Matching in Description Logics: Preliminary Results

Franz Baader
Theoretical Computer Science
RWTH Aachen
52074 Aachen, Germany

Alex Borgida
Dept. of Computer Science
Rutgers University
New Brunswick, NJ, USA

Deborah L. McGuinness
Artificial Intelligence Principles
AT&T Labs–Research
Florham Park, NJ, USA

Abstract

Matching of concepts with variables (concept patterns) is a relatively new operation that has been introduced in the context of concept description languages (description logics), originally to help discard unimportant aspects of large concepts appearing in industrial-strength knowledge bases. This paper proposes a new approach to performing matching, based on a "concept-centered" normal form, rather than the more standard "structural subsumption" normal form for concepts. As a result, matching can be performed (in polynomial time) using *arbitrary* concept patterns of the description language $\mathcal{FL}_\neg$, thus removing restrictions from previous work. The paper also addresses the question of matching problems with additional "side conditions", which were motivated by practical experience.

1 Introduction

The traditional inference problems for Description Logic (DL) systems (like subsumption) are now well-investigated. This means that algorithms are available for solving the subsumption problem and related inference problems in a great variety of DL languages of differing expressive power. In addition, the computational complexity of these inference problems has been investigated in detail. It has turned out, however, that building and maintaining large DL knowledge bases requires support by additional inference capabilities, which have not been considered in the DL literature until very recently. The present paper is concerned with such a new inference service, namely, *matching* of concept descriptions.

Matching of Description Logic concepts was introduced in the CLASSIC system (version 2), under the name of "*filtering*", as a technique for specifying which aspects of a concept should be selected for printing or explanation. The need for this facility became apparent when dealing with large knowledge bases, involving concepts whose description spans multiple pages of output: in many cases, such concepts carried details that either were obviously true (e.g., the age of a person is a number) or were intended for some internal function (e.g., graphical display) rather than domain modeling. In either case, both the printing and the explaining of results provided by the more traditional inference services [8, 7] required pruning. In projects using CLASSIC, pruning of the descriptions resulted in concepts that were approximately an order of magnitude smaller. In small applications such as [9], this actually saved 3–5 pages of printout; in larger applications such as [11, 10] it might save up to 30 pages.

This pruning mechanism was first formalized in [7] as a purely syntactic *match* involving terms/concepts with variables, and then given a semantics and a syntactic implementation in [3]. Given a concept pattern D (i.e., a concept description containing variables) and a concept description C without variables, the matching problem introduced in [3] asks for a substitution σ (of the variables by concept descriptions) such that $C \sqsubseteq \sigma(D)$. More precisely, one is interested in a "minimal" solution of the matching problem, i.e., σ should satisfy the property that there does not exist a substitution δ such that $C \sqsubseteq \delta(D) \sqsubset \sigma(D)$. For example, the minimal matcher of the pattern $D := \forall \mathsf{research\text{-}interests}.X$ against the description C:

$$\forall \mathsf{pets}.\mathsf{Cat} \sqcap \forall \mathsf{research\text{-}interests}.\mathsf{ArtInt} \sqcap \forall \mathsf{hobbies}.\mathsf{Gardening}$$

assigns ArtInt 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 (or explained), if the pattern matches successfully against a specific concept. If there is no match, nothing is displayed.)

In some cases, this pruning effect can be improved by imposing additional side conditions on the solutions of matching problems. For example, the information that the research interests lie in the area of Artificial Intelligence may not provide interesting information if our knowledge base is concerned only with AI researchers. A *side condition* stating that the solutions for the variable X must be *subsumed* by $\mathsf{KnowlRep}$ would make sure that matching succeeds only if the research interests belong to (a subfield of) Knowledge Representation. Thus, the description C from above no longer matches the pattern D, whereas C':

$$\forall \mathsf{pets.Cat} \sqcap \forall \mathsf{research\text{-}interests.DL} \sqcap \forall \mathsf{hobbies.Gardening}$$

would still yield a solution (provided that DL is defined by a description that is subsumed by $\mathsf{KnowlRep}$). Side conditions become especially useful once we have individual role fillers that can match variables, since they allow us to state complex conditions on the matching individual.

In some cases we would like to have a matching process which succeeds only if the variable X is substituted for by a value that is *strictly subsumed* by some description (or pattern). The utility of such strict side-conditions might be more clearly seen in an example where the concept Person is known to have Number restrictions on the age attribute, and we are interested in seeing the value restriction for age only if it represents some *additional* (i.e., stricter) constraint. Another point worth noting is that according to the standard Description Logic semantics, every description is subsumed by all concepts of the form $\forall R.\top$, where $\top$ denotes the universal concept. Hence the pattern D above (concerning research interests) matches every concept. Side conditions requiring the value substituted for a variable to be strictly subsumed by $\top$ prevent such "trivial" matches.

Matching algorithms for a DL containing most of the constructs available in CLASSIC are introduced in [7] and [3]. These algorithms are based on the role-centered normal form[1] of concept descriptions usually employed by structural subsumption algorithms. The main drawback of these algorithms is that they cannot treat arbitrary matching problems since they require the concept pattern to be in structural normal form.

In [2], Baader and Narendran consider unification of concept descriptions in $\mathcal{FL}_0$, which allows for conjunction ($\sqcap$), value restriction ($\forall R.C$), and the top concept ($\top$). Matching modulo equivalence, i.e., the question whether, for a given pattern D and a description C, there exists a substitution σ such that $C \equiv \sigma(D)$, can be seen as a special case of unification where one of the descriptions (namely C) does not contain variables. Since $C \sqsubseteq \sigma(D)$ iff $C \equiv \sigma(C \sqcap D)$, matching modulo subsumption (as introduced above) is an instance of matching modulo equivalence. The polynomial matching algorithm described in [2] does not impose restrictions on the form of the patterns. However, it is restricted to the small language $\mathcal{FL}_0$.

In the present paper, we show that this algorithm can be extended to treat matching in languages allowing for inconsistent concept descriptions, namely $\mathcal{FL}_\perp$, which extends $\mathcal{FL}_0$ by the bottom concept ($\perp$), and $\mathcal{FL}_\neg$, which extends $\mathcal{FL}_\perp$ by primitive negation ($\neg A$, where A is an atomic concept). In addition, we consider matching under additional conditions on the variable bindings, which also arose in examples in [9, 7] and were responsible for about 25% of our space savings in our deployed example. In this paper, we consider two different variants of these "side conditions": subsumption conditions and strict subsumption conditions. Subsumption conditions are of the form $X \sqsubseteq E$, where X is a variable and E is a pattern (i.e., it may contain variables), and they restrict the matchers to substitutions σ satisfying $\sigma(X) \sqsubseteq \sigma(E)$. It should be noted that such a side condition is not a matching problem since variables may occur on both sides. We shall see, however, that in many cases matching under subsumption conditions can be reduced in polynomial time to matching without subsumption conditions. In contrast, strict subsumption conditions may increase the complexity of the matching problem. Such conditions are of the form $X \sqsubset E$, where X is a variable and E is a pattern, and they restrict the matchers to substitutions σ satisfying $\sigma(X) \sqsubseteq \sigma(E)$ and $\sigma(X) \not\equiv \sigma(E)$. Even for the small languages $\mathcal{FL}_0$, matching under strict subsumption conditions is NP-hard.

Except for the proofs of the two main lemmas, which provide us with our polynomial matching algorithm, we omit all the proofs. Detailed proofs can be found in [1].

2 Formal preliminaries

In this section, we first introduce syntax and semantics of the description languages considered in this paper. Then, we formally introduce matching problems, and state some simple results about matching problems and their solutions.

Definition 1 Let $\mathcal{C}$ and $\mathcal{R}$ be disjoint finite sets representing the set of *atomic concepts* and the set of *atomic roles*. The set of all *$\mathcal{FL}_\neg$-concept descriptions* over $\mathcal{C}$ and $\mathcal{R}$ is inductively defined as follows:

- Every element of $\mathcal{C}$ is a concept description (atomic concept).
- The symbols $\top$ (top concept) and $\perp$ (bottom concept) are concept descriptions.
- If $A \in \mathcal{C}$, then $\neg A$ is a concept description (atomic negation).
- If C and D are concept descriptions, then $C \sqcap D$ is a concept description (concept conjunction).

[1] We call this normal form "role-centered" since it groups sub-descriptions by role names, whereas the concept-centered normal form used in this paper groups value restrictions by concept names.

- If C is a concept description and $R \in \mathcal{R}$ is an atomic role, then $\forall R.C$ is a concept description (value restriction).

In the sublanguage $\mathcal{FL}_0$ of $\mathcal{FL}_\neg$, atomic negation and $\bot$ may not be used, whereas in $\mathcal{FL}_\bot$ only atomic negation is disallowed.

The following definition provides a model-theoretic semantics for $\mathcal{FL}_\neg$ and its sublanguages:

Definition 2 An *interpretation* I consists of a nonempty set Δ^I, the domain of the interpretation, and an interpretation function that assigns to every atomic concept $A \in \mathcal{C}$ a set $A^I \subseteq \Delta^I$, and to every atomic role $R \in \mathcal{R}$ a binary relation $R^I \subseteq \Delta^I \times \Delta^I$. The interpretation function is extended to complex concept descriptions as follows:

$$\begin{array}{rcl} \top^I &:=& \Delta^I, \\ \bot^I &:=& \emptyset, \\ (\neg A)^I &:=& \Delta^I \setminus A^I, \\ (C \sqcap D)^I &:=& C^I \cap D^I, \\ (\forall R.C)^I &:=& \{d \in \Delta^I \mid \forall e \in \Delta^I: \\ && (d,e) \in R^I \to e \in C^I\}. \end{array}$$

Based on this semantics, subsumption and equivalence of concept descriptions is defined as follows: Let C and D be $\mathcal{FL}_\neg$-concept descriptions.

- C is *subsumed* by D ($C \sqsubseteq D$) iff $C^I \subseteq D^I$ for all interpretations I.
- C is *equivalent* to D ($C \equiv D$) iff $C^I = D^I$ for all interpretations I.
- C is *strictly subsumed* by D ($C \sqsubset D$) iff $C \sqsubseteq D$ and $C \not\equiv D$.

In order to define matching of concept descriptions, we must introduce the notion of a concept pattern and of substitutions operating on patterns. For this purpose, we introduce an additional set of symbols $\mathcal{X}$ (concept variables), which is disjoint from $\mathcal{C} \cup \mathcal{R}$.

Definition 3 The set of all $\mathcal{FL}_\neg$*-concept patterns* over $\mathcal{C}$, $\mathcal{R}$, and $\mathcal{X}$ is inductively defined as follows:

- Every concept variable $X \in \mathcal{X}$ is a pattern.
- Every $\mathcal{FL}_\neg$-concept description over $\mathcal{C}$ and $\mathcal{R}$ 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 $R \in \mathcal{R}$ is an atomic role, then $\forall R.C$ is a concept pattern.

Thus, concept variables can be used like atomic concepts, with the only difference being that atomic negation may not be applied to variables.

A *substitution* σ is a mapping from $\mathcal{X}$ into the set of all $\mathcal{FL}_\neg$-concept descriptions. This mapping is extended to concept patterns in the obvious way, i.e.,

- $\sigma(A) := A$ and $\sigma(\neg A) := \neg A$ for all $A \in \mathcal{C}$,
- $\sigma(\top) := \top$ and $\sigma(\bot) := \bot$,
- $\sigma(C \sqcap D) := \sigma(C) \sqcap \sigma(D)$, and
- $\sigma(\forall R.C) := \forall R.\sigma(C)$.

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

Obviously, the result of applying a substitution to an $\mathcal{FL}_\neg$-concept pattern is an $\mathcal{FL}_\neg$-concept description.[2] An $\mathcal{FL}_0$-substitution maps concept variables to $\mathcal{FL}_0$-concept descriptions. $\mathcal{FL}_\bot$-substitutions are defined analogously.

Subsumption can be extended to substitutions as follows. The substitution σ is subsumed by the substitution τ ($\sigma \sqsubseteq \tau$) iff $\sigma(X) \sqsubseteq \tau(X)$ for all variables $X \in \mathcal{X}$.

Definition 4 An $\mathcal{FL}_\neg$*-matching problem* is of the form $C \equiv^? D$ where C is an $\mathcal{FL}_\neg$-concept description and D is an $\mathcal{FL}_\neg$-concept pattern. A *solution* or *matcher* of this problem is a substitution σ such that $C \equiv \sigma(D)$.

A *subsumption condition* in $\mathcal{FL}_\neg$ is of the form $X \sqsubseteq^? E$ where X is a concept variable and E is an $\mathcal{FL}_\neg$-concept pattern. The substitution σ satisfies this condition iff $\sigma(X) \sqsubseteq \sigma(E)$.

A *strict subsumption condition* in $\mathcal{FL}_\neg$ is of the form $X \sqsubset^? E$ where X is a concept variable and E is an $\mathcal{FL}_\neg$-concept pattern. The substitution σ satisfies this condition iff $\sigma(X) \sqsubset \sigma(E)$.

Matching problems and (strict) subsumption conditions in $\mathcal{FL}_0$ and $\mathcal{FL}_\bot$ are defined analogously. Note that also the solutions are then constrained to belong to the respective sublanguage.

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 iff it is a solution of all the matching problems $C_i \equiv^? D_i$ contained in the system. However, it is easy to see that solving systems of matching problems can be reduced (in linear time) to solving a single matching problem.

Lemma 5 *Let $R_1, \ldots, R_m$ be distinct atomic roles. Then σ solves the system $\{C_1 \equiv^? D_1, \ldots, C_m \equiv^? D_m\}$ iff it solves the single matching problem*

$$\forall R_1.C_1 \sqcap \cdots \sqcap \forall R_m.C_m \equiv^? \forall R_1.D_1 \sqcap \cdots \sqcap \forall R_m.D_m.$$

[2] Note that this would not be the case if we had allowed the application of negation to concept variables.

Consequently, we may (without loss of generality) restrict our attention to single matching problems with or without finite sets of (strict) subsumption conditions.

In [3, 7], a different type of matching problems has been considered. We will refer to those problems as matching modulo subsumption in order to distinguish them from the matching problems modulo equivalence introduced above.

Definition 6 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 7 *The substitution σ solves the matching problem $C \sqsubseteq^? D$ iff it solves $C \equiv^? C \sqcap D$.*

For $\mathcal{FL}_\neg$, and 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 8 *Let $C \sqsubseteq^? D$ be a matching problem modulo subsumption in $\mathcal{FL}_\neg$, and let $\sigma_\top$ be the substitution that replaces each variable by $\top$. Then $C \sqsubseteq^? D$ has a solution iff $\sigma_\top$ solves $C \sqsubseteq^? D$.*

Thus, solvability of matching problems modulo subsumption in $\mathcal{FL}_\neg$ and its sublanguages is not an interesting new problem. This changes, however, if we consider such matching problems together with additional (strict) subsumption conditions. In fact, these conditions may exclude the trivial solution $\sigma_\top$. In addition, one is usually not interested in an arbitrary solution of the matching problem $C \sqsubseteq^? D$, but rather in computing a "minimal" solution:

Definition 9 Let $C \sqsubseteq^? D$ be a matching problem modulo subsumption. The solution σ of $C \sqsubseteq^? D$ is called *minimal* iff there does not exist a substitution δ such that $C \sqsubseteq \delta(D) \sqsubset \sigma(D)$.

Lemma 10 *Let $C \sqsubseteq^? D$ be an $\mathcal{FL}_\neg$-matching problem modulo subsumption. If σ is the least solution of $C \sqsubseteq^? D$ w.r.t. subsumption of substitutions, i.e., $\sigma \sqsubseteq \delta$ for all solutions δ, then σ is also a minimal solution.*

It should be noted that talking about *the* least solution is a slight abuse of language since the least solution of a given matching problem is unique only up to equivalence: if σ and τ are both least solutions of the same matching problem then they subsume each other, which means that $\sigma(X) \equiv \tau(X)$ for all variables $X \in \mathcal{X}$.

The converse of Lemma 10 need not hold. For example, for the matching problem $\forall R.A \sqsubseteq^? \forall R.A \sqcap \forall R.X$, the substitutions $\sigma := \{X \mapsto A\}$ and $\tau := \{X \mapsto \top\}$ are both minimal solutions, but τ obviously cannot be a least solution. This example also demonstrates that minimal solutions of a given matching problem need not be unique up to equivalence.

3 Matching in $\mathcal{FL}_\bot$

The purpose of this section is to show that solvability of $\mathcal{FL}_\bot$-matching problems can be decided in polynomial time. In addition, for matching problems modulo subsumption we can compute a minimal solution in polynomial time. Our algorithm is based on a "concept-centered" normal form for $\mathcal{FL}_\bot$-concept descriptions.

First, let us recall the concept-centered normal form for $\mathcal{FL}_0$-concept descriptions introduced in [2]. It is easy to see that any $\mathcal{FL}_0$-concept description can be transformed into an equivalent description that is either $\top$ or a (nonempty) conjunction of descriptions of the form $\forall R_1.\cdots\forall R_m.A$ for $m \geq 0$ (not necessarily distinct) atomic roles $R_1, \ldots, R_m$ and an atomic concept $A \neq \top$. We abbreviate $\forall R_1.\cdots\forall R_m.A$ by $\forall R_1 \ldots R_m.A$, where $R_1 \ldots R_m$ is considered as a word over the alphabet $\Sigma := \mathcal{R}$ of all atomic roles. In addition, instead of $\forall w_1.A \sqcap \ldots \sqcap \forall w_\ell.A$ we write $\forall L.A$ where $L := \{w_1, \ldots, w_\ell\}$ is a finite set of words over Σ. The term $\forall\emptyset.A$ is considered to be equivalent to $\top$. Using these abbreviations, any pair of $\mathcal{FL}_0$-concept descriptions C, D containing the atomic concepts $A_1, \ldots, A_k$ can be rewritten as

$$\begin{aligned} C &\equiv \forall U_1.A_1 \sqcap \ldots \sqcap \forall U_k.A_k, \\ D &\equiv \forall V_1.A_1 \sqcap \ldots \sqcap \forall V_k.A_k, \end{aligned}$$

where U_i, V_i are finite sets of words over the alphabet of all atomic roles. This normal form provides us with the following characterization of equivalence of $\mathcal{FL}_0$-concept descriptions [2]:

Lemma 11 *Let C, D be $\mathcal{FL}_0$-concept descriptions with normal forms as introduced above. Then $C \equiv D$ iff $U_i = V_i$ for all $i, 1 \leq i \leq k$.*

This characterization can in turn be used to reduce matching of $\mathcal{FL}_0$-concept descriptions to a certain formal language problem, which can easily be shown to be solvable in polynomial time (see [2] for details).

If we treat $\bot$ like an arbitrary atomic concept, $\mathcal{FL}_\bot$-concept descriptions C, D can still be represented in the form[3]

$$\begin{aligned} C &\equiv \forall U_0.\bot \sqcap \forall U_1.A_1 \sqcap \ldots \sqcap \forall U_k.A_k, \\ D &\equiv \forall V_0.\bot \sqcap \forall V_1.A_1 \sqcap \ldots \sqcap \forall V_k.A_k. \end{aligned}$$

[3]We shall call this the $\mathcal{FL}_0$-normal form of the descriptions.

However, equivalence of the descriptions no longer corresponds to equality of the languages U_i and V_i. The reason is that $\forall R_1.\cdots\forall R_m.\bot$ is subsumed by any value restriction of the form $\forall R_1.\cdots\forall R_m.\forall R_{m+1}.\cdots\forall R_{m+n}.A$. This fact is taken into account by the following characterization of equivalence of $\mathcal{FL}_\bot$-concept descriptions:

Lemma 12 *Let C, D be $\mathcal{FL}_\bot$-concept descriptions with $\mathcal{FL}_0$-normal forms as introduced above. Then*

$$C \equiv D \quad \textit{iff} \quad \begin{array}{l} U_0{\cdot}\Sigma^* = V_0{\cdot}\Sigma^* \textit{ and} \\ U_i \cup U_0{\cdot}\Sigma^* = V_i \cup V_0{\cdot}\Sigma^* \\ \textit{for all } i, 1 \leq i \leq k, \end{array}$$

where Σ^* is the set of all words over the alphabet of all atomic roles and $\cdot$ stands for concatenation.

If D is an $\mathcal{FL}_\bot$-pattern containing the variables $X_1, \ldots, X_\ell$, then its $\mathcal{FL}_0$-normal form is of the form

$$\begin{array}{rcl} D & \equiv & \forall V_0.\bot \sqcap \forall V_1.A_1 \sqcap \ldots \sqcap \forall V_k.A_k \sqcap \\ & & \forall W_1.X_1 \sqcap \ldots \sqcap \forall W_\ell.X_\ell. \end{array}$$

If we want to match D with the description C (with normal form as above), we must solve the following "formal language" equations (where $X_{j,i}$ are interpreted as variables for finite sets of words):

$$(\bot) \quad U_0{\cdot}\Sigma^* = V_0{\cdot}\Sigma^* \cup W_1{\cdot}X_{1,0}{\cdot}\Sigma^* \cup \ldots \cup W_\ell{\cdot}X_{\ell,0}{\cdot}\Sigma^*,$$

and for all $i, 1 \leq i \leq k$,

$$(A_i) \quad U_i \cup U_0{\cdot}\Sigma^* = V_i \cup W_1{\cdot}X_{1,i} \cup \ldots \cup W_\ell{\cdot}X_{\ell,i} \cup U_0{\cdot}\Sigma^*.$$

Theorem 13 *Let C be an $\mathcal{FL}_\bot$-concept description and D an $\mathcal{FL}_\bot$-concept pattern with $\mathcal{FL}_0$-normal forms as introduced above. Then the matching problem $C \equiv^? D$ has a solution iff the formal language equations $(\bot)$ and $(A_1), \ldots, (A_k)$ are each solvable.*

Example 14 As a running example, we will consider the problem of matching the pattern

$$D := X_1 \sqcap (\forall R.X_1) \sqcap (\forall S.X_2)$$

against the description

$$C := \forall R.((\forall S.A_1) \sqcap (\forall R.\bot)) \sqcap \forall S.\forall S.\bot.$$

The $\mathcal{FL}_\bot$-normal forms of C and D are

$$\begin{array}{rcl} C & \equiv & \forall\{RR, SS\}.\bot \sqcap \forall\{RS\}.A_1, \\ D & \equiv & \forall\emptyset.\bot \sqcap \forall\emptyset.A_1 \sqcap \forall\{\varepsilon, R\}.X_1 \sqcap \forall\{S\}.X_2. \end{array}$$

Thus, the matching problem $C \equiv^? D$ is translated into the following two equations:

$$\begin{array}{ll} (\bot) & \{RR, SS\}{\cdot}\Sigma^* = \emptyset{\cdot}\Sigma^* \cup \{\varepsilon, R\}{\cdot}X_{1,0}{\cdot}\Sigma^* \cup \\ & \qquad \{S\}{\cdot}X_{2,0}{\cdot}\Sigma^* \\ (A_1) & \{RS\} \cup \{RR, SS\}{\cdot}\Sigma^* = \emptyset \cup \{\varepsilon, R\}{\cdot}X_{1,1} \cup \\ & \qquad \{S\}{\cdot}X_{2,1} \cup \{RR, SS\}{\cdot}\Sigma^* \end{array}$$

If we want to utilize Theorem 13 for deciding matching problems in $\mathcal{FL}_\bot$, we must show how solvability of the equations $(\bot)$, (A_1), ..., (A_k) can be tested. First, we address the problem of solving equation $(\bot)$.

Lemma 15 *Equation $(\bot)$ has a solution iff replacing $X_{j,0}{\cdot}\Sigma^*$ by the sets*

$$\bigcap_{w \in W_j} w^{-1}{\cdot}(U_0{\cdot}\Sigma^*)$$

solves equation $(\bot)$.[4]

Proof. To show the *only-if direction*, we assume that the assignment $X_{1,0} := M_{1,0}, \ldots, X_{\ell,0} := M_{\ell,0}$ solves equation $(\bot)$.

First, we prove that $M_{j,0}{\cdot}\Sigma^* \subseteq \bigcap_{w \in W_j} w^{-1}{\cdot}(U_0{\cdot}\Sigma^*)$ holds for all $j, 1 \leq j \leq \ell$. Thus, let $v \in M_{j,0}{\cdot}\Sigma^*$ and $w \in W_j$. Since $W_j{\cdot}M_{j,0}{\cdot}\Sigma^* \subseteq U_0{\cdot}\Sigma^*$, we know that $wv \in U_0{\cdot}\Sigma^*$, and thus $v \in w^{-1}{\cdot}(U_0{\cdot}\Sigma^*)$. This shows that $M_{j,0}{\cdot}\Sigma^* \subseteq w^{-1}{\cdot}(U_0{\cdot}\Sigma^*)$ for all $w \in W_j$, and thus $M_{j,0}{\cdot}\Sigma^* \subseteq \bigcap_{w \in W_j} w^{-1}{\cdot}(U_0{\cdot}\Sigma^*)$.

As an immediate consequence, we obtain

$$\begin{array}{rcl} U_0{\cdot}\Sigma^* & = & V_0{\cdot}\Sigma^* \cup W_1{\cdot}M_{1,0}{\cdot}\Sigma^* \cup \ldots \cup W_\ell{\cdot}M_{\ell,0}{\cdot}\Sigma^* \\ & \subseteq & V_0{\cdot}\Sigma^* \cup W_1{\cdot}\displaystyle\bigcap_{w \in W_1} w^{-1}{\cdot}(U_0{\cdot}\Sigma^*) \cup \ldots \cup \\ & & W_\ell{\cdot}\displaystyle\bigcap_{w \in W_\ell} w^{-1}{\cdot}(U_0{\cdot}\Sigma^*). \end{array}$$

It remains to be shown that the inclusion in the other direction holds as well. Obviously, we have $V_0{\cdot}\Sigma^* \subseteq U_0{\cdot}\Sigma^*$ since there exists a solution of $(\bot)$. To conclude the proof of the only-if direction, assume that $u \in W_j$ and $v \in \bigcap_{w \in W_j} w^{-1}{\cdot}(U_0{\cdot}\Sigma^*)$. We must show that $uv \in U_0{\cdot}\Sigma^*$. Obviously, $u \in W_j$ implies $v \in u^{-1}{\cdot}(U_0{\cdot}\Sigma^*)$, and thus $uv \in U_0{\cdot}\Sigma^*$.

To prove the *if direction*, it is sufficient to show that there exist finite sets of words $L_{j,0}$ $(j = 1, \ldots, \ell)$ such that $L_{j,0}{\cdot}\Sigma^* = \bigcap_{w \in W_j} w^{-1}{\cdot}(U_0{\cdot}\Sigma^*)$. This is an immediate consequence of the fact that languages of the form $L{\cdot}\Sigma^*$ for finite L are closed under (binary) intersection and left quotients (see (1) and (2) of Lemma 16 below). $\square$

Lemma 16 *Let U, V be finite languages and w a word.*

1. *There exists a finite language L_1 such that $L_1{\cdot}\Sigma^* = w^{-1}{\cdot}(U{\cdot}\Sigma^*)$.*
2. *There exists a finite language L_2 such that $L_2{\cdot}\Sigma^* = U{\cdot}\Sigma^* \cap V{\cdot}\Sigma^*$.*
3. *$U{\cdot}\Sigma^* \cup V{\cdot}\Sigma^* = (U \cup V){\cdot}\Sigma^*$ and $U{\cdot}(V{\cdot}\Sigma^*) = (U{\cdot}V){\cdot}\Sigma^*$.*

[4] For a word w and a set of words L we have $w^{-1}{\cdot}L := \{u \mid wu \in L\}$. This language is called a *left quotient* of L.

For the matching problem of Example 14, we replace $X_1 \cdot \Sigma^*$ by

$$R^{-1} \cdot (\{RR, SS\} \cdot \Sigma^*) \cap \varepsilon^{-1} \cdot (\{RR, SS\} \cdot \Sigma^*) = \{R\} \cdot \Sigma^* \cap \{RR, SS\} \cdot \Sigma^* = \{RR\} \cdot \Sigma^*$$

and $X_2 \cdot \Sigma^*$ by

$$S^{-1} \cdot (\{RR, SS\} \cdot \Sigma^*) = \{S\} \cdot \Sigma^*.$$

It is easy to see that this replacement solves the equation. The finite languages $L_{j,0}$ are defined as $L_{1,0} := \{RR\}$ and $L_{2,0} := \{S\}$.

Now, let us consider the equations (A_i) for $1 \leq i \leq k$.

Lemma 17 *Equation (A_i) has a solution iff replacing the variables $X_{j,i}$ by the sets $\widehat{L}_{j,i} := \bigcap_{w \in W_j} w^{-1} \cdot (U_i \cup U_0 \cdot \Sigma^*)$ yields a solution of (A_i).*

Proof. The proof of the *only-if direction* is very similar to the proof of this direction for Lemma 15. In particular, one can show that any assignment $X_{1,i} := M_{1,i}, \ldots, X_{\ell,i} := M_{\ell,i}$ that solves (A_i) satisfies $M_{j,i} \subseteq \widehat{L}_{j,i}$.

To prove the *if direction*, it is sufficient to show that there exist finite sets of words $L_{j,i}$ such that $W_j \cdot L_{j,i} \cup U_0 \cdot \Sigma^* = W_j \cdot \widehat{L}_{j,i} \cup U_0 \cdot \Sigma^*$.

We have $\widehat{L}_{j,i} = \bigcap_{w \in W_j} (w^{-1} \cdot U_i \cup w^{-1} \cdot (U_0 \cdot \Sigma^*))$. By applying distributivity of intersection over union, this intersection of unions can be transformed into a union of intersections. Except for the intersection $\bigcap_{w \in W_j} w^{-1} \cdot (U_0 \cdot \Sigma^*)$, all the intersection expressions in this union contain at least one language $w^{-1} U_i$ for a word $w \in W_j$. Since U_i is finite, this shows that $\bigcap_{w \in W_j} w^{-1} \cdot (U_0 \cdot \Sigma^*)$ is the only (possibly) infinite language in the union. Consequently, if we define $L_{j,i} := \widehat{L}_{j,i} \setminus \bigcap_{w \in W_j} w^{-1} \cdot (U_0 \cdot \Sigma^*)$, then $L_{j,i}$ is a finite language.

In order to prove that $W_j \cdot \widehat{L}_{j,i} \cup U_0 \cdot \Sigma^* = W_j \cdot L_{j,i} \cup U_0 \cdot \Sigma^*$, it is sufficient to show that $u \in W_j$ and $v \in \widehat{L}_{j,i} \setminus L_{j,i}$ implies $uv \in U_0 \cdot \Sigma^*$. By definition of $L_{j,i}$, we know that $v \in \bigcap_{w \in W_j} w^{-1} \cdot (U_0 \cdot \Sigma^*)$, and thus $u \in W_j$ implies $uv \in U_0 \cdot \Sigma^*$. □

For the matching problem of Example 14, we have

$$\begin{aligned}
\widehat{L}_{1,1} &= R^{-1} \cdot (\{RS\} \cup \{RR, SS\} \cdot \Sigma^*) \cap \\
&\quad\ \varepsilon^{-1} \cdot (\{RS\} \cup \{RR, SS\} \cdot \Sigma^*) \\
&= (\{S\} \cup \{R\} \cdot \Sigma^*) \cap (\{RS\} \cup \{RR, SS\} \cdot \Sigma^*) \\
&= \{RS\} \cup \{RR\} \cdot \Sigma^*, \\
\widehat{L}_{2,1} &= S^{-1} \cdot (\{RS\} \cup \{RR, SS\} \cdot \Sigma^*) \\
&= \{S\} \cdot \Sigma^*.
\end{aligned}$$

Again, it is easy to see that replacing the variables $X_{j,1}$ by $\widehat{L}_{j,1}$ yields a solution of equation (A_1). The finite languages $L_{j,1}$ are defined as $L_{1,1} := \{RS\}$ and $L_{2,1} := \emptyset$.

Lemma 15 and 17 provide us with a polynomial algorithm for deciding solvability of matching problems in $\mathcal{FL}_\perp$.

Theorem 18 *Solvability of matching problems in $\mathcal{FL}_\perp$ can be decided in polynomial time.*

The proofs of Lemma 15 and 17 also show how to compute a matcher of a given solvable $\mathcal{FL}_\perp$-matching problem in polynomial time. In fact, if the matching problem is solvable, then the following substitution σ is a matcher:

$$\begin{aligned}
\sigma \ := \ \{X_1 &\mapsto \forall L_{1,0}.\perp \sqcap \textstyle\bigsqcap_{i=1}^{k} \forall L_{1,i}.A_i, \\
&\ldots, \\
X_\ell &\mapsto \forall L_{\ell,0}.\perp \sqcap \textstyle\bigsqcap_{i=1}^{k} \forall L_{\ell,i}.A_i\},
\end{aligned}$$

where the languages $L_{j,0}$ $(1 \leq j \leq \ell)$ are defined as in the proof of Lemma 15, and the languages $L_{j,i}$ $(1 \leq j \leq \ell,\ 1 \leq i \leq k)$ are defined as in the proof of Lemma 17. It should be noted that the language $L_{j,i} = \widehat{L}_{j,i} \setminus \bigcap_{w \in W_j} w^{-1} \cdot (U_0 \cdot \Sigma^*)$ is a subset of $\bigcup_{v \in W_j} v^{-1} U_i$, and thus its size is polynomial in the size of the matching problem.

For the matching problem of Example 14, we thus obtain the matcher

$$\{X_1 \mapsto (\forall R.\forall R.\perp) \sqcap (\forall R.\forall S.A_1),\ X_2 \mapsto \forall S.\perp\}.$$

Lemma 19 *Assume that the given $\mathcal{FL}_\perp$-matching problem $C \equiv^? D$ is solvable. Then the substitution σ defined above is the* least *solution of $C \equiv^? D$.*

This lemma, together with Lemma 10, immediately implies the following theorem:

Theorem 20 *Let $C \sqsubseteq^? D$ be a solvable matching problem modulo subsumption. Then the least solution of $C \equiv^? C \sqcap D$ is a minimal solution of $C \sqsubseteq^? D$, and this solution can be computed in polynomial time.*

4 Matching in $\mathcal{FL}_\neg$

The results for matching in $\mathcal{FL}_\perp$ can easily be extended to the language $\mathcal{FL}_\neg$. In principle, negated atomic concepts are treated like new atomic concepts. The fact that $A \sqcap \neg A$ is inconsistent (i.e., equivalent to $\perp$) is taken care of by extending the language in the value restriction for the concept $\perp$ appropriately.

To be more precise, let C, D be $\mathcal{FL}_\perp$-concept descriptions, and $A_1, \ldots, A_k$ the atomic concepts occurring in C, D. By treating the negated atomic concepts $\neg A_i$ like new atomic concepts, we can transform C and D into their $\mathcal{FL}_0$-normal forms:

$$\begin{array}{rcl} C & \equiv & \forall U_0.\bot \sqcap \forall U_1.A_1 \sqcap \ldots \sqcap \forall U_k.A_k \sqcap \\ & & \forall U_{k+1}.\neg A_1 \sqcap \ldots \sqcap \forall U_{2k}.\neg A_k, \\ D & \equiv & \forall V_0.\bot \sqcap \forall V_1.A_1 \sqcap \ldots \sqcap \forall V_k.A_k \sqcap \\ & & \forall V_{k+1}.\neg A_1 \sqcap \ldots \sqcap \forall V_{2k}.\neg A_k. \end{array}$$

If we define

$$\widehat{U}_0 := U_0 \cup \bigcup_{i=1}^{k} (U_i \cap U_{k+i}) \quad \text{and} \quad \widehat{V}_0 := V_0 \cup \bigcup_{i=1}^{k} (V_i \cap V_{k+i}),$$

then Lemma 12 can be generalized to $\mathcal{FL}_\neg$ as follows:

Lemma 21 *Let C, D be $\mathcal{FL}_\neg$-concept descriptions with $\mathcal{FL}_0$-normal forms as introduced above. Then*

$$\begin{array}{lll} C \equiv D & \textit{iff} & \widehat{U}_0 \cdot \Sigma^* = \widehat{V}_0 \cdot \Sigma^* \textit{ and} \\ & & U_i \cup \widehat{U}_0 \cdot \Sigma^* = V_i \cup \widehat{V}_0 \cdot \Sigma^* \\ & & \textit{for all } i, 1 \leq i \leq 2k, \end{array}$$

where Σ^* is the set of all words over the alphabet of all atomic roles.

Consequently, all the results for matching in $\mathcal{FL}_\bot$ carry over to $\mathcal{FL}_\neg$: we simply have to replace k by $2k$ and the sets U_0, V_0 by $\widehat{U}_0, \widehat{V}_0$.

Theorem 22 *Let $C \equiv^? D$ be an $\mathcal{FL}_\neg$-matching problem. Solvability of $C \equiv^? D$ can be tested in polynomial time. If $C \equiv^? D$ is solvable, then a least solution of $C \equiv^? D$ can be computed in polynomial time.*

5 Matching under side conditions

In this section, we first consider strict subsumption conditions, and then briefly mention some results for (non-strict) subsumption conditions.

Strict subsumption conditions

Recall that a strict subsumption condition is of the form $X \sqsubset^? E$ where X is a concept variable and E is a concept pattern. If the concept patterns of a set of strict subsumption conditions do not contain variables (i.e., the expressions E on the right-hand sides of the strict subsumption conditions are concept descriptions), then it is sufficient to compute a least solution of the matching problem, and then test whether this solution also solves the strict subsumption conditions.

Theorem 23 *Let $C \equiv^? D$ be an $\mathcal{FL}_\neg$-matching problem, and $X_1 \sqsubset^? E_1, \ldots, X_n \sqsubset^? E_n$ be strict subsumption conditions such that $E_1, \ldots, E_n$ are $\mathcal{FL}_\neg$-concept descriptions. Then solvability of $C \equiv^? D$ under these conditions is decidable in polynomial time.*

If the right-hand sides of strict subsumption conditions may contain variables, then solvability becomes NP-hard, even for the language $\mathcal{FL}_0$. This can be shown by reducing 3SAT [5] to the matching problem under strict subsumption conditions.

Theorem 24 *Matching under strict subsumption conditions is NP-hard, even for the small language $\mathcal{FL}_0$.*

It should be noted that our reduction of 3SAT to matching under strict subsumption conditions depends on the fact that we consider matching modulo equivalence, rather than matching modulo subsumption. Thus, it is still open whether the NP-hardness result also holds for matching modulo subsumption under strict subsumption conditions.

Theorem 24 provides us only with a hardness result for matching under strict subsumption conditions. Another open question is how to extend the matching algorithm for $\mathcal{FL}_\neg$ to an algorithm that can also handle strict subsumption conditions.

Subsumption conditions

If the subsumption conditions do not introduce cyclic variable dependencies, then a matching problem with subsumption conditions can be reduced to an ordinary matching problem. To be more precise, the sequence of subsumption conditions $X_1 \sqsubseteq^? E_1, \ldots, X_n \sqsubseteq^? E_n$ is *acyclic* iff for all $i, 1 \leq i \leq n$, the pattern E_i does not contain the variables $X_i, \ldots, X_n$. Given such an acyclic sequence of subsumption conditions, we can define a substitution[5] σ inductively as follows:

$$\begin{array}{rcl} \sigma(X_1) & := & Y_1 \sqcap E_1 \quad \text{and} \\ \sigma(X_i) & := & Y_i \sqcap \sigma(E_i) \quad (1 < i \leq n), \end{array}$$

where the Y_i are new variables. We can show that the matching problem $C \equiv^? D$ is solvable under the subsumption conditions $X_1 \sqsubseteq^? E_1, \ldots, X_n \sqsubseteq^? E_n$ iff $C \equiv^? \sigma(D)$ is solvable without subsumption conditions. Unfortunately, the new pattern $\sigma(D)$ may be exponentially larger than the original matching problem with subsumption conditions. However, we conjecture that a compact representation of $\sigma(D)$ may be used to obtain a polynomial algorithm for matching under acyclic subsumption conditions in $\mathcal{FL}_\neg$.

The reduction we have just described is independent of the DL used for constructing the patterns and descriptions. For $\mathcal{FL}_0$, we can go one step further: cyclic subsumption conditions can here be reduced to acyclic ones. Let us illustrate this by two examples: (1) If σ satisfies $X_1 \sqsubseteq^? X_2 \sqcap E_1', X_2 \sqsubseteq^? X_1 \sqcap E_2'$, then $\sigma(X_1) \equiv \sigma(X_2)$,

[5] Strictly speaking, this is not a substitution as introduced in Section 2 since variables are mapped to patterns, and not just to descriptions. It should be clear, however, that the notion of a substitution can be extended appropriately.

which means that we can identify both variables; (2) If σ satisfies $X \sqsubseteq^? \forall R.X$, then $\sigma(X) \equiv \top$, and thus X can be replaced by $\top$.

6 Future Work

Our goal is to extend the results on matching to cover languages at least as expressive as the DL considered in [3]. This requires extending the language to include range constructors (min and max), an individual set constructor (one-of), number restrictions (at-least and at-most), and a fills constructor. For the constructors min, max, and one-of, this mainly requires an appropriate treatment of disjointness, which we have already achieved by our treatment of primitive negation. Number restrictions and the fills construct are more challenging extensions; however, we expect to be able to exploit the characterization of subsumption in cyclic $\mathcal{ALN}$-terminologies provided in [6] for our purposes.

Another motivation for investigating matching modulo equivalence may be found in merging heterogeneous databases. Consider a situation where there is a master ontology along with new database schemas that need to be integrated into the master ontology. In this situation, the integrator would like to know how the new schemas may be mapped onto the master ontology. Our idea is to represent the ontology and the schemas in an appropriate DL, and to view the problem of finding such a mapping as a matching problem of the concepts of the new schema onto the concepts of the master ontology.

7 Conclusion

We have been motivated by the need to prune complicated structures in order to provide manageable object presentations and explanations. The pruning problem can be viewed as a matching problem where there is a comparison between a pattern describing the interesting portions of the object and the larger object itself. Only those portions of the object that match the pattern of interest should be presented. We began with the filtering work introduced in CLASSIC and the theoretical work on the unification of concept terms and generated a formal treatment of matching in the description logic language $\mathcal{FL}_\perp$. We presented results concerning the solvability of the problem including polynomial decidability and (for solvable problems) polynomial computability of a least solution. We also extended the work to include matching under additional side constraints on the variables in the matching patterns. We showed that matching modulo equivalence with strict side conditions is NP-hard even for the small language $\mathcal{FL}_0$.

References

[1] F. Baader, A. Borgida, and D. L. McGuinness. Matching in description logics: Preliminary results. In M.-L. Mugnier, M. Chein, editors, *Proceedings of the Sixth International Conference on Conceptual Structures (ICCS'98)*, *Lecture Notes in Artificial Intelligence*. Springer-Verlag, 1998. To appear.

[2] F. Baader and P. Narendran. Unification of concept terms in description logics. In H. Prade, editor, *Proceedings of the 13th European Conference on Artificial Intelligence (ECAI-98)*, pages 331–335, Brighton, UK, 1998. John Wiley & Sons Ltd. An extended version has appeared as Technical Report LTCS-98-06.

[3] 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, Cambridge, MA (USA), 1996.

[4] 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.

[5] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* W.H. Freeman and Company, New York, 1979.

[6] R. Küsters. Characterizing the semantics of terminological cycles in $\mathcal{ALN}$ using finite automata. In *Proceedings of the Sixth International Conference on Principles of Knowledge Representation and Reasoning (KR'98)*, Trento, Italy, 1998.

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

[8] D. L. McGuinness and A. Borgida. Explaining subsumption in Description Logic. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence, IJCAI'95*, pages 816–821, Montréal, Canada, 1995. Morgan Kaufmann.

[9] D. L. McGuinness, L. Alperin Resnick, and C. Isbell. Description Logic in practice: A CLASSIC application. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence, IJCAI'95*, pages 2045–2046, Montréal, Canada, 1995. Morgan Kaufmann. Video Presentation.

[10] D. L. McGuinness and J.R. Wright. An industrial strength Description Logic-based configurator platform. *IEEE Expert, Special Issue on Configuration*, 1998. To appear.

[11] J. R. Wright, E. S. Weixelbaum, G. T. Vesonder, K. Brown, S. R. Palmer, J. I. Berman, and H. H. Moore. A knowledge-based configurator that supports sales, engineering, and manufacturing at AT&T network systems. *AI Magazine*, 14(3):69–80, 1993.