Matching with respect to general concept inclusions in the Description Logic $\mathcal{EL}$

Franz Baader and Barbara Morawska*
{baader,morawska}@tcs.inf.tu-dresden.de

Theoretical Computer Science, TU Dresden, Germany

Abstract. Matching concept descriptions against concept patterns was introduced as a new inference task in Description Logics (DLs) almost 20 years ago, motivated by applications in the Classic system. For the DL $\mathcal{EL}$, it was shown in 2000 that matching without a TBox is NP-complete. In this paper we show that matching in $\mathcal{EL}$ w.r.t. general TBoxes (i.e., finite sets of general concept inclusions, GCIs) is in NP by introducing a goal-oriented matching algorithm that uses non-deterministic rules to transform a given matching problem into a solved form by a polynomial number of rule applications. We also investigate some tractable variants of the matching problem w.r.t. general TBoxes.

1 Introduction

The DL $\mathcal{EL}$, which offers the constructors conjunction ($\sqcap$), existential restriction ($\exists r.C$), and the top concept ($\top$), has recently drawn considerable attention since, on the one hand, important inference problems such as the subsumption problem are polynomial in $\mathcal{EL}$, even in the presence of general concept inclusions (GCIs) [12]. On the other hand, though quite inexpressive, $\mathcal{EL}$ can be used to define biomedical ontologies, such as the large medical ontology SNOMED CT.[1]

Matching of concept descriptions against concept patterns is a non-standard inference task in Description Logics, which was originally motivated by applications of the Classic system [9]. In [11], Borgida and McGuinness proposed matching as a means to filter out the unimportant aspects of large concept descriptions appearing in knowledge bases of Classic. Subsequently, matching (as well as the more general problem of unification) was also proposed as a tool for detecting redundancies in knowledge bases [8] and to support the integration of knowledge bases by prompting interschema assertions to the integrator [10].

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

* Supported by DFG under grant BA 1122/14-2
[1] see http://www.ihtsdo.org/snomed-ct/

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 $C \equiv^? D$ since $C \equiv \sigma(D)$.

The original paper by Borgida and McGuinness actually considered matching modulo subsumption rather than matching modulo equivalence: such a problem is of the form $C \sqsubseteq^? D$, and a matcher is a substitution σ satisfying $C \sqsubseteq \sigma(D)$. Obviously, any matcher modulo equivalence is also a matcher modulo subsumption, but not vice versa. For example, the substitution $\sigma_\top := \{X \mapsto \top\}$ is a *matcher modulo subsumption* of the matching problem $C \sqsubseteq^? D$, but it is not a matcher modulo equivalence of $C \equiv^? D$. For both cases of matching, the original definitions were formulated for concept descriptions without any TBox, i.e., the subsumption or equivalence that has to be achieved by an application of the matcher does not take a TBox into account. The reason was that at that time TBoxes were usually acyclic, and thus could be reduced away by unfolding.

The first results on matching in DLs were concerned with sublanguages of the Classic description language, which does not allow for existential restrictions of the kind used above. A polynomial-time algorithm for computing matchers modulo subsumption for a rather expressive DL was introduced in [11]. The main drawback of this algorithm was that it required the concept patterns to be in structural normal form, and thus it was not able to handle arbitrary matching problems. In addition, the algorithm was incomplete, i.e., it did not always find a matcher, even if one existed. For the DL $\mathcal{ALN}$, a polynomial-time algorithm for matching modulo subsumption and equivalence was presented in [6]. This algorithm is complete and it applies to arbitrary patterns. In [5], matching in DLs with existential restrictions was investigated for the first time. In particular, it was shown that in $\mathcal{EL}$ the matching problem (i.e., the problem of deciding whether a given matching problem has a matcher or not) is polynomial for matching modulo subsumption, but NP-complete for matching modulo equivalence.

Unification is a generalization of matching where both sides of the problem are patterns and thus the substitution needs to be applied to both sides. In [8] it was shown that the unification problem in the DL $\mathcal{FL}_0$, which offers the constructors conjunction ($\sqcap$), value restriction ($\forall r.C$), and the top concept ($\top$), is ExpTime-complete. In contrast, unification in $\mathcal{EL}$ is "only" NP-complete [7]. In the results for matching and unification mentioned until now, there was no TBox involved, i.e., equivalence and subsumption was considered with respect to the empty TBox. For unification in $\mathcal{EL}$, first attempts were made to take *general TBoxes*, i.e., finite sets of general concept inclusions (GCIs), into account. However, the results obtained so far, which are again NP-completeness results, are restricted to general TBoxes that satisfy a certain restriction on cyclic dependencies between concepts [2,3].

For matching, we solve the general case in this paper: we show that matching in $\mathcal{EL}$ w.r.t. general TBoxes is NP-complete by introducing a goal-oriented matching algorithm that uses non-deterministic rules to transform a given matching problem into a solved form by a polynomial number of rule applications. The matching problems considered in this paper are actually generalizations of matching modulo equivalence and matching modulo subsumption. For the special case of matching modulo subsumption, we show that the problem is tractable also in the presence of GCIs. The same is true for the dual problem where the pattern is on the side of the subsumee rather than on the side of the subsumer.

Due to space constraints, we cannot provide complete proofs of our results. They can be found in [1].

2 The Description Logics $\mathcal{EL}$

The expressiveness of a DL is determined both by the formalism for describing concepts (the concept description language) and the terminological formalism, which can be used to state additional constraints on the interpretation of concepts and roles in a so-called TBox.

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

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

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

A *general concept inclusion axiom (GCI)* is of the form $C \sqsubseteq D$ for concept descriptions C, D. An interpretation $\mathcal{I}$ *satisfies* such an axiom $C \sqsubseteq D$ iff $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$. A *general* $\mathcal{EL}$*-TBox* is a finite set of GCIs. An interpretation is a *model* of a general $\mathcal{EL}$-TBox if it satisfies all its GCIs.

A concept description C is *subsumed* by a concept description D w.r.t. a general TBox $\mathcal{T}$ (written $C \sqsubseteq_{\mathcal{T}} D$) if every model of $\mathcal{T}$ satisfies the GCI $C \sqsubseteq D$. We say that C is *equivalent* to D w.r.t. $\mathcal{T}$ ($C \equiv_{\mathcal{T}} D$) if $C \sqsubseteq_{\mathcal{T}} D$ and $D \sqsubseteq_{\mathcal{T}} C$. If $\mathcal{T}$ is empty, we also write $C \sqsubseteq D$ and $C \equiv D$ instead of $C \sqsubseteq_{\mathcal{T}} D$ and $C \equiv_{\mathcal{T}} D$, respectively. As shown in [12], subsumption w.r.t. general $\mathcal{EL}$-TBoxes is decidable in polynomial time.

An $\mathcal{EL}$-concept description is an *atom* if it is an existential restriction or a concept name. The atoms of an $\mathcal{EL}$-concept description C are the subdescriptions of C that are atoms, and the top-level atoms of C are the atoms occurring in the top-level conjunction of C. Obviously, any $\mathcal{EL}$-concept description is the conjunction of its top-level atoms, where the empty conjunction corresponds to $\top$. The atoms of a general $\mathcal{EL}$-TBox $\mathcal{T}$ are the atoms of all the concept descriptions occurring in GCIs of $\mathcal{T}$.

We say that a subsumption between two atoms is *structural* if their top-level structure is compatible. To be more precise, following [2] we define structural subsumption between atoms as follows: the atom C is *structurally subsumed* by the atom D w.r.t. $\mathcal{T}$ ($C \sqsubseteq^{\mathsf{s}}_{\mathcal{T}} D$) iff one of the following holds:

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

It is easy to see that subsumption w.r.t. $\emptyset$ between two atoms implies structural subsumption w.r.t. $\mathcal{T}$, which in turn implies subsumption w.r.t. $\mathcal{T}$. The matching algorithms presented below crucially depend on the following characterization of subsumption w.r.t. general $\mathcal{EL}$-TBoxes first stated in [2]:

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

1. *there is an index $i \in \{1, \ldots, n\}$ such that $C_i \sqsubseteq^{\mathsf{s}}_{\mathcal{T}} D_j$ or*
2. *there are atoms $A_1, \ldots, A_k, B$ of $\mathcal{T}$ ($k \geq 0$) such that*
   (a) $A_1 \sqcap \cdots \sqcap A_k \sqsubseteq_{\mathcal{T}} B$,
   (b) for every $\eta \in \{1, \ldots, k\}$ there is $i \in \{1, \ldots, n\}$ with $C_i \sqsubseteq^{\mathsf{s}}_{\mathcal{T}} A_\eta$, and
   (c) $B \sqsubseteq^{\mathsf{s}}_{\mathcal{T}} D_j$.

3 Matching in $\mathcal{EL}$

In addition to the set N_C of concept names (which must not be replaced by substitutions), we introduce a set N_V of concept variables (which may be replaced by substitutions). *Concept patterns* are now built from concept names and concept variables by applying the constructors of $\mathcal{EL}$. A *substitution* σ maps every concept variable to an $\mathcal{EL}$-concept description. It is extended to concept patterns in the usual way:

- $\sigma(A) := A$ for all $A \in N_C \cup \{\top\}$,
- $\sigma(C \sqcap D) := \sigma(C) \sqcap \sigma(D)$ and $\sigma(\exists r.C) := \exists r.\sigma(C)$.

An $\mathcal{EL}$-concept pattern C is *ground* if it does not contain variables, i.e., if it is a concept description. Obviously, a ground concept pattern is not modified by applying a substitution.

Definition 2. *Let $\mathcal{T}$ be a general $\mathcal{EL}$-TBox.*[2] *An $\mathcal{EL}$-*matching problem *w.r.t. $\mathcal{T}$ is a finite set $\Gamma = \{C_1 \sqsubseteq^? D_1, \ldots, C_n \sqsubseteq^? D_n\}$ of subsumptions between $\mathcal{EL}$-concept patterns, where for each $i, 1 \leq i \leq n$, C_i or D_i is ground. A substitution σ is a* matcher *of Γ w.r.t. $\mathcal{T}$ if σ* solves *all the subsumptions in Γ, i.e. if $\sigma(C_1) \sqsubseteq_{\mathcal{T}} \sigma(D_1), \ldots, \sigma(C_n) \sqsubseteq_{\mathcal{T}} \sigma(D_n)$. We say that Γ is* matchable *w.r.t. $\mathcal{T}$ if it has a matcher.*

[2] Note that the GCIs in $\mathcal{T}$ are built using concept descriptions, and thus do not contain variables.

Matching problems modulo equivalence and subsumption are special cases of the matching problems introduced above:

- The $\mathcal{EL}$-matching problem Γ is a *matching problem modulo equivalence* if $C \sqsubseteq^? D \in \Gamma$ implies $D \sqsubseteq^? C \in \Gamma$. This coincides with the notion of matching modulo equivalence considered in [6,5], but extended to a non-empty general TBox.
- The $\mathcal{EL}$-matching problem Γ is a *left-ground matching problem modulo subsumption* if $C \sqsubseteq^? D \in \Gamma$ implies that C is ground. This coincides with the notion of matching modulo subsumption considered in [6,5], but again extended to a non-empty general TBox.
- The $\mathcal{EL}$-matching problem Γ is a *right-ground matching problem modulo subsumption* if $C \sqsubseteq^? D \in \Gamma$ implies that D is ground. To the best of our knowledge, this notion of matching has not been investigated before.

We will show in the following that the general case of matching, as introduced in Definition 2, and thus also matching modulo equivalence, is NP-complete, whereas the two notions of matching modulo subsumption are tractable, even in the presence of GCIs.

4 Matching modulo subsumption

The case of *left-ground matching problems modulo subsumption* can be treated as sketched in [5] for the case without a TBox. Given a general $\mathcal{EL}$-TBox $\mathcal{T}$ and two substitutions σ, τ, we define: $\sigma \sqsubseteq_{\mathcal{T}} \tau$ iff $\sigma(X) \sqsubseteq_{\mathcal{T}} \tau(X)$ for all $X \in N_V$.

Consequently, if $\sigma_\top$ denotes the substitution satisfying $\sigma_\top(X) = \top$ for all $X \in N_V$, then $\sigma \sqsubseteq_{\mathcal{T}} \sigma_\top$ holds for all substitutions σ. Since the concept constructors of $\mathcal{EL}$ are monotonic w.r.t. subsumption, this implies $\sigma(D) \sqsubseteq_{\mathcal{T}} \sigma_\top(D)$ for all concept patterns D.

Lemma 3. *Let $\Gamma = \{C_1 \sqsubseteq^? D_1, \ldots, C_n \sqsubseteq^? D_n\}$ be a left-ground matching problem modulo subsumption. Then Γ has a matcher w.r.t. $\mathcal{T}$ iff $\sigma_\top$ is a matcher of Γ w.r.t. $\mathcal{T}$.*

Proof. The "if" direction is trivial. Conversely, assume that σ is a matcher of Γ w.r.t. $\mathcal{T}$. Then we have, for all $i, 1 \leq i \leq n$, that $\sigma_\top(C_i) = C_i = \sigma(C_i) \sqsubseteq_{\mathcal{T}} \sigma(D_i) \sqsubseteq_{\mathcal{T}} \sigma_\top(D_i)$, which shows that $\sigma_\top$ is a matcher of Γ w.r.t. $\mathcal{T}$. □

The lemma shows that it is sufficient to test whether the substitution $\sigma_\top$ is a matcher of Γ, i.e., whether $\sigma_\top(C_i) \sqsubseteq_{\mathcal{T}} \sigma_\top(D_i)$ holds for all $i, 1 \leq i \leq n$. Since in $\mathcal{EL}$ subsumption w.r.t. general TBoxes is decidable in polynomial time, this yields a polynomial-time algorithm for left-ground matching modulo subsumption in $\mathcal{EL}$.

Theorem 4. *Let Γ be a left-ground $\mathcal{EL}$-matching problem modulo subsumption and $\mathcal{T}$ a general $\mathcal{EL}$-TBox. Then we can decide in polynomial time whether Γ has a matcher w.r.t. $\mathcal{T}$ or not.*

The case of *right-ground matching problems modulo subsumption* can be treated similarly. However, since $\mathcal{EL}$ does not have the bottom concept $\bot$ as a concept constructor, we cannot simply define $\sigma_\bot$ as the substitution satisfying $\sigma_\bot(X) = \bot$ for all $X \in N_V$, and then show that that the right-ground matching problems modulo subsumption, Γ, has a matcher w.r.t. $\mathcal{T}$ iff $\sigma_\bot$ is a matcher of Γ w.r.t. $\mathcal{T}$. Instead, we need to define $\sigma_\bot$ in a more complicated manner.

Given a general $\mathcal{EL}$-TBox $\mathcal{T}$ and a right-ground matching problems modulo subsumption $\Gamma = \{C_1 \sqsubseteq^? D_1, \ldots, C_n \sqsubseteq^? D_n\}$, we use $\bot(\Gamma, \mathcal{T})$ to denote the $\mathcal{EL}$-concept description that is the conjunction of all the atoms of $\mathcal{T}$ and of $D_1, \ldots, D_n$. We now define $\sigma_{\bot(\Gamma,\mathcal{T})}$ as the substitution satisfying $\sigma_{\bot(\Gamma,\mathcal{T})}(X) = \bot(\Gamma, \mathcal{T})$ for all $X \in N_V$

Lemma 5. *Let $\Gamma = \{C_1 \sqsubseteq^? D_1, \ldots, C_n \sqsubseteq^? D_n\}$ be a right-ground matching problem modulo subsumption. Then Γ has a matcher w.r.t. $\mathcal{T}$ iff $\sigma_{\bot(\Gamma,\mathcal{T})}$ is a matcher of Γ w.r.t. $\mathcal{T}$.*

Proof. The "if" direction is trivial. To see the "only-if" direction, assume that σ is a matcher of Γ w.r.t. $\mathcal{T}$. We need to show that this implies the $\sigma_{\bot(\Gamma,\mathcal{T})}$ is also a matcher of Γ w.r.t. $\mathcal{T}$, i.e., that it satisfies $\sigma_{\bot(\Gamma,\mathcal{T})}(C) \sqsubseteq_\mathcal{T} D$ for every subsumption $C \sqsubseteq^? D \in \Gamma$.

More generally, we consider subsumptions $C \sqsubseteq^? D$ where C is a subpattern of a pattern occurring in Γ or $\mathcal{T}$ and D is an atom of $\mathcal{T}$ or $D_1, \ldots, D_n$. We show the following claim:

Claim: *For every such subsumption $C \sqsubseteq^? D$, it holds that $\sigma(C) \sqsubseteq_\mathcal{T} D$ implies $\sigma_{\bot(\Gamma,\mathcal{T})}(C) \sqsubseteq_\mathcal{T} D$.*

Before proving the claim, let us show that this implies that $\sigma_{\bot(\Gamma,\mathcal{T})}$ solves Γ w.r.t. $\mathcal{T}$. In fact, any subsumption in Γ is of the form $C \sqsubseteq^? E_1 \sqcap \ldots \sqcap E_k$ where C is a subpattern of a pattern occurring in Γ, and $E_1, \ldots, E_k$ are atoms of one of the D_i. In addition, a substitution solves $C \sqsubseteq^? E_1 \sqcap \ldots \sqcap E_k$ w.r.t. $\mathcal{T}$ iff it solves all the subsumptions $C \sqsubseteq^? E_i$ for $i = 1, \ldots, k$.

We prove the claim by induction on the size $|C|$ of the left-hand side C of the subsumption $C \sqsubseteq^? D$. Let $C = F_1 \sqcap \ldots \sqcap F_\ell$, where $F_1, \ldots, F_\ell$ are atoms. We distinguish the following three cases:

1. If there is an index $i \in \{1, \ldots, \ell\}$ such that F_i is a variable, then $\sigma_{\bot(\Gamma,\mathcal{T})}(F_i) \sqsubseteq D$ since D occurs as a conjunct in $\bot(\Gamma, \mathcal{T})$. This implies $\sigma_{\bot(\Gamma,\mathcal{T})}(C) \sqsubseteq_\mathcal{T} D$.
2. If there is an index $i \in \{1, \ldots, \ell\}$ such that F_i is ground and $\sigma(F_i) \sqsubseteq_\mathcal{T} D$, then $\sigma_{\bot(\Gamma,\mathcal{T})}(F_i) = F_i = \sigma(F_i) \sqsubseteq_\mathcal{T} D$. This again implies $\sigma_{\bot(\Gamma,\mathcal{T})}(C) \sqsubseteq_\mathcal{T} D$.
3. Assume that the above two cases do not hold. Using Lemma 1, we can distinguish two more cases, depending on whether the first or the second condition of the lemma applies.
   (a) If the first condition applies, then there is an index $i \in \{1, \ldots, \ell\}$ such that $F_i \sqsubseteq^{\mathsf{s}}_\mathcal{T} D$. Since F_i is neither ground nor a variable, we know that F_i is a non-ground existential restriction. Thus, $F_i = \exists r.F'$, $D = \exists r.(D_1 \sqcap \ldots \sqcap D_m)$ with $D_1, \ldots, D_m$ atoms, and $\sigma(F') \sqsubseteq_\mathcal{T} D_i$ for all

$i \in \{1, \ldots, m\}$. Since F' is a subpattern of C, D_i are atoms of D, and $|F'| < |C|$, we can apply the induction hypothesis to the subsumptions $F' \sqsubseteq^? D_i$. This yields $\sigma_{\bot(\Gamma,\mathcal{T})}(F') \sqsubseteq_{\mathcal{T}} D_i$ for all $i \in \{1, \ldots, m\}$, and thus $\sigma_{\bot(\Gamma,\mathcal{T})}(C) \sqsubseteq_{\mathcal{T}} D$.

(b) If the second condition applies, then there are atoms $A_1, \ldots, A_k, B$ of $\mathcal{T}$ such that $A_1 \sqcap \cdots \sqcap A_k \sqsubseteq_{\mathcal{T}} B \sqsubseteq_{\mathcal{T}} D$ and for each $\eta \in \{1, \ldots, k\}$, there is $j \in \{1, \ldots, \ell\}$ such that
   i. F_j is a concept variable and $\sigma(F_j) \sqsubseteq_{\mathcal{T}} A_\eta$, or
   ii. F_j is ground and $F_j \sqsubseteq_{\mathcal{T}} A_\eta$, or
   iii. $F_j = \exists r.F'$, $A_\eta = \exists r.A'$ and $\sigma(F') \sqsubseteq_{\mathcal{T}} A'$.

   It is sufficient to show that the subsumption relationships in 3(b)i and 3(b)iii also hold if we replace σ by $\sigma_{\bot(\Gamma,\mathcal{T})}$. For 3(b)i this can be shown as in 1 and for 3(b)iii as in 3a.

This completes the proof of the claim, and thus of the lemma. □

Since the size of $\bot(\Gamma, \mathcal{T})$ is polynomial in the size of Γ and $\mathcal{T}$, this lemma yields a polynomial-time decision procedure for right-ground matching modulo subsumption.

Theorem 6. *Let Γ be a right-ground $\mathcal{EL}$-matching problem modulo subsumption and $\mathcal{T}$ a general $\mathcal{EL}$-TBox. Then we can decide in polynomial time whether Γ has a matcher w.r.t. $\mathcal{T}$ or not.*

5 The general case

NP-hardness for the general case follows from the known NP-hardness result for matching modulo equivalence without a TBox [5]. In the following, we show that matching in $\mathcal{EL}$ w.r.t. general TBoxes is in NP by introducing a goal-oriented matching algorithm that uses non-deterministic rules to transform a given matching problem into a solved form by a polynomial number of rule applications.

Let $\mathcal{T}$ be a general $\mathcal{EL}$-TBox and Γ_0 an $\mathcal{EL}$-matching problem. We can assume without loss of generality that all the subsumptions $C \sqsubseteq^? D$ in Γ_0 are such that either C or D is non-ground. In fact, if both C and D are ground, then the following holds:

– If $C \sqsubseteq_{\mathcal{T}} D$, then Γ_0 has a matcher w.r.t. $\mathcal{T}$ iff $\Gamma_0 \setminus \{C \sqsubseteq^? D\}$ has a matcher w.r.t. $\mathcal{T}$.
– If $C \not\sqsubseteq_{\mathcal{T}} D$, then Γ_0 does not have a matcher w.r.t. $\mathcal{T}$.

Consequently, we can either remove all the offending ground subsumptions without changing the solvability status of the problem, or immediately decide non-solvability. Using the fact that $C \sqsubseteq_{\mathcal{T}} D_1 \sqcap D_2$ iff $C \sqsubseteq_{\mathcal{T}} D_1$ and $C \sqsubseteq_{\mathcal{T}} D_2$, we can additionally normalize Γ_0 such that the right-hand side of each subsumption in Γ_0 is an atom. We call an $\mathcal{EL}$-matching problem *normalized* if $C \sqsubseteq^? D \in \Gamma_0$ implies that (i) either C or D is non-ground, and (ii) D is an atom.

Eager Solving (variable on the right):

Condition: A subsumption $C \sqsubseteq^? X \in \Gamma$ where $X \in N_V$. **Action:** – If there is some subsumption of the form $X \sqsubseteq^? D \in \Gamma$ such that $C \not\sqsubseteq_{\mathcal{T}} D$, then the rule application fails. – Otherwise, mark $C \sqsubseteq^? X$ as "solved."

Eager Solving (variable on the left):

Condition: A subsumption $X \sqsubseteq^? D \in \Gamma$ where $X \in N_V$. **Action:** – If there is some subsumption of the form $C \sqsubseteq^? X \in \Gamma$ such that $C \not\sqsubseteq_{\mathcal{T}} D$, then the rule application fails. – Otherwise, mark $X \sqsubseteq^? D$ as "solved."

Fig. 1. Eager Rules

Thus, assume that Γ_0 is a normalized $\mathcal{EL}$-matching problem. Our algorithm starts with $\Gamma := \Gamma_0$, and then applies non-deterministic rules to Γ. A non-failing application of a rule may add subsumptions to Γ. Note, however, that a subsumption is only added if it is not yet present. New subsumptions that are added are marked as "unsolved," as are initially all the subsumptions of Γ_0. A rule application may *fail*, which means that this attempt of solving the matching problem was not successful. A non-failing rule application marks one of the subsumptions in the matching problem as "solved." Rules are applied until all subsumptions are marked "solved" or an attempt to apply a rule has failed.

Our definition of the rules uses a function $Dec(\dots)$ on subsumptions of the form $C \sqsubseteq^? D$, where C and D are atoms and D is not a variable. A call of $Dec(C \sqsubseteq^? D)$ returns a (possibly empty) set of subsumptions or it fails:

1. $Dec(C \sqsubseteq^? D) := \{C \sqsubseteq^? D\}$, if C is a variable.
2. If $D_1, \dots, D_n$ are atoms, then $Dec(\exists r.C' \sqsubseteq^? \exists r.(D_1 \sqcap \dots \sqcap D_n))$ fails if there is an $i \in \{1, \dots, n\}$ such that both sides of $C' \sqsubseteq^? D_i$ are ground and $C' \not\sqsubseteq_{\mathcal{T}} D_i$. Otherwise, $Dec(\exists r.C' \sqsubseteq^? \exists r.(D_1 \sqcap \dots \sqcap D_n)) := \{C' \sqsubseteq^? D_i \mid 1 \leq i \leq n$ and C' or D_i is non-ground$\}$.
3. If $C = \exists r.C'$ and $D = \exists s.D'$ for roles $s \neq r$, then $Dec(C \sqsubseteq^? D)$ fails.
4. If $C = A$ is a concept name and $D = \exists r.D'$ an existential restriction, then $Dec(C \sqsubseteq^? D)$ fails.
5. If $D = A$ is a concept name and $C = \exists r.C'$ an existential restriction, then $Dec(C \sqsubseteq^? D)$ fails.
6. If both C and D are ground and $C \not\sqsubseteq_{\mathcal{T}} D$ then $Dec(C \sqsubseteq^? D)$ fails, and otherwise returns $\emptyset$.

Algorithm 7. Let Γ_0 be a normalized $\mathcal{EL}$-matching problem. Starting with $\Gamma := \Gamma_0$, apply the rules of Figure 1 and Figure 2 exhaustively in the following order:

Decomposition:

Condition: This rule applies to $\mathfrak{s} = C_1 \sqcap \cdots \sqcap C_n \sqsubseteq^? D \in \Gamma$.
Action: Its application chooses an index $i \in \{1, \ldots, n\}$ and calls $Dec(C_i \sqsubseteq^? D)$. If this call does not fail, then it adds the returned subsumptions to Γ, and marks $\mathfrak{s}$ as *solved*. If $Dec(C_i \sqsubseteq^? D)$ fails, it returns "failure."

Mutation :

Condition: This rule applies to $\mathfrak{s} = C_1 \sqcap \cdots \sqcap C_n \sqsubseteq^? D$ in Γ.
Action: Its application chooses atoms $A_1, \ldots, A_k, B$ of $\mathcal{T}$. If $A_1 \sqcap \cdots \sqcap A_k \sqsubseteq_{\mathcal{T}} B$ does not hold, then it returns "failure." Otherwise, it performs the following two steps:

- Choose for each $\eta \in \{1, \ldots, k\}$ an $i \in \{1, \ldots, n\}$ and call $Dec(C_i \sqsubseteq^? A_\eta)$. If this call does not fail, it adds the returned subsumptions to Γ. Otherwise, if $Dec(C_i \sqsubseteq^? A_\eta)$ fails, the rule returns "failure."
- If it has not failed before and $Dec(B \sqsubseteq^? D)$ does not fail, it adds the returned subsumptions to Γ. Otherwise, if $Dec(B \sqsubseteq^? D)$ fails, it returns "failure."

If these steps did not fail, then the rule marks $\mathfrak{s}$ as *solved*.

Fig. 2. Non-deterministic rules

(1) **Eager rule application:** If an eager rule from Figure 1 applies to an unsolved subsumption, apply it. If the rule application fails, stop and return "failure."
(2) **Non-deterministic rule application:** If no eager rule is applicable, let $\mathfrak{s}$ be an unsolved subsumption in Γ. Choose one of the non-deterministic rules of Figure 2, and apply it to $\mathfrak{s}$. If this rule application fails, then stop and return "failure."

If no more rule applies and the algorithm has not stopped returning "failure," then return "success."

In (2), the choice which unsolved subsumption to consider next is don't care non-deterministic. However, choosing which rule to apply to the chosen subsumption is don't know non-deterministic. Additionally, the application of a non-deterministic rules may require don't know non-deterministic choices to be made. If a non-deterministic rule is applied to a subsumption $\mathfrak{s}$, then neither its left-hand side nor its right-hand side is a variable. In fact, a subsumption that has a variable on one of its sides is solved by one of the eager rules, which have precedence over the non-deterministic rules.

It is easy to see that the subsumptions added by the non-deterministic rules satisfy the normalization conditions (i) and (ii), and thus all the sets Γ generated during a run of the algorithm are normalized $\mathcal{EL}$-matching problems. The next lemma states an important property ensured by the presence of the eager rules.

Lemma 8. *If Γ is a matching problem generated during a non-failing run of the algorithm, and both $C \sqsubseteq^? X \in \Gamma$ and $X \sqsubseteq^? D \in \Gamma$ are solved, then $C \sqsubseteq_{\mathcal{T}} D$.*

Proof. Obviously, one of the two subsumptions was solved after the other. This means that, when it was solved by the application of an eager rule, the other one was already present. Since we consider a non-failing run, the application of the eager rule did not fail, which yields $C \sqsubseteq_{\mathcal{T}} D$. □

Any run of the algorithm *terminates after a polynomial number of steps*. The main reason for this is that there are only *polynomially many subsumptions* that can occur in the matching problems Γ generated during a run.

Lemma 9. *Let Γ be a matching problem generated during a run of Algorithm 7. Then any subsumption occurring in Γ is of one of the following forms:*

1. *A subsumption contained in the original input matching problem Γ_0.*
2. *A subsumption of the form $C \sqsubseteq^? D$ where C, D are subpatterns of concept patterns occurring in Γ_0.*
3. *A subsumption of the form $C \sqsubseteq^? A$ or $A \sqsubseteq^? C$ where A is an atom of $\mathcal{T}$ and C is a subpattern of a concept pattern occurring in Γ_0.*

Since any rule application either fails while trying to solve an unsolved subsumption (in which case the algorithm stops immediately) or actually solves an unsolved subsumption, there can be only polynomially many rule applications during a run. In addition, it is easy to see that each rule application can be realized in polynomial time, with a polynomial number of possible non-deterministic choices. This shows that Algorithm 7 is indeed an NP-algorithm. It remains to show that it is sound and complete.

To show *soundness*, assume that Γ is a matching problem obtained after termination of a non-failing run of the algorithm. Since the run terminated without failure, all the subsumptions in Γ are solved. We use the subsumptions of the form $X \sqsubseteq^? C \in \Gamma$ to define a substitution σ_Γ. Note that the fact that Γ is a normalized $\mathcal{EL}$-matching problem implies that C is a ground pattern, i.e., a concept description. For each variable $X \in N_V$, we define

$$S_X^\Gamma := \{C \mid X \sqsubseteq^? C \in \Gamma\},$$

and denote the conjunction of all the elements of S_X^Γ as $\sqcap S_X^\Gamma$, where the empty conjunction is $\top$. The substitution σ_Γ is now defined as

$$\sigma_\Gamma(X) := \sqcap S_X^\Gamma \text{ for all } X \in N_V.$$

Lemma 10. *σ_Γ is a matcher of Γ w.r.t. $\mathcal{T}$.*

Since the input matching problem Γ_0 is contained in Γ, this lemma shows that σ_Γ is a matcher also of Γ_0 w.r.t. $\mathcal{T}$. This completes the proof of soundness.

Regarding *completeness*, we can use a given matcher of Γ_0 w.r.t. $\mathcal{T}$ to guide the application of the non-deterministic rules such that a non-failing run is generated (see [1] for details).

Lemma 11. *Let σ be a matcher of Γ_0 w.r.t. $\mathcal{T}$. Then there is a non-failing and terminating run of Algorithm 7 producing a matching problem Γ such that σ is a matcher of Γ w.r.t. $\mathcal{T}$.*

This lemma provides the final step towards showing that Algorithm 7 is an NP-decision procedure for matching w.r.t. general TBoxes in $\mathcal{EL}$.

Theorem 12. *The problem of deciding whether a given $\mathcal{EL}$-matching problem has a matcher w.r.t. a given general $\mathcal{EL}$-TBox or not is NP-complete.*

Let us illustrate the working of the algorithm with a small example. We consider the TBox $\mathcal{T} := \{C \sqsubseteq A, C \sqsubseteq \exists s.C, \exists s.B \sqsubseteq \exists s.C\}$ and the matching problem $\Gamma := \{X \sqcap B \sqsubseteq^? \exists s.A, \exists s.B \sqsubseteq^? \exists s.X\}$. Obviously, this problem is neither left- nor right-ground, and thus we need to use Algorithm 7 to solve it. In the beginning, all the subsumptions in Γ are unsolved, and no eager rule is applicable.

In order to apply a non-deterministic rule, the algorithm chooses one of the unsolved subsumptions. Let us assume that this is the first one, i.e., $X \sqcap B \sqsubseteq^? \exists s.A$. Now, we have a (don't know non-deterministic) choice between applying *Decomposition* or *Mutation*. Consider the case where *Decomposition* is applied in such a way that it produces $Dec(X \sqsubseteq^? \exists s.A) = \{X \sqsubseteq^? \exists s.A\}$. The unsolved subsumption $X \sqsubseteq^? \exists s.A$ is then added to Γ, while $X \sqcap B \sqsubseteq^? \exists s.A$ is marked as "solved."

Now, the algorithm applies *Eager Solving (variable on the left)* to $X \sqsubseteq^? \exists s.A$. Since there are no subsumptions with right-hand side X, the rule application does not fail and $X \sqsubseteq^? \exists s.A$ is marked as "solved."

The algorithm then chooses the only unsolved subsumption left: $\exists s.B \sqsubseteq^? \exists s.X$. Again, there is the choice between applying *Decomposition* and *Mutation*. Let us assume that *Decomposition* is chosen, which yields $Dec(\exists s.B \sqsubseteq^? \exists s.X) = \{B \sqsubseteq^? X\}$. The subsumption $\exists s.B \sqsubseteq^? \exists s.X$ is marked as "solved" and the unsolved subsumption $B \sqsubseteq^? X$ is added to Γ.

Now *Eager Solving (variable on the right)* is applied to this subsumption, which leads to failure since $B \not\sqsubseteq_{\mathcal{T}} \exists s.A$.

Backtracking to the last choice point, the algorithm applies *Mutation* to $\exists s.B \sqsubseteq^? \exists s.X$. Let us assume that it chooses the atoms $\exists s.B, \exists s.C$ of $\mathcal{T}$, which is a good choice since $\exists s.B \sqsubseteq_{\mathcal{T}} \exists s.C$. *Mutation* then yields $Dec(\exists s.B \sqsubseteq^? \exists s.B) = \emptyset$ and $Dec(\exists s.C \sqsubseteq^? \exists s.X) = \{C \sqsubseteq^? X\}$. The subsumption $\exists s.B \sqsubseteq^? \exists s.X$ is then marked as "solved" and the unsolved subsumption $C \sqsubseteq^? X$ is added to Γ.

Finally, *Eager Solving (variable on the right)* is applied to this subsumption, which does not fail since $C \sqsubseteq_{\mathcal{T}} \exists s.A$.

Since now all subsumptions are solved, no more rules apply, and the algorithm returns "success." The matcher computed by this run of the algorithm (as defined in the proof of soundness) is $\{X \mapsto \exists s.A\}$.

6 Conclusion

We have extended the known results for matching in $\mathcal{EL}$ [5] to the case where subsumption and equivalence is considered w.r.t. a non-empty general TBox, i.e., a non-empty set of GCIs. For the DL $\mathcal{FL}_0$, matching without GCIs is polynomial, and this remains true even in the extension $\mathcal{ALN}$ of $\mathcal{FL}_0$. It would be interesting to see how one can solve matching problems w.r.t. general TBoxes in these DLs. Since already subsumption in $\mathcal{FL}_0$ w.r.t. general TBoxes is ExpTime-complete [4], the complexity of solving such matching problems is at least ExpTime-hard. Another interesting open problem is unification in $\mathcal{EL}$ w.r.t. general TBoxes.

References

1. Baader, F., , Morawska, B.: Matching with respect to general concept inclusions in the description logic $\mathcal{EL}$. LTCS-Report 14-03, Chair of Automata Theory, Institute of Theoretical Computer Science, Technische Universität Dresden, Dresden, Germany (2014), see http://lat.inf.tu-dresden.de/research/reports.html.
2. Baader, F., Borgwardt, S., Morawska, B.: Extending unification in $\mathcal{EL}$ towards general TBoxes. In: Proc. of the 13th Int. Conf. on Principles of Knowledge Representation and Reasoning (KR 2012). pp. 568–572. AAAI Press (2012)
3. Baader, F., Borgwardt, S., Morawska, B.: A goal-oriented algorithm for unification in $\mathcal{ELH}_{R^+}$ w.r.t. cycle-restricted ontologies. In: Thielscher, M., Zhang, D. (eds.) Pro. of 25th Australasian Joint Conf. on Artificial Intelligence (AI'12). Lecture Notes in Artificial Intelligence, vol. 7691, pp. 493–504. Springer-Verlag (2012)
4. Baader, F., Brandt, S., Lutz, C.: Pushing the $\mathcal{EL}$ envelope. In: Kaelbling, L.P., Saffiotti, A. (eds.) Proc. of the 19th Int. Joint Conf. on Artificial Intelligence (IJCAI 2005). pp. 364–369. Morgan Kaufmann, Los Altos, Edinburgh (UK) (2005)
5. Baader, F., Küsters, R.: Matching in description logics with existential restrictions. In: Proc. of the 7th Int. Conf. on Principles of Knowledge Representation and Reasoning (KR 2000). pp. 261–272 (2000)
6. Baader, F., Küsters, R., Borgida, A., McGuinness, D.L.: Matching in description logics. J. of Logic and Computation 9(3), 411–447 (1999)
7. Baader, F., Morawska, B.: Unification in the description logic $\mathcal{EL}$. Logical Methods in Computer Science 6(3) (2010)
8. Baader, F., Narendran, P.: Unification of concept terms in description logics. J. of Symbolic Computation 31(3), 277–305 (2001)
9. Borgida, A., Brachman, R.J., McGuinness, D.L., Alperin Resnick, L.: CLASSIC: A structural data model for objects. In: Proc. of the ACM SIGMOD Int. Conf. on Management of Data. pp. 59–67 (1989)
10. Borgida, A., Küsters, R.: What's not in a name? Initial explorations of a structural approach to integrating large concept knowledge-bases. Tech. Rep. DCS-TR-391, Rutgers University (1999)
11. Borgida, A., McGuinness, D.L.: Asking queries about frames. In: Proc. of the 5th Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR'96). pp. 340–349 (1996)
12. Brandt, S.: Polynomial time reasoning in a description logic with existential restrictions, GCI axioms, and—what else? In: de Mántaras, R.L., Saitta, L. (eds.) Proc. of the 16th Eur. Conf. on Artificial Intelligence (ECAI 2004). pp. 298–302 (2004)