Matching in Description Logics

Franz Baader* and Ralf Küsters
Theoretical Computer Science
RWTH Aachen
52074 Aachen, Germany
e-mail: {baader, kuesters}@informatik.rwth-aachen.de

Alex Borgida†
Dept. of Computer Science
Rutgers University
New Brunswick, NJ, USA
e-mail: borgida@cs.rutgers.edu

Deborah L. McGuinness‡
Department of Computer Science
Stanford University
Gates Building
Stanford University, Ca. 94305
e-mail: dlm@ksl.stanford.edu

Abstract

Matching concepts against patterns (concepts with variables) is a relatively new operation that has been introduced in the context of concept description languages (description logics). Their original goals was to help filter out unimportant aspects of complicated concepts appearing in large industrial knowledge bases. We propose 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 $\mathcal{ALN}$, thus removing restrictions from previous work. The paper also addresses the question of matching problems with additional "side conditions", which were motivated by practical needs.

1 Introduction

Knowledge representation systems based on Description Logic (DL systems) can be used to represent the knowledge of an application domain in a structured and formally well-understood way [13, 4, 12, 37, 9]. In such systems, the important notions of the domain can be described by *concept descriptions*, i.e., expressions that are built from atomic concepts (unary predicates) and atomic roles (binary predicates) using the concept constructors provided by the Description Logic language (DL language) of the system. The atomic concepts and the concept descriptions represent sets of individuals, whereas roles represent binary relations between individuals. For example,

*Partially supported by the EC Working Group CCL II.
†Partially supported by NSF Grant IRI 9619979.
‡Work was done while the author was at AT&T Labs – Research.

using the atomic concept Woman and the atomic role child, the concept of all *women having only daughters* (i.e., women such that all their children are again women) can be represented by the concept description

$$\mathsf{Woman} \sqcap \forall \mathsf{child}.\mathsf{Woman}.$$

DL systems provide their users with various inference capabilities that allow them to deduce implicit knowledge from the explicitly represented knowledge. For instance, the *subsumption* algorithm allows one to determine subconcept-superconcept relationships: C is subsumed by D ($C \sqsubseteq D$) if and only if all instances of C are also instances of D, i.e., the first description is always interpreted as a subset of the second description. For example, the concept description Woman obviously subsumes the concept description $\mathsf{Woman} \sqcap \forall\mathsf{child}.\mathsf{Woman}$. With the help of the subsumption algorithm, a newly introduced concept description can automatically be placed at the correct position in the hierarchy of the already existing concept descriptions. Two concept descriptions C, D are *equivalent* ($C \equiv D$) if and only if they subsume each other, i.e., if and only if they always represent the same set of individuals. For example, the descriptions $\mathsf{Woman} \sqcap \forall\mathsf{child}.\mathsf{Woman}$ and $(\forall\mathsf{child}.\mathsf{Woman}) \sqcap \mathsf{Woman}$ are equivalent since $\sqcap$ is interpreted as set intersection, which is commutative.

The traditional inference problems for DL systems (like subsumption) are now well-investigated, which 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 (see, e.g., [24, 36, 33, 22, 1, 3, 21, 14, 11, 7, 2, 8]). In addition, the computational complexity of these inference problems has been investigated in detail [24, 32, 34, 17, 16, 19, 35, 18].

It has turned out, however, that building and maintaining large DL knowledge bases requires additional support in the form of inferences that have not been considered in the DL literature until very recently [27]. The present paper is concerned with one such new inference service, namely, *matching* of concept descriptions, which was motivated by the problem of pruning large descriptions.

Pruning as a motivation

In industrial applications, objects and their descriptions may become too large and complex to view in traditional ways. Simply printing (descriptions of) objects in small applications such as configuring stereo systems [28, 29] can easily take 10 pages, while printing objects in industrial applications such as configuring telecommunications equipment [38, 30, 31] might take five times as much space. In addition, if explanation facilities [26, 25] are introduced and a naive explanation is presented of all deductions, the system can produce five times as much output again. It quickly becomes clear that object descriptions need to be pruned if users are to be able to inspect objects and not be overwhelmed with irrelevant details.

We have observed that information may not be worthy of display for many reasons. Information may be obviously true because it is commonly known definitional knowledge, (e.g., the age of a person must be a number), or because it is common knowledge in the domain, (e.g., the state field of an address must be filled with a state in the US if the application is only concerned with US citizens). Information may also not be worth presenting because it is information only relevant to an internal function, (e.g., information describing where to display an object in a graphical presentation), or because it is otherwise determined to be non-informative or not of interest to typical users (e.g., healthy eaters typically do not want to see the sugar content of particular foods). However, information that may not be of general interest, can, under certain conditions, become critical (e.g., if a food is known to fill the "eats" role for a diabetic's meal, then the sugar content becomes significant). Thus, the context of the information becomes a critical component in determining what should be presented.

Normally, users would need to retrieve descriptions of object portions and then verify that they are interesting, by using functions from the application programming interface (API) of the knowledge base management system (KBMS). For example, they might retrieve the value restriction on an individual's age and then check to see if it is strictly subsumed by the concept Number.

This approach, which leaves the solution outside the KBMS, is less desirable than one in which the specification of what is interesting is stored as part of the knowledge base itself [10]. The advantage of the second alternative is that such specifications can be saved, organized, and re-used (e.g., through inheritance), even by naive users. McGuinness introduced the ability to provide "pruned views" of objects in the CLASSIC system (version 2), under the name of "*filtering*". The problem of filtering was viewed as a matching problem: taking a description of interesting object portions and matching that against existing object descriptions. Matching patterns were associated with classes and then used to filter all subclasses and instances of the class. The patterns were defined once by a domain-literate person and then all users could use them as the default pruning mechanism. The initial implementation had implicit variables in matching patterns and also relied to some extent on a library of test filters. This implementation has been used in small applications [28, 29] to save 3–5 pages of output (sometimes reducing the object to 25 percent of its former size). In larger applications [38, 30, 31] it can easily save 30 pages of output per object.

Matching as a declarative solution

Even for matching filters attached to classes, one has a choice of using a variety of specification techniques. As usual in information-intensive applications (e.g., databases), a *declarative* specification of filters should be preferred to a more *procedural* one: it is usually more concise and elegant because it is likely to support formal analysis and thence optimization by the KBMS.

A more declarative version of matching filters can be provided by introducing *variables* into concepts, thus producing "concept patterns" [25]. The pruning mechanism was initially described as a purely syntactic *match* involving concept patterns [25], and then given a formal semantics and a provably sound syntactic implementation [10]. Given a concept pattern D (i.e., a concept description containing variables) and a concept description C without variables, the matching problem introduced by Borgida and McGuinness [10] 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

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

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 (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 be particularly interesting 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 KR 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, when augmented by this side condition, whereas

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

would still yield a solution (provided that DL can be inferred to be subsumed by KR).

In some cases it would be useful 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 be seen more clearly in an example where the concept Person is known to have Number as restriction 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) in fact 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 were introduced by McGuinness [25], and generalized in Borgida and McGuinness [10] to any DL supporting a certain type of subsumption algorithms (called "structural" subsumption algorithms). These matching algorithms are based on the role-centered structural normal form[1] of concept descriptions usually employed by structural subsumption algorithms. The main drawback of these algorithms is that, in an effort at generality, they require the concept pattern itself to be in structural normal form, and thus place strong restrictions on the occurrence of variables. The reason is that it is not possible to normalize arbitrary patterns, and thus certain natural concept patterns must be disallowed. For example, since at most one variable may occur "in the same place", the pattern in Example 23 would not be admissible. This makes it difficult to build composite patterns from simpler, previously defined ones. In addition, these algorithms do not always find a matcher, even if it exists, due to an incomplete treatment of the top ($\top$) and the bottom ($\bot$) concepts (see Example 41).

[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 article groups value restrictions by concept names (see Section 3).

Baader and Narendran [6] consider unification of concept descriptions of the language $\mathcal{FL}_0$, 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 $C \sqsubseteq \sigma(D)$ if and only if $C \equiv \sigma(C \sqcap D)$, matching modulo subsumption (as introduced above) is an instance of matching modulo equivalence. The polynomial matching algorithm described by Baader and Narendran [6] does not impose restrictions on the form of the patterns. However, it is restricted to the small language $\mathcal{FL}_0$.

The new results

We shall show that Baader and Narendran's 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 ($\perp$), $\mathcal{FL}_\neg$, which extends $\mathcal{FL}_\perp$ by primitive negation ($\neg A$, where A is an atomic concept), and $\mathcal{ALN}$, which extends $\mathcal{FL}_\neg$ by number restrictions. The reasons for starting with a detailed treatment of the small language $\mathcal{FL}_\perp$, and then extending this treatment in two steps to the larger languages, are mainly of a didactic nature. It should, however, also be noted that, for matching, positive results (such as decidability in polynomial time) do not automatically transfer from a given language to its sublanguages. In fact, a matching problem of the smaller language that does not have a solution in this language may well have one in the larger language.[2]

In addition to pure matching problems, we also consider matching under additional conditions on the variable bindings, which also arose in practical examples [28, 25] 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 to matching without subsumption conditions. It is not yet clear whether this reduction leads to an increase of the complexity. In contrast, strict subsumption conditions definitely 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 $\sigma(X) \not\equiv \sigma(E)$. We shall show that, even for the small language $\mathcal{FL}_0$, matching under strict subsumption conditions is NP-hard.

2 Formal preliminaries

In this section, we first introduce the 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.

[2] We will come back to this point in the conclusion.

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 $\mathcal{ALN}$*-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 $\bot$ (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).
- 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).
- If $R \in \mathcal{R}$ is an atomic role and $n \geq 0$ is a nonnegative integer, then $(\leq n\, R)$ and $(\geq n\, R)$ are concept descriptions (number restrictions).

In the sublanguage $\mathcal{FL}_0$ of $\mathcal{ALN}$, number restrictions, atomic negation, and $\bot$ may not be used, in $\mathcal{FL}_\bot$ atomic negation and number restriction may not be used, and in $\mathcal{FL}_\neg$ only number restrictions are disallowed.

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

Definition 2 An *interpretation* I consists of a nonempty set Δ^I, the domain of the interpretation, and an interpretation function $\cdot^I$ 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:

$$\begin{aligned}
\top^I &:= \Delta^I, \\
\bot^I &:= \emptyset, \\
(\neg A)^I &:= \Delta^I \setminus A^I, \\
(C \sqcap D)^I &:= C^I \cap D^I, \\
(\forall R.C)^I &:= \{d \in \Delta^I \mid \forall e \in \Delta^I\colon (d,e) \in R^I \rightarrow e \in C^I\}, \\
(\leq n\, R)^I &:= \{d \in \Delta^I \mid card(\{e \in \Delta^I \mid (d,e) \in R^I\}) \leq n\}, \\
(\geq n\, R)^I &:= \{d \in \Delta^I \mid card(\{e \in \Delta^I \mid (d,e) \in R^I\}) \geq n\}.
\end{aligned}$$

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

- C is *subsumed* by D ($C \sqsubseteq D$) if and only if $C^I \subseteq D^I$ for all interpretations I.
- C is *equivalent* to D ($C \equiv D$) if and only if $C^I = D^I$ for all interpretations I.
- C is *strictly subsumed* by D ($C \sqsubset D$) if and only if $C \sqsubseteq D$ 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 $\mathcal{ALN}$*-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 $\mathcal{ALN}$-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. $\mathcal{FL}_0$-, $\mathcal{FL}_\bot$- and $\mathcal{FL}_\neg$-patterns are defined analogously.

A *substitution* σ is a mapping from $\mathcal{X}$ into the set of all $\mathcal{ALN}$-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 $\sigma(\bot) := \bot$,
- $\sigma(C \sqcap D) := \sigma(C) \sqcap \sigma(D)$,
- $\sigma(\forall R.C) := \forall R.\sigma(C)$,
- $\sigma(\geq n\ R) := (\geq n\ R)$, and $\sigma(\leq n\ R) := (\leq n\ R)$.

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 $\mathcal{ALN}$-concept pattern is an $\mathcal{ALN}$-concept description.[3] An $\mathcal{FL}_0$-substitution maps concept variables to $\mathcal{FL}_0$-concept descriptions, and $\mathcal{FL}_\bot$- and $\mathcal{FL}_\neg$-substitutions are defined analogously.

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

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

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

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

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

Matching problems and (strict) subsumption conditions in $\mathcal{FL}_0$, $\mathcal{FL}_\perp$, and $\mathcal{FL}_\neg$ are defined analogously. Note that the solutions are then also 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 if and only if 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 $\{C_1 \equiv^? D_1, \ldots, C_m \equiv^? D_m\}$ if and only if it solves the single matching problem

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

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

Borgida and McGuinness [10, 25] have considered a different type of matching problems. We will refer to those problems as matching problems modulo subsumption in order to distinguish them from the matching problems modulo equivalence introduced above.

Definition 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 $C \sqsubseteq^? D$ if and only if it solves $C \equiv^? C \sqcap D$.

For $\mathcal{ALN}$, 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 $\mathcal{ALN}$, and let $\sigma_\top$ be the substitution that replaces each variable by $\top$. Then $C \sqsubseteq^? D$ has a solution if and only if $\sigma_\top$ solves $C \sqsubseteq^? D$.

Thus, solvability of matching problems modulo subsumption in $\mathcal{ALN}$ 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* if and only if there does not exist a substitution δ such that $C \sqsubseteq \delta(D) \sqsubset \sigma(D)$.

Lemma 10 Let $C \sqsubseteq^? D$ be an $\mathcal{ALN}$-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.

Proof. This is an immediate consequence of the following fact, which can easily be proved by induction on the structure of $\mathcal{ALN}$-concept patterns: If $\sigma \sqsubseteq \delta$, then $\sigma(D) \sqsubseteq \delta(D)$ for any $\mathcal{ALN}$-concept pattern D. ∎

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 by Baader and Narendran [6]. 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. If $m = 0$, then this is the empty word ε, and thus $\forall \varepsilon.A$ is our "abbreviation" for A. 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. 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

$$C \equiv \forall U_1.A_1 \sqcap \ldots \sqcap \forall U_k.A_k \quad \text{and} \quad D \equiv \forall V_1.A_1 \sqcap \ldots \sqcap \forall V_k.A_k,$$

where U_i, V_i are finite sets of words over the alphabet of all atomic roles. By convention, the term $\forall \emptyset.A$ is considered to be equivalent to $\top$, and hence the concept $\top$ itself can be represented by making all the coefficients, V_i, be empty sets. This normal form provides us with the following characterization of equivalence of $\mathcal{FL}_0$-concept descriptions [6]:

Lemma 11 Let C, D be $\mathcal{FL}_0$-concept descriptions with normal forms as introduced above. Then $C \equiv D$ if and only if $U_i = V_i$ 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 [6].

If we treat $\bot$ like an arbitrary atomic concept, $\mathcal{FL}_\bot$-concept descriptions C, D can still be represented in the form[4]

$$C \equiv \forall U_0.\bot \sqcap \forall U_1.A_1 \sqcap \ldots \sqcap \forall U_k.A_k \text{ and } D \equiv \forall V_0.\bot \sqcap \forall V_1.A_1 \sqcap \ldots \sqcap \forall V_k.A_k.$$

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

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

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

Proof. Assume that the right-hand side of the equivalence stated in the lemma holds. It is sufficient to show that this implies $C \sqsubseteq D$ (since $D \sqsubseteq C$ then follows by symmetry). Considering the normal form of D this means that we must show that for all $w \in V_0$ we have (1) $C \sqsubseteq \forall w.\bot$, and for all $i, 1 \leq i \leq k$, and all $w \in V_i$ we have (2) $C \sqsubseteq \forall w.A_i$. Thus, let $w \in V_0$. By assumption, $V_0 \subseteq V_0{\cdot}\Sigma^* = U_0{\cdot}\Sigma^*$, which implies that there exist a word $u \in U_0$ and $v \in \Sigma^*$ such that $w = uv$. Thus, the normal form for C contains the conjunct $\forall u.\bot$. Since $\forall u.\bot \sqsubseteq \forall uv.\bot$ for any word v we have established that (1) holds. Property (2) can be shown similarly.

Conversely, assume that the right-hand side of the equivalence stated in the lemma does not hold, i.e., (1) $U_0{\cdot}\Sigma^* \neq V_0{\cdot}\Sigma^*$, or for some $i, 1 \leq i \leq k$, (2) $U_i \cup U_0{\cdot}\Sigma^* \neq V_i \cup V_0{\cdot}\Sigma^*$.

First, we assume that (1) holds. Without loss of generality we may assume that there exists a word $w := R_1 \ldots R_m \in \Sigma^*$ such that $w \in U_0{\cdot}\Sigma^*$ and $w \notin V_0{\cdot}\Sigma^*$. We claim that this implies $D \not\sqsubseteq C$, and thus $C \not\equiv D$.

In order to prove this claim, we construct an interpretation I as follows: the domain $\Delta^I := \{d_0, \ldots, d_m\}$ consists of $m+1$ distinct individuals; the interpretation of atomic concepts A_i is given by $A_i^I := \Delta^I$; finally, the atomic roles are interpreted as $S^I := \{(d_{i-1}, d_i) \mid S = R_i\}$. It is easy to see that this interpretation satisfies $d_0 \in (\forall u.A_i)^I$ for all words $u \in \Sigma^*$ (since $A_i^I = \Delta^I$), and $d_0 \in (\forall u.\bot)^I$ for all words u that are not a prefix of $w = R_1 \ldots R_m$. Consequently, $d_0 \in (\forall u.A_i)^I$ for all $u \in V_i$. In addition, $w \notin V_0{\cdot}\Sigma^*$ implies that no word in V_0 is a prefix of w, and thus $d_0 \in (\forall u.\bot)^I$ for all words $u \in V_0$. This shows that $d_0 \in D^I$. However, by construction, $d_0 \notin (\forall w.\bot)^I$, which implies $d_0 \notin C^I$.

Second, we assume that (1) does not hold, i.e., $U_0{\cdot}\Sigma^* = V_0{\cdot}\Sigma^*$, and that (2) holds. Without loss of generality we may assume that there exists a word $w := R_1 \ldots R_m \in$

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

Σ^* such that $w \in U_i$ and $w \notin V_i \cup V_0 \cdot \Sigma^*$. Again, we claim that this implies $D \not\sqsubseteq C$, and thus $C \not\equiv D$.

In order to prove this claim, we construct an interpretation I as follows: the domain $\Delta^I := \{d_0, \ldots, d_m\}$ consists of $m+1$ distinct individuals; the interpretation of atomic concepts A_j for $j \neq i$ is given by $A_j^I := \Delta^I$; the interpretation of A_i is $A_i^I := \Delta^I \setminus \{d_m\}$; finally, the atomic roles are interpreted as $S^I := \{(d_{i-1}, d_i) \mid S = R_i\}$. By construction $d_0 \notin (\forall w.A_i)^I$, and thus $d_0 \notin C^I$. On the other hand, it is easy to show (using arguments that are similar to the ones employed in the first case) that $d_0 \in D^I$. ∎

If D is an $\mathcal{FL}_\perp$-pattern containing the atomic concepts $A_1 \ldots A_k$ and the variables $X_1, \ldots, X_\ell$, then its $\mathcal{FL}_0$-normal form is of the form

$$D \equiv \forall V_0.\bot \sqcap \forall V_1.A_1 \sqcap \ldots \sqcap \forall V_k.A_k \sqcap \forall W_1.X_1 \sqcap \ldots \sqcap \forall W_\ell.X_\ell.$$

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 $i, 1 \le i \le k$,

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

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

PROOF. Let

$$\sigma := \{X_1 \mapsto \forall L_{1,0}.\bot \sqcap \bigsqcap_{i=1}^{k} \forall L_{1,i}.A_i, \ \ldots, \ X_\ell \mapsto \forall L_{\ell,0}.\bot \sqcap \bigsqcap_{i=1}^{k} \forall L_{\ell,i}.A_i\}$$

be a substitution.[5] By employing elementary equivalences between concept descriptions we can show that the $\mathcal{FL}_0$-normal form of $\sigma(D)$ is

$$\begin{aligned} \sigma(D) \quad \equiv \quad & \forall (V_0 \cup W_1 \cdot L_{1,0} \cup \cdots \cup W_\ell \cdot L_{\ell,0}).\bot \sqcap \\ & \bigsqcap_{i=1}^{k} \forall (V_i \cup W_1 \cdot L_{1,i} \cup \cdots \cup W_\ell \cdot L_{\ell,i}).A_i. \end{aligned}$$

Lemma 12 implies that $C \equiv \sigma(D)$ if and only if

$$U_0 \cdot \Sigma^* \quad = \quad (V_0 \cup W_1 \cdot L_{1,0} \cup \cdots \cup W_\ell \cdot L_{\ell,0}) \cdot \Sigma^*, \tag{1}$$

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

$$\begin{aligned} U_i \cup U_0 \cdot \Sigma^* \quad = \quad & V_i \cup W_1 \cdot L_{1,i} \cup \cdots \cup W_\ell \cdot L_{\ell,i} \cup \\ & (V_0 \cup W_1 \cdot L_{1,0} \cup \cdots \cup W_\ell \cdot L_{\ell,0}) \cdot \Sigma^*. \end{aligned} \tag{2}$$

[5] Without loss of generality we restrict our attention to the images of variables occurring in D, and assume that σ introduces only atomic concepts occurring in C or D.

Since concatenation distributes over union, (1) corresponds to the fact that the assignment $X_{1,0} := L_{1,0}, \ldots, X_{\ell,0} := L_{\ell,0}$ solves equation ($\bot$). In addition, if we already know that (1) holds, then (2) corresponds to the fact that the assignment $X_{1,i} := L_{1,i}, \ldots, X_{\ell,i} := L_{\ell,i}$ solves equation (A_i).

This shows how a solution σ of the matching problem $C \equiv^? D$ yields solutions of the equations ($\bot$), (A_1), ..., (A_k), and conversely how solutions of these equations can be used to construct a matcher σ. ∎

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

$$C \equiv \forall\{RR, SS\}.\bot \sqcap \forall\{RS\}.A_1 \quad \text{and} \quad D \equiv \forall\emptyset.\bot \sqcap \forall\emptyset.A_1 \sqcap \forall\{\varepsilon, R\}.X_1 \sqcap \forall\{S\}.X_2.$$

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

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

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 if and only if replacing $X_{j,0}\cdot\Sigma^*$ by the sets

$$\widehat{L}_{j,0} := \bigcap_{w \in W_j} w^{-1}\cdot(U_0\cdot\Sigma^*)$$

solves equation ($\bot$).[6]

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 $j, 1 \le j \le \ell$. 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

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

[6]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.

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). ∎

The following lemma shows that languages of the form $L{\cdot}\Sigma^*$ for finite L are closed under left quotients, intersection, union, and left concatenation with finite languages.

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 $U{\cdot}(V{\cdot}\Sigma^*) = (U{\cdot}V){\cdot}\Sigma^*$.

Proof. (1) Since $(uv)^{-1}L = v^{-1}{\cdot}(u^{-1}{\cdot}L)$ for all languages L, it is sufficient to consider the case where w has length 1, i.e., $w \in \Sigma$. We distinguish two cases:

- If the empty word ε belongs to U, then $U{\cdot}\Sigma^* = \Sigma^* = w^{-1}{\cdot}\Sigma^*$, and thus we can take $L_1 := \{\varepsilon\}$.
- If $\varepsilon \notin U$, then our assumption that $w \in \Sigma$ implies that $w^{-1}{\cdot}(U{\cdot}\Sigma^*) = (w^{-1}{\cdot}U) \cdot \Sigma^*$, and thus we can take $L_1 := w^{-1}{\cdot}U$, which is finite since U is finite.

(2) It is easy to see that we can take $L_2 := (U \cap V{\cdot}\Sigma^*) \cup (V \cap U{\cdot}\Sigma^*)$.
(3) is trivial. ∎

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 equation $(\bot)$. 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 if and only if 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

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

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.

PROOF. Obviously, Lemma 15 and 17 provide us with an effective method for testing matching problems in $\mathcal{FL}_\perp$ for solvability. It remains to be shown that this test can be realized in polynomial time. First, note that the combined size[7] of the finite languages U_i and V_i is linear in the size of the concept description and the pattern. Thus, the size of the equations $(\perp)$ and (A_i) is polynomial in the size of the original matching problem. Both for equation $(\perp)$ and for equation (A_i) we compute a "candidate" for a solution and then test whether it really is a solution.

First, let us consider equation $(\perp)$. Given the finite language U_0, we can construct (in polynomial time) a *deterministic* finite automaton that accepts the left-hand side $U_0\cdot\Sigma^*$ of equation $(\perp)$, and whose size is linear in the size of U_0.

Regarding the right-hand side of equation $(\perp)$, it is easy to see that computing the candidate and inserting it into the right-hand side can be done in polynomial time. To be more precise, we can compute (in polynomial time) a *deterministic* finite automaton accepting the (regular) language obtained by inserting the candidate solution into the right-hand side of equation $(\perp)$, and the size of this automaton is polynomial in the size of the equation. In fact, in order to construct this automaton, we start with very simple finite deterministic automata for $U_0\cdot\Sigma^*$ and $V_0\cdot\Sigma^*$. In principle, these automata have the form of a tree (representing the finite language U_0

[7] As *size* of a finite language we take the sum of the length of the words occurring in the language.

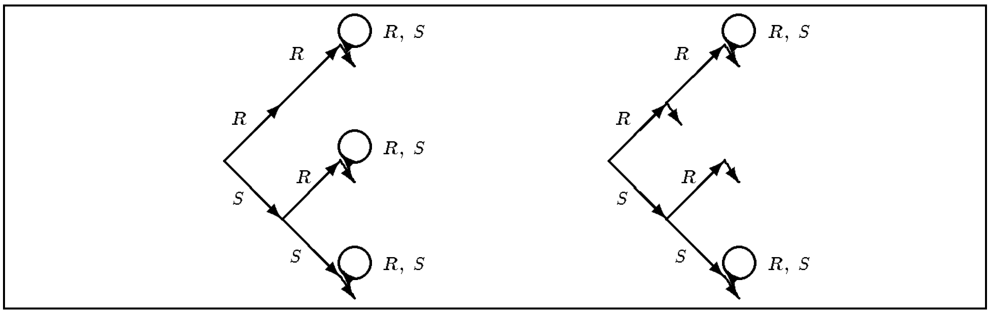

Figure 1: Tree-like automata for the languages $\{RR, SR, SS\}\cdot\{R, S\}^*$ and $\{R, SR\} \cup \{RR, SS\}\cdot\{R, S\}^*$. The root of the tree is the initial state and the final states are marked by exiting arrows without destination.

or V_0) with loops at the leaves (representing the Σ^* at the end), where the root is the initial state and the leaves are the final states (see the left-hand side of Fig. 1 for an example). It is easy to see that computing the left quotient and the intersection of languages represented by such tree-like automata can be realized as linear operations on tree-like automata. Thus, computing the languages $\widehat{L}_{j,0}$, and thus the candidate solution, is polynomial. Inserting the candidate solution into the right-hand side of the equation is also polynomial since the concatenation $W_j\cdot\widehat{L}_{j,0}$ can be realized by a quadratic operation: $W_j\cdot\widehat{L}_{j,0}$ can be represented as union of the languages $\{w\}\cdot\widehat{L}_{j,0}$ where $w \in W_j$. Now, computing a tree-like automaton corresponding to $\{w\}\cdot\widehat{L}_{j,0}$ is a linear operation. In addition, union can be realized as a linear operation on tree-like automata as well.

Since equivalence of regular languages given by *deterministic* finite automata can be decided in time polynomial in the size of the automata,[8] this shows that solvability of equation ($\bot$) can be tested in polynomial time.

The equations (A_i) can be treated similarly. We just have to extend our argument regarding closure properties of tree-like automata from automata representing languages of the form $L\cdot\Sigma^*$ for finite L to languages of the form $L \cup L'\cdot\Sigma^*$ for finite L, L' (see the right-hand side of Fig. 1 for an example of such an extended tree-like automaton). ∎

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

$$\sigma := \{X_1 \mapsto \forall L_{1,0}.\bot \sqcap \bigsqcap_{i=1}^{k} \forall L_{1,i}.A_i,\ \ldots,\ X_\ell \mapsto \forall L_{\ell,0}.\bot \sqcap \bigsqcap_{i=1}^{k} \forall L_{\ell,i}.A_i\},$$

where the languages $L_{j,0}$ ($1 \le j \le \ell$) are defined as in the proof of Lemma 15, and the languages $L_{j,i}$ ($1 \le j \le \ell$, $1 \le i \le k$) are defined as in the proof of Lemma 17.

Lemma 19 The substitution σ defined above can be computed in polynomial time.

[8]Note that this would not be the case for *nondeterministic* finite automata.

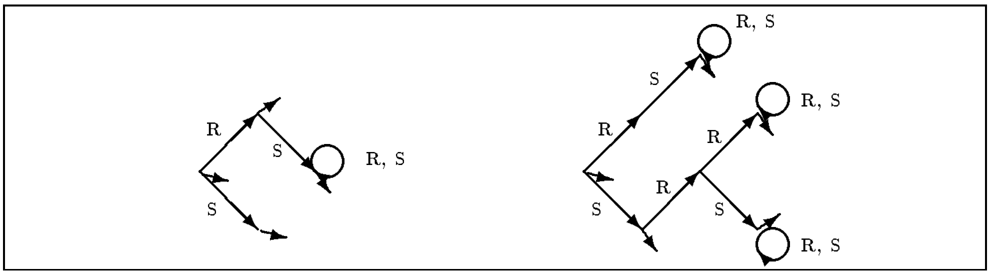

Figure 2: The complement tree-like automata for the automata in Fig. 1.

PROOF. It is sufficient to show that the languages $L_{j,i}$ can be computed in polynomial time. For $i = 0$ this has already been shown in the proof of Theorem 18 since $L_{j,0}$ can easily be read off the tree-like automata for $\widehat{L}_{j,0}$. For $i > 0$ we have $L_{j,i} = \widehat{L}_{j,i} \setminus \widehat{L}_{j,0}$. We know that both for $\widehat{L}_{j,i}$ and for $\widehat{L}_{j,0}$ we can compute tree-like automata in polynomial time. Since intersection is a linear operation on tree-like automata, it remains to be shown that the complement of the language accepted by a tree-like automaton can also be accepted by a tree-like automaton, and that this complement automaton can be computed in polynomial time. This can be achieved by first completing the tree-like automaton by additional sink states; then iteratively removing all leaves that are final states; and finally exchanging final and non-final states (see Fig. 2 for two examples). ∎

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

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

Lemma 20 Assume that the given $\mathcal{FL}_\bot$-matching problem $C \equiv^? D$ is solvable. Then the substitution σ defined above is the *least* solution of $C \equiv^? D$.

PROOF. Assume that

$$\delta := \{X_1 \mapsto \forall M_{1,0}.\bot \sqcap \prod_{i=1}^{k} \forall M_{1,i}.A_i,\ \ldots,\ X_\ell \mapsto \forall M_{\ell,0}.\bot \sqcap \prod_{i=1}^{k} \forall M_{\ell,i}.A_i\}$$

is another solution of $C \equiv^? D$. Consequently, the assignment $X_{1,0} := M_{1,0}, \ldots, X_{\ell,0} := M_{\ell,0}$ solves equation ($\bot$), and the assignment $X_{1,i} := M_{1,i}, \ldots, X_{\ell,i} := M_{\ell,i}$ solves (A_i). As shown in the proofs of Lemma 15 and 17, this implies that $M_{j,0}{\cdot}\Sigma^* \subseteq L_{j,0}{\cdot}\Sigma^*$ $(1 \le j \le \ell)$ and $M_{j,i} \subseteq \widehat{L}_{j,i}$ $(1 \le j \le \ell,\ 1 \le i \le k)$.

As in the proof of Lemma 12, we can infer $\sigma(X_j) \sqsubseteq \delta(X_j)$ from $M_{j,0}{\cdot}\Sigma^* \subseteq L_{j,0}{\cdot}\Sigma^*$ and $M_{j,i} \cup M_{j,0}{\cdot}\Sigma^* \subseteq L_{j,i} \cup L_{j,0}{\cdot}\Sigma^*$. We already know that the first inclusion holds. For the second inclusion, it remains to be shown that $M_{j,i} \subseteq L_{j,i} \cup L_{j,0}{\cdot}\Sigma^*$. This is an immediate consequence of $M_{j,i} \subseteq \widehat{L}_{j,i}$ since $\widehat{L}_{j,i} = L_{j,i} \cup \bigcap_{w \in W_j} w^{-1}{\cdot}(U_0{\cdot}\Sigma^*)$ and $L_{j,0}{\cdot}\Sigma^* = \bigcap_{w \in W_j} w^{-1}{\cdot}(U_0{\cdot}\Sigma^*)$. ∎

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

Theorem 21 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 Extension to larger languages

In this section, we show that our approach for solving matching problems in $\mathcal{FL}_\bot$ can be extended to the larger languages $\mathcal{FL}_\neg$ and $\mathcal{ALN}$.

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

In order to extend the results for matching in $\mathcal{FL}_\bot$ to the larger language $\mathcal{FL}_\neg$, we treat negated atomic concepts like new atomic concepts. The fact that $A \sqcap \neg A$ is inconsistent (i.e., equivalent to $\bot$) is taken care of by extending the language in the value restriction for the concept $\bot$ appropriately.

To be more precise, let C, D be $\mathcal{FL}_\neg$-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:

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

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

$$C \equiv D \qquad \text{iff} \qquad \begin{array}{l} \widehat{U}_0 \cdot \Sigma^* = \widehat{V}_0 \cdot \Sigma^* \text{ and} \\ U_i \cup \widehat{U}_0 \cdot \Sigma^* = V_i \cup \widehat{V}_0 \cdot \Sigma^* \text{ for all } i, 1 \le i \le 2k. \end{array}$$

Since the formulation of this lemma is just a syntactic variant of the one of Lemma 12 (where k is replaced by $2k$ and the sets U_0, V_0 by $\widehat{U}_0, \widehat{V}_0$), one might conjecture that Theorem 13 can be generalized accordingly. Unfortunately, this is not the case, as demonstrated by the following example.

Example 23 Let R, S, T be three distinct atomic roles. We consider the problem of matching the pattern

$$(\forall R.(X_1 \sqcap X_2)) \sqcap (\forall S.X_1) \sqcap (\forall T.X_2)$$

against the description

$$(\forall R.\bot) \sqcap (\forall S.A_1) \sqcap (\forall T.\neg A_1).$$

Obviously, this matching problem can be solved by simply replacing X_1 by A_1 and X_2 by $\neg A_1$. However, if we construct the equations $(\bot)$, (A_1), and $(\neg A_1)$ according to the way it is done in Section 3, with the only difference that U_0, V_0 are replaced by $\widehat{U}_0, \widehat{V}_0$,[9] then we obtain

$$\begin{array}{ll}(\bot) & \{R\}\cdot\Sigma^* = \{R,S\}\cdot X_{1,0}\cdot\Sigma^* \cup \{R,T\}\cdot X_{2,0}\cdot\Sigma^*,\\ (A_1) & \{S\} \cup \{R\}\cdot\Sigma^* = \{R,S\}\cdot X_{1,1} \cup \{R,T\}\cdot X_{2,1} \cup \{R\}\cdot\Sigma^*,\\ (\neg A_1) & \{T\} \cup \{R\}\cdot\Sigma^* = \{R,S\}\cdot X_{1,2} \cup \{R,T\}\cdot X_{2,2} \cup \{R\}\cdot\Sigma^*.\end{array}$$

Obviously, the equation (A_1) can be solved by $X_{1,1} := \{\varepsilon\}$ and $X_{2,1} := \emptyset$, and the equation $(\neg A_1)$ by $X_{1,2} := \emptyset$ and $X_{2,2} := \{\varepsilon\}$. However, the equation $(\bot)$ is *not solvable.*

The reason for the problem exhibited by this example is that the value restriction $\forall R.\bot$ required by the description cannot directly be generated from the pattern by insertion of $\bot$, but instead by an interaction of A_1 and $\neg A_1$ in the instantiated pattern. In fact, the solutions of the equations (A_1) and $(\neg A_1)$ defined above satisfy

$$R \in (\{R,S\}\cdot\{\varepsilon\} \cup \{R,T\}\cdot\emptyset) \cap (\{R,S\}\cdot\emptyset \cup \{R,T\}\cdot\{\varepsilon\}),$$

which provides us with the word R (and thus the language $R\cdot\Sigma^*$) missing on the right-hand side of $(\bot)$.

In order to formulate this solution to the problem in the general case, we consider the generic $\mathcal{FL}_\neg$-matching problem $C \equiv^? D$, where the $\mathcal{FL}_0$-normal forms of C, D are

$$\begin{array}{rcl}C & \equiv & \forall U_0.\bot \sqcap \forall U_1.A_1 \sqcap \ldots \sqcap \forall U_k.A_k \sqcap \forall U_{k+1}.\neg A_1 \sqcap \ldots \sqcap \forall U_{2k}.\neg A_k,\\ D & \equiv & \forall V_0.\bot \sqcap \forall V_1.A_1 \sqcap \ldots \sqcap \forall V_k.A_k \sqcap \forall V_{k+1}.\neg A_1 \sqcap \ldots \sqcap \forall V_{2k}.\neg A_k \sqcap\\ & & \forall W_1.X_1 \sqcap \ldots \sqcap \forall W_\ell.X_\ell.\end{array}$$

The sets $\widehat{U}_0, \widehat{V}_0$ are assumed to be defined as above Lemma 22. If we want to match D with the description C, then we must solve the following formal language equations:

$$(\bot) \quad \widehat{U}_0\cdot\Sigma^* = V_0\cdot\Sigma^* \cup W_1\cdot X_{1,0}\cdot\Sigma^* \cup \ldots \cup W_\ell\cdot X_{\ell,0}\cdot\Sigma^* \cup \bigcup_{i=1}^{k} \mathrm{Int}(A_i, \neg A_i)\cdot\Sigma^*,$$

where

$$\begin{array}{rcl}\mathrm{Int}(A_i, \neg A_i) & := & (V_i \cup W_1\cdot X_{1,i} \cup \ldots \cup W_\ell\cdot X_{\ell,i}) \cap\\ & & (V_{k+i} \cup W_1\cdot X_{1,k+i} \cup \ldots \cup W_\ell\cdot X_{\ell,k+i}),\end{array}$$

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

$$(A_i) \quad U_i \cup \widehat{U}_0\cdot\Sigma^* = V_i \cup W_1\cdot X_{1,i} \cup \ldots \cup W_\ell\cdot X_{\ell,i} \cup \widehat{U}_0\cdot\Sigma^*$$

and

$$(\neg A_i) \quad U_{k+i} \cup \widehat{U}_0\cdot\Sigma^* = V_{k+i} \cup W_1\cdot X_{1,k+i} \cup \ldots \cup W_\ell\cdot X_{\ell,k+i} \cup \widehat{U}_0\cdot\Sigma^*.$$

[9]Note that for this particular matching problem, $U_0 = \widehat{U}_0$ and $V_0 = \widehat{V}_0$.

Theorem 24 Let C be an $\mathcal{FL}_\neg$-concept description and D an $\mathcal{FL}_\neg$-concept pattern with $\mathcal{FL}_0$-normal forms as introduced above. Then the matching problem $C \equiv^? D$ has a solution if and only if the system of formal language equations $(\bot)$ and $(A_1), \ldots, (A_k)$, $(\neg A_1), \ldots, (\neg A_k)$ is solvable.

PROOF. Let

$$\begin{array}{rclcl} \sigma & := & \{X_1 & \mapsto & \forall L_{1,0}.\bot \sqcap \prod_{i=1}^{k} \forall L_{1,i}.A_i \sqcap \prod_{i=1}^{k} \forall L_{1,k+i}.\neg A_i, \\ & & & \vdots & \\ & & X_\ell & \mapsto & \forall L_{\ell,0}.\bot \sqcap \prod_{i=1}^{k} \forall L_{\ell,i}.A_i \sqcap \prod_{i=1}^{k} \forall L_{\ell,k+i}.\neg A_i\} \end{array}$$

be a substitution.[10] Again, by employing elementary equivalences between concept descriptions we can show that the $\mathcal{FL}_0$-normal form of $\sigma(D)$ is

$$\begin{array}{rcl} \sigma(D) & \equiv & \forall\,(V_0 \cup W_1 \cdot L_{1,0} \cup \cdots \cup W_\ell \cdot L_{\ell,0})\,.\bot \sqcap \\ & & \prod_{i=1}^{k} \forall\,(V_i \cup W_1 \cdot L_{1,i} \cup \cdots \cup W_\ell \cdot L_{\ell,i})\,.A_i \sqcap \\ & & \prod_{i=1}^{k} \forall\,(V_{k+i} \cup W_1 \cdot L_{1,k+i} \cup \cdots \cup W_\ell \cdot L_{\ell,k+i})\,.A_{k+i}. \end{array}$$

Thus, Lemma 22 implies that σ satisfies $C \equiv \sigma(D)$ if and only if the assignment $X_{j,i} := L_{j,i}$ ($j = 1, \ldots, \ell, i = 1, \ldots, 2k$) solves the system of formal language equations $(\bot), (A_1), \ldots, (A_k), (\neg A_1), \ldots, (\neg A_k)$. ∎

It remains to be shown how solvability of the system $(\bot), (A_1), \ldots, (A_k), (\neg A_1), \ldots, (\neg A_k)$ can be tested. In contrast to the formal language equations considered for matching in $\mathcal{FL}_\bot$, the equations of this system cannot be solved separately since $(\bot)$ contains variables also occurring in the other equations. Nevertheless, the approach employed in the previous section for solving the equations separately also applies to the system to be considered here.

Lemma 25 The system of equations $(\bot), (A_1), \ldots, (A_k), (\neg A_1), \ldots, (\neg A_k)$ has a solution if and only if

1. replacing the variables $X_{j,i}$ by the sets $\widehat{L}_{j,i} := \bigcap_{w \in W_j} w^{-1} \cdot (U_i \cup \widehat{U}_0 \cdot \Sigma^*)$ yields a solution of (A_i) for $i = 1, \ldots, k$,
2. replacing the variables $X_{j,k+i}$ by the sets $\widehat{L}_{j,k+i} := \bigcap_{w \in W_j} w^{-1} \cdot (U_{k+i} \cup \widehat{U}_0 \cdot \Sigma^*)$ yields a solution of $(\neg A_i)$ for $i = 1, \ldots, k$, and
3. replacing $X_{j,0} \cdot \Sigma^*$ by the sets $\widehat{L}_{j,0} := \bigcap_{w \in W_j} w^{-1} \cdot (\widehat{U}_0 \cdot \Sigma^*)$ together with the assignments considered in 1. and 2. solves equation $(\bot)$.

[10] Without loss of generality we may assume that σ introduces only atomic concepts occurring in C or D. In fact, additional atomic concepts or negated atomic concepts introduced by a solution of the matching problem can simply be replaced by $\bot$ (see the proof of Proposition 35).

PROOF. To show the *only-if* direction, assume that the assignment $X_{j,i} := M_{j,i}$ ($j = 1, \ldots, \ell$, $i = 0, \ldots, 2k$) solves the system $(\bot), (A_1), \ldots, (A_k), (\neg A_1), \ldots, (\neg A_k)$.

As in the proof of Lemma 17 we can show that $M_{j,i} \subseteq \widehat{L}_{j,i}$ holds for all $j = 1, \ldots, \ell$ and $i = 1, \ldots, 2k$, and that replacing the variables $X_{j,i}$ (resp. $X_{j,k+i}$) by the sets $\widehat{L}_{j,i}$ (resp. $\widehat{L}_{j,k+i}$) solves equation (A_i) (resp. $(\neg A_i)$).

As in the proof of Lemma 15 we can show that $M_{j,0} \cdot \Sigma^* \subseteq \widehat{L}_{j,0}$ holds for all $j = 1, \ldots, \ell$. Together with the inclusions $M_{j,i} \subseteq \widehat{L}_{j,i}$ for $j = 1, \ldots, \ell$ and $i = 1, \ldots, 2k$, this implies that the left-hand side $\widehat{U}_0 \cdot \Sigma^*$ of equation $(\bot)$ is contained in the set obtained by replacing in the right-hand side of $(\bot)$ the variables $X_{j,i}$ by $\widehat{L}_{j,i}$ ($j = 1, \ldots, \ell, i = 1, \ldots, 2k$) and $X_{j,0} \cdot \Sigma^*$ by $\widehat{L}_{j,0}$ ($j = 1, \ldots, \ell$).

To conclude the proof of the *only-if* direction, it remains to be shown that the inclusion in the other direction holds as well. Obviously, $V_0 \cdot \Sigma^* \subseteq \widehat{U}_0 \cdot \Sigma^*$ and $W_j \cdot \widehat{L}_{j,0} \subseteq \widehat{U}_0 \cdot \Sigma^*$ can be shown as in the proof of Lemma 15. Thus, assume that

$$w \in \left(V_i \cup W_1 \cdot \widehat{L}_{1,i} \cup \ldots \cup W_\ell \cdot \widehat{L}_{\ell,i}\right) \cap \left(V_{k+i} \cup W_1 \cdot \widehat{L}_{1,k+i} \cup \ldots \cup W_\ell \cdot \widehat{L}_{\ell,k+i}\right).$$

Since we already know that the equations (A_i) and $(\neg A_i)$ are solved by the assignment $X_{j,i} := \widehat{L}_{j,i}$, this implies that $w \in (U_i \cup \widehat{U}_0 \cdot \Sigma^*) \cap (U_{k+i} \cup \widehat{U}_0 \cdot \Sigma^*)$. By definition of $\widehat{U}_0$, we have $U_i \cap U_{k+i} \subseteq \widehat{U}_0$, and thus $w \in \widehat{U}_0 \cdot \Sigma^*$.

In the proofs of the *if* direction of Lemma 15 and Lemma 17 we have shown how to construct finite languages $L_{j,i}$ ($j = 1, \ldots, \ell$, $i = 0, \ldots, 2k$) such that

- $L_{j,0} \cdot \Sigma^* = \widehat{L}_{j,0}$ for all $j = 1, \ldots, \ell$, and
- $L_{j,i} \cup L_{j,0} \cdot \Sigma^* = \widehat{L}_{j,i}$ for all $j = 1, \ldots, \ell$ and $i = 1, \ldots, 2k$ (see the proof of Lemma 20).

To proof the *if* direction of the present lemma, it remains to be shown that the assignment $X_{j,i} := L_{j,i}$ solves the system $(\bot), (A_1), \ldots, (A_k), (\neg A_1), \ldots, (\neg A_k)$.

For the equations (A_i) (resp. $(\neg A_i)$), this is an immediate consequence of the following facts:

- the assignment $X_{j,i} := \widehat{L}_{j,i}$ (resp. $X_{j,k+i} := \widehat{L}_{j,k+i}$) solves (A_i) (resp. $(\neg A_i)$),
- $\widehat{L}_{j,i} = L_{j,i} \cup L_{j,0} \cdot \Sigma^*$, and
- $W_j \cdot L_{j,0} \cdot \Sigma^* = W_j \cdot \widehat{L}_{j,0} \subseteq \widehat{U}_0 \cdot \Sigma^*$.

For the equation $(\bot)$, let L_i be the language obtained by instantiating $\mathrm{Int}(A_i, \neg A_i)$ with the languages $L_{j,i}$, and $\widehat{L}_i$ the language obtained by instantiating $\mathrm{Int}(A_i, \neg A_i)$ with the languages $\widehat{L}_{j,i}$. Since $L_{j,0} \cdot \Sigma^* = \widehat{L}_{j,0}$, it remains to be shown that any word w in $\widehat{L}_i \setminus L_i$ belongs to $W_1 \cdot L_{1,0} \cdot \Sigma^* \cup \ldots \cup W_\ell \cdot L_{\ell,0} \cdot \Sigma^*$. This is, however, an easy consequence of the fact that $\widehat{L}_{j,i} = L_{j,i} \cup L_{j,0} \cdot \Sigma^*$, which implies that such a word w must belong to $W_j \cdot L_{j,0} \cdot \Sigma^*$ for some $j, 1 \leq j \leq \ell$. ∎

As in the proof of Theorem 18 we can show that this lemma provides us with a polynomial algorithm for deciding solvability of matching problems in $\mathcal{FL}_\neg$. In addition, as in the proof of Lemma 20 we can show that the solution σ obtained from the sets $L_{j,i}$ is the least solution of the matching problem, and that this solution can be computed in polynomial time.

Theorem 26 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.

Corollary 27 Let $C \sqsubseteq^? D$ be a solvable $\mathcal{FL}_\neg$-matching problem modulo subsumption. Then a minimal solution of $C \sqsubseteq^? D$ can be computed in polynomial time.

4.2 Matching in $\mathcal{ALN}$

Let C, D be $\mathcal{ALN}$-concept descriptions, $\mathcal{C}$ the set of atomic concepts occurring in C and D, $\mathcal{N}_\geq$ the set of at-least restrictions in C and D, and $\mathcal{N}_\leq$ the set of at-most restrictions in C and D. Without loss of generality we assume that $\mathcal{N}_\geq$ does not contain at-least restrictions of the form $(\geq 0\ R)$ since these restrictions can be replaced by $\top$.

By treating negated atomic concepts and number restrictions like atomic concepts, we can transform C and D into their $\mathcal{FL}_0$-normal forms

$$\begin{aligned} C \;\equiv\;& \forall U_\bot.\bot \sqcap \bigsqcap_{A\in\mathcal{C}} \forall U_A.A \sqcap \bigsqcap_{A\in\mathcal{C}} \forall U_{\neg A}.\neg A \sqcap \\ & \bigsqcap_{(\geq n\ R)\in\mathcal{N}_\geq} \forall U_{\geq nR}.(\geq n\ R) \sqcap \bigsqcap_{(\leq n\ R)\in\mathcal{N}_\leq} \forall U_{\leq nR}.(\leq n\ R), \end{aligned} \tag{3}$$

$$\begin{aligned} D \;\equiv\;& \forall V_\bot.\bot \sqcap \bigsqcap_{A\in\mathcal{C}} \forall V_A.A \sqcap \bigsqcap_{A\in\mathcal{C}} \forall V_{\neg A}.\neg A \sqcap \\ & \bigsqcap_{(\geq n\ R)\in\mathcal{N}_\geq} \forall V_{\geq nR}.(\geq n\ R) \sqcap \bigsqcap_{(\leq n\ R)\in\mathcal{N}_\leq} \forall V_{\leq nR}.(\leq n\ R). \end{aligned} \tag{4}$$

In the following, we will also use the notation $U_{\geq nR}$ (resp. $U_{\leq nR}$, U_A, $U_{\neg A}$) for at-least restrictions (resp. at-most restrictions, atomic concepts and negated atomic concepts) not contained in $\mathcal{N}_\geq$ (resp. $\mathcal{N}_\leq$, $\mathcal{C}$). In this case, these sets are assumed to be empty. The same holds for V in place of U.

If C is an $\mathcal{FL}_\neg$-concept description, then the set $\widehat{U}_0{\cdot}\Sigma^*$ (as defined in Section 4.1) is equal to the set $\{w \in \Sigma^* \mid C \sqsubseteq \forall w.\bot\}$ of so-called C-excluding words. The following definition generalizes this notion to $\mathcal{ALN}$-concept descriptions:

Definition 28 For an $\mathcal{ALN}$-concept description C, we define $E_C := \{w \in \Sigma^* \mid C \sqsubseteq \forall w.\bot\}$, and call this set the set of *C-excluding words.*

In order to provide a syntactic description of E_C we need one more notation.

Definition 29 A word $w = R_1 \cdots R_n \in \Sigma^*$ is *required by the $\mathcal{ALN}$-concept description C* (with $\mathcal{FL}_0$-normal form as in (3)) *starting from* $v = R_1 \cdots R_m$, $m \leq n$, if and only if for all i, $m \leq i < n$, there are numbers $k_{i+1} \geq 1$ such that $vR_{m+1} \cdots R_i \in U_{\geq k_{i+1} R_{i+1}}$.

Note that both n and m in this definition may be 0. Thus, the empty word ε is required (starting from ε) by any $\mathcal{ALN}$-concept description. The intuition underlying the notion of required words is clarified in the next lemma.

Lemma 30 Assume that $w = R_1 \cdots R_n$ is required by C starting from $v = R_1 \cdots R_m$, $m \leq n$, and that I is an interpretation such that $d \in C^I$ and $(d, e) \in R_1^I \circ \ldots \circ R_m^I$. Then e has an $(R_{m+1} \cdots R_n)$-successor in I, i.e., there is an individual f such that $(e, f) \in R_{m+1}^I \circ \ldots \circ R_n^I$.

Example 31 Let us illustrate the above definition using the following $\mathcal{ALN}$-concept descriptions:

$$\begin{aligned} C &:= \forall\{RS, R\}.(\geq 2\ S) \sqcap \forall\{RS\}.(\leq 1\ S) \sqcap \forall\{S\}.A_1, \\ D &:= \forall\{R\}.(\geq 2\ S) \sqcap \forall\{R\}.(\leq 1\ S) \sqcap \forall\{RRS, S\}.A_1 \sqcap (\leq 1\ R). \end{aligned}$$

It is easy to see that both RS and RSS are required by C starting from R. The concept D also requires RS starting from R, but it does not require RSS.

Using the notion of required words, exclusion can be characterized as follows [23]:

Theorem 32 For an $\mathcal{ALN}$-concept description C (with $\mathcal{FL}_0$-normal form as in (3)) it holds that $w \in E_C$ if and only if

1. there exists a prefix $v \in \Sigma^*$ of w and a word $v' \in \Sigma^*$ such that vv' is required by C starting from v and
   (a) $vv' \in U_\perp$, or
   (b) there is an atomic concept A with $vv' \in U_A \cap U_{\neg A}$, or
   (c) there are number restrictions $(\geq \ell\ R)$ and $(\leq r\ R)$ such that $\ell > r$ and $vv' \in U_{\geq \ell R} \cap U_{\leq rR}$; or
2. there exists a prefix vR of w (with $v \in \Sigma^*$, $R \in \Sigma$) such that $v \in U_{\leq 0R}$.

Furthermore, it can be shown [5]:

Proposition 33 For an $\mathcal{ALN}$-concept description C one can compute (in polynomial time) a finite set of words U such that $E_C = U{\cdot}\Sigma^*$.

Using Theorem 32 (1c), it is not hard to verify that, for the concept description C and D of Example 31, the sets of excluding words are $E_C = E_D = R\{R, S\}^*$. Thus, in both cases we can take $U := \{R\}$ in the above proposition.

Again, equivalence[11] of $\mathcal{ALN}$-concept descriptions can be characterized in terms of certain regular languages [23]:

Theorem 34 Let C, D be $\mathcal{ALN}$-concept descriptions with $\mathcal{FL}_0$-normal forms as introduced in (3) and (4), respectively. Then $C \equiv D$ if and only if for all $A \in \mathcal{C}$, $(\geq n\ R) \in \mathcal{N}_\geq$, and $(\leq n\ R) \in \mathcal{N}_\leq$ we have

$$\begin{aligned} E_C &= E_D, \\ U_A \cup E_C &= V_A \cup E_D, \\ U_{\neg A} \cup E_C &= V_{\neg A} \cup E_D, \\ \textstyle\bigcup_{m \geq n} U_{\geq mR} \cup E_C &= \textstyle\bigcup_{m \geq n} V_{\geq mR} \cup E_D, \text{ and} \\ \textstyle\bigcup_{m \leq n} U_{\leq mR} \cup E_C{\cdot}R^{-1} &= \textstyle\bigcup_{m \leq n} V_{\leq mR} \cup E_D{\cdot}R^{-1}, \end{aligned}$$

where, for $L \subseteq \Sigma^*$, we define $L{\cdot}R^{-1} := \{w \in \Sigma^* \mid wR \in L\}$.

[11] For subsumption, it can be shown that $C \sqsubseteq D$ if and only if set inclusion "$\supseteq$" instead of equality "$=$" holds between the languages.

The intuition underlying these equations is that the languages on the left- and right-hand sides represent all value-restrictions satisfied by C and D, respectively. For example, $C \sqsubseteq \forall w.A$ iff $w \in U_A \cup E_C$. The set of excluding words E_C is necessary in this characterization because of $\forall w.\bot \sqsubseteq \forall w.A$. For at-most restrictions we must use $E_C \cdot R^{-1}$ instead of E_C since $\forall wR.\bot \sqsubseteq \forall w.(\leq n\ R)$ for any $n \geq 0$.

Let us illustrate Theorem 34 using the concept descriptions C, D introduced in Example 31. According to the definition of C and D and the sets of excluding words we have already computed for them, Theorem 34 says that $C \equiv D$ holds if and only if the following identities are true:

$$\begin{array}{lrcl}
(\bot) & R\{R,S\}^* & = & R\{R,S\}^* \\
(A_1) & \{S\} \cup R\{R,S\}^* & = & \{RRS, S\} \cup R\{R,S\}^* \\
(\geq 2\ S) & \{RS, R\} \cup R\{R,S\}^* & = & \{R\} \cup R\{R,S\}^* \\
(\leq 1\ S) & \{RS\} \cup R\{R,S\}^* \cdot S^{-1} & = & \{R\} \cup R\{R,S\}^* \cdot S^{-1} \\
(\leq 1\ R) & \emptyset \cup R\{R,S\}^* \cdot R^{-1} & = & \{\varepsilon\} \cup R\{R,S\}^* \cdot R^{-1}
\end{array}$$

It is easy to see that these identities are indeed true, and thus we can conclude that C and D are equivalent. The identity for $(\leq 1\ R)$ shows that we really must use $E_C \cdot R^{-1}$ instead of E_C and $E_D \cdot R^{-1}$ instead of E_D. In fact, $\emptyset \cup R\{R,S\}^* \neq \{\varepsilon\} \cup R\{R,S\}^*$, and thus, using E_C, E_D in place of $E_C \cdot R^{-1}, E_D \cdot R^{-1}$, we would have concluded (incorrectly) that C and D are not equivalent.

The characterization of equivalence provided by Theorem 34 can again be used to reduce a given $\mathcal{ALN}$-matching problem $C \equiv^? D$ to a system of formal language equations. In the sequel, we assume that the $\mathcal{FL}_0$-normal form of C is given as in (3) and the one for the $\mathcal{ALN}$-concept pattern D is

$$\begin{aligned}
D \quad \equiv \quad & \forall V_\bot.\bot \sqcap \bigsqcap_{A \in \mathcal{C}} \forall V_A.A \sqcap \bigsqcap_{A \in \mathcal{C}} \forall V_{\neg A}.\neg A \sqcap \\
& \bigsqcap_{(\geq n\ R) \in \mathcal{N}_\geq} \forall V_{\geq nR}.(\geq n\ R) \sqcap \bigsqcap_{(\leq n\ R) \in \mathcal{N}_\leq} \forall V_{\leq nR}.(\leq n\ R) \sqcap \\
& \bigsqcap_{i=1}^{\ell} \forall W_i.X_i.
\end{aligned} \tag{5}$$

Proposition 35 The matching problem $C \equiv^? D$ has a solution σ iff it has a solution $\widehat{\sigma}$ that does not introduce new atomic concepts or number restrictions.

Proof. The *if direction* is trivial. For the *only-if direction*, we distinguish two cases, depending on whether the new concept is atomic or a number restriction.

(1) First, assume that σ introduces exactly one new atomic concept $B \notin \mathcal{C}$. Thus, the $\mathcal{FL}_0$-normal form of $\sigma(D)$ has the form

$$\begin{aligned}
\sigma(D) \quad \equiv \quad & \forall V'_\bot.\bot \sqcap \bigsqcap_{A \in \mathcal{C}} \forall V'_A.A \sqcap \bigsqcap_{A \in \mathcal{C}} \forall V'_{\neg A}.\neg A \sqcap \\
& \bigsqcap_{(\geq n\ R) \in \mathcal{N}_\geq} \forall V'_{\geq nR}.(\geq n\ R) \sqcap \bigsqcap_{(\leq n\ R) \in \mathcal{N}_\leq} \forall V'_{\leq nR}.(\leq n\ R) \sqcap \\
& \forall V'_B.B \sqcap \forall V'_{\neg B}.\neg B.
\end{aligned} \tag{6}$$

We obtain $\widehat{\sigma}$ by replacing every occurrence of B and $\neg B$ in σ by $\bot$. Then, it is easy to see that

$$\begin{aligned}
\widehat{\sigma}(D) \quad \equiv \quad & \forall (V'_\bot \cup V'_B \cup V'_{\neg B}).\bot \sqcap \bigsqcap_{A \in \mathcal{C}} \forall V'_A.A \sqcap \bigsqcap_{A \in \mathcal{C}} \forall V'_{\neg A}.\neg A \sqcap \\
& \bigsqcap_{(\geq n\ R) \in \mathcal{N}_\geq} \forall V'_{\geq nR}.(\geq n\ R) \sqcap \bigsqcap_{(\leq n\ R) \in \mathcal{N}_\leq} \forall V'_{\leq nR}.(\leq n\ R).
\end{aligned} \tag{7}$$

Since $\bot \sqsubseteq B$ and $\bot \sqsubseteq \neg B$ it follows that $\widehat{\sigma}(D) \sqsubseteq \sigma(D)$.

Conversely, since B is a concept name not occurring in C, D, we know that $U_B = U_{\neg B} = \emptyset$. By Theorem 34, we can conclude from $C \equiv \sigma(D)$ that $U_B \cup E_C = V'_B \cup E_C$ as well as $U_{\neg B} \cup E_C = V'_{\neg B} \cup E_C$. This implies $V'_B \cup V'_{\neg B} \subseteq E_C = E_{\sigma(D)}$, and thus $\sigma(D) \sqsubseteq \forall(V'_B \cup V'_{\neg B}).\bot$. Let D' be the concept description obtained from $\sigma(D)$ by removing the conjunct $\forall V'_B.B \sqcap \forall V'_{\neg B}.\neg B$. Obviously, $\sigma(D) \sqsubseteq D'$ and $\widehat{\sigma}(D) = D' \sqcap \forall(V'_B \cup V'_{\neg B}).\bot$. This, together with $\sigma(D) \sqsubseteq \forall(V'_B \cup V'_{\neg B}).\bot$, implies $\sigma(D) \sqsubseteq \widehat{\sigma}(D)$.

If σ introduces more than one new atomic concept, then we simply iterate this argument.

(2) In the second case, we assume that σ introduces exactly one new at-least restriction $(\geq k\ S) \notin \mathcal{N}_\geq$. Thus, the $\mathcal{FL}_0$-normal form of $\sigma(D)$ has the form:

$$\begin{aligned}\sigma(D) \quad \equiv \quad & \forall V'_\bot.\bot \sqcap \bigsqcap_{A\in\mathcal{C}} \forall V'_A.A \sqcap \bigsqcap_{A\in\mathcal{C}} \forall V'_{\neg A}.\neg A \sqcap \\ & \bigsqcap_{(\geq\, n\, R)\in\mathcal{N}_\geq} \forall V'_{\geq nR}.(\geq n\ R) \sqcap \bigsqcap_{(\leq\, n\, R)\in\mathcal{N}_\leq} \forall V'_{\leq nR}.(\leq n\ R) \sqcap \\ & \forall V'_{\geq k S}.(\geq k\ S)\end{aligned} \tag{8}$$

We distinguish two sub-cases:

(a) There is an at-least restriction $(\geq h\ S) \in \mathcal{N}_\geq$ with $h > k$ and there is no $h' < h$ with this property, i.e., we choose the "least" at-least restriction on S in $\mathcal{N}_\geq$ that is "'greater" than $(\geq k\ S)$ (in the sense that the number h occurring in this restriction is larger than k). We obtain $\widehat{\sigma}$ by replacing $(\geq k\ S)$ in σ by $(\geq h\ S)$. Thus,

$$\begin{aligned}\widehat{\sigma}(D) \quad \equiv \quad & \forall V'_\bot.\bot \sqcap \bigsqcap_{A\in\mathcal{C}} \forall V'_A.A \sqcap \bigsqcap_{A\in\mathcal{C}} \forall V'_{\neg A}.\neg A \sqcap \\ & \bigsqcap_{(\geq\, n\, R)\in\mathcal{N}_\geq\setminus\{(\geq\, h\, S)\}} \forall V'_{\geq nR}.(\geq n\ R) \sqcap \\ & \bigsqcap_{(\leq\, n\, R)\in\mathcal{N}_\leq} \forall V'_{\leq nR}.(\leq n\ R) \sqcap \forall(V'_{\geq hS} \cup V'_{\geq kS}).(\geq h\ S).\end{aligned} \tag{9}$$

Since $(\geq h\ S) \sqsubseteq (\geq k\ S)$ we know that $\widehat{\sigma}(D) \sqsubseteq \sigma(D)$.

Because $C \equiv \sigma(D)$, Theorem 34 yields $V'_{\geq kS} \subseteq \bigcup_{m\geq k} U_{\geq mS} \cup E_C$. Furthermore, $\bigcup_{m=k}^{h-1} U_{\geq mS} = \emptyset$ by definition of $(\geq h\ S)$. Consequently, $V'_{\geq kS} \subseteq \bigcup_{m\geq h} U_{\geq mS} \cup E_C$. Therefore, using the characterization of subsumption it can be verified that $C \sqsubseteq \forall V'_{\geq kS}.(\geq h\ S)$, and thus $\sigma(D) \sqsubseteq \forall V'_{\geq kS}.(\geq h\ S)$. Let D' be the concept description obtained from $\sigma(D)$ by removing the conjunct $\forall V'_{\geq kS}.(\geq k\ S)$. Obviously, $\sigma(D) \sqsubseteq D'$ and $\widehat{\sigma}(D) = D' \sqcap \forall V'_{\geq kS}.(\geq h\ S)$. This, together with $\sigma(D) \sqsubseteq \forall V'_{\geq kS}.(\geq h\ S)$, implies $\sigma(D) \sqsubseteq \widehat{\sigma}(D)$.

(b) If there is no greater at-least restriction $(\geq h\ S) \in \mathcal{N}_\geq$ for $(\geq k\ S)$, then $\widehat{\sigma}$ is obtained from σ by replacing all occurrences of $(\geq k\ S)$ in σ by $\bot$. Theorem 34 for C and $\sigma(D)$ yields $V'_{\geq kS} \subseteq E_C$, and thus one can show $\sigma(D) \equiv \widehat{\sigma}(D)$ as in the first part of the proof.

If more than one new at-least restriction is introduced by σ, then the argument presented above can again be iterated.

For at-most restrictions one chooses the greatest at-most restriction in $\mathcal{N}_{\leq}$ that is less than $(\leq k\ S)$. If there is no such at-most restriction, again, $(\leq k\ S)$ is replaced by $\bot$. The proof for at-most restrictions is very similar to the proof for at-least restrictions. ∎

Matching of the pattern D onto the description C can again be reduced to solving a system of formal language equations. First, we organize the variables occurring in the system of equations by defining certain tuples of variables:

$$\begin{array}{rcl} X_{\bot} & := & (X_{1,\bot}, \ldots, X_{\ell,\bot}), \\ X_{\mathcal{C}} & := & (X_{i,A} \,|\, 1 \leq i \leq \ell, A \in \mathcal{C}), \\ X_{\neg} & := & (X_{i,\neg A} \,|\, 1 \leq i \leq \ell, A \in \mathcal{C}), \\ X_{\geq} & := & (X_{i,\geq nR} \,|\, 1 \leq i \leq \ell, (\geq n\ R) \in \mathcal{N}_{\geq}), \\ X_{\leq} & := & (X_{i,\leq nR} \,|\, 1 \leq i \leq \ell, (\leq n\ R) \in \mathcal{N}_{\leq}). \end{array}$$

An assignment of finite languages $L_{i,\bot}$ to $X_{i,\bot}$, $L_{i,A}$ to $X_{i,A}$, $L_{i,\neg A}$ to $X_{i,\neg A}$, $L_{i,\geq nR}$ to $X_{i,\geq nR}$, and $L_{i,\leq nR}$ to $X_{i,\leq nR}$ defines the following substitution σ:

$$\begin{array}{rcl} \sigma(X_i) & := & \forall L_{i,\bot}.\bot \sqcap \bigsqcap_{A\in\mathcal{C}} \forall L_{i,A}.A \sqcap \bigsqcap_{A\in\mathcal{C}} \forall L_{i,\neg A}.\neg A \sqcap \\ & & \bigsqcap_{(\geq n\ R)\in\mathcal{N}_{\geq}} \forall L_{i,\geq nR}.(\geq n\ R) \sqcap \bigsqcap_{(\leq n\ R)\in\mathcal{N}_{\leq}} \forall L_{i,\leq nR}.(\leq n\ R) \end{array} \tag{10}$$

for $i = 1, \ldots, \ell$.

For a given assignment, the operator $E_D(X_{\bot}, X_{\mathcal{C}}, X_{\neg}, X_{\geq}, X_{\leq})$ yields the set $E_{\sigma(D)}$ of $\sigma(D)$-excluding words, where σ is the substitution defined by the assignment.

The following equations correspond to the matching problem $C \equiv^? D$:

$$(\bot) \quad E_C = E_D(X_{\bot}, X_{\mathcal{C}}, X_{\neg}, X_{\geq}, X_{\leq}),$$

for all $A \in \mathcal{C}$

$$\begin{array}{lrcl} (A) & U_A \cup E_C & = & V_A \cup W_1{\cdot}X_{1,A} \cup \cdots \cup W_\ell{\cdot}X_{\ell,A} \cup E_C, \\ (\neg A) & U_{\neg A} \cup E_C & = & V_{\neg A} \cup W_1{\cdot}X_{1,\neg A} \cup \cdots \cup W_\ell{\cdot}X_{\ell,\neg A} \cup E_C, \end{array}$$

for all $(\geq n\ R) \in \mathcal{N}_{\geq}$

$$\begin{array}{lrcl} (\geq n\ R) & \bigcup_{m\geq n} U_{\geq mR} \cup E_C & = & \bigcup_{m\geq n} V_{\geq mR} \cup \\ & & & W_1{\cdot}X_{1,\geq nR} \cup \cdots \cup W_\ell{\cdot}X_{\ell,\geq nR} \cup E_C, \end{array}$$

and for all $(\leq n\ R) \in \mathcal{N}_{\leq}$

$$\begin{array}{lrcl} (\leq n\ R) & \bigcup_{m\leq n} U_{\leq mR} \cup E_C{\cdot}R^{-1} & = & \bigcup_{m\leq n} V_{\leq mR} \cup \\ & & & W_1{\cdot}X_{1,\leq nR} \cup \cdots \cup W_\ell{\cdot}X_{\ell,\leq nR} \cup E_C{\cdot}R^{-1}. \end{array}$$

Theorem 36 Let C be an $\mathcal{ALN}$-concept description and D an $\mathcal{ALN}$-concept pattern with $\mathcal{FL}_0$-normal forms as introduced in (3) and (5). Then the matching problem $C \equiv^? D$ has a solution if and only if the system of formal language equations $(\bot)$, (A), $(\neg A)$, $(\geq n\, R)$, and $(\leq n\, R)$ is solvable, where $A \in \mathcal{C}$, $(\geq n\, R) \in \mathcal{N}_{\geq}$, and $(\leq n\, R) \in \mathcal{N}_{\leq}$.

PROOF. By Proposition 35 we can restrict our attention to substitutions that do not introduce new atomic concepts or number restrictions. Thus, let σ be a substitution of the form shown in (10). Then,

$$\begin{aligned} \sigma(D) \;\equiv\;& \forall V'_\bot.\bot \;\sqcap\; \prod_{A\in\mathcal{C}} \forall V'_A.A \;\sqcap\; \prod_{A\in\mathcal{C}} \forall V'_{\neg A}.\neg A \;\sqcap \\ & \prod_{(\geq n\ R)\in\mathcal{N}_\geq} \forall V'_{\geq nR}.(\geq n\ R) \;\sqcap \prod_{(\leq n\ R)\in\mathcal{N}_\leq} \forall V'_{\leq nR}.(\leq n\ R), \end{aligned} \tag{11}$$

where

$$\begin{aligned} V'_\bot &:= V_\bot \cup W_1{\cdot}L_{1,\bot} \cup \dots \cup W_\ell{\cdot}L_{\ell,\bot}, \\ V'_A &:= V_A \cup W_1{\cdot}L_{1,A} \cup \dots \cup W_\ell{\cdot}L_{\ell,A}, \\ V'_{\neg A} &:= V_{\neg A} \cup W_1{\cdot}L_{1,\neg A} \cup \dots \cup W_\ell{\cdot}L_{\ell,\neg A}, \\ V'_{\geq nR} &:= V_{\geq nR} \cup W_1{\cdot}L_{1,\geq nR} \cup \dots \cup W_\ell{\cdot}L_{\ell,\geq nR}, \\ V'_{\leq nR} &:= V_{\leq nR} \cup W_1{\cdot}L_{1,\leq nR} \cup \dots \cup W_\ell{\cdot}L_{\ell,\leq nR}. \end{aligned}$$

Since for $n \geq m$ we have $(\geq n\,R) \sqsubseteq (\geq m\,R)$, we can without loss of generality assume that $L_{i,\geq mR} \supseteq L_{i,\geq nR}$ for all $n \geq m$ and $1 \leq i \leq \ell$. Analogously, we may assume that $L_{i,\leq mR} \supseteq L_{i,\leq nR}$ for all $n \leq m$ and $1 \leq i \leq \ell$. Consequently, we have

$$\begin{aligned} \bigcup_{m\geq n} V'_{\geq mR} &= \bigcup_{m\geq n} V_{\geq mR} \cup W_1{\cdot} \bigcup_{m\geq n} L_{1,\geq mR} \cup \dots \cup W_\ell{\cdot} \bigcup_{m\geq n} L_{\ell,\geq mR} \\ &= \bigcup_{m\geq n} V_{\geq mR} \cup W_1{\cdot}L_{1,\geq nR} \cup \dots \cup W_\ell{\cdot}L_{\ell,\geq nR}. \end{aligned}$$

An analogous identity holds for at-most restrictions. These identities, together with Theorem 34, imply that $C \equiv \sigma(D)$ if and only if

$$\begin{array}{lrcl} (\bot)' & E_C &=& E_{\sigma(D)}, \\ (A)' & U_A \cup E_C &=& V_A \cup W_1{\cdot}L_{1,A} \cup \dots \cup W_\ell{\cdot}L_{\ell,A} \cup E_C, \\ (\neg A)' & U_{\neg A} \cup E_C &=& V_{\neg A} \cup W_1{\cdot}L_{1,\neg A} \cup \dots \cup W_\ell{\cdot}L_{\ell,\neg A} \cup E_C, \\ (\geq n\ R)' & \bigcup_{m\geq n} U_{\geq mR} \cup E_C &=& \bigcup_{m\geq n} V_{\geq mR} \cup \\ & & & W_1{\cdot}L_{1,\geq nR} \cup \dots \cup W_\ell{\cdot}L_{\ell,\geq nR} \cup E_C, \\ (\leq n\ R)' & \bigcup_{m\leq n} U_{\leq mR} \cup E_C{\cdot}R^{-1} &=& \bigcup_{m\leq n} V_{\leq mR} \cup \\ & & & W_1{\cdot}L_{1,\leq nR} \cup \dots \cup W_\ell{\cdot}L_{\ell,\leq nR} \cup E_C{\cdot}R^{-1}. \end{array}$$

We are now ready to proof the statement of the theorem.

First, assume that the substitution σ solves $C \equiv^? D$. Without loss of generality, we may assume that σ is of the form shown in (10), and that it satisfies $L_{i,\geq mR} \supseteq L_{i,\geq nR}$ for all $n \geq m$ and $1 \leq i \leq \ell$, and $L_{i,\leq mR} \supseteq L_{i,\leq nR}$ for all $n \leq m$ and $1 \leq i \leq \ell$. Because of $C \equiv \sigma(D)$ we know that the identities $(\bot)'$, $(A)'$, $(\neg A)'$, $(\geq n\ R)'$, and $(\leq n\,R)'$ are satisfied, which shows that the system of formal language equations $(\bot)$, (A), $(\neg A)$, $(\geq n\ R)$, and $(\leq n\ R)$ for $A \in \mathcal{C}$, $(\geq n\ R) \in \mathcal{N}_\geq$, and $(\leq n\ R) \in \mathcal{N}_\leq$ is solvable.

Conversely, assume that the system $(\bot)$, (A), $(\neg A)$, $(\geq n\ R)$, and $(\leq n\ R)$ is solved by the languages $L_{i,\bot}$, $L_{i,A}$, $L_{i,\neg A}$, $L_{i,\geq nR}$, $L_{i,\leq nR}$. Because of the union on the left-hand side of the equations for number restrictions, it is easy to see that we can assume without loss of generality that the following inclusion relationships hold: $L_{i,\geq mR} \supseteq L_{i,\geq nR}$ for all $n \geq m$ and $1 \leq i \leq \ell$, and $L_{i,\leq mR} \supseteq L_{i,\leq nR}$ for all $n \leq m$ and $1 \leq i \leq \ell$. If the substitution σ is defined as in (10) using the languages of the solution of $(\bot)$, (A), $(\neg A)$, $(\geq n\ R)$, and $(\leq n\ R)$, then it follows that the identities $(\bot)'$, $(A)'$, $(\neg A)'$, $(\geq n\ R)'$, and $(\leq n\ R)'$ are satisfied, and thus $C \equiv \sigma(D)$. ∎

In order to compute candidate solutions of the system of equations corresponding to $C \equiv^? D$, we generalize the definitions of the languages $\widehat{L}_{j,i}$ introduced in Lemma 25:

$$\begin{aligned}
\widehat{L}_{i,\bot} &:= \bigcap_{w\in W_i} w^{-1}E_C \\
\widehat{L}_{i,A} &:= \bigcap_{w\in W_i} w^{-1}(U_A \cup E_C), \\
\widehat{L}_{i,\neg A} &:= \bigcap_{w\in W_i} w^{-1}(U_{\neg A} \cup E_C), \\
\widehat{L}_{i,\geq nR} &:= \bigcap_{w\in W_i} w^{-1}(\bigcup_{m\geq n} U_{\geq mR} \cup E_C), \\
\widehat{L}_{i,\leq nR} &:= \bigcap_{w\in W_i} w^{-1}(\bigcup_{m\leq n} U_{\leq mR} \cup E_C{\cdot}R^{-1})
\end{aligned}$$

for all $1 \leq i \leq \ell$, $A \in \mathcal{C}$, $(\geq n\ R) \in \mathcal{N}_{\geq}$, and $(\leq n\ R) \in \mathcal{N}_{\leq}$. Using these (possibly infinite) languages we define finite languages that are a solution of the system of equations corresponding to $C \equiv^? D$, provided that there exists a solution (see Lemma 37).

By Proposition 33 we know that E_C is of the form $U{\cdot}\Sigma^*$. Therefore, as a consequence of Lemma 16, for all $1 \leq i \leq \ell$, there exists a language $L_{i,\bot}$ such that, $L_{i,\bot}{\cdot}\Sigma^* = \widehat{L}_{i,\bot}$. Furthermore, we define

$$\begin{aligned}
L_{i,A} &:= \widehat{L}_{i,A} \setminus \widehat{L}_{i,\bot}, \\
L_{i,\neg A} &:= \widehat{L}_{i,\neg A} \setminus \widehat{L}_{i,\bot}, \\
L_{i,\geq nR} &:= \widehat{L}_{i,\geq nR} \setminus \widehat{L}_{i,\bot}, \\
L_{i,\leq nR} &:= \widehat{L}_{i,\leq nR} \setminus \widehat{L}_{i,\bot},
\end{aligned}$$

for all $1 \leq i \leq \ell$, $A \in \mathcal{C}$, $(\geq n\ R) \in \mathcal{N}_{\geq}$, and $(\leq n\ R) \in \mathcal{N}_{\leq}$. Applying the fact that E_C is of the form $U{\cdot}\Sigma^*$, it is easy to see that $E_C{\cdot}R^{-1} = U{\cdot}\Sigma^* \cup U{\cdot}R^{-1}$. Similar to the proof of Lemma 17 one can now show that the languages $L_{\cdot,\cdot}$ introduced above are finite.

These languages can be used to determine whether the system of equations corresponding to $C \equiv^? D$ has a solution or not.

Lemma 37 The system of equations $(\bot)$, (A), $(\neg A)$, $(\geq n\ R)$, and $(\leq n\ R)$, where $A \in \mathcal{C}$, $(\geq n\ R) \in \mathcal{N}_{\geq}$, and $(\leq n\ R) \in \mathcal{N}_{\leq}$, has a solution if and only if

1. replacing the variables $X_{i,A}$, $1 \leq i \leq \ell$, by the sets $L_{i,A}$ yields a solution of equation (A) for all $A \in \mathcal{C}$,

2. replacing the variables $X_{i,\neg A}$, $1 \leq i \leq \ell$, by the sets $L_{i,\neg A}$ yields a solution of equation $(\neg A)$ for all $A \in \mathcal{C}$,

3. replacing the variables $X_{i,\geq nR}$, $1 \leq i \leq \ell$, by the sets $L_{i,\geq nR}$ yields a solution of equation $(\geq n\ R)$ for all at-least restrictions $(\geq n\ R) \in \mathcal{N}_{\geq}$,

4. replacing the variables $X_{i,\leq nR}$, $1 \leq i \leq \ell$, by the sets $L_{i,\leq nR}$ yields a solution of equation $(\leq n\ R)$ for all at-most restrictions $(\leq n\ R) \in \mathcal{N}_{\leq}$,

5. replacing the variables $X_{i,\bot}$, $1 \leq i \leq \ell$, by the sets $L_{i,\bot}$ together with the assignments considered in 1.–4. solves equation $(\bot)$.

Proof. The *if direction* of this lemma is trivial. To show the *only-if direction*, let $M_{i,\bot}$, $M_{i,A}$, $M_{i,\neg A}$, $M_{i,\geq nR}$, and $M_{i,\leq nR}$ denote the languages assigned by a solution of the equation system, and let σ_M be the substitution defined by this solution.

Claim 1: $M_{i,\bot} \subseteq \widehat{L}_{i,\bot}$, $M_{i,A} \subseteq \widehat{L}_{i,A}$, $M_{i,\neg A} \subseteq \widehat{L}_{i,\neg A}$, $M_{i,\geq nR} \subseteq \widehat{L}_{i,\geq nR}$, and $M_{i,\leq nR} \subseteq \widehat{L}_{i,\leq nR}$.

Proof of the claim. For A, $\neg A$, $(\geq n\ R)$, and $(\leq n\ R)$ this can be shown as in the proof of Lemma 15. Obviously, we have $W_i{\cdot}M_{i,\bot} \subseteq E_{\sigma_M(D)} \equiv E_C$. Thus, $w \in W_i$ and $v \in M_{i,\bot}$ imply $v \in w^{-1}{\cdot}E_C$. Since this holds for all $w \in W_i$, we have $v \in \widehat{L}_{i,\bot}$, and therefore, $M_{i,\bot} \subseteq \widehat{L}_{i,\bot}$, which completes the proof of Claim 1. $\diamond$

Let $\sigma_\bot$ be the substitution defined by using the languages $L_{i,\bot}$ in place of $M_{i,\bot}$, and the languages $M_{i,A}$, $M_{i,\neg A}$, $M_{i,\geq nR}$, $M_{i,\leq nR}$.

Claim 2: $\sigma_\bot \sqsubseteq \sigma_M$ and $\sigma_\bot(D) \equiv \sigma_M(D) \equiv C$.

Proof of the claim. Claim 1 yields $M_{i,\bot} \subseteq \widehat{L}_{i,\bot}$. Since $\widehat{L}_{i,\bot} = L_{i,\bot}{\cdot}\Sigma^*$, this implies $\forall L_{i,\bot}.\bot \sqsubseteq \forall M_{i,\bot}.\bot$. As an easy consequence, we obtain $\sigma_\bot \sqsubseteq \sigma_M$.

Now, $\forall L_{i,\bot}.\bot \sqsubseteq \forall M_{i,\bot}.\bot$, together with the definition of the substitution $\sigma_\bot$, yields $\sigma_\bot(D) \equiv \sigma_M(D) \sqcap \bigsqcap_{i=1}^{\ell} \forall W_i{\cdot}L_{i,\bot}.\bot$. By definition of $L_{i,\bot}$, we know that $W_i{\cdot}L_{i,\bot} \subseteq E_C \equiv E_{\sigma_M(D)}$. This yields $\sigma_\bot(D) \equiv \sigma_M(D)$, and thus completes the proof of Claim 2. $\diamond$

Let σ' be the substitution defined by the languages $M'_{i,\bot} := L_{i,\bot}$, $M'_{i,A} := M_{i,A} \setminus \widehat{L}_{i,\bot}$, $M'_{i,\neg A} := M_{i,\neg A} \setminus \widehat{L}_{i,\bot}$, $M'_{i,\geq nR} := M_{i,\geq nR} \setminus \widehat{L}_{i,\bot}$, and $M'_{i,\leq nR} := M_{i,\leq nR} \setminus \widehat{L}_{i,\bot}$.

Claim 3: $\sigma' \sqsubseteq \sigma_\bot$ and $\sigma_\bot \sqsubseteq \sigma'$, i.e., σ' and $\sigma_\bot$ are equivalent.

Proof of the claim. If $w \in M_{i,A} \cap \widehat{L}_{i,\bot}$, then (by definition of $L_{i,\bot}$) there is a word $v \in L_{i,\bot}$ and $v' \in \Sigma^*$ such that $w = vv'$. Furthermore, $\forall w.A \sqcap \forall v.\bot \equiv \forall v.\bot$. This also holds for $\neg A$, $(\geq n\ R)$ and $(\leq n\ R)$ in place of A. As an easy consequence of this observation we obtain that σ' and $\sigma_\bot$ are equivalent. $\diamond$

Claims 2 and 3 imply that the languages $M'_{i,\bot}$, $M'_{i,A}$, $M'_{i,\neg A}$, $M'_{i,\geq nR}$, $M'_{i,\leq nR}$ also yield a solution of the equation system. Moreover, by Claim 1 and the definition of the languages $M'_{\cdot,\cdot}$ and $L_{\cdot,\cdot}$ it follows that $M'_{i,\bot} \subseteq L_{i,\bot}$, $M'_{i,A} \subseteq L_{i,A}$, $M'_{i,\neg A} \subseteq L_{i,\neg A}$, $M'_{i,\geq nR} \subseteq L_{i,\geq nR}$, and $M'_{i,\leq nR} \subseteq L_{i,\geq nR}$. Thus, for the languages $L_{\cdot,\cdot}$, the $\subseteq$-direction of the equations (A), $(\neg A)$, $(\geq n\ R)$, and $(\leq n\ R)$ holds.

As in the proof of Lemma 15 it can be shown that for the languages $\widehat{L}_{\cdot,\cdot}$ the inclusion in the other direction holds as well. Consequently, this is also the case for the languages $L_{\cdot,\cdot} \subseteq \widehat{L}_{\cdot,\cdot}$. To sum up, we have shown that, for the languages $L_{\cdot,\cdot}$, the equations (A), $(\neg A)$, $(\geq n\ R)$, and $(\leq n\ R)$ are satisfied. It remains to be shown that $(\bot)$ is also satisfied for these languages.

Let σ be the substitution defined by the languages $L_{i,\bot}$, $L_{i,A}$, $L_{i,\neg A}$, $L_{i,\geq nR}$, $L_{i,\leq nR}$, and let $\sigma(D)$ be of the form shown in (11). Obviously, $\sigma \sqsubseteq \sigma'$ since $M'_{i,\bot} \subseteq L_{i,\bot}$, $M'_{i,A} \subseteq L_{i,A}$, $M'_{i,\neg A} \subseteq L_{i,\neg A}$, $M'_{i,\geq nR} \subseteq L_{i,\geq nR}$, and $M'_{i,\leq nR} \subseteq L_{i,\geq nR}$. Thus, $\sigma(D) \sqsubseteq \sigma'(D) \equiv C$. This implies $E_{\sigma(D)} \supseteq E_C$. To show $E_{\sigma(D)} \subseteq E_C$, we assume that $w \in E_{\sigma(D)}$. According to Theorem 32 we must distinguish two cases:

(1) There is a prefix v of w and a word $v' = R_1 \cdots R_n \in \Sigma^*$ such that vv' is required by $\sigma(D)$ starting from v, i.e., there are at-least restrictions $(\geq m_{i+1}\ R_{i+1})$, $m_{i+1} \geq 1$ such that $vR_1 \cdots R_i \in V'_{\geq m_{i+1} R_{i+1}}$, $0 \leq i < n$. Furthermore, $vv' \in V'_\bot$, or there is an $A \in \mathcal{C}$ with $vv' \in V'_A \cap V'_{\neg A}$, or there are number restrictions $(\geq k\ R)$, $(\leq r\ R)$, $k > r$, with $vv' \in V'_{\geq kR} \cap V'_{\leq rR}$.

Because the respective equations are satisfied, we already know that

$$
\begin{aligned}
V'_{\geq m_{i+1} R_{i+1}} &\subseteq \bigcup_{m \geq m_{i+1}} U_{\geq m R_{i+1}} \cup E_C, \\
V'_A &\subseteq U_A \cup E_C, \\
V'_{\neg A} &\subseteq U_{\neg A} \cup E_C, \\
V'_{\geq kR} &\subseteq \bigcup_{m \geq k} U_{\geq mR} \cup E_C, \text{ and} \\
V'_{\leq rR} &\subseteq \bigcup_{m \leq r} U_{\leq mR} \cup E_C \cdot R^{-1}.
\end{aligned}
$$

Moreover, by definition of $L_{i,\bot}$, the inclusion $V'_\bot \subseteq E_C$ holds. To conclude that $w \in E_C$ we must distinguish two cases:

(a) If there is no proper prefix v'' of v' such that $vv'' \in E_C$, then vv' is required by C starting from v. Furthermore, it holds that $vv' \in E_C$, or $vv' \in (U_A \cup E_C) \cap (U_{\neg A} \cup E_C)$, or $vv' \in (\bigcup_{m \geq k} U_{\geq mR} \cup E_C) \cap (\bigcup_{m \leq r} U_{\leq mR} \cup E_C \cdot R^{-1})$. In all three cases it follows that $vv' \in E_C$. By Lemma 30 this implies $v \in E_C$. Since $\forall v.\bot \sqsubseteq \forall w.\bot$ we know $w \in E_C$.

(b) If v'' is the shortest prefix of v' such that $vv'' \in E_C$, then vv'' is required by C starting from v. Again, by Lemma 30 it follows that $v \in E_C$, and thus $w \in E_C$.

(2) There exists a prefix vR of w (where $v \in \Sigma^*$ and $R \in \Sigma$) such that $v \in V'_{\leq 0R}$. Because the equation $(\leq 0\ R)$ is satisfied, it follows that $V'_{\leq 0R} \subseteq U_{\leq 0R} \cup E_C \cdot R^{-1}$. If $v \in U_{\leq 0R}$, then Theorem 32 yields $w \in E_C$. If $v \in E_C \cdot R^{-1}$, then $C \sqsubseteq \forall v.(\leq 0\ R)$. Obviously, this also implies $w \in E_C$. ∎

By Proposition 33 we know that E_C is of the form $U \cdot \Sigma^*$ for a finite language U of polynomial size, and we have already observed that $E_C \cdot R^{-1} = (U \cup U \cdot R^{-1}) \cdot \Sigma^*$. Consequently, using Proposition 33, we can compute a finite set $V \subseteq \Sigma^*$ in time polynomial in the size of C such that $E_C \cdot R^{-1} = V \cdot \Sigma^*$. Thus, as shown in Lemma 19, the languages $L_{\cdot,\cdot}$ can be computed in time polynomial in the size of the matching problem $C \equiv^? D$. Furthermore, as in the proof of Theorem 18 we can show that inserting these candidate solutions into the equations (A), $(\neg A)$, $(\geq n\, R)$, and $(\leq n\, R)$, and testing whether they solve these equations can be realized in polynomial time. Finally, because $\sigma(D)$ (where σ is defined as in the proof of Lemma 37) can be computed in time polynomial in the size of C and D, this also holds for the language V with $V{\cdot}\Sigma^* = E_{\sigma(D)}$. This shows that the problem $E_C = E_{\sigma(D)}$ is decidable in time polynomial in the size of C and D.

Theorem 38 Solvability of matching problems in $\mathcal{ALN}$ can be decided in polynomial time.

In the proof of Proposition 35 we have shown that, for an arbitrary solution σ'_M of $C \equiv^? D$, there is a solution σ_M that does not introduce new atomic concepts or number restrictions. More precisely, the proof of Proposition 35 shows that new atomic concepts can be replaced by $\perp$, at-least restrictions can be replaced by greater at-least restrictions or by $\perp$, and at-most restrictions can be replaced by smaller at-most restrictions or by $\perp$. Thus, $\sigma_M \sqsubseteq \sigma'_M$. Furthermore, in the proof of Lemma 37 we have verified that σ_M satisfies $\sigma_M \sqsupseteq \sigma_\perp \equiv \sigma' \sqsupseteq \sigma$. Consequently, we can again compute the least solution of the matching problem.

Lemma 39 Assume that the given $\mathcal{ALN}$-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 40 Let $C \sqsubseteq^? D$ be a solvable $\mathcal{ALN}$-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.

We conclude this section with an example that illustrates the matching algorithm for $\mathcal{ALN}$ described above.

Example 41 Let C be an $\mathcal{ALN}$-concept description (describing a class of rather unhappy persons) and D an $\mathcal{ALN}$-concept pattern having the following $\mathcal{FL}_0$-normal forms:

$$\begin{aligned} C &\equiv \forall \mathsf{friends}.\perp \sqcap \forall \mathsf{enemies}.\mathsf{Rich}, \\ D &\equiv \forall\{\mathsf{friends}\}.(\geq 2\,\mathsf{enemies}) \sqcap \\ &\quad\ \forall\{\mathsf{friends\ friends}, \mathsf{enemies}\}.X_1 \sqcap \forall\{\mathsf{friends\ enemies}\}.X_2. \end{aligned}$$

It is easy to see that $E_C = \{\mathsf{friends}\}{\cdot}\Sigma^*$ where $\Sigma = \{\mathsf{friends}, \mathsf{enemies}\}$. Thus, the equations (Rich) and $(\geq 2\,\mathsf{enemies})$ have the following form:

$$\begin{aligned} (\mathsf{Rich}) \qquad \{\mathsf{enemies}\} \cup \{\mathsf{friends}\}{\cdot}\Sigma^* &= \emptyset \cup \{\mathsf{friends}\}{\cdot}\Sigma^* \cup \\ &\quad\ \{\mathsf{friends\ friends}, \mathsf{enemies}\}{\cdot}X_{1,\mathsf{Rich}} \cup \\ &\quad\ \{\mathsf{friends\ enemies}\}{\cdot}X_{2,\mathsf{Rich}}, \\ (\geq 2\,\mathsf{enemies}) \qquad \emptyset \cup \{\mathsf{friends}\}{\cdot}\Sigma^* &= \{\mathsf{friends}\} \cup \{\mathsf{friends}\}{\cdot}\Sigma^* \cup \\ &\quad\ \{\mathsf{friends\ friends}, \mathsf{enemies}\}{\cdot}X_{1,\geq 2\mathsf{enemies}} \cup \\ &\quad\ \{\mathsf{friends\ enemies}\}{\cdot}X_{2,\geq 2\mathsf{enemies}}. \end{aligned}$$

It is easy to see that the approach for finding candidate solutions of these equations described above yields $L_{1,\mathsf{Rich}} = \{\varepsilon\}$, $L_{2,\mathsf{Rich}} = \emptyset$, $L_{1,\geq 2\mathsf{enemies}} = \emptyset$, and $L_{2,\geq 2\mathsf{enemies}} = \emptyset$, which indeed solve ($\mathsf{Rich}$) and ($\geq 2\ \mathsf{enemies}$). In addition, $\widehat{L}_{1,\perp} = \emptyset$ and $\widehat{L}_{2,\perp} = \Sigma^*$, and thus $L_{1,\perp} = \emptyset$ and $L_{2,\perp} = \{\varepsilon\}$.

The substitution σ induced by these finite languages replaces X_1 by Rich and X_2 by $\perp$. It remains to be shown that the equation ($\perp$) is satisfied by this substitution, i.e., that $E_C = E_{\sigma(D)}$ holds. We have

$$\begin{aligned}\sigma(D) \quad \equiv \quad & \forall\{\mathsf{friends}\}.(\geq 2\ \mathsf{enemies}) \sqcap \\ & \forall\{\mathsf{friends\ friends}, \mathsf{enemies}\}.\mathsf{Rich} \sqcap \forall\{\mathsf{friends\ enemies}\}.\perp.\end{aligned}$$

Because the word $\mathsf{friends\ enemies}$ is required by $\sigma(D)$ starting from $\mathsf{friends}$ and $\sigma(D)$ contains the value restriction $\forall\{\mathsf{friends\ enemies}\}.\perp$, we know that $\mathsf{friends}$ is an element of $E_{\sigma(D)}$ by Theorem 32. Consequently, every word that starts with the letter $\mathsf{friends}$ is in $E_{\sigma(D)}$. In addition, it is easy to see that $\mathsf{enemies}$ and the empty word ε do not belong to $E_{\sigma(D)}$. To sum up, we have $E_{\sigma(D)} = \{\mathsf{friends}\}{\cdot}\Sigma^* = E_C$.

This shows that σ is indeed a solution of the matching problem. It should be noted that the matching algorithm introduced by Borgida and McGuinness [10] does not find this solution.

5 Matching under side conditions

In the following, we will show that strict subsumption conditions increase the computational complexity of matching. Non-strict subsumption conditions can often be eliminated, but it is not yet clear whether this elimination leads to an increase in the complexity of the problem. For strict subsumption conditions we obtain a polynomiality result if the right-hand sides of the conditions are restricted to concept descriptions rather than patterns.

5.1 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 42 Let $C \equiv^? D$ be an $\mathcal{ALN}$-matching problem, and $X_1 \sqsubset^? E_1, \ldots, X_n \sqsubset^? E_n$ strict subsumption conditions such that $E_1, \ldots, E_n$ are $\mathcal{ALN}$-concept descriptions. Then solvability of $C \equiv^? D$ under these conditions is decidable in polynomial time. The same holds for the smaller languages $\mathcal{FL}_0$, $\mathcal{FL}_\perp$, and $\mathcal{FL}_\neg$.

If the right-hand sides of strict subsumption conditions may contain variables, then solvability becomes NP-hard, even for the language $\mathcal{FL}_0$. It should be noted that this does not automatically imply NP-hardness for the larger languages $\mathcal{FL}_\perp$, $\mathcal{FL}_\neg$, and $\mathcal{ALN}$, though we strongly conjecture that the hardness result also holds for them.

The hardness result for $\mathcal{FL}_0$ will be shown by reducing 3SAT [20] to the matching problem under strict subsumption conditions. Recall that matching modulo subsumption can be reduced to matching modulo equivalence, and that a system of matching problems can be coded into a single matching problem. For this reason, we may, without loss of generality, construct a problem that consists of matching problems modulo subsumption, matching problems modulo equivalence, and strict subsumption conditions.

Theorem 43 Matching under strict subsumption conditions is NP-hard, even for the small language $\mathcal{FL}_0$.

Proof. Let A be an arbitrary concept name. For every propositional variable p occurring in the 3SAT problem, we introduce three concept variables, namely X_p, $X_{\overline{p}}$, and Z_p, and two roles R_p and $R_{\overline{p}}$. Using these concept variables and roles, we construct the matching statement

$$\forall R_p.A \sqcap \forall R_{\overline{p}}.A \quad \sqsubseteq \quad Z_p, \tag{12}$$

and the strict subsumption condition

$$Z_p \quad \sqsubset \quad \forall R_p.X_p \sqcap \forall R_{\overline{p}}.X_{\overline{p}}. \tag{13}$$

It is easy to see that the subsumption relationship between $\forall R_p.A \sqcap \forall R_{\overline{p}}.A$ and $\forall R_p.X_p \sqcap \forall R_{\overline{p}}.X_{\overline{p}}$ enforced by (12) and (13) implies that any solution θ of (12) and (13) satisfies:

$$(\theta(X_p) \equiv A \vee \theta(X_p) \equiv \top) \wedge (\theta(X_{\overline{p}}) \equiv A \vee \theta(X_{\overline{p}}) \equiv \top).$$

In addition, the fact that this subsumption relationship must be strict implies

$$\theta(X_p) \equiv \top \vee \theta(X_{\overline{p}}) \equiv \top.$$

Finally, if this solution also satisfies the matching statement

$$A \quad \equiv \quad X_p \sqcap X_{\overline{p}}, \tag{14}$$

then we know that not both variables can be replaced by $\top$, i.e.,

$$\theta(X_p) \equiv A \vee \theta(X_{\overline{p}}) \equiv A.$$

This shows that, if we take $\top$ as the truth value 1 and A as the truth value 0, then any solution assigns either 0 or 1 to X_p, and the opposite truth value to $X_{\overline{p}}$.

It remains to be shown that 3-clauses and the corresponding truth conditions can be encoded. We introduce a concept variable Z_c and three roles $R_{c,1}, R_{c,2}, R_{c,3}$ for every 3-clause c in our 3SAT problem, and represent the clause by a matching problem together with a strict subsumption condition. For example, assume that $c := p \vee \neg q \vee r$. Then c is represented by the matching statement

$$\forall R_{c,1}.A \sqcap \forall R_{c,2}.A \sqcap \forall R_{c,3}.A \quad \sqsubseteq \quad Z_c,$$

and the strict subsumption condition

$$Z_c \quad \sqsubset \quad \forall R_{c,1}.X_p \sqcap \forall R_{c,2}.X_{\overline{q}} \sqcap \forall R_{c,3}.X_r.$$

Obviously, the strict inclusion implied by these two statements can only be satisfied by a substitution θ if it assigns $\top$ to at least one of the variables X_p, $X_{\overline{q}}$, and X_r. This completes our reduction. ∎

Note that (14) is a matching problem modulo equivalence that cannot be represented by a matching problem modulo subsumption. Thus, it is still open whether the NP-hardness result also holds for matching modulo subsumption under strict subsumption conditions.

Theorem 43 only provides a hardness result for matching under strict subsumption conditions. Thus, another open question is how to extend the matching algorithm for $\mathcal{FL}_0$ (or one of the larger languages considered in this paper) to an algorithm that can also handle strict subsumption conditions.

5.2 Subsumption conditions

Recall that a subsumption condition is of the form $X \sqsubseteq^? E$ where X is a concept variable and E is a concept pattern. 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.

Definition 44 The sequence of subsumption conditions $X_1 \sqsubseteq^? E_1, \ldots, X_n \sqsubseteq^? E_n$ is *acyclic* if and only if for all i, $1 \leq i \leq n$, the pattern E_i does not contain the variables $X_i, \ldots, X_n$.

A set of subsumption conditions is called *acyclic* if and only if the subsumption conditions can be arranged in an acyclic sequence.

Note that we may (without loss of generality) assume that $X_1, \ldots, X_n$ are all the variables occurring in the patterns $D, E_1, \ldots, E_n$ since, for an additional variable Z occurring in one of the patterns, we can simply add the subsumption condition $Z \sqsubseteq^? \top$ to the beginning of the sequence.

Given such an acyclic sequence of subsumption conditions, we can define a substitution[12] σ inductively as follows:

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

where the Y_i are new variables.

Proposition 45 The matching problem $C \equiv^? D$ is solvable under the acyclic subsumption conditions $X_1 \sqsubseteq^? E_1, \ldots, X_n \sqsubseteq^? E_n$ if and only if $C \equiv^? \sigma(D)$ is solvable without subsumption conditions.

Proof. To show the *if* direction, we assume that τ' is a solution of the matching problem $C \equiv^? \sigma(D)$. We construct a new substitution τ by induction on i:

$$\tau(X_1) := \tau'(Y_1) \sqcap E_1 \quad \text{and} \quad \tau(X_i) := \tau'(Y_i) \sqcap \tau(E_i) \quad (1 < i \leq n).$$

[12] 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.

Since the sequence of subsumption conditions $X_1 \sqsubseteq^? E_1, \ldots, X_n \sqsubseteq^? E_n$ is acyclic, E_1 does not contain variables, and E_i may only contain the variables $X_1, \ldots, X_{i-1}$. Thus, we have $E_1 = \tau(E_1)$ and $\tau(E_i)$ is well-defined by induction. It remains to be shown that τ solves $C \equiv^? D$ under the subsumption conditions $X_1 \sqsubseteq^? E_1, \ldots, X_n \sqsubseteq^? E_n$.

Since $\tau(X_i) = \tau'(Y_i) \sqcap \tau(E_i) \sqsubseteq \tau(E_i)$, the subsumption conditions are satisfied by definition of τ and the fact that $E_1 = \tau(E_1)$.

By induction on i, it is easy to show that $\tau(X_i) = \tau'(\sigma(X_i))$ holds for all $i, 1 \leq i \leq n$. Since we have assumed that the patterns do not contain variables other than $X_1, \ldots, X_n$, this implies $\tau(D) = \tau'(\sigma(D))$. Finally, $\tau'(\sigma(D)) \equiv C$ since τ' solves $C \equiv^? \sigma(D)$.

To show the *only-if* direction, we assume that τ is a solution of the matching problem $C \equiv^? D$ that satisfies the subsumption conditions $X_1 \sqsubseteq^? E_1, \ldots, X_n \sqsubseteq^? E_n$. The new substitution τ' is defined by $\tau'(Y_i) := \tau(X_i)$. First, we show (by induction on i) that $\tau(X_i) \equiv \tau'(\sigma(X_i))$ holds for all $i, 1 \leq i \leq n$.

For $i = 1$ we have

$$\tau'(\sigma(X_1)) = \tau'(Y_1 \sqcap E_1) = \tau'(Y_1) \sqcap \tau'(E_1) = \tau(X_1) \sqcap E_1 = \tau(X_1) \sqcap \tau(E_1) \equiv \tau(X_1).$$

The last equivalence holds since $\tau(X_1) \sqsubseteq \tau(E_1)$ by the assumption that τ satisfies the subsumption conditions. In the induction step, we have

$$\tau'(\sigma(X_i)) = \tau'(Y_i \sqcap \sigma(E_i)) = \tau'(Y_i) \sqcap \tau'(\sigma(E_i)) \equiv \tau(X_i) \sqcap \tau(E_i) \equiv \tau(X_i).$$

Thus, $\tau'(\sigma(D)) \equiv \tau(D) \equiv C$, which shows that τ' solves the matching problem $C \equiv^? \sigma(D)$. ∎

Unfortunately, the new pattern $\sigma(D)$ may be exponentially larger than the original matching problem with subsumption conditions.

Example 46 Let R, S be distinct atomic roles and A an atomic concept. We consider the acyclic subsumption conditions

$$X_1 \sqsubseteq^? A, \quad X_2 \sqsubseteq^? \forall R.X_1 \sqcap \forall S.X_1, \quad \ldots, \quad X_n \sqsubseteq^? \forall R.X_{n-1} \sqcap \forall S.X_{n-1}.$$

Let the substitution σ be defined as described above. It is easy to see that the size of $\sigma(X_n)$ is exponential in n. In fact, the $\mathcal{FL}_0$-normal form of $\sigma(X_n)$ is

$$\sigma(X_n) \equiv Y_n \sqcap \forall L_1.Y_{n-1} \sqcap \forall L_2.Y_{n-2} \sqcap \ldots \sqcap \forall L_{n-1}.Y_1 \sqcap \forall L_{n-1}.A,$$

where L_i denotes the set of all words of length i over the alphabet $\Sigma := \{R, S\}$. The set L_{n-1} alone already contains 2^{n-1} different words.

This example also suggests the use of a compact representation of (the $\mathcal{FL}_0$-normal form of) the pattern $\sigma(D)$. In the example, we can represent L_i as the i-fold concatenation of L_1. This yields a polynomial representation of the exponentially large languages L_i. It is easy to see that such a compact representation of $\sigma(D)$ is always possible. However, it is not yet clear whether the computations required by our solvability test for matching problems are still polynomial in the size of this compact representation, though we strongly conjecture that this is the case.

The reduction described in Lemma 45 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.

Proposition 47 For every $\mathcal{FL}_0$-matching problem with subsumption conditions there exists an equivalent[13] $\mathcal{FL}_0$-matching problem with acyclic subsumption conditions whose size is polynomial in the size of the original problem.

PROOF. Let $C \equiv^? D$ be an $\mathcal{FL}_0$-matching problem and $\Gamma := \{X_1 \sqsubseteq^? E_1, \ldots, X_n \sqsubseteq^? E_n\}$ a set of $\mathcal{FL}_0$-subsumption conditions. The set Γ defines a dependency graph G_Γ, whose nodes are the variables $X_1, \ldots, X_n$, and whose edges are defined as follows: there is an edge from X_i to X_j with label W if and only if E_i contains a value restriction of the form $\forall W.X_j$ (where W is a word over the set of role names). For example, the condition $X_2 \sqsubseteq \forall R.\forall S.X_1 \sqcap X_3$ induces an edge with label RS from X_2 to X_1, and an edge with label ε from X_2 to X_3. Obviously, the set Γ is acyclic if and only if the graph G_Γ is acyclic. We distinguish between two different types of cycles in G_Γ, and show how they can be eliminated.

First, assume that there is a path from X_i to X_i whose label (i.e., the concatenation of the labels of its edges) is a nonempty word W. Then any substitution σ satisfying Γ also satisfies the subsumption relation $\sigma(X_i) \sqsubseteq \forall W.\sigma(X_i)$. Since W is nonempty and $\sigma(X_i)$ must be an $\mathcal{FL}_0$-concept description, this is only possible if $\sigma(X_i) = \top$. Let τ be the substitution that replaces X_i by $\top$. We eliminate X_i by applying τ both to Γ and to the matching problem $C \equiv^? D$. It should be noted that this transforms the subsumption condition $X_i \sqsubseteq E_i$ into the matching problem $\top \sqsubseteq \tau(E_i)$, which is equivalent to $\top \equiv^? \tau(E_i)$. However, as shown in Lemma 5, the two matching problems $C \equiv^? \tau(D)$ and $\top \equiv^? \tau(E_i)$ can be transformed into a single matching problem.

Second, assume that there is a path from X_i to X_i with label ε. If this path has length 1, then E_i is of the form $X_i \sqcap E_i'$. Since a substitution satisfies $X_i \sqsubseteq X_i \sqcap E_i'$ if and only if it satisfies $X_i \sqsubseteq E_i'$, such a cycle can easily be eliminated. Finally, if the cyclic path involves also another variable, say X_j, then any substitution σ satisfying Γ also satisfies $\sigma(X_i) \equiv \sigma(X_j)$, and thus we can eliminate X_i by replacing it by X_j. ∎

6 Future Work

Our goal is to extend the results on matching to cover languages at least as expressive as CLASSIC. This requires extending the language to include range constructors (min and max), an individual set constructor (one-of), and a fills constructor. We believe that this should be an easy extension of the results presented in this paper. In fact, min, max, and one-of mainly require an appropriate treatment of disjointness, which we have already achieved by our treatment of atomic negation. The fills construct is similar to number restrictions in that it states the existence of a certain role successor.

The work on strict and non-strict subsumption conditions will be continued. One way of showing decidability of matching under strict subsumption conditions could be to extend the results on unification of concept terms [6] to disunification, i.e., problems that may contain both equations and negated equations [15]. For non-strict subsumption conditions we will try to show that a compact representation of the pattern $\sigma(D)$ can be used to obtain a polynomiality result.

Another motivation for investigating matching modulo equivalence may be found in merging heterogeneous databases. Consider a situation where there is a master

[13] Equivalent means that the problems have the same set of solutions.

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 languages $\mathcal{FL}_\bot$, $\mathcal{FL}_\neg$, and $\mathcal{ALN}$. We presented results concerning the solvability of the problem including polynomial decidability and (for solvable problems) polynomial computability of a least solution.

We have mentioned in the introduction that positive results for matching (such as decidability in polynomial time) do not automatically transfer from a given language to its sublanguages since a matching problem of the smaller language that does not have a solution in this language may well have one in the larger language. For example, in the sublanguage of $\mathcal{FL}_0$ that does not allow for the top concept $\top$, the matching problem $A \equiv^? A \sqcap \forall R.X$ obviously does not have a solution, whereas it is solvable by $\{X \mapsto \top\}$ in $\mathcal{FL}_0$. As an easy consequence of the results presented in this paper, one can show, however, that this phenomenon cannot occur between the languages $\mathcal{FL}_0$, $\mathcal{FL}_\bot$, $\mathcal{FL}_\neg$, and $\mathcal{ALN}$ (see, in particular, the proof of Proposition 35).

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 subsumption conditions is NP-hard for the small language $\mathcal{FL}_0$. It should be noted that the phenomenon mentioned above does occur between $\mathcal{FL}_0$ and $\mathcal{FL}_\bot$ if subsumption conditions are allowed. For example, the set of subsumption conditions $\{X \sqsubseteq^? \forall R.X,\ X \sqsubseteq^? A\}$ is not satisfiable in $\mathcal{FL}_0$, but it can be satisfied by the $\mathcal{FL}_\bot$-substitution $\{X \mapsto \bot\}$. Thus, the NP-hardness result for matching modulo equivalence with strict subsumption conditions in $\mathcal{FL}_0$ does not imply hardness of this problem for the larger languages $\mathcal{FL}_\bot$, $\mathcal{FL}_\neg$, and $\mathcal{ALN}$, though we strongly conjecture that the hardness result also holds for them.

References

[1] F. Baader. Augmenting concept languages by transitive closure of roles: An alternative to terminological cycles. In *Proceedings of the 12th International Joint Conference on Artificial Intelligence*, pages 446–451, Sydney, Australia, 1991.

[2] F. Baader, M. Buchheit, and B. Hollunder. Cardinality restrictions on concepts. *Artificial Intelligence*, 88(1–2):195–213, 1996.

[3] F. Baader and P. Hanschke. A scheme for integrating concrete domains into concept languages. In *Proceedings of the 12th International Joint Conference on Artificial Intelligence*, pages 452–457, Sydney, Australia, 1991.

[4] F. Baader and B. Hollunder. A terminological knowledge representation system with complete inference algorithms. In M. Richter and H. Boley, editors, *Proceedings of the First International Workshop on Processing Declarative Knowledge*, volume 567 of *Lecture Notes in Computer Science*, pages 67–85, Kaiserslautern (Germany), 1991. Springer-Verlag.

[5] F. Baader, R. Küsters, and R. Molitor. Structural subsumption from an automata theoretic point of view. In *Proceedings of the 1998 International Workshop on Description Logics (DL'98)*, Trento, Italy, 1998. An extended version has appeared as Technical Report LTCS-98-04, LuFg Theoretical Computer Science, RWTH Aachen, Germany (see http://www-lti.informatik.rwth-aachen.de/Forschung/Papers.html).

[6] 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.

[7] F. Baader and U. Sattler. Description logics with symbolic number restrictions. In W. Wahlster, editor, *Proceedings of the Twelfth European Conference on Artificial Intelligence (ECAI-96)*, pages 283–287, Budapest, Hungary, 1996. John Wiley & Sons Ltd.

[8] F. Baader and U. Sattler. Number restrictions on complex roles in description logics. In *Proceedings of the Fifth International Conference on the Principles of Knowledge Representation and Reasoning (KR-96)*, Cambridge, MA (USA), 1996. Morgan Kaufmann, Los Altos.

[9] A. Borgida. Description Logics in Data Management. *IEEE Transactions on Knowledge and Data Engineering*, 7(5): 671–682, 1995.

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

[11] A. Borgida and P. Patel-Schneider. A semantics and complete algorithm for subsumption in the CLASSIC description logic. *Journal of Artificial Intelligence Research*, 1:277–308, 1994.

[12] 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.

[13] R. J. Brachman and J. G. Schmolze. An overview of the KL-ONE knowledge representation system. *Cognitive Science*, 9(2):171–216, 1985.

[14] M. Buchheit, F. M. Donini, and A. Schaerf. Decidable reasoning in terminological knowledge representation systems. *Journal of Artificial Intelligence Research*, 1:109–138, 1993.

[15] H. Comon. Disunification: A survey. In J.-L. Lassez and G. Plotkin, editors, *Computational Logic: Essays in Honor of Alan Robinson*, pages 322–359. MIT Press, Cambridge, MA, 1991.

[16] F. Donini, M. Lenzerini, D. Nardi, and W. Nutt. The complexity of concept languages. In *Proceedings of the 2nd International Conference on Principles of Knowledge Representation and Reasoning*, pages 151–162, Cambridge MA (USA), 1991.

[17] F. Donini, M. Lenzerini, D. Nardi, and W. Nutt. Tractable concept languages. In *Proceedings of the 12th International Joint Conference on Artificial Intelligence*, pages 458–463, Sydney, Australia, 1991.

[18] F. Donini, M. Lenzerini, D. Nardi, and A. Schaerf. Deduction in concept languages: From subsumption to instance checking. *Journal of Logic and Computation*, 4(4):423–452, 1994.

[19] F. Donini, B. Hollunder, M. Lenzerini, A.M. Spaccamela, D. Nardi, and W. Nutt. The complexity of existential quantification in concept languages. *Journal of Artificial Intelligence*, 53:309–327, 1992.

[20] 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.

[21] B. Hollunder and F. Baader. Qualifying number restrictions in concept languages. In *Proceedings of the 2nd International Conference on Principles of Knowledge Representation and Reasoning*, pages 335–346, Cambridge MA (USA), 1991.

[22] B. Hollunder, W. Nutt, and M. Schmidt-Schauß. Subsumption algorithms for concept description languages. In *Proceedings of the 9th European Conference on Artificial Intelligence*, pages 348–353, Stockholm, Sweden, 1990.

[23] 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)*, pages 499–510, Trento, Italy, 1998.

[24] H. J. Levesque and R. J. Brachman. Expressiveness and tractability in knowledge representation and reasoning. *Computational Intelligence*, 3:78–93, 1987.

[25] 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.

[26] 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.

[27] D. L. McGuinness and P. F. Patel-Schneider. Usability issues in Description Logic systems. In *Proceedings of the 15th National Conference on Artificial Intelligence, AAAI'98*, Madison, Wisconsin, 1998.

[28] 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.

[29] D. L. McGuinness, C. Isbell, M. Parker, P. F. Patel-Schneider, L. Alperin Resnick, and C. Welty. A Description Logic-based configurator on the Web. *ACM Sigart Bulletin*, 9(2):20–22, 1998.

[30] D. L. McGuinness and J. R. Wright. An industrial strength Description Logic-based configurator platform. *IEEE Intelligent Systems*, 13(4): 66–77, 1998.

[31] D. L. McGuinness and J. R. Wright. Conceptual modeling for configuration: A description logic-based approach. *Artificial Intelligence for Engineering Design, Analysis, and Manufacturing Journal*, 12:333–344, 1998.

[32] B. Nebel. Computational complexity of terminological reasoning in BACK. *Journal of Artificial Intelligence*, 34(3):371–383, 1988.

[33] B. Nebel. *Reasoning and Revision in Hybrid Representation Systems*, volume 422 of *Lecture Notes in Computer Science*. Springer–Verlag, 1990.

[34] B. Nebel. Terminological reasoning is inherently intractable. *Journal of Artificial Intelligence*, 43(2):235–249, 1990.

[35] A. Schaerf. On the complexity of the instance checking problem in concept languages with existential quantification. *Journal of Intelligent Information Systems*, 2:265–278, 1993.

[36] M. Schmidt-Schauß and G. Smolka. Attributive concept descriptions with complements. *Journal of Artificial Intelligence*, 47:1–26, 1991.

[37] W. A. Woods and J. G. Schmolze. The KL-ONE family. *Computers and Mathematics with Applications, special issue on knowledge representation*, 23(2–5):133–177, 1991.

[38] 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.