}$ is a subtree of $\mathcal{G}(\sigma'(D))$. Intuitively, $\mathcal{G}$ comprises only those parts of $\mathcal{G}(\sigma'(D))$ that are necessary to guarantee $\mathcal{G} \sqsubseteq C$. Second, using $\mathcal{G}$ we define the desired matcher σ.

Just as for $\mathcal{FLE}$, we need the $\forall$-mapping φ on $\mathcal{G}(\sigma'(D))$ defined as in Section 6.1. According to Proposition 75, there exists an injective homomorphism ψ from $\mathcal{G}(C \setminus \top)$ into the $\bot$-extension $\mathcal{H} := \varphi(\mathcal{G}(\sigma'(D)))_\bot$ of $\varphi(\mathcal{G}(\sigma'(D)))$.

The first "approximation" of $\mathcal{G}$ is

$$\mathcal{G}' := \varphi^{-1}(\psi(\mathcal{G}(C \setminus \top))).$$

For $\mathcal{FLE}$, this would already be the rooted subtree $\mathcal{G}$ of $\mathcal{G}(\sigma'(D))$ we are interested in. However, $\mathcal{G}'$ is not sufficient in the presence of inconsistent concepts. In order to see this point, assume that v is a node in $\mathcal{G}(C \setminus \top)$ labeled $\bot$. Then, $\mathcal{G}'$ contains nodes $n \in \varphi^{-1}(\psi(v))$. But the labels of n in $\mathcal{G}(\sigma'(D))$ need not contain $\bot$. Thus, $\mathcal{G}'(n)$ might not contain $\bot$, and hence, $\mathcal{G}' \sqsubseteq C$ does not hold in general. The inconsistency at the node $\psi(v)$ in $\mathcal{H}$ might rather be caused by an interaction of certain subtrees in $\varphi(\mathcal{G}(\sigma'(D)))$. One needs to extend $\mathcal{G}'$ by those subtrees. Before, extending $\mathcal{G}'$ accordingly to $\mathcal{G}$ we will identify the subtrees in $\varphi(\mathcal{G}(\sigma'(D)))$ that contribute to the inconsistency in $\psi(v)$.

So, let v be a node in $\mathcal{G}(C \setminus \top)$ labeled $\bot$. By definition, the node $\psi(v)$ in $\mathcal{H}$ has $\bot$ in its label. Let $V_{\psi(v)}$ be the subset of all nodes w in $\varphi(\mathcal{G}(\sigma'(D)))$ with a predecessor w' such that w' (i) has $\psi(v)$ as its successor (see Section 4 for the definition of predecessor and successor); and (ii) if the path from the root of $\varphi(\mathcal{G}(\sigma'(D)))$ to $\psi(v)$ is labeled $r_1 \cdots r_n$ and the path to w' is labeled $r_1 \cdots r_m$, $m \leq n$, then the path from w' to w is an $\forall$-path labeled $r_{m+1} \cdots r_n$. Observe that by definition $\psi(v) \in V_{\psi(v)}$. Now, we define

$$C_{\psi(v)} := \bigsqcap_{w \in V_{\psi(v)}} C_{\varphi(\mathcal{G}(\sigma'(D)))_w}$$

The following lemma states that $C_{\psi(v)}$ in fact represents the parts of $\varphi(\mathcal{G}(\sigma'(D)))$ that contribute to the inconsistency in $\psi(v)$.

Lemma 92 $C_{\psi(v)} \equiv \bot$.

By definition of φ, we can conclude

Lemma 93 $C_{\psi(v)} \equiv \bigsqcap_{w \in \varphi^{-1}(V_{\psi(v)})} C_{\mathcal{G}(\sigma'(D))_w}$.

According to Lemma 93, the first idea would be to extend $\mathcal{G}'$ by $\mathcal{G}(\sigma'(D))_w$ for every $w \in \varphi^{-1}(V_{\psi(v)})$. However, the size of such an extension might not be polynomially bounded in the size of the matching problem $C \equiv^? D$. However, by our assumption we know that $\mathit{depth}(C_{\psi(v)}) \leq \mathit{depth}(C) + \mathit{depth}(D)$. Then, $C_{\psi(v)}$ comprised an inconsistent $\mathcal{FLE}$-subdesription $C'_{\psi(v)}$ of size polynomially bounded in $\mathit{depth}(C) + \mathit{depth}(D)$. Thus, by Lemma 93 $\mathcal{G}(C'_{\psi(v)})$ consists of rooted subtrees $\mathcal{G}_{v,w}$ of the description trees $\mathcal{G}(\sigma'(D))_w$, $w \in \varphi^{-1}(V_{\psi(v)})$. Note that some of the trees $\mathcal{G}_{v,w}$ might correspond to $\top$. Clearly, these trees do not contribute to $\mathcal{G}(C'_{\psi(v)})$. Now, $\mathcal{G}'$ is extended to $\mathcal{G}$ by adding all $\mathcal{G}_{v,w}$ to $\mathcal{G}'$ in case $\mathcal{G}_{v,w} \not\equiv \top$. Of course, all predecessors of w in $\mathcal{G}(\sigma'(D))$ must be added as well.

Formally, $\mathcal{G}' = (V', E', v_0, \ell')$ is extended to $\mathcal{G} = (V, E, v_0, \ell)$ as follows: Let $W := \{v \in \mathcal{G}(C \setminus \top) \mid \bot \in \mathcal{G}(C \setminus \top)(v)\}$ and $W_v := \{w \mid w \in \varphi^{-1}(V_{\psi(v)}), \mathcal{G}_{v,w} \not\equiv \top\}$. Now, for $v \in W$, $w \in W_v$, let $\mathcal{G}_{v,w} := (V_{v,w}, E_{v,w}, w, \ell_{v,w})$ and $Q := \bigcup_{v \in W} \bigcup_{w \in W_v} V_{v,w}$. Finally, let $\mathcal{G}(\sigma'(D)) = (V'', E'', v_0, \ell'')$. Then,

- $V := V' \cup Q \cup \{w \mid w$ is a predecessor in $\mathcal{G}(\sigma'(D))$ for some node in $Q\}$;
- $E := E'' \cap V \times (N_R \cup \forall N_R) \times V$;
- $\ell(n) := \ell'(n) \cup \bigcup_{v \in W, w \in W_v} \ell_{v,w}(n)$ where $\ell'(n) := \ell_{v,w}(n) := \emptyset$ in case $n \notin V'$ resp. $n \notin V_{v,w}$.

With $\mathcal{G}$ thus defined, we are able to show

Lemma 94 $\mathcal{G} \sqsubseteq C$.

Proof. We show that ψ is an homomorphism from $\mathcal{G}(C \setminus \top)$ into the $\bot$-extension $\varphi(\mathcal{G})_\bot$ of $\varphi(\mathcal{G})$. Since $\varphi(\mathcal{G}) \equiv \mathcal{G}$ (cf. Lemma 77) and $\mathcal{G}(C \setminus \top) \equiv C$, this completes the proof.

We first show that ψ is an homomorphism from $\mathcal{G}(C \setminus \top)$ into $\varphi(\mathcal{G})$ when the labels of the nodes are not taken into account, i.e., for the time being, we ignore the second condition in Definition 55: By definition, ψ is an injective homomorphism from $\mathcal{G}(C \setminus \top)$ into $\mathcal{H}$ the $\bot$-extension of $\varphi(\mathcal{G}(\sigma'(D)))$. Thus, when ignoring the labels ψ is a homomorphism from $\mathcal{G}(C \setminus \top)$ into $\varphi(\mathcal{G}(\sigma'(D)))$. But then, since $\mathcal{G}' = \varphi^{-1}(\psi(\mathcal{G}(C \setminus \top)))$ is a rooted subtree of $\mathcal{G}$, ψ is also a homomorphism from $\mathcal{G}(C \setminus \top))$ into $\varphi(\mathcal{G})$ when, again, ignoring the labels of the nodes.

It remains to take the labels of the nodes in $\mathcal{G}(C \setminus \top)$ into account. For every concept name P, negation $\neg P$, or $\bot$ in the label of a node $v \in \mathcal{G}(C \setminus \top)$, we need to show $P \in \varphi(\mathcal{G})(\psi(v))$ resp. $\neg P, \bot \in \varphi(\mathcal{G})(\psi(v))$. However, it turns out that for $\bot$ we must take the $\bot$-extension $\varphi(\mathcal{G})_\bot$ of $\varphi(\mathcal{G})$.

First, assume $P \in \mathcal{G}(C \setminus \top)(v)$. The case $\neg P \in \mathcal{G}(C \setminus \top)(v)$ can be dealt with analogously. We know that $\psi(\mathcal{G}(C \setminus \top))$ is a rooted subtree of $\mathcal{H}$. In particular, $\mathcal{H}$ coincides with $\varphi(\mathcal{G}(\sigma'(D)))$ except that some labels of nodes in $\mathcal{H}$ are extended by $\bot$. Consequently, there must exist a node v' in the set $\varphi^{-1}(\psi(v))$ with $P \in \mathcal{G}(\sigma'(D))(v')$. Then, by construction $P \in \mathcal{G}(v')$.

Now, assume that $\bot \in \mathcal{G}(C \setminus \top)(v)$. By construction of $\mathcal{G}$, we can add $\bot$ to the label of $\psi(v)$ in $\varphi(\mathcal{G})$ without changing the semantics of $\varphi(\mathcal{G})$.

This shows that ψ is an homomorphism from $\mathcal{G}(C \setminus \top)$ into the $\bot$-extension of $\varphi(\mathcal{G})$. ∎

Now, we show that $\mathcal{G}$ is indeed "small". Let $V_D(X)$ and $\mathcal{G}_{\sigma'(X),w}$ be defined as in Section 6.2.

Lemma 95 The size of $\mathcal{G}$ is polynomially bounded in the size of the matching problem $C \equiv^? D$.

Proof. We first investigate the size of $\mathcal{G}' = \varphi^{-1}(\psi(\mathcal{G}(C \setminus \top)))$. By Lemma 72, $|\mathcal{G}(C \setminus \top)| \leq |\mathcal{G}(C)|$. Furthermore, since ψ is a homomorphism on $\mathcal{G}(C \setminus \top)$ it is $|\psi(\mathcal{G}(C \setminus \top))| \leq |\mathcal{G}(C \setminus \top)|$.

Let $v \in \psi(\mathcal{G}(C \setminus \top))$ and $w \in V_D(X)$ for a variable X in D. Then, because $\sigma'(X)$ is in $\forall$-normal form, we know by definition of φ that $\varphi^{-1}(v)$ contains at most one node in $\mathcal{G}_{\sigma'(X),w}$. Thus, the number of nodes that are in $\mathcal{G}_{\sigma'(X),w}$ and

$\mathcal{G}'$ is linearly bounded in the size of $\psi(\mathcal{G}(C \setminus \top))$. Furthermore, the number of variables and nodes $w \in V_D(X)$ is linear in the size of D. Consequently, intersecting the union of nodes in $\mathcal{G}_{\sigma'(X),w}$, for the variables X in D and $w \in V_D(X)$, with the nodes in $\mathcal{G}'$ yields a set of nodes polynomially bounded in the size of the matching problem. Finally, observe that $\mathcal{G}(\sigma'(D))$ is obtained by instantiating $\mathcal{G}(D)$ at the node w by $\mathcal{G}_{\sigma'(X),w}$ for all variables X in D and $w \in V_D(X)$. As a result, the number of nodes in $\mathcal{G}'$ is polynomially bounded in the size of the matching problem.

Now, let v' be a node in $\mathcal{G}'$. Then there exists a node v in $\psi(\mathcal{G}(C \setminus \top))$ such that $v' \in \varphi^{-1}(v)$. By definition, the label of v' in $\mathcal{G}'$ is a subset of the label of v.

This shows that the size of $\mathcal{G}'$ is polynomially bounded in the size of the matching problem.

By definition of $\mathcal{G}$, it is easy to see that $|\mathcal{G}| - |\mathcal{G}'|$ is polynomially bounded in the size of the matching problem, which completes the proof. ∎

In order to derive the matcher σ from $\mathcal{G}$ we need to define the intersection of subtrees.

Definition 96 Let $\mathcal{H} = (V, E, v_0, \ell)$ be an $\mathcal{ALE}$-description tree and let $\mathcal{H}_1 = (V_1, E_1, v_1, \ell_1)$ and $\mathcal{H}_2 = (V_2, E_2, v_2, \ell_2)$ be subtrees of $\mathcal{H}$. Then, *restricting* $\mathcal{H}_1$ *to* $\mathcal{H}_2$ yields the following subtree ${\mathcal{H}_1}_{\mathcal{H}_2} = (V', E', v_1, \ell')$ of $\mathcal{H}_1$:

- $V' := V_1 \cap V_2$ if $V_1 \cap V_2 \neq \emptyset$; otherwise ${\mathcal{H}_1}_{\mathcal{H}_2}$ is defined to be a tree only consisting of v_1 without any edges and with empty label;
- $E' := E \cap V' \times (N_R \cup \forall N_R) \times V'$;
- $\ell'(n) := \ell_1(n) \cap \ell_2(n)$ for all $n \in V'$.

Obviously, ${\mathcal{H}_1}_{\mathcal{H}_2}$ is a subtree of $\mathcal{H}_1$, and thus, $|{\mathcal{H}_1}_{\mathcal{H}_2}| \leq |\mathcal{H}_1|$. With that notion at hand and the definition of $\mathcal{G}$, σ is defined as follows:

Definition 97 For all variables X in D we define

$$\sigma(X) := \bigsqcap_{w \in V_D(X)} C_{(\mathcal{G}_{\sigma'(X),w})_{\mathcal{G}}}.$$

Lemma 98 The substitution σ specified in Definition 97 is a matcher for $C \equiv^? D$ of size polynomially bounded in the size of the matching problem. Also, σ only contains identifiers used in C.

Proof. By construction, we know that $\mathcal{G}$ is isomorphic to a rooted subtree of $\mathcal{G}(\sigma(D))$. Thus, $\sigma(D) \sqsubseteq \mathcal{G}$. By Lemma 94, this means $\sigma(D) \sqsubseteq C$.

On the other hand, for every variable X and $w \in V_D(X)$, $\sigma'(X) \equiv \mathcal{G}_{\sigma'(X),w} \sqsubseteq C_{(\mathcal{G}_{\sigma'(X),w})_{\mathcal{G}}}$. Consequently, $\sigma'(X) \sqsubseteq \sigma(X)$. Then, Lemma 5 guarantees $\sigma'(D) \sqsubseteq \sigma(D)$. This shows that $\sigma(D) \equiv C$.

By the assumption that σ' only uses identifiers already used in C, it is plain to see that σ only uses identifiers already used in C.

It remains to investigate the size of σ. By Lemma 95, the size of $\mathcal{G}$ is polynomially bounded in the size of the matching problem. Moreover, we know that $|(\mathcal{G}_{\sigma'(X),w})_{\mathcal{G}}| \leq |\mathcal{G}|$ and the number of variables X and nodes in $V_D(X)$ is linearly bounded in the size of D. From that, it is easy to derive that the size of σ is polynomially bounded in the size of the matching problem. ∎

Theorem 90 is an immediate consequence of Lemma 98. From Theorem 90 and the hardness result for deciding the solvability of $\mathcal{ALE}$-matching problems modulo equivlance we obtain the following corollary.

Corollary 99 Deciding the solvability of $\mathcal{ALE}$-matching problems is an NP-complete problem.

8 Computing matchers in $\mathcal{ALE}$

After having investigated the complexity of deciding the solvability of matching problems, we show how matchers can actually be computed. Just as for $\mathcal{EL}$, $\mathcal{ALE}$-matching problems may have several solutions. So, the first step is to formally characterize those sets which contain the potentially most interesting matchers. The description of these sets essentially coincides with the one presented in Section 3.4. Then, in the two subsequent sections we show how to compute the sets of matchers thus defined.

8.1 Solutions of $\mathcal{ALE}$-matching problems

Just as for $\mathcal{EL}$ (cf. Section 3.4) the set of interesting matchers of a given $\mathcal{ALE}$-matching problem is formally captured using the orderings $\sqsubseteq_i$, $\sqsubseteq_s$, and $\preceq_d$. For the readers convenience, we recall the definitions of these orderings and the related notions for matching problems modulo subsumption. The definitions can be extended to matching modulo equivalence in the obvious way.

Definition 100 Let $C \sqsubseteq^? D$ be an $\mathcal{ALE}$-matching problem modulo subsumption, and let σ, τ be solutions of this problem. Then, we define:

- σ is *i-subsumed* ("i" for "instance") by τ ($\sigma \sqsubseteq_i \tau$) if and only if $\sigma(D) \sqsubseteq \tau(D)$.
- σ is *s-subsumed* ("s" for "substitution") by τ ($\sigma \sqsubseteq_s \tau$) if and only if $\sigma(X) \sqsubseteq \tau(X)$ for all variables $X \in D$;
- σ is *submatcher* of τ ($\sigma \preceq_d \tau$) if and only if $\sigma(X) \preceq_d \tau(X)$ for all variables $X \in D$.[6]

[6]Of course, $\preceq_d$ is used in the sense specified in Definition 65. Moreover, when restricting to $\mathcal{FLE}$-matching problems, the only difference is that $\preceq_d$ is defined with respect to $\mathcal{FLE}$-subdesriptions.

Again, two matchers σ, τ are *i-equivalent* iff $\sigma \sqsubseteq_i \tau$ and $\tau \sqsubseteq_i \sigma$; σ is *i-minimal* iff for every matcher τ, $\tau \sqsubseteq_i \sigma$ implies $\tau \equiv_i \sigma$. Equivalence and minimality for the other two orderings are defined analogously. As before, d-minimal matchers are also called *reduced*. In particular, the images of these matchers are reduced as well. However, as already shown in Section 3.4, this is only a necessary not a sufficient condition for a matcher to be reduced. Thus, for a given matcher it is not sufficent to simply reduce the images of the matcher in order to obtain a reduced matcher.

With these orderings at hand, the set of matchers we want to compute for an $\mathcal{ALE}$-matching problem modulo subsumption contains for every i-equivalence class of i-minimal matchers the set of reduced matchers of that class. More precisely, we are only interested in those matchers where the images of the variables are in $\forall$-normal form. In the following, we will refer to these kind of matchers as *matchers in $\forall$-normal form*. But then, by Theorem 70 we know that reduced and s-equivalent matchers in $\forall$-normal form coincide up to commutativity and assoziativity of concept conjunction. So, it is sufficient to compute the reduced matchers up to s-equivalence.

Just as for $\mathcal{EL}$, our approach for computing i-minimal and reduced matchers of $C \sqsubseteq^? D$ prodeeds in two steps (which we consider in more detail in the next two sections):

1. Compute the set of all i-minimal matchers of $C \sqsubseteq^? D$ up to i-equivalence (i.e., one matcher of for each i-equivlanece class).

2. For each i-minimal matcher σ computed in the first step, compute the d-minimal matchers in $\forall$-normal form up to s-equivalence of the matching problem $\sigma(D) \equiv^? D$.

Of course, if we are interested in matching modulo equivlanece in the first place, we just apply the second step to $C \equiv^? D$.

8.2 Computing i-minimal matchers in $\mathcal{ALE}$

In this section, we present an algorithm for computing s-complete sets for $\mathcal{ALE}$-matching problems modulo subsumption, similar to the one in Section 3.5 for $\mathcal{EL}$. As already mentioned there, given an s-complete set one can extract representatives for each i-equivalence class of i-minimal matchers using the subsumption algorithm for $\mathcal{ALE}$.

Just as for $\mathcal{EL}$, the algorithm can easily be extended to one that computes s-complete sets for matching modulo equivalence, which, according to Lemma 9 is a more general problem.

The $\mathcal{ALE}$-matching algorithm for computing s-complete sets

The main idea of the matching algorithm for $\mathcal{ALE}$ is the same as for $\mathcal{EL}$, i.e., based on homomorphisms from the description tree of the concept pattern D into the description tree of the concept description C one defines certain substitutions and checks if these substitutions are solutions of the matching problem.

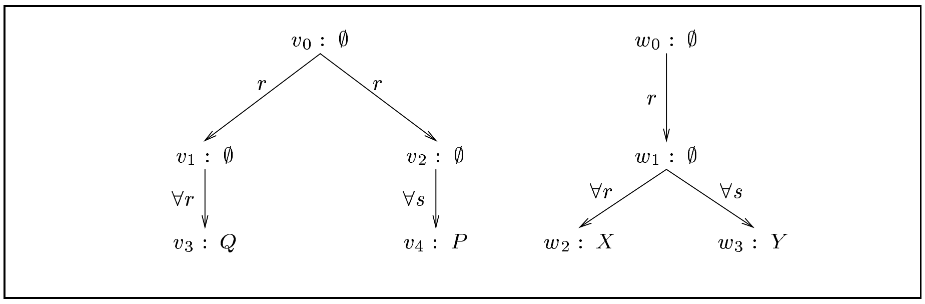

Figure 7: The description trees for C and D.

However, as illustrated by the following example, we now need to compute complete sets for a set of so-called $\top$-patterns of the original pattern D, which are obtained from D by replacing certain variables by $\top$.

Example 101 Let

$$C := (\exists r.\forall r.Q) \sqcap (\exists r.\forall s.P) \quad \sqsubseteq^? \quad \exists r.(\forall r.X \sqcap \forall s.Y) =: D$$

be an $\mathcal{ALE}$-matching problem. The description trees for C and D are depicted in Figure 7. Obviously, $\sigma := \{X \mapsto Q, Y \mapsto \top\}$ and $\tau := \{X \mapsto \top, Y \mapsto P\}$ are solutions for the given matching problem. However, there is no homomorphism from the tree for D into the one for C: The node w_1 can be mapped either on v_1 or v_2. In the former case, w_2 can be mapped on v_3, but then there is no way to map w_3. In the latter case, w_3 must be mapped on v_4, but then there is no node w_2 can be mapped on.

The example shows that there need not to be a homomorphism between a description tree corresponding to a pattern and a description tree for a concept description, although the matching problem is solvable. On the other hand, as shown in [3], subsumption of $\mathcal{ALE}$-concept descriptions can be characterized in terms of homomorphisms (see also Theorem 62). Still, this requires the normalization of both the subsumee and the subsumer. In particular, the subsumer must be turned into $\top$-normal form. But as far as matching is concerned, a normalization of the concept pattern (subsumer) would depend on the substitutions for the variables, which we do not no in advance. So there is no unique way to normalize a concept pattern. We illustrate this problem by Example 101.

Both instances $\sigma(D)$, $\tau(D)$ of D are not $\top$-normalized since they contain the subdescriptions $\forall s.\top$ and $\forall r.\top$, respectively. The description tree for the $\top$-normalized concept description $\sigma(D)$ would not include the node w_3 and the $\forall s$-edge leading to it. In this case, there is a homomorphism from this description tree into the one for C. Analogously, for $\tau(D)$, w_2 would be deleted, which again allows to define a homomorphism.

As a result, in order to compute matchers using hormormophisms one has to consider a *set* of so-called $\top$-patterns of D. These patterns are obtained by replacing a subset of variables in D by $\top$. Furthermore, before turning such a $\top$-pattern into a description tree one has to normalize the pattern using the $\top$-rule. Referring to the example, we get the following normalized $\top$-patterns for D: D, $\exists r.(\forall r.X)$, $\exists r.(\forall s.Y)$, and $\exists r.\top$. Matching these patterns against C, our matching algorithm computes the following sets of solutions: $\emptyset$, $\{\sigma\}$, $\{\tau\}$, and $\{\{X \mapsto \top, Y \mapsto \top\}\}$. The union of these sets provides us with an s-complete set of solutions for $C \sqsubseteq^? D$.

Another way to overcome the problem posed by Example 101, is to introduce *partial* homomorphisms instead of total ones. The idea is that the partial homomorphisms allow to leave the image of certain nodes of a description tree for D undefined if there is no corresponding node in C. In Example 101, this means that the image of w_2 and w_3 might be undefined. We will not pursue this approach in the sequel, however.

In the sequel, the matching algorithm sketched above is formally described.

Definition 102 A concept pattern D' is called $\top$*-pattern* of D if D' is obtained from D by substituting a subset of variables in D by $\top$.

Using this notion the $\mathcal{ALE}$-matching algorithm for computing s-complete sets for matching modulo equivalence can be specified as depicted in Figure 8. Deleting the last line of the algorithm, provides us with the variant for computing s-complete sets for matching modulo subsumption.

```
Input: $\mathcal{ALE}$-matching problem $C \equiv^? D$
Output: s-complete set $\mathcal{C}$ of matchers for $C \equiv^? D$

$\mathcal{C} := \emptyset$
For all $\top$-pattern $D'$ of $D$ do
  For all homomorphisms $\varphi$ from $\mathcal{G}^{\top}_{D'}$ into $\mathcal{G}_C$
    Define $\sigma$ by
      $\sigma(X) := lcs(C_{(\mathcal{G}_C)_{\varphi(w)}} \mid X \in \ell(w))$ for all variables $X$ in $D'$ and
      $\sigma(X) := \top$ for all variables $X$ in $D$ not in $D'$
    If $C \sqsupseteq \sigma(D')$ then $\mathcal{C} := \mathcal{C} \cup \{\sigma\}$
```

Figure 8: The $\mathcal{ALE}$-matching algorithm.

Soundness of the $\mathcal{ALE}$-matching algorithm

The proof will proceed similarly to the one for $\mathcal{EL}$ in Section 3.5 and just as for $\mathcal{EL}$, it is based on two lemmas.

The first lemma says that for all substitutions σ computed by the algorithm in Figure 8, it is $C \sqsubseteq \sigma(D)$. We will use this fact to show that for computing s-complete sets for matching modulo subsumption one can use the algorithm depicted in Figure 8 when simply deleting the last line.

Lemma 103 Let D' be a $\top$-pattern of D, φ be a homomorphism from $\mathcal{G}^{\top}_{D'}$ into $\mathcal{G}_C$, and let σ be a substitution computed by the matching algorithm in Figure 8 w.r.t. D' and φ. Then, $C \sqsubseteq \sigma(D)$.

Proof. We first show that $C \sqsubseteq \sigma(D')$. Let X be a variable in D'. Then for every node w in D' with X in its label there exists a node v in $\mathcal{G}_C$ with $\varphi(w) = v$. By the definition of σ it follows $C_{\mathcal{G}_C(v)} \sqsubseteq \sigma(X)$. Thus, according to Theorem 62 there exists a homomorphism from $\mathcal{G}^{\top}_{\sigma(X)}$ into $\mathcal{G}_C(v)$.

For this reason, analogously to Lemma 40, one can extend φ to a homomorphism from $\mathcal{G}$ into $\mathcal{G}_C$ where $\mathcal{G}$ is obtaiend by instantiating $\mathcal{G}^{\top}_{D'}$ by (copies of) $\mathcal{G}^{\top}_{\sigma(X)}$ for all variables X in D'. It is easy to see that $\mathcal{G} \equiv \sigma(D')$. Therefore, by Remark 63 we can conclude $C \sqsubseteq \sigma(D')$.

Now, from the definition of a $\top$-pattern and since $\sigma(Y) = \top$ for all variables in D that are not in D' we know $\sigma(D) \equiv \sigma(D')$. This shows $C \sqsubseteq \sigma(D)$ and completes the proof of the lemma. ∎

Analogously to Lemma 41 we now show

Lemma 104 If σ' is a matcher for $C \equiv^? D$, then there exists a matcher σ in the set $\mathcal{C}$ of matchers computed by the matching algorithm depicted in Figure 8 with $\sigma \sqsubseteq_s \sigma'$.

Proof. We know $C \sqsubseteq \sigma'(D)$. Let $T := \{X \mid \sigma'(X) \equiv \top\}$ be a subset of variables in D. Furthermore, let D' be the $\top$-pattern of D where the variables in T are substituted by $\top$. We can conclude $\sigma'(D) \equiv \sigma'(D')$. Thus, $C \sqsubseteq \sigma'(D')$. Theorem 62 implies that there exists a homomorphism φ' from $\mathcal{G}^{\top}_{\sigma'(D')}$ into $\mathcal{G}_C$.

Now, let $\mathcal{G}$ be the description tree obtained by instantiating $\mathcal{G}^{\top}_{D'}$ by (copies of) $\mathcal{G}^{\top}_{\sigma(X)}$ for all variables X in D'. We claim that $\mathcal{G}$ is isomorphic to $\mathcal{G}^{\top}_{\sigma'(D')}$:

Computing $\mathcal{G}^{\top}_E$ for some concept pattern E corresponds to iteratively deleting the nodes in $\mathcal{G}(E)$ which i) are connected with their direct predecessors by means of an $\forall$-edge, which ii) have an empty label, and which iii) have no outgoing edges.

Now, the description tree $\mathcal{G}(\sigma'(D'))$ is obtained by instantiating $\mathcal{G}(D')$ by $\mathcal{G}(\sigma'(X))$ for all variables in D'. By definition of D', we know that $\sigma'(X) \not\equiv \top$ for all variables X in D'. Thus, the nodes in the subtree $\mathcal{G}(D')$ of $\mathcal{G}(\sigma'(D'))$ containing variables are not deleted when $\top$-normalizing $\mathcal{G}(\sigma'(D'))$. For this reason, one can obtain $\mathcal{G}^{\top}_{\sigma'(D')}$ by first $\top$-normalizing $\mathcal{G}(D')$, which yields $\mathcal{G}^{\top}_{D'}$, and then instantiating $\mathcal{G}^{\top}_{D'}$ by $\mathcal{G}^{\top}_{\sigma'(X)}$. This shows that $\mathcal{G}$ is isomophic to $\mathcal{G}^{\top}_{\sigma'(D')}$.

But then, $\mathcal{G}^{\top}_{D'}$ is a subtree of $\mathcal{G}^{\top}_{\sigma'(D')}$. Therefore, restricting φ' to $\mathcal{G}^{\top}_{D'}$, provides us with a homomorphism φ from $\mathcal{G}^{\top}_{D'}$ into $\mathcal{G}_C$.

Let σ be the substitution computed by the matching algorithm in Figure 8 w.r.t. D' and φ. It remains to show $\sigma \sqsubseteq_s \sigma'$:

If X is a variable in D but not in D', then we know $\sigma'(X) \equiv \top$. Thus, $\sigma(X) = \top \sqsubseteq \sigma'(X)$.

Now, let X be a variable in D'. Then, there exists a node w in D' with X in its label. Let $v := \varphi(w)$. When restricting φ' to $(\mathcal{G}^{\top}_{\sigma'(D')})_w$, one gets an homomorphism from $(\mathcal{G}^{\top}_{\sigma'(D')})_w$ into $(\mathcal{G}_C)_v$. Since $\mathcal{G}$ is isomorphic to $\mathcal{G}^{\top}_{\sigma'(D')}$, we can conclude from the definition of $\mathcal{G}$ that $\mathcal{G}^{\top}_{\sigma'(X)}$ is (isomorphic to) a subtree of $(\mathcal{G}^{\top}_{\sigma'(D')})_w$. Therefore, there exists a homomorphism from $\mathcal{G}^{\top}_{\sigma'(X)}$ into $(\mathcal{G}_C)_v$. By Theorem 62, we know $C_{(\mathcal{G}_C)_v} \sqsubseteq \sigma'(X)$. Hence, $\sigma(X) \sqsubseteq \sigma'(X)$. ∎

Note that the proof of this lemma makes heavy use of the fact that for the concept pattern D (or the $\top$-patterns D') only a "minor" normalization, namely, $\top$-normalization, is necessary, as opposed to the full normalization which is needed for C. As we will see in the next section, this has a major impact on complexity issues.

Lemma 103 and Lemma 104, analogously to $\mathcal{EL}$ (Section 3.5), provide us with the following theorem.

Theorem 105 For a matching problem modulo equivalence, the set $\mathcal{C}$ computed by the algorithm depicted in Figure 8 is s-complete.

Furthermore, note that the Lemmas 103 and 104 do not make use of the fact that the algorithm in Figure 8 checks $C \sqsupseteq \sigma(D')$. Also, Lemma 104 only uses the fact that σ' satisfies $C \sqsubseteq \sigma'(D)$. As a result, we obtain

Theorem 106 For a matching problem modulo subsumption, the algorithm depicted in Figure 8 when deleting the last line computes an s-complete set $\mathcal{C}$ of matchers.

Complexity of computing s-complete and i-complete sets

Just as for $\mathcal{EL}$ (Section 3.5), one can prove:

Corollary 107
1. For $\mathcal{ALE}$- and $\mathcal{FLE}$-matching problems modulo equivalence
   (a) the cardinality of a (minimal) s-complete set of matchers might grow exponentially in the size of the matching problem;
   (b) the size of s-minimal matchers might grow exponentially in the size of the matching problem.
2. For $\mathcal{ALE}$- and $\mathcal{FLE}$-matching problems modulo subsumption
   (a) the cardinality of (minimal) s-complete and i-complete sets of matchers might grow exponentially in the size of the matching problem;
   (b) the size of s-minimal and i-minimal matchers might grow exponentially in the size of the matching problem.

As already mentioned in Section 3.5, a minimal i-complete set for matching modulo equivalence contains at most one element. Furthermore, by Theorem 78 for $\mathcal{FLE}$ resp. Theorem 90 for $\mathcal{ALE}$, we know that this matcher can polynomially be bounded in the size of the matching problem.

Again, as in Section 3.5, the algorithm in Figure 8 can be used to prove matching upper bounds for the exponential lower bounds mentioned in Corollary 107.

Since the size of $\mathcal{G}^{\top}_{D'}$ for some $\top$-pattern D' of D is linear in D and since $\mathcal{G}_C$ is of size at most exponential in C the number of mappings from $\mathcal{G}^{\top}_{D'}$ into $\mathcal{G}_C$ is exponentially bounded. Furthermore, the number of $\top$-patterns of D is exponentially bounded by the size of D. As shown in [3], $lcs(C_{(\mathcal{G}_C)_{\varphi(w)}} \mid X \in \ell(w))$ corresponds to the product of the description trees $(\mathcal{G}_C)_{\varphi(w)}$. Due to the fact that the number of nodes w in $\mathcal{G}^{\top}_{D'}$ is linear in the size of the matching problem and the size of $(\mathcal{G}_C)_{\varphi(w)}$ is at most exponential in C the lcs can be computed in time exponential in the size of the matching problem $C \equiv^? D$. Thus, the size of every substitution computed by the matching algorithm is at most exponential in the size of the matching problem. To sum up,

Corollary 108 1. For $\mathcal{ALE}$- and $\mathcal{FLE}$-matching problems modulo equivalence

   (a) the cardinality of a (minimal) s-complete set of matchers can exponentially be bounded in the size of the matching problem; and

   (b) the size of s-minimal matchers can exponentially be bounded in the size of the matching problem.

2. For $\mathcal{ALE}$- and $\mathcal{FLE}$-matching problems modulo subsumption

   (a) the cardinality of (minimal) s-complete and i-complete sets of matchers can exponentially be bounded in the size of the matching problem; and

   (b) the size of s-minimal and i-minimal matchers can exponentially be bounded in the size of the matching problem.

We now consider the complexity of the matching algorithm itself. As shown in [12], subsumption of $\mathcal{ALE}$-concept descriptions is NP-complete. Since a substitution σ computed by the matching algorithm is of size at most exponential in the size of the matching problem $C \equiv^? D$ it follows that $C \sqsupseteq \sigma(D)$ can be decided in non-deterministic exponential time in the size of the matching problem. Thus, it can be decided in space exponential in the size of the matching algorithm. Recalling that the number of $\top$-patterns D' of D and the number of mappings from $\mathcal{G}^{\top}_{D'}$ into $\mathcal{G}_C$ is at most exponential in the size of D and that the lcs specified in the matching algorithm can be computed in time exponential in the size of the matching problem, the matching algorithms in Figure 8 runs in space exponential in the size of the matching problem. As mentioned above, for i-complete sets it is sufficient to compute only one matcher of size polynomially bounded in the size of the matching problem. Thus, i-complete sets can be computed by an exponential time algorithm by simply enumerating all substitutions up to the polynomially bound and testing for equivalence, which in $\mathcal{ALE}$ and $\mathcal{FLE}$ can be done by an NP-algorithm.

For matching modulo subsumption we can dispense with the last subsumption test in the algorithm in Figure 8. Thus, for this problem we obtain an exponential time algorithm for computing s-complete (and therefore, i-complete) sets of matchers. We summarize these complexity results in the following corollary.

Corollary 109

1. For $\mathcal{ALE}$- and $\mathcal{FLE}$-matching problems modulo equivalence
   - (a) s-complete sets can be computed by an algorithm using exponential space in the size of the matching problem; and
   - (b) i-complete sets can be computed by an exponential time algorithm.
2. For $\mathcal{ALE}$- and $\mathcal{FLE}$-matching problems modulo subsumption both s- and i-complete sets can be computed by exponential time algorithms.

Given s-complete (i-complete) sets, minimal sets can be computed by using the subsumption algorithms for $\mathcal{FLE}$ and $\mathcal{ALE}$. Note, however, that the size of the matchers in these sets (and therefore, the subsumption problems to consider) might already be of exponential size in the size of the matching problem. Consequently, such an algorithm only provides us with an exponential space algorithm for computing minimal complete sets.

8.3 Computing d-minimal matchers in $\mathcal{ALE}$

In this section, we present a (naïve) algorithm for computing d-complete sets of matchers for $\mathcal{ALE}$- and $\mathcal{FLE}$-matching problems modulo equivalence. Recall that d-complete sets contain (at least) all d-minimal (i.e., reduced) matchers in $\forall$-normal form up to s-equivalence.

The algorithm is based on the following lemma.

Lemma 110 For a $\mathcal{ALE}$-matching problem $C \equiv^? D$, the size of the d-minimal matchers in $\forall$-normal form can polynomially be bounded in the size of the matching problem.

PROOF. Let σ'' be some matcher of $C \equiv^? D$ in $\forall$-normal form. By Proposition 91, there exists a matcher σ', with $\sigma' \preceq_d \sigma''$ and $depth(\sigma'(X)) \leq depth(C)$ for all variables X in D. Clearly, because of $\sigma' \preceq_d \sigma''$, σ' is in $\forall$-normal form as well. Now, by Lemma 98 σ specified in Definition 97 is a matcher of $C \equiv^? D$ and of size polynomially bounded in the size of the matching problem. Furthermore, by Definition 97 it is easy to see that σ can be represented in such a way that $\sigma \preceq_d \sigma'$. ∎

Note that the construction of σ in Section 7.2 also works for $\mathcal{FLE}$-matching problems. In fact, the proof is easier since one does not need to take inconsistent concepts into account. Thus, the above lemma also holds for matching problems in $\mathcal{FLE}$.

Now, a naïve algorithm for computing (minimal) d-complete sets can simply compute all possible substitutions (in $\forall$-normal form) up to the polynomial bound and filter out those that do not solve the matching problem (and that are not d-minimal). In order to obtain minimal d-complete sets one must also take care of s-equivalent matchers. It is easy to see that d-minimality can be tested by a polynomially time algorithm with an oracle for checking subsumption. Thus, by Lemma 110, we obtain

Corollary 111 (Minimal) d-complete sets for $\mathcal{ALE}$- and $\mathcal{FLE}$-matching problems modulo equivalence can be compute by an exponential time algorithm in the size of the matching problem.

The approach for computing d-complete sets pursued in Section 3.6 using s-maximal matchers does not work for $\mathcal{ALE}$, since the set of s-maximal matcher in $\mathcal{ALE}$ might be infinite. Consider for example the $\mathcal{ALE}$-matching problem $\exists r.A \sqcap \exists r.\neg A \equiv^? X \sqcap Y$. Then, the substitutions $\sigma_n := \{X \mapsto \exists r. \cdots .\exists r.A, Y \mapsto \exists r. \cdots .\exists r.\neg A\}$ for $\exists$-chains of length n are s-incomparable, s-maximal matchers. For $\mathcal{FLE}$, we conjecture however, that Theorem 50 still holds. So, the approach pursued in Section 3.6 might work for $\mathcal{FLE}$.

Just as for $\mathcal{EL}$ (cf. Section 3.6), we can show that the cardinality of a d-complete set can grow exponentially in the size of the matching problem. However, Lemma 110 shows that the size of the matchers in d-complete sets can polynomially be bounded in the size of the matching problem. To sum up, for the size of d-complete sets we obtain

Corollary 112 For $\mathcal{ALE}$- and $\mathcal{FLE}$-matching problems modulo equivalence

- the cardinality of d-complete sets might grow exponentially in the size of the matching problem;
- the size of matchers in d-complete sets can polynomially be bounded.

Observe that when computing d-complete sets for matchers σ in i-equivalence classes of i-minimal matchers, the matching problem $\sigma(D) \equiv^? D$ might already be of size exponential in the size of the original matching problem $C \sqsubseteq^? D$ (see the steps 1. and 2. at the end of Section 8.1).

9 Conclusion and future work

In this work, we have seen that adding existential restrictions to description logis increases the complexity of both deciding the solvability of matching problems and computing matchers for these problems. Also, as opposed to the languages considered in the literature for matching so far, matching problems do not have unique least matchers anymore in the presence of existential restrictions. Using orderings on matchers, we formally answered the question of what are good sets of matchers.

The remaining technical challenge is to design a practical algorithm for computing d-minimal matchers for the languages $\mathcal{ALE}$ as well as $\mathcal{ALN}$ and its extension to the CLASSIC description language.

We will also evaluate the usefulness of matching for removing redundancies in knowledge bases within our process engineering application [6]. In order to apply matching to the problem of integrating knowledge bases [9], we first need to extend the matching algorithm to an algorithm that takes schemas (i.e., certain types of inclusion axioms) into account.

References

[1] F. Baader, R. Küsters, A. Borgida, and D. McGuinness. Matching in description logics. *Journal of Logic and Computation*, 9(3):411–447, 1999.

[2] F. Baader, R. Küsters, and R. Molitor. Computing least common subsumer in description logics with existential restrictions. LTCS-Report 98-09, LuFG Theoretical Computer Science, RWTH Aachen, Germany, 1998. See http://www-lti.informatik.rwth-aachen.de/Forschung/Papers.html.

[3] F. Baader, R. Küsters, and R. Molitor. Computing least common subsumer 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.

[4] F. Baader, R. Küsters, and R. Molitor. Rewriting concepts using terminologies – revisited. LTCS-Report 99-12, LuFG Theoretical Computer Science, RWTH Aachen, Germany, 1999. See http://www-lti.informatik.rwth-aachen.de/Forschung/Reports.html.

[5] F. Baader and P. Narendran. Unification of concept terms in description logics. In *Proceedings of the 13th Biennial European Conference on Artificial Intelligence (ECAI-98)*. Brighton, UK, 1998.

[6] F. Baader and U. Sattler. Knowledge representation in process engineering. In *Proceedings of the International Workshop on Description Logics*, Cambridge (Boston), MA, U.S.A., 1996. AAAI Press/The MIT Press.

[7] F. Baader and W. Snyder. Unification theory. In J.A. Robinson and A. Voronkov, editors, *Handbook of Automated Reasoning*. Elsevier Science Publishers, 1999. To appear.

[8] A. Borgida, R. J. Brachman, D. L. McGuinness, and L. A. Resnick. CLASSIC: A structural data model for objects. In *Proceedings of the 1989 ACM SIGMOD International Conference on Management of Data*, pages 59–67, Portland, OR, 1989.

[9] A. Borgida and R. Küsters. What's not in a name? Initial explorations of a structural approach to integrating large concept knowledge-bases. Technical Report DCS-TR-391, Rutgers University, USA, 1999. Available via ftp://ftp.cs.rutgers.edu/pub/technical-reports/.

[10] A. Borgida and D. L. McGuinness. Asking queries about frames. In *Proceedings of the Fifth International Conference on Principles of Knowledge Representation and Reasoning (KR'96)*, pages 340–349, San Francisco, Calif., 1996. Morgan Kaufmann.

[11] R. J. Brachman, D. L. McGuinness, P. F. Patel-Schneider, L. A. Resnick, and A. Borgida. Living with CLASSIC: When and how to use a KL-ONE-like language. In J. Sowa, editor, *Principles of Semantic Networks*, pages 401–456. Morgan Kaufmann, San Mateo, Calif., 1991.

[12] F.M. Donini, B. Hollunder, M. Lenzerini, A. Marchetti, D. Nardi, and W. Nutt. The complexity of existential quantification in concept languages. *Artificial Intelligence*, 2–3:309–327, 1992.

[13] M.R. Garey and D.S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* Freeman, San Francisco, 1979.

[14] D.L. McGuinness. *Explaining Reasoning in Description Logics.* PhD thesis, Department of Computer Science, Rutgers University, October, 1996. Also available as a Rutgers Technical Report LCSR-TR-277.