Matching in Description Logics with Existential Restrictions

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

Abstract

Matching of concepts with variables (concept patterns) is a relatively new operation that has been introduced in the context of description logics, originally to help filter out unimportant aspects of large concepts appearing in industrial-strength knowledge bases. Previous work on this problem has produced polynomial-time matching algorithms for sublanguages of the DL used in CLASSIC. Consequently, these algorithms cannot handle existential restrictions. In this paper, we consider matching in DLs allowing for existential restrictions. We describe decision procedures that test solvability of matching problems as well as algorithms for computing complete sets of matchers. Unfortunately, these algorithms are no longer polynomial-time, even for the small language $\mathcal{EL}$, which allows for the top concept, conjunction and existential restrictions.

1 Motivation

Matching concepts against patterns is a relatively new inference problem for DLs, which has been introduced in [5, 7] to support the pruning of large concept descriptions. Given a concept pattern D (i.e., a concept description containing variables) and a concept description C without variables, the *matching problem* $C \sqsubseteq^? D$ asks for a substitution σ (of the variables by concept descriptions) such that $C \sqsubseteq \sigma(D)$. More precisely, one is interested in a matcher σ such that the instance $\sigma(D)$ of D is as small as possible, i.e., σ should satisfy the property that there does not exist a substitution δ such that $C \sqsubseteq \delta(D) \sqsubset \sigma(D)$. We will call such a matcher *i-minimal* (instance minimal).

For example, the i-minimal matcher of the pattern

$$D := \forall \mathsf{research\text{-}interests}.X$$

against the description

$$C := \forall \mathsf{pets}.\mathsf{Cat} \sqcap \forall \mathsf{research\text{-}interests}.\mathsf{AI}$$

assigns AI to the variable X, and thus finds the scientific interests (in this case Artificial Intelligence) described in the concept. (The concept pattern can be thought of as a "format statement", describing what information is to be displayed, if the pattern matches successfully against a specific concept. If there is no match, nothing is displayed.)

A polynomial-time algorithm for computing an i-minimal matcher for a rather expressive DL (extending $\mathcal{ALN}$ by existential restrictions and some other operators) was introduced in [5]. The main drawback of this algorithm is that it requires the concept pattern to be in structural normal form, and thus it cannot handle arbitrary matching problems. In addition, due to an incomplete treatment of the top- ($\top$) and the bottom- ($\bot$) concepts, it does not always find a matcher, even if one exists.

For the DL $\mathcal{ALN}$, a polynomial-time matching algorithm that applies to arbitrary matching problems and always computes an i-minimal matcher (if the problem is solvable at all) was presented in [2]. Actually, this algorithm solves *matching problems modulo equivalence* ($C \equiv^? D$) instead of the *matching problems modulo subsumption* ($C \sqsubseteq^? D$) introduced above: σ is a matcher of $C \equiv^? D$ iff $C \equiv \sigma(D)$. Since σ is a matcher of $C \sqsubseteq^? D$ iff it is one of $C \equiv^? C \sqcap D$, matching modulo subsumption is a special case of matching modulo equivalence. Moreover, the matcher σ computed by the algorithm is the *least* matcher w.r.t. subsumption $\sqsubseteq_s$ of substitutions, where $\sigma \sqsubseteq_s \tau$ iff $\sigma(X) \sqsubseteq \tau(X)$ for all variables X. Note that the least matcher is also i-minimal since $\sigma \sqsubseteq_s \tau$ implies $\sigma(D) \sqsubseteq \tau(D)$.

The purpose of this paper is to transfer these results to DLs allowing for existential restrictions ($\exists R.C$). A more detailed presentation and all proofs can be found in [1]. In order to get a feel for the new problems caused by existential restrictions, we start with the small DL

$\mathcal{EL}$, which allows for the constructors top-concept, conjunction ($\sqcap$), and existential restriction. The results obtained for $\mathcal{EL}$ are then extended to the DL $\mathcal{ALE}$, which additionally allows for the constructors bottom-concept, atomic negation ($\neg A$ for concept names A), and value restriction ($\forall r.C$). We start with formally introducing the notions already mentioned above and mentioning some simple properties.

2 Preliminaries

In the following, let N_C and N_R be two disjoint sets of concept names and roles names, respectively. In order to define concept patterns, we additionally need the set $\mathcal{X}$ of concept variables, which is disjoint from $N_C \cup N_R$. Roughly speaking, an $\mathcal{ALE}$-concept pattern is an $\mathcal{ALE}$-concept description over the concept names $N_C \cup \mathcal{X}$ and the role names N_R. However, although $\mathcal{ALE}$ allows for negation of atomic concepts, negated concept variables are disallowed.

Definition 1 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 and 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. For example, to define $\mathcal{EL}$-concept patterns, we replace $\mathcal{ALE}$ in the second item by $\mathcal{EL}$, and remove the value restriction $\forall R.C$ from the fourth item. The following notion can also be restricted to sublanguages of $\mathcal{ALE}$ in the obvious way. 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.,

- $\sigma(E) := E$ for all $E \in N_C$,
- $\sigma(\top) := \top$ and $\sigma(\bot) := \bot$,
- $\sigma(C \sqcap D) := \sigma(C) \sqcap \sigma(D)$,
- $\sigma(\forall R.C) := \forall R.\sigma(C)$ and $\sigma(\exists R.C) := \exists R.\sigma(C)$.

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 $E \sqcap (\forall R.E) \sqcap F \sqcap \forall R.(E \sqcap \forall R.E)$.

It is easy to see that the result of applying a substitution to an $\mathcal{ALE}$-concept pattern is an $\mathcal{ALE}$-concept description. (Note that this would no longer be the case if negation were allowed in front of concept variables.)

Now, matching problems modulo subsumption ($C \sqsubseteq^? D$) and modulo equivalence ($C \equiv^? D$) and their solutions (called matchers) are defined as already stated in the first section.

For a given matching problem, we are not interested in all matcher, but rather in matchers that are "minimal" w.r.t. subsumption. There are two possibilities for comparing the matchers: either by comparing the instances of the right-hand side that they induce, or by comparing the substitutions themselves. The following definition can analogously be stated for matching modulo equivalence.

Definition 2 Let $C \sqsubseteq^? D$ be a matching problem, and let σ and τ be solutions of this problem. Then we define

1. σ is *s-subsumed* ("s" for "substitution") by τ ($\sigma \sqsubseteq_s \tau$) iff $\sigma(X) \sqsubseteq \tau(X)$ for all variables X occurring in D.
2. σ is *i-subsumed* ("i" for "instance") by τ ($\sigma \sqsubseteq_i \tau$) iff $\sigma(D) \sqsubseteq \tau(D)$.

A matcher σ of $C \sqsubseteq^? D$ is called *s-minimal* iff there is no matcher τ of the problem such that $\tau \sqsubset_s \sigma$, (i.e., $\tau \sqsubseteq_s \sigma$ and $\sigma \not\sqsubseteq_s \tau$); *i-minimal* matchers are defined analogously.

The following two examples show that s-minimal matchers are not necessarily i-minimal, and that conversely i-minimality does not imply s-minimality. (1) For the matching problem $\exists R.A \sqcap \exists R.B \sqsubseteq^? \exists R.X \sqcap \exists R.Y$, the matcher $\sigma := \{X \mapsto A, Y \mapsto A\}$ is s-minimal, but not i-minimal since with $\tau := \{X \mapsto A, Y \mapsto B\}$ we have $\tau \sqsubset_i \sigma$. (2) For the matching problem $\exists R.A \sqsubseteq^? \exists R.A \sqcap \exists R.X$ the matcher $\sigma := \{X \mapsto \top\}$ is i-minimal, but not s-minimal since the matcher $\tau := \{X \mapsto A\}$ satisfies $\tau \sqsubset_s \sigma$.

In contrast to the case for $\mathcal{ALN}$, solvable $\mathcal{ALE}$-matching problems need not have a unique s-minimal (i-minimal) matcher, as illustrated by the following example: the $\mathcal{ALE}$-matching problem $\exists R.A \sqcap \exists R.B \sqsubseteq^? \exists R.X$ has two s-minimal solutions $\sigma := \{X \mapsto A\}$ and $\tau := \{X \mapsto B\}$, which are also i-minimal since they lead to two minimal instances $\sigma(\exists R.X) = \exists R.A$ and $\tau(\exists R.X) = \exists R.B$ of the pattern. (Note that $\{X \mapsto A \sqcap B\}$ is not a solution of the matching problem.)

For this reason, our matching algorithm computes so-called complete sets of matchers. A set $\mathcal{C}$ of matchers is *s-complete* iff for all matchers σ of the matching problem there exists $\sigma' \in \mathcal{C}$ such that $\sigma' \sqsubseteq_s \sigma$; *i-complete sets* are defined analogously.[1]

The matching algorithm presented below will compute s-complete sets as opposed to i-complete sets. The following (simple) observations justify this decision: From $\sqsubseteq_s \subseteq \sqsubseteq_i$ we can deduce that every s-complete set is also

[1]The notion of a complete set of solutions is also used in unification theory [4], but there the solutions (i.e., unifiers) are compared w.r.t. the instantiation quasi-ordering.

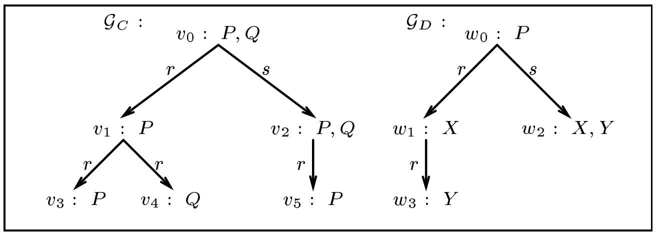

Figure 1: $\mathcal{EL}$-description trees.

i-complete (the converse is not true). Furthermore, every s-complete set contains all s-minimal and i-minimal matchers modulo s-equivalence (i.e., s-subsumption in both directions). Consequently, given an s-complete set, it is easy to extract the s- and i-minimal matchers using the subsumption algorithm for the DL in question.

3 The matching algorithm for $\mathcal{EL}$

This algorithm is based on the characterization of subsumption between $\mathcal{EL}$-concepts via homomorphisms between the corresponding description trees. This characterization has been introduced in [3] for the purpose of computing the *least common subsumer* (lcs) of $\mathcal{EL}$-concepts. Intuitively, the $\mathcal{EL}$-description tree $\mathcal{G}_C$ corresponding to an $\mathcal{EL}$-concept description C is just a graphical representation of the syntactic structure of the description. For example, the description tree corresponding to the $\mathcal{EL}$-concept description

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

is depicted on the left-hand side of Fig. 1. For an $\mathcal{EL}$-concept C and a node v in the corresponding description tree $\mathcal{G}_C$, we denote the subconcept of C corresponding to v by C_v. In our example, we have $C_{v_1} = P \sqcap \exists r.P \sqcap \exists r.Q$.

A *homomorphism* from the $\mathcal{EL}$-description tree $\mathcal{H}$ into the $\mathcal{EL}$-description tree $\mathcal{G}$ is a mapping φ from the nodes of $\mathcal{H}$ into the nodes of $\mathcal{G}$ such that (i) the root of $\mathcal{H}$ is mapped to the root of $\mathcal{G}$; (ii) if $\mathcal{H}$ contains an edge from v to w with label r, then $\mathcal{G}$ contains an edge from $\varphi(v)$ to $\varphi(w)$ with label r; (iii) if the concept name A belongs to the label of the node v in $\mathcal{H}$, then it also belongs to the label of $\varphi(v)$ in $\mathcal{G}$. Given two $\mathcal{EL}$-concept descriptions C, D, we have $C \sqsubseteq D$ iff there exists a homomorphism from $\mathcal{G}_D$ into $\mathcal{G}_C$ [3].

The notion of an $\mathcal{EL}$-description tree can be extended to concept patterns by simply treating variables like concept names. For example, the concept pattern $D := P \sqcap \exists r.(X \sqcap \exists r.Y) \sqcap \exists s.(X \sqcap Y)$ yields the description tree depicted on the right-hand side of Fig. 1. When extending the notion of a homomorphism to description trees representing concept patterns, we simply ignore the concept variables, i.e., (iii) is required only for non-variable concept names. In our example, there are exactly two homomorphisms φ_1, φ_2 from $\mathcal{G}_D$ into $\mathcal{G}_C$. Both map w_i onto v_i for $i = 0, 1, 2$, and we have $\varphi_1(w_3) = v_3$ and $\varphi_2(w_3) = v_4$.

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 $\mathcal{G}_D$;
$\mathcal{C} := \emptyset$;
For all homomorphisms φ from $\mathcal{G}_D$ into $\mathcal{G}_C$ do
  Define σ by
    $\sigma(X) := lcs\{C_{\varphi(v)} \mid X \in \text{label of } v \text{ in } \mathcal{G}_D\}$
    for all variables X in D;
  If $C \sqsupseteq \sigma(D)$ then $\mathcal{C} := \mathcal{C} \cup \{\sigma\}$;

Figure 2: The $\mathcal{EL}$-matching algorithm.

The matching algorithm described in Fig. 2 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 $C \sqsupseteq \sigma(D)$. (Obviously, 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 $C_{\varphi(v)}$ where v is such that X occurs in the label of v. Since there may exist several nodes v 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. $\sqsubseteq_s$. Recall that E is the *least common subsumer* of $E_1, \ldots, E_n$ iff i) E subsumes $E_1, \ldots, E_n$ and ii) E is the least concept description w.r.t. subsumption that satisfies i). An algorithm for computing the lcs of $\mathcal{EL}$-concepts has been described in [3]. The size of the lcs is at most exponential in the size of the concepts $E_1, \ldots, E_n$.

In our example, the homomorphism φ_1 yields the substitution σ_1:

$$\begin{array}{rcl} \sigma_1(X) & := & lcs\{P \sqcap \exists r.P \sqcap \exists r.Q,\ P \sqcap Q \sqcap \exists r.P\} \\ & \equiv & P \sqcap \exists r.P, \\ \sigma_1(Y) & := & lcs\{P \sqcap Q \sqcap \exists r.P,\ P\} \\ & \equiv & P, \end{array}$$

whereas φ_2 yields the substitution σ_2:

$$\begin{array}{rcl} \sigma_2(X) & := & lcs\{P \sqcap \exists r.P \sqcap \exists r.Q,\ P \sqcap Q \sqcap \exists r.P\} \\ & \equiv & P \sqcap \exists r.P, \\ \sigma_2(Y) & := & lcs\{P \sqcap Q \sqcap \exists r.P,\ Q\} \\ & \equiv & Q. \end{array}$$

Since $C \not\sqsupseteq \sigma_1(D)$ and $C \sqsupseteq \sigma_2(D)$, the output of the algorithm is $\{\sigma_2\}$.

4 The complexity of matching in $\mathcal{EL}$

We can show that our matching algorithm always computes an s-complete (and thus also i-complete) set of matchers and that it runs in time exponential in the size of the matching problem [1].

Theorem 3 For every $\mathcal{EL}$-matching problem there exists a s-complete (i-complete) set of matchers with size at most exponential in the size of the matching problem, and this set can be computed in exponential time.

The following example demonstrates that this upper bound is optimal.

Example 4 Consider the $\mathcal{EL}$ matching problem $\exists r.A \sqcap \exists r.B \equiv^? \exists r.X_1 \sqcap \cdots \sqcap \exists r.X_n$, where A, B are concept names and X_i, $1 \leq i \leq n$, are concept variables.

For a word $w = a_1 \cdots a_n \in \{A, B\}^n$ we define $\sigma_w(X_i) := a_i$ for every $1 \leq i \leq n$. Obviously, if w contains both A and B, σ_w is a matcher of the matching problem. It is not hard to verify that each of these matchers σ_w must be contained (modulo equivalence) in every s-complete set. Since there are exponentially many words of length n containing both A and B, we can conclude that every s-complete set is exponential in the size of the matching problem.

Proposition 5 The cardinality of s-complete sets of matchers may grow exponentially in the sizes of the $\mathcal{EL}$-matching problems.

Since the lcs of a sequence of $\mathcal{EL}$-concept description may grow exponentially in the size of the sequence, not only the cardinality of a complete set of matchers, but also the size of a single matcher may grow exponentially.

Deciding solvability of matching problems in $\mathcal{EL}$ The algorithm described in Fig. 2 always computes an s-complete set of matchers. Consequently, the matching problem has a solution iff this set is non-empty. This provides us with an exponential time decision procedure for matching modulo equivalence in $\mathcal{EL}$. However, it can be shown [1] that every solvable $\mathcal{EL}$-matching problem has a matcher of size polynomially bounded in the size of the matching problem. As a result, there even exists a non-deterministic polynomial decision procedure. Furthermore, we can show that the problem of matching modulo equivalence in $\mathcal{EL}$ is NP-hard by a reduction of SAT [6] to the matching problem. In contrast, solvability of matching problems *modulo subsumption* can be decided in polynomial time since it can be reduced to subsumption by replacing all variables in the concept pattern by $\top$.

Proposition 6 Deciding solvability of an $\mathcal{EL}$-matching problem modulo equivalence is NP-complete.

In [1], NP-hardness is proved by a reduction of SAT that uses only a fixed number of concept names and role names. Here we give a simpler reduction for which, however, the number of concept names and role names grows with the given formula.

Let $\phi = p_1 \wedge \cdots \wedge p_m$ be a propositional formula 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 role names $r_1, \ldots, r_n$ and $s_1, \ldots, s_m$.

First, we show that we can specify a matching problem such that X_i must be replaced by A and $\overline{X}_i$ by B (corresponding to $x_i = \mathit{true}$) or vice versa (corresponding to $x_i = \mathit{false}$). This matching problem is given by $C \equiv^? D$, where

$$\begin{aligned} C &:= \exists r_1.A \sqcap \exists r_1.B \sqcap \cdots \sqcap \exists r_n.A \sqcap \exists r_n.B, \\ D &:= \exists r_1.X_1 \sqcap \exists r_1.\overline{X}_1 \sqcap \cdots \sqcap \exists r_n.X_n \sqcap \exists r_n.\overline{X}_n. \end{aligned}$$

The matchers of this problem are exactly the substitutions that replace X_i by A and $\overline{X}_i$ by B, or vice versa.

In order to encode ϕ, we first introduce a concept pattern D_{p_i} for each conjunct p_i. For example, if $p_i = x_1 \vee \overline{x}_2 \vee x_3 \vee \overline{x}_4$, then $D_{p_i} := X_1 \sqcap \overline{X}_2 \sqcap X_3 \sqcap \overline{X}_4 \sqcap B$. The whole formula is then represented by the matching problem $C' \equiv^? D'$, where

$$\begin{aligned} C' &:= \exists s_1.(A \sqcap B) \sqcap \cdots \sqcap \exists s_m.(A \sqcap B), \\ D' &:= \exists s_1.D_{p_1} \sqcap \cdots \sqcap \exists s_m.D_{p_m}. \end{aligned}$$

This matching problem ensures that, among the variables in D_{p_i}, at least one must be replaced by A. This corresponds to the fact that, within one conjunct p_i, there must be at least one literal that evaluates to *true*. Note that we need the concept B in D_{p_i} to cover the case where all variables in D_{p_i} are substituted by A.

We combine the two matching problems introduced above into a single problem $C \sqcap C' \equiv^? D \sqcap D'$. It is easy to verify that ϕ is satisfiable iff this matching problem is solvable.

5 Extension to $\mathcal{ALE}$

In order to handle $\mathcal{ALE}$-matching problems, the matching algorithm for $\mathcal{EL}$ must be modified in two respects. First, the concept description C must be normalized using the rules in [3] before translating it into a description tree $\mathcal{G}_C$. In principle, these rules propagate value restrictions onto existential restrictions, and replace $\forall r.\top$ by $\top$. As shown in [3], the size of this normalized description may be exponential in the size of the original description. Second, instead of computing all homomorphisms between $\mathcal{G}_D$ and $\mathcal{G}_C$, one must consider all possible concepts D' obtained by replacing some of the variables in D by $\top$, and compute all homomorphisms between $\mathcal{G}_{D'}$ and

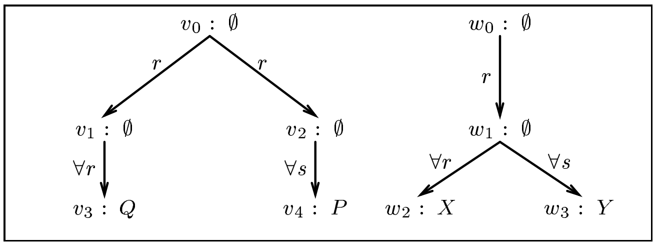

Figure 3: The description trees for C and D.

$\mathcal{G}_C$. (Of course, the notion of a description tree and of a homomorphism must be adapted to the larger language; see [3] for details.) The need for the latter modification is demonstrated by the following example:

Example 7 Consider the $\mathcal{ALE}$-matching problem $C \sqsubseteq^? D$, where

$$C := (\exists r.\forall r.Q) \sqcap (\exists r.\forall s.P) \text{ and } D := \exists r.(\forall r.X \sqcap \forall s.Y).$$

The description trees for C and D are depicted in Figure 3. In addition to the edges representing existential restrictions ($\exists$-edge), these tress contain $\forall$-edges, which represent value-restrictions. (The trees in our example need not be normalized since they are already in normal form.) Obviously, $\sigma := \{X \mapsto Q, Y \mapsto \top\}$ and $\tau := \{X \mapsto \top, Y \mapsto P\}$ are solutions of 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 to v_1 or to v_2. In the former case, w_2 can be mapped to v_3, but there is no appropriate image for w_3. In the latter case, w_3 can be mapped to v_4, but then there is no appropriate node w_2 can be mapped to.

Now, consider the concept pattern D' obtained from D by replacing Y by $\top$. The normalization process removes the node w_3 (which corresponds to the fact that $\forall s.\top \equiv \top$), and thus the normalized description tree $\mathcal{G}_{D'}$ consists of the nodes w_0, w_1, w_2. Obviously, $\{w_0 \mapsto v_0, w_1 \mapsto v_1, w_2 \mapsto v_3\}$ is a homomorphism between $\mathcal{G}_{D'}$ and $\mathcal{G}_C$. This homomorphism corresponds to the matcher σ. The matcher τ can be obtained analogously (by replacing X by $\top$).

From a conceptual point of view, it is thus not hard to turn our matching algorithm for $\mathcal{EL}$ into one that can also handle $\mathcal{ALE}$. However, the run-time complexity of the algorithm increases considerably. The following results for matching in $\mathcal{ALE}$ are proved in [1]:

Theorem 8
1. There is an algorithm for computing s-complete sets of $\mathcal{ALE}$-matchers, which uses at most exponential space.
2. Solvability of $\mathcal{ALE}$-matching problems modulo equivalence is NP-hard, and it can be decided in nondeterministic exponential time.
3. Solvability of $\mathcal{ALE}$-matching problems modulo subsumption is an NP-complete problem.

6 Conclusion and future work

We have seen that existential restrictions have two unpleasant effects w.r.t. matching: (1) There need no longer exist a unique least matcher, and (2) the complexity of computing complete sets of matchers and of deciding solvability of matching problems is no longer polynomial. In contrast, for the language $\mathcal{ALN}$, which does not allow for existential restrictions, solvability of matching problems can be decided in polynomial time, and a solvable matching problem always has a least solution [2].

For $\mathcal{ALE}$, the complexity of our matching algorithm is not optimal w.r.t. the known lower bounds. Thus, our short-term goal is to obtain tighter complexity bounds for matching in that DL. We will also try to extend the results to DLs allowing for number restrictions, and—in the long run—to DLs allowing for full negation. We conjecture, however, that DLs with full negation will require techniques quite different from the ones used in this work.

References

[1] F. Baader and R. Küsters. Matching in description logics with existential restrictions. Technical Report LTCS-Report 99-07, LuFg Theoretical Computer Science, RWTH Aachen, Germany, 1999. See http://www-lti.informatik.rwth-aachen.de/Forschung/Papers.html.

[2] F. Baader, R. Küsters, A. Borgida, and D.L. McGuinness. Matching in description logics. *Journal of Logic and Computation*, 9, 1999.

[3] F. Baader, R. Küsters, and R. Molitor. Computing least common subsumer in description logics with existential restrictions. In *Proceedings of the 16th International Joint Conference on Artificial Intelligence (IJCAI'99)*, 1999.

[4] F. Baader and J.H. Siekmann. Unification theory. In D.M. Gabbay, C.J. Hogger, and J.A. Robinson, editors, *Handbook of Logic in Artificial Intelligence and Logic Programming*, pages 41–125. Oxford University Press, Oxford, UK, 1994.

[5] 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.

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

[7] D.L. McGuinness. *Explaining Reasoning in Description Logics.* PhD thesis, Department of Computer Science, Rutgers University, October, 1996.