EPiC Series in Computing
Volume 57, 2018, Pages 76–94
LPAR-22. 22nd International Conference on Logic for Programming, Artificial Intelligence and Reasoning

Matching in the Description Logic $\mathcal{FL}_0$ with respect to General TBoxes*

Franz Baader, Oliver Fernández Gil, and Pavlos Marantidis

Theoretical Computer Science, TU Dresden
01062 Dresden, Germany
firstname.lastname@tu-dresden.de

Abstract

Matching concept descriptions against concept patterns was introduced as a new inference task in Description Logics two decades ago, motivated by applications in the Classic system. Shortly afterwards, a polynomial-time matching algorithm was developed for the DL $\mathcal{FL}_0$. However, this algorithm cannot deal with general TBoxes (i.e., finite sets of general concept inclusions). Here we show that matching in $\mathcal{FL}_0$ w.r.t. general TBoxes is in ExpTime, which is the best possible complexity for this problem since already subsumption w.r.t. general TBoxes is ExpTime-hard in $\mathcal{FL}_0$. We also show that, w.r.t. a restricted form of TBoxes, the complexity of matching in $\mathcal{FL}_0$ can be lowered to PSpace.

1 Introduction

Matching is the special case of unification where one of the expressions to be unified has no variables and thus remains unchanged under substitutions. In Description Logic (DL), matching of concept descriptions against concept patterns was originally introduced in [11] as a non-standard inference task that can be used to filter out the unimportant aspects of large concept descriptions appearing in knowledge bases of the system Classic [9]. Subsequently, matching (as well as the more general problem of unification) was also proposed as a tool for detecting redundancies in knowledge bases [8] and to support the integration of knowledge bases by prompting interschema assertions to the integrator [10].

All three applications have in common that one wants to search the knowledge base for concepts having a certain (not completely specified) form. This "form" can be expressed with the help of so-called *concept patterns*, i.e., concept descriptions containing variables (which stand for concept descriptions). For example, assume that we want to find concepts that are concerned with humans that share some characteristic with all their children. This can be expressed by the pattern $D := \mathsf{Human} \sqcap X \sqcap \forall \mathsf{has\text{-}child}.X$ where X is a variable standing for the common characteristic. The concept description $C := \mathsf{Human} \sqcap \mathsf{Tall} \sqcap \forall \mathsf{has\text{-}child}.\mathsf{Tall}$ matches this pattern in the sense that, if we replace the variable X by the concept description Tall, the pattern

*Supported by the DFG Research Training Group 1763 (QuantLA) and the DFG grant BA 1122/20-1.

becomes *equivalent* to the concept description C. Thus, the substitution $\sigma := \{X \mapsto \mathsf{Tall}\}$ is a *matcher* of the matching problem $C \equiv^? D$ since $C \equiv \sigma(D)$.

Both matching and unification have been investigated in detail for the inexpressive DLs $\mathcal{FL}_0$ (with concept constructors top $\top$, conjunction $C \sqcap D$, value restriction $\forall r.C$) and $\mathcal{EL}$ (with concept constructors $\top$, $C \sqcap D$, existential restriction $\exists r.C$). Whereas in $\mathcal{EL}$ both matching [4] and unification [6] are NP-complete problems, the complexity of these problems differs significantly for $\mathcal{FL}_0$: matching is polynomial, but unification is ExpTime-complete [8]. These results were shown for the case without a background TBox, i.e., where equivalence $\equiv$ between concept descriptions must be achieved, rather than equivalence $\equiv_\mathcal{T}$ w.r.t. a TBox $\mathcal{T}$ consisting of general concept inclusions (GCI). For the DL $\mathcal{EL}$ it was proved in [7] that the presence of TBoxes does not change the complexity of the matching problem: it stays in NP. For unification in $\mathcal{EL}$ w.r.t. TBoxes an NP upper bound could until now only be shown for a restricted form of TBoxes [1].

Matching in $\mathcal{FL}_0$ in the presence of TBoxes has not been investigated until now. In this paper, we close this gap by showing that it is an ExpTime-complete problem. Since already subsumption in $\mathcal{FL}_0$ w.r.t. TBoxes is ExpTime-complete [2], ExpTime-hardness of this problem is clear. The first main contribution of this paper is thus to show the ExpTime upper bound. We do this by first showing an ExpTime upper bound for the problem of testing whether an $\mathcal{FL}_0$ matching problem has a matcher in the extended logic $\mathcal{FL}_{reg}$. Basically, in $\mathcal{FL}_{reg}$ one can use regular languages to express infinite conjunctions of value restrictions. Our proof of the ExpTime upper bound depends on a fine-grained analysis of the complexity of subsumption of $\mathcal{FL}_{reg}$ concept descriptions w.r.t. an $\mathcal{FL}_0$ TBox, which uses automata on words and trees. The second step is then to show that an $\mathcal{FL}_0$ matching problem has an $\mathcal{FL}_0$ matcher iff it has an $\mathcal{FL}_{reg}$ matcher. The second main contribution of this paper is to show that the complexity of the matching problem can be lowered from ExpTime to PSpace if one considers TBoxes of a restricted form where the role depth on the left-hand side of a GCI is not larger than the role depth on the right-hand side.

2 Preliminaries

We start by introducing the DLs $\mathcal{FL}_0$ and $\mathcal{FL}_{reg}$, and then we will consider the matching problem. While we are mainly interested in matching in $\mathcal{FL}_0$, we will use $\mathcal{FL}_{reg}$ matchers as intermediate solutions.

The DLs $\mathcal{FL}_0$ and $\mathcal{FL}_{reg}$

Concept descriptions of the DL $\mathcal{FL}_0$ are built from concept and role names using the concept constructors *conjunction* ($\sqcap$), *value restriction* ($\forall r.C$) and the *top concept* ($\top$). To be more precise, $\mathcal{FL}_0$ concept descriptions are obtained from disjoint sets $\mathsf{N_C}$ and $\mathsf{N_R}$ of concept and role names, respectively, using the following syntax rules:

$$C ::= \top \mid A \mid C \sqcap C \mid \forall r.C,$$

where $A \in \mathsf{N_C}$, $r \in \mathsf{N_R}$, and C is an $\mathcal{FL}_0$ concept description. An $\mathcal{FL}_0$ *TBox* is a finite set of *general concept inclusions (GCIs)*, which are expressions of the form $C \sqsubseteq D$ where C, D are $\mathcal{FL}_0$ concept descriptions.

The DL $\mathcal{FL}_{reg}$ extends $\mathcal{FL}_0$ by allowing the use of regular languages L over the alphabet of all role names $\mathsf{N_R}$ to express infinite conjunctions of value restrictions. Basically, the value restriction $\forall L.C$ stands for the (possibly infinite) conjunction $\bigsqcap_{w \in L} \forall w.C$, where (for $w =$

$r_1 \dots r_k \in \mathsf{N_R}^*$) the expression $\forall w.C$ is an abbreviation for $\forall r_1.\cdots\forall r_k.C$. To be more precise, $\mathcal{FL}_{reg}$ *concept descriptions* are obtained from disjoint sets $\mathsf{N_C}$ and $\mathsf{N_R}$ of concept and role names, respectively, using the following syntax rules:

$$C ::= \top \mid A \mid C \sqcap C \mid \forall L.C,$$

where $A \in \mathsf{N_C}$, L is a regular language over $\mathsf{N_R}$, and C is an $\mathcal{FL}_{reg}$ concept description. Here we assume that the regular language L is given by a regular expression or a non-deterministic finite automaton (NFA). Since the singleton language $\{r\}$ for $r \in \mathsf{N_R}$ is regular, every $\mathcal{FL}_0$ concept description is also an $\mathcal{FL}_{reg}$ concept description.

The semantics of $\mathcal{FL}_0$ and $\mathcal{FL}_{reg}$ is defined using interpretations as in first-order logic. An interpretation $\mathcal{I} = (\Delta^\mathcal{I}, \cdot^\mathcal{I})$ consists of a non-empty domain $\Delta^\mathcal{I}$ and an interpretation function $\cdot^\mathcal{I}$, which assigns subsets $A^\mathcal{I} \subseteq \Delta^\mathcal{I}$ to concept names $A \in \mathsf{N_C}$ and binary relations $r^\mathcal{I} \subseteq \Delta^\mathcal{I} \times \Delta^\mathcal{I}$ to role names $r \in \mathsf{N_R}$. The function $\cdot^\mathcal{I}$ is inductively extended to arbitrary $\mathcal{FL}_0$ and $\mathcal{FL}_{reg}$ concept descriptions as follows:

$$\begin{aligned}
&\top^\mathcal{I} := \Delta^\mathcal{I}, \quad (C \sqcap D)^\mathcal{I} := C^\mathcal{I} \cap D^\mathcal{I}, \quad (\forall r.C)^\mathcal{I} := \{x \in \Delta^\mathcal{I} \mid \forall y.((x,y) \in r^\mathcal{I} \implies y \in C^\mathcal{I})\},\\
&(\forall \varepsilon.C)^\mathcal{I} := C^\mathcal{I}, \quad (\forall w.C)^\mathcal{I} := (\forall r_1.\cdots\forall r_k.C)^\mathcal{I} \text{ where } w = r_1 \dots r_k \in \mathsf{N_R}^+,\\
&(\forall L.C)^\mathcal{I} := \textstyle\bigcap_{w \in L} (\forall w.C)^\mathcal{I} \text{ and in particular } (\forall \emptyset.C)^\mathcal{I} = \Delta^\mathcal{I}.
\end{aligned}$$

The interpretation $\mathcal{I}$ is a *model* of the $\mathcal{FL}_0$ TBox $\mathcal{T}$ if $C^\mathcal{I} \subseteq D^\mathcal{I}$ for all $C \sqsubseteq D \in \mathcal{T}$. Given an $\mathcal{FL}_0$ TBox $\mathcal{T}$ and two $\mathcal{FL}_{reg}$ concept descriptions C, D, we say that C *is subsumed by* D w.r.t. $\mathcal{T}$ (denoted as $C \sqsubseteq_\mathcal{T} D$) if $C^\mathcal{I} \subseteq D^\mathcal{I}$ for all models $\mathcal{I}$ of $\mathcal{T}$. These two concept descriptions are *equivalent* w.r.t. $\mathcal{T}$ (written $C \equiv_\mathcal{T} D$) if $C \sqsubseteq_\mathcal{T} D$ and $D \sqsubseteq_\mathcal{T} C$. If $\mathcal{T}$ is the empty TBox, we simply write $C \sqsubseteq D$ and $C \equiv D$ instead of $C \sqsubseteq_\emptyset D$ and $C \equiv_\emptyset D$.

Subsumption between $\mathcal{FL}_{reg}$ concept descriptions w.r.t. an $\mathcal{FL}_0$ TBox is an ExpTime-complete problem. The ExpTime upper bound follows from the known ExpTime upper bound for propositional dynamic logic (PDL) [15, 12] since $\mathcal{FL}_{reg}$ concept descriptions can be translated into PDL and GCIs can be internalized [16]. The corresponding ExpTime lower bound already holds for subsumption between $\mathcal{FL}_0$ concept descriptions w.r.t. an $\mathcal{FL}_0$ TBox [2].

In case the TBox is empty, subsumption between $\mathcal{FL}_0$ concept descriptions is polynomial [13] and between $\mathcal{FL}_{reg}$ concept descriptions it is PSpace-complete [5]. The upper bounds can, for example, be shown using an appropriate normal form. For $\mathcal{FL}_{reg}$ concept descriptions, this normal form can be obtained by applying the following equivalences as rewrite rules from left to right:

$$\begin{aligned}
&\forall L.(E \sqcap F) \equiv \forall L.E \sqcap \forall L.F, \quad \forall L.A \sqcap \forall L'.A \equiv \forall (L \cup L').A,\\
&\forall L_1.\forall L_2.\dots\forall L_n.E \equiv \forall (L_1 \cdot L_2 \cdot \dots \cdot L_n).E,
\end{aligned}$$

where E, F are $\mathcal{FL}_{reg}$ concept descriptions, $A \in \mathsf{N_C}$, and $L, L_1, \dots, L_n$ are regular languages given as NFAs or regular expressions. In the case of NFAs, one needs to use their closure under union and concatenation to obtain the new automata.

Using these rules, given $\mathcal{FL}_{reg}$ concept descriptions C, D can be transformed in polynomial time into the following *normal form*:

$$C \equiv \forall K_1.A_1 \sqcap \dots \sqcap \forall K_k.A_k, \quad D \equiv \forall L_1.A_1 \sqcap \dots \sqcap \forall L_k.A_k, \tag{1}$$

where $A_1, \dots, A_k$ are the concept names occurring in C and D, and $K_1, \dots, K_k, L_1, \dots, L_k$ are regular languages (given as regular expressions or NFAs). In case the concept name A_i occurs in C, but not in D, then $L_i = \emptyset$, and thus $\forall L_i.A_i \equiv \top$. Concept names occurring in D, but not in C, are treated analogously.

Using these normal forms, subsumption between the $\mathcal{FL}_{reg}$ concept descriptions C, D can now be characterized as follows [5]:

$$C \sqsubseteq D \text{ iff } L_i \subseteq K_i \text{ holds for all } i, 1 \leq i \leq k.$$

Since the inclusion problem for regular languages (given as regular expressions or NFAs) is PSpace-complete, this yields both the PSpace upper and the PSpace lower bound for subsumption of $\mathcal{FL}_{reg}$ concept descriptions. In the special case of $\mathcal{FL}_0$ concept descriptions, the languages are finite and have a very simple representation (basically, the enumeration of the words contained in them), which allows for checking inclusion in polynomial time [8].

Later on, we will need to consider such normal forms, but where the regular languages are given as *deterministic* finite automata (DFAs). We call this normal form then *deterministic normal form (DNF)*. A DNF of an $\mathcal{FL}_{reg}$ concept description can obviously be obtained from its normal form by constructing DFAs from the regular expressions or NFAs, which may however result in DFAs that are exponentially larger than the original regular expressions or NFAs. In general, this normal form is not unique since different DFAs may accept the same language. One can make it unique (up to isomorphism of automata) by using minimal DFAs.

For the case of subsumption between $\mathcal{FL}_0$ concept descriptions w.r.t. a non-empty $\mathcal{FL}_0$ TBox, a language-based characterization was developed that is similar to the one given above [3]. Here, we extend it to $\mathcal{FL}_{reg}$ concept descriptions. Given an $\mathcal{FL}_{reg}$ concept description C and an $\mathcal{FL}_0$ TBox $\mathcal{T}$, the *value restriction set* of C w.r.t. $\mathcal{T}$ is defined as:

$$\mathcal{L}_\mathcal{T}(C) := \{(w, A) \in \mathsf{N_R}^* \times \mathsf{N_C} \mid C \sqsubseteq_\mathcal{T} \forall w.A\}.$$

We denote as $\mathcal{L}_\mathcal{T}(C, A)$ the language $\{w \mid (w, A) \in \mathcal{L}_\mathcal{T}(C)\}$. In the same way as for $\mathcal{FL}_0$ [14], these value restriction sets can be used to characterize subsumption.

Lemma 1. *Let $\mathcal{T}$ be an $\mathcal{FL}_0$ TBox and C, D two $\mathcal{FL}_{reg}$ concept descriptions. Then, $C \sqsubseteq_\mathcal{T} D$ iff $\mathcal{L}_\mathcal{T}(D) \subseteq \mathcal{L}_\mathcal{T}(C)$.*

Matching

To define the matching problem w.r.t. an $\mathcal{FL}_0$ TBox, we first need to introduce the notions of *concept patterns* and *substitutions*. We consider a set of concept variables $\mathsf{N_X}$ disjoint from $\mathsf{N_C}$ and $\mathsf{N_R}$. An $\mathcal{FL}_0$ concept pattern is an $\mathcal{FL}_0$ concept description defined over the set of concept names $\mathsf{N_C} \cup \mathsf{N_X}$ and the set of role names $\mathsf{N_R}$.

Informally, a matching problem in $\mathcal{FL}_0$ asks, given an $\mathcal{FL}_0$ concept description C and an $\mathcal{FL}_0$ concept pattern D, whether the variables occurring in D can be replaced by concept descriptions such that the resulting expression is equivalent to C. The meaning of "replacing" in the previous sentence is formalized using the notion of a substitution. An $\mathcal{FL}_{reg}$ *substitution* σ is a mapping assigning $\mathcal{FL}_{reg}$ concept descriptions $\sigma(X)$ to variables $X \in \mathsf{N_X}$. The application of such a substitution σ to concept patterns is inductively defined as follows:

$$\sigma(\top) := \top, \quad \sigma(A) := A \text{ for all } A \in \mathsf{N_C},$$
$$\sigma(C \sqcap D) := \sigma(C) \sqcap \sigma(D), \quad \sigma(\forall r.C) := \forall r.\sigma(C).$$

We say that σ is an $\mathcal{FL}_0$ *substitution* if $\sigma(X)$ is an $\mathcal{FL}_0$ concept description for all $X \in \mathsf{N_X}$.

Definition 2. Let $\mathcal{T}$ be an $\mathcal{FL}_0$ TBox. An $\mathcal{FL}_0$ *matching problem* w.r.t. $\mathcal{T}$ is an equation of the form $C \equiv^?_\mathcal{T} D$, where C is an $\mathcal{FL}_0$ concept description and D an $\mathcal{FL}_0$ concept pattern. The $\mathcal{FL}_{reg}$ substitution σ is an $\mathcal{FL}_{reg}$ *matcher* of this problem if $C \equiv_\mathcal{T} \sigma(D)$. If this matcher is an $\mathcal{FL}_0$ substitution, then we call it an $\mathcal{FL}_0$ *matcher*.

Given a matching problem $C \equiv^?_\mathcal{T} D$, only the σ-images of variables occurring in D are relevant. For this reason, we will assume in the following that $\mathsf{N_X}$ consists of exactly these variables.

Example 3. Let C and D respectively be the following $\mathcal{FL}_0$ concept description and $\mathcal{FL}_0$ concept pattern (in normal form):

$$C := \forall\{r, s\}.A \sqcap \forall\{s\}.B, \qquad D := \forall\{rr\}.A \sqcap \forall\{r, s\}.X_1 \sqcap \forall\{s\}.X_2.$$

It is easy to see that $\mathcal{L}_\emptyset(C, A) = \{r, s\}$. Since D has the conjunct $\forall\{rr\}.A$, we know that the word rr belongs to $\mathcal{L}_\emptyset(\sigma(D), A)$ for all substitutions σ. By the characterization of subsumption given in Lemma 1, this implies that the matching problem $C \equiv^? D$ has no matcher w.r.t. the empty TBox.

However, if we consider $C \equiv^? D$ w.r.t. TBox $\mathcal{T} := \{A \sqsubseteq \forall r.A, \forall s.B \sqsubseteq A\}$, there actually exists an $\mathcal{FL}_0$ matcher for this problem. Notice that the GCI $\forall s.B \sqsubseteq A$ implies that $C \sqsubseteq_\mathcal{T} A$. Moreover, $A \sqsubseteq \forall r.A$ yields that $A \sqsubseteq_\mathcal{T} \forall w.A$ for all $w \in \{r\}^*$. Hence, it follows that $\mathcal{L}_\mathcal{T}(C, A) = \{s\} \cup \{r\}^*$ and $\mathcal{L}_\mathcal{T}(C, B) = \{s\}$. By setting $\sigma(X_1) := A$ and $\sigma(X_2) := B$, we have that $\mathcal{L}_\mathcal{T}(C, A) = \mathcal{L}_\mathcal{T}(\sigma(D), A)$ and $\mathcal{L}_\mathcal{T}(C, B) = \mathcal{L}_\mathcal{T}(\sigma(D), B)$. Thus, Lemma 1 implies that σ is a matcher for $C \equiv^?_\mathcal{T} D$.

The polynomial-time algorithm for deciding whether an $\mathcal{FL}_0$ matching problem has an $\mathcal{FL}_0$ matcher w.r.t. the empty TBox introduced in [8] is based on the observation that such a problem has a matcher iff a certain candidate substitution is a matcher. The algorithm thus computes this candidate substitution and checks whether it is indeed a matcher. Basically, our matching algorithm proceeds in the same way, but we need to overcome two problems. First, the candidate substitution is an $\mathcal{FL}_{reg}$ substitution rather than an $\mathcal{FL}_0$ substitution. Thus, we actually check whether the problem has an $\mathcal{FL}_{reg}$ matcher. However, we then show that the existence of an $\mathcal{FL}_{reg}$ matcher also implies the existence of an $\mathcal{FL}_0$ matcher. Second, the candidate matcher may already be of exponential size. Thus, if we just use the result that subsumption in $\mathcal{FL}_{reg}$ w.r.t. an $\mathcal{FL}_0$ TBox is in ExpTime, we obtain a doubly-exponential upper bound for the overall complexity of checking whether the candidate substitution really is a matcher. In order to bring this upper bound down to ExpTime, we need a more fine-grained analysis of the complexity of the subsumption problem, which we provide in the next section.

3 Subsumption in $\mathcal{FL}_{reg}$ w.r.t. an $\mathcal{FL}_0$ TBox

Given $\mathcal{FL}_{reg}$ concept descriptions C, D and an $\mathcal{FL}_0$ TBox $\mathcal{T}$, we are interested in the complexity of deciding whether $C \sqsubseteq_\mathcal{T} D$ holds or not. Basically, we will use the characterization of subsumption given in Lemma 1 to obtain a subsumption algorithm. However, it turns out that a model-theoretic variant of this characterization is more appropriate to achieve a fine-grained complexity analysis that distinguishes between the size of C, D and the size of $\mathcal{T}$. For subsumption between $\mathcal{FL}_0$ concept descriptions w.r.t. an $\mathcal{FL}_0$ TBox $\mathcal{T}$, such a semantic characterization has been introduced in [14, 3].

Definition 4. Let $\mathcal{T}$ be an $\mathcal{FL}_0$ TBox and C an $\mathcal{FL}_{reg}$ concept description. An interpretation $\mathcal{I} = (\Delta^\mathcal{I}, \cdot^\mathcal{I})$ is called a *functional interpretation* if $\Delta^\mathcal{I} = \mathsf{N_R}^*$ and $r^\mathcal{I} := \{(u, ur) \mid u \in \mathsf{N_R}^*\}$ for all $r \in \mathsf{N_R}$. The functional interpretation $\mathcal{I}$ is called a

- *functional model of* C if $\varepsilon \in C^\mathcal{I}$,
- *functional model of* $\mathcal{T}$ if $\mathcal{I}$ is a model of $\mathcal{T}$,

- *functional model of* C *w.r.t.* $\mathcal{T}$ if $\varepsilon \in C^{\mathcal{I}}$ and $\mathcal{I}$ is a model of $\mathcal{T}$.

Calling such interpretations *functional* is justified by the fact that they interpret roles as (total) functions: for every $u \in \mathsf{N_R}^*$ and every $r \in \mathsf{N_R}$, the word ur is the unique r-successor of u. As an immediate consequence of this functional interpretation of roles, we have for all $A \in \mathsf{N_C}$ and $u, w \in \mathsf{N_R}^*$:

$$w \in (\forall u.A)^{\mathcal{I}} \text{ iff } wu \in A^{\mathcal{I}}.$$

We define inclusion and intersection of functional interpretations as follows:

- $\mathcal{I} \subseteq \mathcal{J}$ if $A^{\mathcal{I}} \subseteq A^{\mathcal{J}}$ for all $A \in \mathsf{N_C}$;
- $\mathcal{I} \cap \mathcal{J}$ is the unique functional interpretation that satisfies $A^{\mathcal{I}\cap\mathcal{J}} = A^{\mathcal{I}} \cap A^{\mathcal{J}}$ for all $A \in \mathsf{N_C}$.

It is easy to see that the above classes of functional models are closed under intersection, i.e., if $\mathcal{I}$ and $\mathcal{J}$ are both functional models of C w.r.t. $\mathcal{T}$ (and likewise of C, or of $\mathcal{T}$), then so is their intersection $\mathcal{I} \cap \mathcal{J}$. This actually not only holds for binary intersection, but also for arbitrary intersection of functional models. In particular, this implies that there must exist a *least functional model* of C w.r.t. $\mathcal{T}$, i.e., a functional model $\mathcal{J}$ of C w.r.t. $\mathcal{T}$ such that $\mathcal{J} \subseteq \mathcal{I}$ holds for all functional models $\mathcal{I}$ of C w.r.t. $\mathcal{T}$. There is a close connection between the least functional models and the value restriction sets introduced in the previous section.

Proposition 5. *Given an $\mathcal{FL}_{reg}$ concept description C and an $\mathcal{FL}_0$ TBox $\mathcal{T}$, let $\mathcal{I}_{C,\mathcal{T}} = (\mathsf{N_R}^*, \cdot^{\mathcal{I}_{C,\mathcal{T}}})$ be the functional interpretation satisfying $A^{\mathcal{I}_{C,\mathcal{T}}} = \{w \in \mathsf{N_R}^* \mid (w, A) \in \mathcal{L}_{\mathcal{T}}(C)\}$ for all $A \in \mathsf{N_C}$. Then, $\mathcal{I}_{C,\mathcal{T}}$ is the least functional model of C w.r.t. $\mathcal{T}$.*

The proof of this proposition is identical to the one given in [3] for the case where C is an $\mathcal{FL}_0$ concept description. Combining this result with Lemma 1 we can immediately conclude the following.

Corollary 6. *Let $\mathcal{T}$ be an $\mathcal{FL}_0$ TBox and C, D $\mathcal{FL}_{reg}$ concept descriptions. Then $C \sqsubseteq_{\mathcal{T}} D$ iff $\mathcal{I}_{D,\mathcal{T}} \subseteq \mathcal{I}_{C,\mathcal{T}}$.*

In order to test the condition $\mathcal{I}_{D,\mathcal{T}} \subseteq \mathcal{I}_{C,\mathcal{T}}$, we want to represent these least functional models using tree automata. To do this, we first need to introduce labeled trees and looping automata recognizing such trees. Let L be a finite set of labels and $\Sigma = \{\sigma_1, \ldots, \sigma_n\}$ a finite set of symbols. An infinite L-labeled Σ-tree t is a mapping $t : \Sigma^* \to L$ that assigns a label $t(w) \in L$ to each node $w \in \Sigma^*$. Intuitively, the nodes of a Σ-tree t correspond to finite words in Σ^*, where the empty word ε represents the root of t and every node w has n children corresponding to the words $w\sigma_1, \ldots, w\sigma_n$. The set of all infinite L-labeled Σ-trees is denoted as $T^{\omega}_{\Sigma,L}$.

Functional interpretations can be represented as $\mathsf{N_R}$-trees with labels from the set $2^{\mathsf{N_C}}$. More precisely, given a functional interpretation $\mathcal{I}$, the $2^{\mathsf{N_C}}$-labeled $\mathsf{N_R}$-tree $t_{\mathcal{I}}$ corresponding to $\mathcal{I}$ is defined as $t_{\mathcal{I}}(w) := \{A \in \mathsf{N_C} \mid w \in A^{\mathcal{I}}\}$. Conversely, any tree $t \in T^{\omega}_{\mathsf{N_R},2^{\mathsf{N_C}}}$ induces a functional interpretation $\mathcal{I}_t$ where $A^{\mathcal{I}_t} := \{w \in \mathsf{N_R}^* \mid A \in t(w)\}$ for every $A \in \mathsf{N_C}$. These two mappings are bijections that are inverse to each other. In the following, we will not always distinguish between a functional interpretation and its tree representation. For example, we will say that an automaton recognizes a functional interpretation $\mathcal{I}$ rather than use the (more exact) expression that it recognizes the tree representation $t_{\mathcal{I}}$ of $\mathcal{I}$.

Definition 7. A *looping tree automaton (LTA)* is a tuple $\mathcal{A} = (\Sigma, Q, L, \Delta, I)$ where $\Sigma = \{\sigma_1, \ldots, \sigma_n\}$ is a finite set of symbols, Q is a finite set of states, L is a finite set of labels, $\Delta \subseteq Q \times L \times Q^n$ is the transition relation and $I \subseteq Q$ the set of initial states. A run of $\mathcal{A}$ on a tree $t \in T^{\omega}_{\Sigma,L}$

is a Q-labeled Σ-tree $\rho : \Sigma^* \to Q$ such that $\rho(\varepsilon) \in I$ and $(\rho(w), t(w), \rho(w\sigma_1), \dots, \rho(w\sigma_n)) \in \Delta$ for all $w \in \mathsf{N_R}^*$. The tree language $\mathcal{L}(\mathcal{A})$ *recognized* by $\mathcal{A}$ is the set of all trees t such that $\mathcal{A}$ has a run on t.

In [3] it is shown how to construct an LTA $\mathcal{A}_{C,\mathcal{T}}$ that recognizes the functional models of an $\mathcal{FL}_0$ concept description C w.r.t. an $\mathcal{FL}_0$ TBox $\mathcal{T}$. If we set $C = \top$, then this automaton actually recognizes the functional models of $\mathcal{T}$. Thus, the construction and the results in [3] (in particular, Definition 8 and Lemma 9 of [3]) provide us with the following results.

Proposition 8. *Given an $\mathcal{FL}_0$ TBox $\mathcal{T}$, we can construct an LTA $\mathcal{A}_\mathcal{T}$ such that $\mathcal{L}(\mathcal{A}_\mathcal{T}) = \{t_\mathcal{I} \mid \mathcal{I}$ is a functional model of $\mathcal{T}\}$. The size of $\mathcal{A}_\mathcal{T}$ is exponential in the size of $\mathcal{T}$, and it can be constructed in exponential time.*

Now, consider an $\mathcal{FL}_{reg}$ concept description $C = \forall L_1.A_1 \sqcap \dots \sqcap \forall L_k.A_k$ in deterministic normal form, and let $\mathcal{A}_1, \dots, \mathcal{A}_k$ be the DFAs recognizing the languages $L_1, \dots, L_k$. In the following, we assume that these DFAs are of the form $\mathcal{A}_i = (Q_i, \mathsf{N_R}, q_i^0, \delta_i, F_i)$ where Q_i is the set of states, $\mathsf{N_R}$ the alphabet, q_i^0 the initial state, $\delta_i : Q_i \times \mathsf{N_R} \to Q_i$ the transition function, and $F_i \subseteq Q_i$ the set of final states. We can use these DFAs to construct an LTA $\mathcal{A}_C$ that recognizes the functional models of C. To be more precise, we define $\mathcal{A}_C := (P, \Sigma, L, \Delta, \{p^0\})$, where $P := Q_1 \times \dots \times Q_k$, $\Sigma := \mathsf{N_R} = \{r_1, \dots, r_n\}$, $L := 2^{\{A_1,\dots,A_k\}}$, $p^0 := (q_1^0, \dots, q_k^0)$, and

$$\begin{aligned}\Delta := \{(p, \ell, p_1, \dots, p_n) \mid p = (q_1, \dots, q_k) \in P,\ \ &\{A_i \mid 1 \le i \le k, q_i \in F_i\} \subseteq \ell, \\ &p_i = (\delta_1(q_1, r_i), \dots, \delta_k(q_k, r_i)) \text{ for } i = 1, \dots, n \quad \}\end{aligned}$$

Lemma 9. *$\mathcal{L}(\mathcal{A}_C) = \{t_\mathcal{I} \mid \mathcal{I}$ is a functional model of $C\}$.*

Proof. Since the automata $\mathcal{A}_i$ are deterministic, the automaton $\mathcal{A}_C$ has at most one run ρ, where $\rho(w) = (\delta_1(q_1^0, w), \dots, \delta_n(q_k^0, w))$ for all $w \in \Sigma^*$. For a given tree t, ρ is indeed a run on the tree t iff the following holds for all $w \in \mathsf{N_R}^*$: $t(w)$ contains all concept names A_i with $\delta_i(q_i^0, w) \in F_i$, i.e., it contains all A_i with $w \in L_i$. Consequently, the tree t is accepted by $\mathcal{A}_C$ iff $w \in A_i^{\mathcal{I}_t}$ holds for all $w \in L_i$. Since $w \in A_i^{\mathcal{I}_t}$ is equivalent to $\varepsilon \in (\forall w.A_i)^{\mathcal{I}_t}$, this shows that $\mathcal{A}_C$ accepts exactly the tree-representations of functional models of C. □

Note that the size of $\mathcal{A}_C$ is bounded by $h(m^k)$, where m is the maximal size of the automata $\mathcal{A}_1, \dots, \mathcal{A}_k$, k is the number of concept names occurring in C, and h is a polynomial. In order to obtain an automaton that recognizes all functional models of C w.r.t. $\mathcal{T}$, we can apply the standard product construction to obtain an automaton recognizing $\mathcal{L}(\mathcal{A}_C) \cap \mathcal{L}(\mathcal{A}_\mathcal{T}) = \{t_\mathcal{I} \mid \mathcal{I}$ is a functional model of C w.r.t. $\mathcal{T}\}$.

Proposition 10. *Given an $\mathcal{FL}_{reg}$ concept description C in DNF and an $\mathcal{FL}_0$ TBox $\mathcal{T}$, we can construct an LTA $\mathcal{A}_{C,\mathcal{T}}$ such that $\mathcal{L}(\mathcal{A}_{C,\mathcal{T}}) = \{t_\mathcal{I} \mid \mathcal{I}$ is a functional model of C w.r.t. $\mathcal{T}\}$. If m is the maximal size of the DFAs used to represent regular languages in C, k is the number of concept names occurring in C or $\mathcal{T}$, and τ is the size of $\mathcal{T}$, then the size of $\mathcal{A}_{C,\mathcal{T}}$ is bounded by $2^{h_1(\tau)} \cdot h_2(m^k)$ for polynomials h_1, h_2.*

Just as in the case of an $\mathcal{FL}_0$ concept description C, the automaton $\mathcal{A}_{C,\mathcal{T}}$ can be transformed into an LTA that accepts exactly the least functional model of C w.r.t. $\mathcal{T}$. This transformation removes states and transitions (see Definition 10 and Theorem 11 in [3]).

Proposition 11. *Given an $\mathcal{FL}_{reg}$ concept description C in DNF and an $\mathcal{FL}_0$ TBox $\mathcal{T}$, we can construct an LTA $\widehat{\mathcal{A}}_{C,\mathcal{T}}$ such that $\mathcal{L}(\widehat{\mathcal{A}}_{C,\mathcal{T}}) = \{t_{\mathcal{I}_{C,\mathcal{T}}}\}$. The size of $\widehat{\mathcal{A}}_{C,\mathcal{T}}$ is bounded by the size of $\mathcal{A}_{C,\mathcal{T}}$.*

Now, let $\mathcal{T}$ be an $\mathcal{FL}_0$ TBox and C, D $\mathcal{FL}_{reg}$ concept descriptions in DNF. According to Corollary 6, we have $C \sqsubseteq_\mathcal{T} D$ iff $\mathcal{I}_{D,\mathcal{T}} \subseteq \mathcal{I}_{C,\mathcal{T}}$. As shown in [3], the latter condition can be reduced to the emptiness problem for an LTA that is obtained from $\mathcal{A}_{C,\mathcal{T}}$ and $\mathcal{A}_{D,\mathcal{T}}$ using an appropriate product construction (see the construction above Corollary 12 in [3]). Since the emptiness problem for LTAs can be decided in linear time, this yields the following fine-grained complexity result for subsumption.

Theorem 12. *There are polynomials h_1, h_2 such that subsumption between $\mathcal{FL}_{reg}$ concept descriptions C, D in DNF w.r.t. an $\mathcal{FL}_0$ TBox $\mathcal{T}$ can be decided in time at most $2^{h_1(\tau)} \cdot h_2(m^k)$ where m is the maximal size of the DFAs used to represent regular languages in C, D, k is the number of concept names occurring in C, D or $\mathcal{T}$, and τ is the size of $\mathcal{T}$.*

If we start with arbitrary $\mathcal{FL}_{reg}$ concept descriptions C, D, then we can construct equivalent normal forms in polynomial time without changing the set of concept names occurring in these concept descriptions. Transforming the regular expressions or NFAs representing the regular languages in these normal forms into equivalent deterministic automata may produce DFAs whose size is exponential in the size of C, D. Thus, the maximal size m of the DFAs occurring in the DNFs of C, D is bounded by 2^s where s is the combined size of C, D. Thus, $h_2(m^k) = h_2((2^s)^k) = h_2(2^{s \cdot k})$ is still single-exponential in the size of C, D.

Corollary 13. *Let C, D be $\mathcal{FL}_{reg}$ concept descriptions, and $\mathcal{T}$ an $\mathcal{FL}_0$ TBox. Then subsumption between C and D w.r.t. $\mathcal{T}$ can be decided in time exponential in the combined size of C, D, and $\mathcal{T}$.*

4 The complexity of matching in $\mathcal{FL}_0$ w.r.t. TBoxes

Since subsumption in $\mathcal{FL}_0$ w.r.t. a TBox is ExpTime-complete [2], matching w.r.t. a TBox is ExpTime-hard. In fact, we have $E \sqsubseteq_\mathcal{T} F$ iff the matching problem $C \equiv^? D$ has a matcher w.r.t. $\mathcal{T}$, where $C = E \sqcap F$ and $D = E$ is a variable-free pattern. This hardness result is, of course, independent of whether we are looking for a matcher in $\mathcal{FL}_{reg}$ or in $\mathcal{FL}_0$. In this section, we will show the corresponding upper bounds, first for the existence of an $\mathcal{FL}_{reg}$ matcher, and then for $\mathcal{FL}_0$ matchers.

Deciding the existence of an $\mathcal{FL}_{reg}$-matcher

By applying the normalization rules described above to the $\mathcal{FL}_0$ concept pattern D, a given $\mathcal{FL}_0$ matching problem $C \equiv^?_\mathcal{T} D$ can be equivalently stated as an equation of the form:

$$C \equiv^?_\mathcal{T} E \sqcap \forall L_1.X_1 \sqcap \ldots \sqcap \forall L_m.X_m \qquad (2)$$

where E is an $\mathcal{FL}_0$ concept description, $L_1, \ldots, L_m$ are finite languages over $\mathsf{N_R}$ (given by the enumeration of their elements), and $X_1, \ldots, X_m$ are the concept variables occurring in D. Generalizing the approach for matching in $\mathcal{FL}_0$ without a TBox [8], we now show that an equation of the form (2) has an $\mathcal{FL}_{reg}$ matcher iff a certain candidate substitution is a matcher. For this, we need to introduce the following notation: given a language L and a word u, we define the *left-quotient* of L w.r.t. u as $u^{-1}L := \{v \mid uv \in L\}$.

Let σ be an $\mathcal{FL}_{reg}$ matcher of (2) such that $\sigma(X_i)$ is in normal form, and assume that $\forall L_{i,j}.A_j$ is the conjunct for the concept name A_j in $\sigma(X_i)$. Then, after applying the substitution σ to the right-hand side of the equation (2), the value restriction $\forall L_i \cdot L_{i,j}.A_j$ is a conjunct on the right-hand side, and thus subsumes C. Lemma 1 thus implies that $L_i \cdot L_{i,j} \subseteq \mathcal{L}_\mathcal{T}(C, A_j)$.

Now, assume that $v \in L_{i,j}$. Then we know that $uv \in \mathcal{L}_\mathcal{T}(C, A_j)$ must hold for every $u \in L_i$, i.e., $v \in u^{-1}\mathcal{L}_\mathcal{T}(C, A_j)$ for all $u \in L_i$. This shows that $L_{i,j} \subseteq \bigcap_{u\in L_i} u^{-1}\mathcal{L}_\mathcal{T}(C, A_j)$. At first sight, this does not help us in our search for matchers since the languages $\mathcal{L}_\mathcal{T}(C, A_j)$ are infinite, and there are thus possibly infinitely many choices for such subsets to consider. However, we can show that we can restrict our attention to the maximal such sets. To be more precise, we define for $i = 1, \ldots, m$ and $j = 1, \ldots, k$ the languages

$$\widehat{L}_{i,j} := \bigcap_{u\in L_i} u^{-1}\mathcal{L}_\mathcal{T}(C, A_j).$$

Since the class of regular languages is closed under building left-quotients and finite intersections, and the languages $\mathcal{L}_\mathcal{T}(C, A_j)$ are regular [14, Theorem 5.21], the languages $\widehat{L}_{i,j}$ are also regular. Thus, we can use them within $\mathcal{FL}_{reg}$ concept descriptions. Consequently, if we define the *candidate substitution* $\widehat{\sigma}$ as

$$\widehat{\sigma}(X_i) := \forall\widehat{L}_{i,1}.A_1 \sqcap \ldots \sqcap \forall\widehat{L}_{i,k}.A_k \quad \text{for } i = 1, \ldots, m,$$

then $\widehat{\sigma}$ is a well-defined $\mathcal{FL}_{reg}$ substitution.

Lemma 14. *The equation (2) has an $\mathcal{FL}_{reg}$ matcher iff the candidate substitution $\widehat{\sigma}$ is a matcher of (2).*

Proof. Since $\widehat{\sigma}$ is an $\mathcal{FL}_{reg}$ substitution, the if-direction of the proof is trivial. To show the other direction, assume that equation (2) has an $\mathcal{FL}_{reg}$ matcher σ, i.e., σ is an $\mathcal{FL}_{reg}$ substitution such that $C \equiv_\mathcal{T} E \sqcap \forall L_1.\sigma(X_1) \sqcap \ldots \sqcap \forall L_m.\sigma(X_m)$. This implies that $C \sqsubseteq_\mathcal{T} E$. Moreover, the construction of $\widehat{\sigma}$ implies that $C \sqsubseteq_\mathcal{T} \forall L_i.\widehat{\sigma}(X_i)$ for all $i, 1 \leq i \leq m$ since $L_i{\cdot}\widehat{L}_{i,j} \subseteq \mathcal{L}_\mathcal{T}(C, A_j)$ holds for all $j, 1 \leq j \leq k$. Consequently, we have $C \sqsubseteq_\mathcal{T} E \sqcap \forall L_1.\widehat{\sigma}(X_1) \sqcap \ldots \sqcap \forall L_m.\widehat{\sigma}(X_m)$.

To see the opposite direction, assume that $\sigma(X_i) = \forall L_{i,1}.A_1 \sqcap \ldots \sqcap \forall L_{i,k}.A_k$ for $i = 1, \ldots, m$. As argued above, the fact that σ is an $\mathcal{FL}_{reg}$ matcher of (2) implies that $L_{i,j} \subseteq \widehat{L}_{i,j}$ holds for all $i, 1 \leq i \leq m$ and $j, 1 \leq j \leq k$. Consequently, we have $\widehat{\sigma}(X_i) \sqsubseteq \sigma(X_i)$ for all $i, 1 \leq i \leq m$, which yields $E \sqcap \forall L_1.\widehat{\sigma}(X_1) \sqcap \ldots \sqcap \forall L_m.\widehat{\sigma}(X_m) \sqsubseteq_\mathcal{T} E \sqcap \forall L_1.\sigma(X_1) \sqcap \ldots \sqcap \forall L_k.\sigma(X_m) \equiv_\mathcal{T} C$. Thus, we can conclude that $\widehat{\sigma}$ is a matcher of (2). □

This lemma reduces deciding whether (2) has an $\mathcal{FL}_{reg}$ matcher to deciding whether $\widehat{\sigma}$ is a matcher of (2). The latter can be checked as follows.

Lemma 15. *The candidate substitution $\widehat{\sigma}$ is a matcher of (2) iff*

1. $C \sqsubseteq_\mathcal{T} E$, and 2. $E \sqcap \forall L_1.\widehat{\sigma}(X_1) \sqcap \ldots \sqcap \forall L_m.\widehat{\sigma}(X_m) \sqsubseteq_\mathcal{T} C$.

The first condition requires testing subsumption of $\mathcal{FL}_0$ concept descriptions w.r.t. an $\mathcal{FL}_0$ TBox, which can be performed in exponential time. The second condition requires testing subsumption of $\mathcal{FL}_{reg}$ concept descriptions w.r.t. an $\mathcal{FL}_0$ TBox. However, we cannot directly apply Corollary 13 since the size of these concept descriptions need not be polynomial in the combined size of C, D, and $\mathcal{T}$. To show that this test can also be performed in exponential time, we must use the more fine-grained complexity result for subsumption of $\mathcal{FL}_{reg}$ concept descriptions in DNF w.r.t. an $\mathcal{FL}_0$ TBox of Theorem 12. To obtain the desired ExpTime upper bound, we thus need to show that there are DFAs recognizing the regular languages in the normal form of $E \sqcap \forall L_1.\widehat{\sigma}(X_1) \sqcap \ldots \sqcap \forall L_m.\widehat{\sigma}(X_m)$ that are of size at most exponential in the combined size of C, D, and $\mathcal{T}$. Assume the normal form of E is $\forall K_1.A_1 \sqcap \ldots \sqcap \forall K_k.A_k$, for finite languages $K_1, \ldots, K_k$. Then the normal form of $E \sqcap \forall L_1.\widehat{\sigma}(X_1) \sqcap \ldots \sqcap \forall L_m.\widehat{\sigma}(X_m)$ is $\forall M_1.A_1 \sqcap \ldots \sqcap \forall M_k.A_k$, where

$$M_j = K_j \cup L_1{\cdot}\widehat{L}_{1,j} \cup \ldots \cup L_m{\cdot}\widehat{L}_{m,j} \quad \text{for } j = 1, \ldots, k.$$

Lemma 16. *For all $j, 1 \leq j \leq k$, there is a DFA recognizing M_j whose size is at most exponential in the combined size of C, D, and $\mathcal{T}$.*

Proof. We start the proof by constructing DFAs of appropriate size for the languages $\widehat{L}_{i,j} = \bigcap_{u \in L_i} u^{-1}\mathcal{L}_\mathcal{T}(C, A_j)$. As shown in [14], the languages $\mathcal{L}_\mathcal{T}(C, A_j)$ are recognized by DFAs $\mathcal{A}_j$ whose sizes are at most exponential in the combined size of C and $\mathcal{T}$. For every $j, 1 \leq j \leq k$ and $u \in L_i$, a DFA $\mathcal{A}_{j,u}$ for $u^{-1}\mathcal{L}_\mathcal{T}(C, A_j)$ is obtained from $\mathcal{A}_j$ by taking as new initial state the state reached with u from the initial state of $\mathcal{A}_j$. The intersection $\bigcap_{u \in L_i} u^{-1}\mathcal{L}_\mathcal{T}(C, A_j)$ can then be realized by building the product automaton $\mathcal{P}_{i,j}$ of the automata $\mathcal{A}_{j,u}$ for all $u \in L_i$. The size of $\mathcal{P}_{i,j}$ is exponential in $|L_i|$ times the combined size of C and $\mathcal{T}$, and thus exponential in the combined size of C, D, and $\mathcal{T}$.

Second, let us consider the concatenation $L_i \cdot \widehat{L}_{i,j}$. For every $u \in L_i$ we can construct a DFA $\mathcal{P}^u_{i,j}$ for $\{u\} \cdot \widehat{L}_{i,j}$ by adding $|u|$ many states "before" the initial state of $\mathcal{P}_{i,j}$. A DFA $\widehat{\mathcal{P}}_{i,j}$ for the union $\bigcup_{u \in L_i} \{u\} \cdot \widehat{L}_{i,j} = L_i \cdot \widehat{L}_{i,j}$ can then again be obtained by a product construction. The exponent $|L_i|$ in the size of $\widehat{\mathcal{P}}_{i,j}$ caused by this product construction becomes a factor in the overall exponent and it is bounded by the size of the pattern D. Consequently, the sizes of the DFAs $\widehat{\mathcal{P}}_{i,j}$ recognizing $L_i \cdot \widehat{L}_{i,j}$ are again exponential in the combined size of C, D, and $\mathcal{T}$.

Finally, the union $K_j \cup L_1 \cdot \widehat{L}_{1,j} \cup \ldots \cup L_m \cdot \widehat{L}_{m,j}$ can again be realized by a product construction, where the exponent $m + 1$ caused by this construction again becomes a factor in the overall exponent and is bounded by the size of the pattern D. □

Together with Theorem 12, this lemma yields the desired ExpTime upper bound.

Theorem 17. *The problem of deciding whether an $\mathcal{FL}_0$ matching problem w.r.t. an $\mathcal{FL}_0$ TBox has an $\mathcal{FL}_{reg}$ matcher or not is* ExpTime*-complete.*

We illustrate our ExpTime decision procedure using the matching problem of Example 3.

Example 18. Consider the matching problem $C \equiv_\mathcal{T} D$, where

$$C := \forall\{r, s\}.A \sqcap \forall\{s\}.B, \quad D := \forall\{rr\}.A \sqcap \forall\{r, s\}.X_1 \sqcap \forall\{s\}.X_2, \quad \mathcal{T} := \{A \sqsubseteq \forall r.A, \forall s.B \sqsubseteq A\}.$$

We have seen in Example 3 that $\mathcal{L}_\mathcal{T}(C, A) = \{s\} \cup \{r\}^*$ and $\mathcal{L}_\mathcal{T}(C, B) = \{s\}$. Since $r^{-1}\mathcal{L}_\mathcal{T}(C, A) \cap s^{-1}\mathcal{L}_\mathcal{T}(C, A) = \{r\}^* \cap \{\varepsilon\} = \{\varepsilon\}$ and $r^{-1}\mathcal{L}_\mathcal{T}(C, B) \cap s^{-1}\mathcal{L}_\mathcal{T}(C, B) = \emptyset \cap \{\varepsilon\} = \emptyset$, the value of the candidate substitution for X_1 is $\widehat{\sigma}(X_1) := A$. Regarding X_2, we have $s^{-1}\mathcal{L}_\mathcal{T}(C, A) = \{\varepsilon\}$ and $s^{-1}\mathcal{L}_\mathcal{T}(C, B) = \{\varepsilon\}$, which yields $\widehat{\sigma}(X_2) := A \sqcap B$. It is easy to see that $\widehat{\sigma}$ is in fact a matcher. Here the candidate substitution is actually an $\mathcal{FL}_0$ substitution, and thus also shows that there is an $\mathcal{FL}_0$ matcher.

In general, this need not be the case. For example, consider the matching problem $C' \equiv_{\mathcal{T}'} D'$, where $C' := A$, $D' := A \sqcap \forall r.X$, and $\mathcal{T}' := \{A \sqsubseteq \forall r.A\}$. We have $\mathcal{L}_\mathcal{T}(C', A) = \{r\}^*$ and $r^{-1}\mathcal{L}_\mathcal{T}(C', A) = \{r\}^*$. Thus, the candidate substitution is defined as $\widehat{\sigma}'(X) := \forall\{r\}^*.A$. This substitution is an $\mathcal{FL}_{reg}$ matcher, but it is not an $\mathcal{FL}_0$ substitution. Nevertheless, the matching problem has $\mathcal{FL}_0$ matchers. For example, for all $n \geq 0$, the subsitution σ_n with $\sigma_n(X) := \forall r^n.A$ is an $\mathcal{FL}_0$ matcher.

Deciding the existence of an $\mathcal{FL}_0$-matcher

We will show that a matching problem of the form (2) has an $\mathcal{FL}_0$ matcher iff it has an $\mathcal{FL}_{reg}$ matcher. We have shown that any matcher σ of (2) with normal form

$$\sigma(X_i) := \forall L_{i,1}.A_1 \sqcap \ldots \sqcap \forall L_{i,k}.A_k \quad \text{for } i = 1, \ldots, m, \tag{3}$$

satisfies $L_{i,j} \subseteq \widehat{L}_{i,j}$ for $i = 1, \ldots, m$ and $j = 1, \ldots, k$. Obviously, this substitution is an $\mathcal{FL}_0$ substitution iff the languages $L_{i,j}$ are finite.

Analogously to Lemma 15 we can thus show the following lemma characterizing the existence of $\mathcal{FL}_0$ matchers.

Lemma 19. *Equation (2) has an $\mathcal{FL}_0$ matcher iff*

1. *$C \sqsubseteq_\mathcal{T} E$, and*
2. *there are finite languages $L_{i,j} \subseteq \widehat{L}_{i,j}$ such that $E \sqcap \forall L_1.\sigma(X_1) \sqcap \ldots \sqcap \forall L_m.\sigma(X_m) \sqsubseteq_\mathcal{T} C$, where σ is defined as in (3).*

The second condition is satisfied if the candidate substitution $\widehat{\sigma}$ satisfies the corresponding condition. Before we can prove this implication, we need to introduce some more notation. A possibly negated $\mathcal{FL}_0$ *concept assertion* is of the form $C(a)$ or $\neg C(a)$ where C is an $\mathcal{FL}_0$ concept description and a is an individual name from a set $\mathsf{N_I}$ of such names. We extend the semantics of $\mathcal{FL}_0$ such that interpretations $\mathcal{I}$ assign elements $a^\mathcal{I} \in \Delta^\mathcal{I}$ to individual names $a \in \mathsf{N_I}$. Given a (finite or infinite) set M of such assertions, we say that M is *consistent* w.r.t. the $\mathcal{FL}_0$ TBox $\mathcal{T}$ if there is a model $\mathcal{I}$ of $\mathcal{T}$ such that $a^\mathcal{I} \in C^\mathcal{I}$ holds for all positive concept assertions $C(a)$ in M and $a^\mathcal{I} \notin C^\mathcal{I}$ holds for all negative concept assertions $\neg C(a)$ in M.

Lemma 20. *If $E \sqcap \forall L_1.\widehat{\sigma}(X_1) \sqcap \ldots \sqcap \forall L_m.\widehat{\sigma}(X_m) \sqsubseteq_\mathcal{T} C$, then Condition 2 in Lemma 19 is satisfied.*

Proof. It is easy to see that $E \sqcap \forall L_1.\widehat{\sigma}(X_1) \sqcap \ldots \sqcap \forall L_m.\widehat{\sigma}(X_m) \sqsubseteq_\mathcal{T} C$ holds in $\mathcal{FL}_{reg}$ iff the following (possibly infinite) set of assertions is inconsistent w.r.t. the $\mathcal{FL}_0$ TBox $\mathcal{T}$:

$$\{E(a), \neg C(a)\} \cup \bigcup_{i=1}^{m} \bigcup_{j=1}^{k} \{(\forall uv.A_j)(a) \mid u \in L_i \wedge v \in \widehat{L}_{i,j}\}.$$

Since $\mathcal{FL}_0$ TBoxes and possibly negated $\mathcal{FL}_0$ concept assertions can clearly be translated into sentences of first-order logic (FOL), compactness of FOL implies that there is a *finite* set $\Gamma \subseteq \bigcup_{i=1}^{m} \bigcup_{j=1}^{k} \{(\forall uv.A_j)(a) \mid u \in L_i \wedge v \in \widehat{L}_{i,j}\}$ such that $\{E(a), \neg C(a)\} \cup \Gamma$ is inconsistent w.r.t. $\mathcal{T}$. We use Γ to define finite subsets $L_{i,j}$ of $\widehat{L}_{i,j}$:

$$L_{i,j} := \{v \in \widehat{L}_{i,j} \mid \text{ there is } u \in L_i \text{ such that } (\forall uv.A_j)(a) \in \Gamma\}.$$

Then Γ is a subset of the set $\Gamma' := \bigcup_{i=1}^{m} \bigcup_{j=1}^{k} \{\forall uv.A(a) \mid u \in L_i, v \in L_{i,j}\}$, which implies that $\{E(a), \neg C(a)\} \cup \Gamma'$ is also inconsistent w.r.t. $\mathcal{T}$. This in turn implies $E \sqcap \forall L_1.\sigma(X_1) \sqcap \ldots \sqcap \forall L_m.\sigma(X_m) \sqsubseteq_\mathcal{T} C$, where σ is defined as in (3). □

We are now ready to show the following equivalence.

Theorem 21. *An $\mathcal{FL}_0$ matching problem has an $\mathcal{FL}_{reg}$ matcher iff it has an $\mathcal{FL}_0$ matcher.*

Proof. Clearly, an $\mathcal{FL}_0$ matcher is also an $\mathcal{FL}_{reg}$ matcher. Conversely, assume that the matching problem is of the form (2) and that it has an $\mathcal{FL}_{reg}$ matcher. Then the two conditions in Lemma 15 are satisfied. The first condition coincides with the first condition in Lemma 19 and, according to Lemma 20, the second condition implies the second condition in Lemma 19. Thus, Lemma 19 yields the existence of an $\mathcal{FL}_0$ matcher. □

As an immediate consequence of Theorem 17, we thus obtain the following complexity result.

Corollary 22. *The problem of deciding whether an $\mathcal{FL}_0$ matching problem w.r.t. an $\mathcal{FL}_0$ TBox has an $\mathcal{FL}_0$ matcher or not is* ExpTime*-complete.*

5 Subsumption and matching w.r.t. forward TBoxes

Let $\mathcal{T}$ be an $\mathcal{FL}_0$ TBox. We assume in the following that the GCIs in $\mathcal{T}$ are of the form

$$\forall v_1.A_1 \sqcap \cdots \sqcap \forall v_s.A_s \sqsubseteq \forall v.A, \tag{4}$$

where $v_1, \ldots, v_s, v \in \mathsf{N_R}^*$ and $A_1, \ldots, A_s, A \in \mathsf{N_C}$. This is without loss of generality since (i) any $\mathcal{FL}_0$ concept description is equivalent to a conjunction of value restrictions of the form $\forall w.B$, and (ii) $C \sqsubseteq D \sqcap E$ iff $C \sqsubseteq D$ and $C \sqsubseteq E$.

Definition 23. Let $\mathcal{T}$ be an $\mathcal{FL}_0$ TBox. Then $\mathcal{T}$ is called a *forward TBox* if all its GCIs are of the form (4) where $|v_i| \leq |v|$ for all $i, 1 \leq i \leq s$.

For example, the GCI $A \sqsubseteq \forall r.A$ can be an element of a forward TBox, but the GCI $\forall r.B \sqsubseteq A$ cannot. We show in the following that restricting to forward TBoxes lowers the complexity of subsumption and matching from ExpTime to PSpace. Actually, the main contribution of this section is developing the PSpace subsumption algorithm. The PSpace matching algorithm can then be obtained from it by a simple modification.

Subsumption in $\mathcal{FL}_0$ w.r.t. forward TBoxes

We assume in the following that all $\mathcal{FL}_0$ concept descriptions are conjunctions of value restrictions of the form $\forall w.B$ for $w \in \mathsf{N_R}^*$ and $B \in \mathsf{N_C}$. Given such a concept description D, we denote with $\widehat{D}$ the set of these value restrictions. For example, if $D = \forall rr.A \sqcap \forall s.A \sqcap \forall s.B$, then $\widehat{D} = \{\forall rr.A, \forall s.A, \forall s.B\}$.

For the same reason that GCIs can be restricted without loss of generality to being of the form (4), we can also restrict the attention to subsumption problems of the form $C \sqsubseteq_{\mathcal{T}} \forall w.A$.

The main idea underlying the PSpace subsumption algorithm presented below is the following: if $C \sqsubseteq_{\mathcal{T}} \forall w_0.A_0$ then either $\forall w_0.A_0 \in \widehat{C}$, or there is some GCI $\forall v_1.A_1 \sqcap \cdots \sqcap \forall v_s.A_s \sqsubseteq \forall v.A_0$ with $w_0 = pv$ for some $p \in \mathsf{N_R}^*$ and $C \sqsubseteq_{\mathcal{T}} \forall pv_i.A_i$ for $i = 1, \ldots, s$. This idea is formalized using the notion of a derivation tree.

Definition 24. Let $\mathcal{T}$ be an $\mathcal{FL}_0$ TBox, C an $\mathcal{FL}_0$ concept description, and $\forall w_0.A_0$ a value restriction. A *derivation tree* for $\forall w_0.A_0$ w.r.t. $\mathcal{T}$ is a finite tree T satisfying the following properties:

1. The nodes of T are labeled with value restrictions, where the root is labeled with $\forall w_0.A_0$.
2. If $\forall w.A$ labels a node k of T and $\forall w_1.A_1, \ldots, \forall w_s.A_s$ are the labels of its children $k_1, \ldots, k_s$, then there is a GCI g in $\mathcal{T}$ of the form $g : \forall v_1.A_1 \sqcap \ldots \sqcap \forall v_s.A_s \sqsubseteq \forall v.A$ and a word $p \in \mathsf{N_R}^*$ such that $w = pv$ and $w_i = pv_i$ for all $i = 1, \ldots, s$. Each child node k_i with label $\forall w_i.A_i$ is assigned the following two additional labels: the *GCI-used* $\mathfrak{g}(k_i) = g$ and the *prefix* $\mathfrak{d}(k_i) = p$. For the root k_0 we set $\mathfrak{g}(k_0) = \bot$ (standing for "no GCI") and $\mathfrak{d}(k_0) = w_0$.

We denote as $\mathfrak{T}_{\mathcal{T}}(\forall w_0.A_0)$ the set of all derivation trees for $\forall w_0.A_0$ w.r.t. $\mathcal{T}$. The set of value restrictions labeling the leaves of such a tree T is denoted as $\mathfrak{L}(T)$. We say that T is a *derivation tree for* $C \sqsubseteq \forall w_0.A_0$ *w.r.t.* $\mathcal{T}$ if $\mathfrak{L}(T) \subseteq \widehat{C}$, and denote the set of such trees with $\mathfrak{T}_{\mathcal{T},C}(\forall w_0.A_0)$. Finally, $\mathfrak{V}_{\mathcal{T},C}$ consists of the value restrictions $\forall w_0.A_0$ such that $\mathfrak{T}_{\mathcal{T},C}(\forall w_0.A_0) \neq \emptyset$.

Derivation trees can be used to obtain the following characterization of subsumption in $\mathcal{FL}_0$ w.r.t. TBoxes.

Lemma 25. *Let $\mathcal{T}$ be an $\mathcal{FL}_0$ TBox, C an $\mathcal{FL}_0$ concept description, and $\forall w_0.A_0$ a value restriction. Then, $C \sqsubseteq_{\mathcal{T}} \forall w_0.A_0$ iff there exists a derivation tree for $C \sqsubseteq \forall w_0.A_0$ w.r.t. $\mathcal{T}$.*

Proof. The if-direction can be shown by a simple induction over the size of the derivation tree.

To prove the only-if direction, we use the set $\mathfrak{V}_{\mathcal{T},C}$ to construct a functional interpretation as follows: let $\mathcal{J}_{\mathcal{T},C}$ be the functional interpretation such that

$$A^{\mathcal{J}_{\mathcal{T},C}} := \{w \mid \forall w.A \in \mathfrak{V}_{\mathcal{T},C}\} \text{ for all } A \in \mathsf{N_C}.$$

We show that $\mathcal{J}_{\mathcal{T},C}$ is a model of $\mathcal{T}$. Let $E \sqsubseteq F \in \mathcal{T}$ and $w \in \Delta^{\mathcal{J}_{\mathcal{T},C}}$ be such that $w \in E^{\mathcal{I}_{\mathcal{T},C}}$. We can assume that $E \sqsubseteq F$ is of the form (4). Since $w \in E^{\mathcal{J}_{\mathcal{T},C}}$, this means that $wv_i \in (A_i)^{\mathcal{J}_{\mathcal{T},C}}$ for all $i, 1 \leq i \leq s$. Hence, by the definition of $\mathcal{J}_{\mathcal{T},C}$, we have that $\forall wv_i.A_i \in \mathfrak{V}_{\mathcal{T},C}$ for all $i, 1 \leq i \leq s$. By definition of $\mathfrak{V}_{\mathcal{T},C}$ we thus know that there exist derivation trees $T_1 \in \mathfrak{T}_{\mathcal{T},C}(\forall wv_1.A_1), \ldots, T_s \in \mathfrak{T}_{\mathcal{T},C}(\forall wv_s.A_s)$. From this, it is clear that one can build a derivation tree $T \in \mathfrak{T}_{\mathcal{T},C}(\forall wv.A)$. This implies that $wv \in A^{\mathcal{I}_{\mathcal{T},C}}$ and thus $w \in (\forall v.A)^{\mathcal{J}_{\mathcal{T},C}}$. This shows that $\mathcal{J}_{\mathcal{T},C}$ is a model of $\mathcal{T}$.

Finally, notice that $\forall w.A \in \mathfrak{V}_{\mathcal{T},C}$ for all $\forall w.A \in \widehat{C}$. Hence, by the definition of $\mathcal{J}_{\mathcal{T},C}$, we have that $\varepsilon \in C^{\mathcal{J}_{\mathcal{T},C}}$. Since $C \sqsubseteq_{\mathcal{T}} \forall w_0.A_0$, this implies that $\varepsilon \in (\forall w_0.A_0)^{\mathcal{J}_{\mathcal{T},C}}$, and thus $w_0 \in A_0^{\mathcal{J}_{\mathcal{T},C}}$. This yields $\forall w_0.A_0 \in \mathfrak{V}_{\mathcal{T},C}$, which shows $\mathfrak{T}_{\mathcal{T},C}(\forall w_0.A_0) \neq \emptyset$ as required. □

Using this lemma, we can try to decide whether $C \sqsubseteq_{\mathcal{T}} \forall w_0.A_0$ as follows. Start with the tree that has just one node k_0 labeled with $\forall w_0.A_0$. If this label belongs to $\widehat{C}$, then stop with success. Otherwise, try to find a GCI that allows to *expand the node* k_0 by adding children with appropriate labels (see 2. in Definition 24). If the subsumption $C \sqsubseteq_{\mathcal{T}} \forall w_0.A_0$ holds, then there must be such a GCI. Thus, if there is none, we can stop with failure. Now assume that we have already generated a derivation tree T for $\forall w_0.A_0$. If $\mathfrak{L}(T) \subseteq \widehat{C}$, then we can stop with success. Otherwise, we pick a leaf whose label does not belong to $\widehat{C}$ and try to expand it using an appropriate GCI. If no such GCI exists, we stop with failure. Otherwise, we continue the expansion process until a failure case occurs or the labels of all leaves belong to $\widehat{C}$.

This approach, as described until now, does not yield a PSpace algorithm for subsumption for two reasons. First, the generated derivation trees may grow to having exponential size. Second, expansion need not terminate unless we install an appropriate cycle check, but then keeping the information necessary to detect cycles may require exponential space.

The main idea to solve the first problem is that we actually need not store the whole derivation tree T. It is sufficient to know the value restrictions in $\mathfrak{L}(T) \setminus \widehat{C}$ (together with the prefix label of the corresponding leaf). In fact, the value restrictions in $\mathfrak{L}(T) \setminus \widehat{C}$ are the ones that require further expansion. In order to ensure that this information can be represented using only polynomial space, we restrict the choice of which leaf is expanded next.

Definition 26. Let T, T' be derivation trees for $\forall w_0.A_0$ w.r.t. $\mathcal{T}$. We write $T \to T'$ if T' is obtained from T by expanding a leaf whose label belongs to $\mathfrak{L}(T) \setminus \widehat{C}$, where among the eligible such leaves we choose one whose prefix label has maximal length.

The following proposition is an easy consequence of Lemma 25 and the fact that the order of leaf expansion is irrelevant.

Proposition 27. *Let $\mathcal{T}$ be an $\mathcal{FL}_0$ TBox, C an $\mathcal{FL}_0$ concept description, and $\forall w_0.A_0$ a value restriction. Then, $C \sqsubseteq_{\mathcal{T}} \forall w_0.A_0$ iff there exists a sequence $T_0 \to T_1 \to \ldots \to T_n$ of derivation trees for $\forall w_0.A_0$ w.r.t. $\mathcal{T}$ such that T_0 has just one node k_0 labeled with $\forall w_0.A_0$ and $T_n \in \mathfrak{T}_{\mathcal{T},C}(\forall w_0.A_0)$.*

When checking for the existence of such a sequence $T_0 \to T_1 \to \ldots \to T_n$, it is sufficient to keep only $\mathfrak{L}(T_i) \setminus \widehat{C}$ as well as the prefix labels of the leaves that yield these value restrictions in memory.[1] Thus, our algorithm works with sets consisting of elements of the form $(\forall w.A, p)$ where $\forall w.A$ is the value restriction label and p is the prefix label of a leaf. We now show that the cardinality of these sets is polynomial in the size of $\mathcal{T}$, C, and $\forall w_0.A_0$, and that their elements come from an at most exponentially large base set. In fact, this then implies that there are only exponentially many such sets.

Given a derivation tree T, we denote the set of prefixes of nodes whose value restrictions belong to $\mathfrak{L}(T) \setminus \widehat{C}$ with $\mathcal{P}(T)$. In addition, we denote the prefix order on words with $\preceq$.

Lemma 28. *Let $\mathcal{T}$ be a forward TBox, and $T_0 \to T_1 \to \ldots \to T_n$ be a sequence of derivation trees for $\forall w_0.A_0$ w.r.t. $\mathcal{T}$ where T_0 is as described in Proposition 27. Then, for each $i, 0 \leq i \leq n$,*

1. *the elements of $\mathcal{P}(T_i)$ are linearly ordered by the prefix order $\preceq$;*
2. *the set $\mathfrak{L}(T_i) \setminus \widehat{C}$ contains at most $|w_0| \cdot t$ distinct value restrictions, where t is the number of distinct value restrictions occurring in the left-hand sides of the GCIs in $\mathcal{T}$.*

Proof. 1. Since T_0 has only one leaf, it trivially satisfies the property required by the lemma. Now assume that $(\mathcal{P}(T_i), \preceq)$ is a totally ordered set. Let ℓ be the leaf of T_i that is expanded when going from T_i to T_{i+1}, $\forall w.A$ its value restriction label and p its prefix label, $k_1, \ldots, k_s$ the children of ℓ in T_{i+1}, and p' the prefix label of these children. It is easy to see that $\mathcal{P}(T_{i+1}) \subseteq \mathcal{P}(T_i) \cup \{p'\}$. In addition, by the definition of node expansion, both p and p' are prefixes of w, and thus $p \preceq p'$ or $p' \prec p$. Since p is of maximal length in $\mathcal{P}(T_i)$ and $\mathcal{P}(T_i)$ is linearly ordered w.r.t. $\preceq$, all the elements of $\mathcal{P}(T_i)$ are prefixes of p. Thus, independently of whether $p \preceq p'$ or $p' \prec p$, the set $\mathcal{P}(T_i) \cup \{p'\}$ is also linearly ordered w.r.t. $\preceq$.

2. The restriction to forward TBoxes implies that each value restriction $\forall w.A$ occurring in a derivation tree of $\forall w_0.A_0$ satisfies $|w| \leq |w_0|$. Since each prefix is a prefix of such a word w, this length restriction also holds for the prefixes. A linearly ordered set of prefixes of length at most $|w_0|$ can clearly have at most $|w_0|$ elements. Finally, if $\forall w.A$ is the value restriction label of a leaf ℓ with prefix label u, then $w = uv_i$ where $\forall v_i.A_i$ occurs on the left-hand sides of some GCI in $\mathcal{T}$. □

The second part of this lemma shows that representing such a value restriction set $\mathfrak{L}(T_i) \setminus \widehat{C}$ requires only polynomial space. In addition, there can be only exponentially many different such sets. In fact, we have seen that the value restrictions $\forall w.A$ occurring in these sets satisfy $|w| \leq |w_0|$. In addition, these words w contain only role names occurring in the input. Thus, there are only exponentially many value restrictions that can potentially occur in these sets, and consequently there are only exponentially many possibilities for choosing a polynomial number of them. Our NPSpace algorithm thus non-deterministically generates a sequence $S_0, S_1, S_2, \ldots$ of such sets reflecting a sequence $T_0 \to T_1 \to T_2 \to \ldots$ of derivation trees, and keeps only the most recent such set in memory. To solve the termination issue, we do not test for cycles (since this would require keeping all the generated sets in memory), but in each step increment an appropriate exponential counter (which needs only polynomial space), which stops the algorithm with failure when it overflows. In fact, such an overflow indicates that there must be a repetition in the sequence (i.e., $i < j$ such that $S_i = S_j$), and thus a shorter sequence would yield the same result.

[1]Strictly speaking, different leaves could be labeled with the same value restriction $\forall w.A$, but clearly we need to expand only one of them into a derivation tree for $C \sqsubseteq \forall w.A$.

Below we formally present this NPSPACE algorithm for deciding subsumption w.r.t. forward $\mathcal{FL}_0$ TBoxes, and prove its correctness. Recall that t denotes the number of distinct value restrictions occurring in the left-hand sides of the GCIs in $\mathcal{T}$.

Algorithm 1: Subsumption in $\mathcal{FL}_0$ with respect to forward TBoxes

```
   Input: Forward $\mathcal{FL}_0$ TBox $\mathcal{T}$, $\mathcal{FL}_0$ concept description $C$, and value restriction $\forall w_0.A_0$.
   Output: "yes" if $C \sqsubseteq_{\mathcal{T}} \forall w_0.A_0$, and "fail" otherwise.
1  if $\forall w_0.A_0 \in \widehat{C}$ then
2      return yes
3  end
4  $S := \{(\forall w_0.A_0, w_0)\}$;
5  $c := 0$   ($c$ is stored in binary);
6  $k := ((|\mathsf{N_R}|+1)\cdot t)^{|w_0|^2\cdot t}$   ($k$ is stored in binary);
7  while $S \neq \emptyset$ and $c \leq k$ do
8      non-deterministically choose $(\forall w.A, p) \in S$ with longest $p$;
9      non-deterministically choose a GCI $g : \forall v_1.A_1 \sqcap \cdots \sqcap \forall v_s.A_s \sqsubseteq \forall v.A$
               such that $w = p'v$ for some $p' \in \mathsf{N_R}^*$;
10     (fail if there is no such GCI);
11     $S := (S \setminus \{(\forall w.A, p)\}) \cup \{(\forall p'v_i.A_i, p') \mid i = 1,\ldots,s$ and $\forall p'v_i.A_i \notin \widehat{C}\}$;
12     $c := c + 1$;
13 end
14 return yes if $S = \emptyset$ and fail otherwise
```

Termination of the procedure is guaranteed due to the use of the counter c. By Lemma 28, the algorithm only needs polynomial space to store the set S. This shows that Algorithm 1 is a *terminating non-deterministic* PSPACE *procedure*. We now proceed to show that this procedure is sound and complete.

Lemma 29 (Soundness)**.** *If Algorithm 1 answers **yes**, then $C \sqsubseteq_{\mathcal{T}} \forall w_0.A_0$ holds.*

Proof. Assume that the algorithm has a successful run performing $n \geq 0$ iterations of the while loop. Let $S_0 = \{(\forall w_0.A_0, w_0)\}$ and $S_1, \ldots, S_n$ be the sets corresponding to S after the i^{th}-iteration of the while loop. The following claim can easily be proved by *induction on $n - i$*:

$$\text{For all } 0 \leq i \leq n \text{ and all } (\forall w.A, p) \in S_i \text{ we have } C \sqsubseteq_{\mathcal{T}} \forall w.A.$$

Since the algorithm answers ***yes***, we have $S_n = \emptyset$, and thus the base case is trivially true. The induction step is an easy consequence of the way the sets S_i are iteratively constructed. Thus, since $S_0 = \{(\forall w_0.A_0, w_0)\}$, we obtain that $C \sqsubseteq_{\mathcal{T}} \forall w_0.A_0$. □

In principle, our proof of completeness considers a sequence $T_0 \to T_1 \to \ldots \to T_n$ of derivation trees for $\forall w_0.A_0$ w.r.t. $\mathcal{T}$ such that T_0 has just one node labeled with $\forall w_0.A_0$ and $T_n \in \mathfrak{T}_{\mathcal{T},C}(\forall w_0.A_0)$. It then transforms this sequence into a sequence of sets $S_0, S_1, \ldots, S_n$ by considering the value restrictions not in $\widehat{C}$ labelling the leaves of the trees T_i (together with the prefix label of the respective leaf). Unfortunately, this sequence of sets need not always be a sequence of sets that can be produced by our algorithm. In fact, it may be the case that T_i contains several leaves with label $\forall w.A$ and maximal prefix u. If one of these leaves is expanded in the transition $T_i \to T_{i+1}$, then the others remain in T_{i+1}, and thus their label $\forall w.A$ and prefix u remains in S_{i+1}. However, one can assume without loss of generality that all such leaves are expanded simultaneously in the same way. Let us denote the transition relation on derivation trees obtained this way by $\to_s$. It is easy to see that Proposition 27 and Lemma 28 also hold with $\to_s$ in place of $\to$.

Lemma 30 (Completeness)**.** *If $C \sqsubseteq_\mathcal{T} \forall w_0.A_0$, then Algorithm 1 answers **yes**.*

Proof. Assume that $C \sqsubseteq_\mathcal{T} \forall w_0.A_0$. If $\forall w_0.A_0 \in \widehat{C}$, then Algorithm 1 immediately answers **yes**. Otherwise, by (the $\rightarrow_s$-variant of) Proposition 27, there exists a sequence $T_0 \rightarrow_s T_1 \rightarrow_s \ldots \rightarrow_s T_n$ of derivation trees for $\forall w_0.A_0$ w.r.t. $\mathcal{T}$ such that T_0 has just one node labeled with $\forall w_0.A_0$ and $T_n \in \mathfrak{T}_{\mathcal{T},C}(\forall w_0.A_0)$. We will now show how $T_0 \rightarrow_s T_1 \rightarrow_s \ldots \rightarrow_s T_n$ can be used to produce a successful run of the algorithm. Let us start by constructing a sequence of sets $S_0, S_1, \ldots, S_n$ as follows:

- $S_0 = \{(\forall w_0.A_0, w_0)\}$.
- For all $0 \leq i < n$, the set S_{i+1} is constructed from S_i as follows. Since $T_i \rightarrow_s T_{i+1}$, the derivation tree T_{i+1} is obtained from T_i by expanding one leaf ℓ_i of T_i with label $\forall w^i.A^i$ (or several such leaves) such that:
  - $\forall w^i.A^i \in \mathfrak{L}(T_i) \setminus \widehat{C}$ and $\mathfrak{d}(\ell_i) = p^i$ is of maximal length,
  - there exists a GCI g_i of the form $\forall v_1.A_1 \sqcap \ldots \sqcap \forall v_s.A_s \sqsubseteq \forall v.A^i$ such that $p'v = w^i$ for some $p' \in \mathsf{N_R}^*$ and $\forall p'v_j.A_j \in \mathfrak{L}(T_{i+1})$ for all $j, 1 \leq j \leq s$.

  Then $S_{i+1} := (S_i \setminus \{(\forall w^i.A^i, p^i)\}) \cup \{(\forall p'v_i.A_i, p') \mid i = 1, \ldots, s \text{ and } \forall p'v_i.A_i \notin \widehat{C}\}$.

Using induction on i, we can show the following for all $i, 0 \leq i \leq n$:

$(\forall w.A, p) \in S_i$ only if there is a leaf ℓ in T_i with label $\forall w.A \in \mathfrak{L}(T_i) \setminus \widehat{C}$ and $\mathfrak{d}(\ell) = p$.

Consequently, $\mathfrak{L}(T_n) \subseteq \widehat{C}$ implies $S_n = \emptyset$. Furthermore, notice that starting with $S = S_i$, an iteration of the while loop resulting in $S = S_{i+1}$ can be achieved by choosing $(\forall w^i.A^i, p^i)$ from S and g_i from $\mathcal{T}$. Hence, if $n \leq k$, the sequence $S_0, \ldots, S_n$ yields a successful run of the algorithm on input $\mathcal{T}$, C and $\forall w_0.A_0$.

In case $n > k$, the counter c would overflow leading to a failing run of the algorithm. However, $S_0, \ldots, S_n$ can be transformed into a sufficiently short sequence $S_0, S_{j_1}, \ldots, S_{j_m}$ inducing a successful run. This is an easy consequence of the following argument. Assuming there are two indices $0 \leq i_1 < i_2 \leq n$ such that $S_{i_1} = S_{i_2}$, we can transform $S_0, \ldots, S_n$ into a shorter sequence $S_0, \ldots, S_{i_1}, S_{i_2+1}, \ldots, S_n$ satisfying the same properties described above for $S_0, \ldots, S_n$. Iteratively applying this argument will result in a sequence $S_0, S_{j_1}, \ldots, S_{j_m}$, where $1 \leq j_i \leq n$ for all $1 \leq i \leq m$ and $S_{j_{i_1}} \neq S_{j_{i_2}}$ for all $0 \leq i_1 < i_2 \leq m$. As seen in Lemma 28, each set S_i contains at most $|w_0| \cdot t$ elements. In addition, there are at most $(|\mathsf{N_R}|+1)^{|w_0|} \cdot t$ many different elements a set S_i can contain.[2] This means that there are at most $((|\mathsf{N_R}|+1)^{|w_0|} \cdot t)^{|w_0| \cdot t} \leq ((|\mathsf{N_R}|+1) \cdot t)^{|w_0|^2 \cdot t}$ different such sets. Hence, $m \leq k$ and thus $S_0, S_{j_1}, \ldots, S_{j_m}$ corresponds to a successful run of Algorithm 1 on input $\mathcal{T}$, C and $\forall w_0.A_0$. □

Using the fact that PSpace = NPSpace, we thus have shown the desired PSpace upper bound for subsumption.

Basically the same algorithm can also be used to handle *backward TBoxes*, which consist of GCIs of the form (4) where $|v_i| \geq |v|$ for all $i, 1 \leq i \leq s$. While for such TBoxes expansion of leaves may increase the length of value restrictions, one can stop with failure whenever a value restriction is generated that is longer than the longest value restriction in $\widehat{C}$.

Theorem 31. *Subsumption in $\mathcal{FL}_0$ w.r.t. forward (backward) TBoxes is in* PSpace.

[2] Note that $(|\mathsf{N_R}|+1)^{|w_0|}$ is an over-approximation of the number of words of length at most $|w_0|$ over the alphabet $\mathsf{N_R}$.

Matching in $\mathcal{FL}_0$ w.r.t. forward TBoxes

We now adapt the NPSPACE subsumption algorithm for forward TBoxes introduced above to the problem of deciding whether a matching problem has an $\mathcal{FL}_0$ matcher. Consider a matching problem of the form (2). According to Lemma 19, we need to check the two conditions stated in this lemma. The first condition is a subsumption test w.r.t. the forward TBox $\mathcal{T}$, and can thus be performed in PSPACE. Regarding the second test, we add elements to the finite language $L_{i,j}$ on the fly while performing the subsumption test $E \sqcap \forall L_1.\sigma(X_1) \sqcap \ldots \sqcap \forall L_m.\sigma(X_m) \sqsubseteq_{\mathcal{T}} C$.

Basically, we run the NPSPACE subsumption algorithm on $E \sqsubseteq \forall w_0.A_0$ for every value restriction $\forall w_0.A_0$ in $\widehat{C}$. But we now have two conditions under which a leaf with label $\forall w.A_j$ need not be expanded: either $\forall w.A_j \in \widehat{E}$, or there exists $i, 1 \leq i \leq m$, $w_1 \in L_i$, and $w_2 \in \widehat{L}_{i,j} = \bigcap_{u \in L_i} u^{-1}\mathcal{L}_{\mathcal{T}}(C, A_j)$ with $w = w_1 w_2$. Checking the second condition requires only polynomial space. In fact, there are only polynomially many pairs w_1, w_2 to be considered. For each of them, we need to check for all $u \in L_i$ whether $uw_2 \in \mathcal{L}_{\mathcal{T}}(C, A_j)$. Each of these tests is a subsumption test $C \sqsubseteq_{\mathcal{T}} \forall uw_2.A_j$, which needs only polynomial space. Note that for backward TBoxes $\mathcal{T}$, the languages $\mathcal{L}_{\mathcal{T}}(C, A_j)$ are actually finite and contain only words not longer than the longest value restriction in $\widehat{C}$.

Algorithm 2 below formally describes the decision procedure sketched above to solve the matching problem w.r.t. forward TBoxes.

Algorithm 2: Matching in $\mathcal{FL}_0$ with respect to forward TBoxes

```
Input: A forward $\mathcal{FL}_0$ TBox $\mathcal{T}$ and a matching problem of the form (2).
Output: "yes" if $C \equiv^?_{\mathcal{T}} E \sqcap \forall L_1.X_1 \sqcap \ldots \sqcap \forall L_m.X_m$ has an $\mathcal{FL}_0$ matcher, and "fail" otherwise.
if $C \not\sqsubseteq_{\mathcal{T}} E$ then
    fail
end
foreach $\forall w_0.A_0 \in \widehat{C} \setminus \widehat{E}$ do
    $S := \{(\forall w_0.A_0, w_0)\}$;
    $c := 0$   ($c$ is stored in binary);
    $k := ((|\mathsf{N_R}| + 1) \cdot t)^{|w_0|^2 \cdot t}$   ($k$ is stored in binary);
    while $S \neq \emptyset$ and $c \leq k$ do
        non-deterministically choose $(\forall w.A_j, p) \in S$ with longest $p$;
        if there exists $i, 1 \leq i \leq m$, $w_1 \in L_i$, $w_2 \in \widehat{L}_{i,j} = \bigcap_{u \in L_i} u^{-1}\mathcal{L}_{\mathcal{T}}(C, A_j)$ with $w = w_1 w_2$ then
            $S := S \setminus \{(\forall w.A_j, p)\}$
        else
            non-deterministically choose a GCI $g : \forall v_1.A_1 \sqcap \cdots \sqcap \forall v_s.A_s \sqsubseteq \forall v.A$
                    such that $w = p'v$ for some $p' \in \mathsf{N_R}^*$;
            (fail if there is no such GCI);
            $S := (S \setminus \{(\forall w.A, p)\}) \cup \{(\forall p'v_i.A_i, p') \mid i = 1, \ldots, s$ and $\forall p'v_i.A_i \notin \widehat{E}\}$;
        end
    end
    fail if $S \neq \emptyset$
end
return yes
```

Termination of the procedure is guaranteed because of the counter c and the fact that there are finitely many value restrictions in $\widehat{C}$. As argued above, only polynomial space is required to perform all the checks involved. Hence, Algorithm 2 is a terminating non-deterministic PSPACE procedure.

Lemma 32 (Soundness). *If Algorithm 2 answers* ***yes****, then $C \equiv^?_{\mathcal{T}} E \sqcap \forall L_1.X_1 \sqcap \ldots \sqcap \forall L_m.X_m$ has an $\mathcal{FL}_0$ matcher.*

Proof. For all $1 \leq i \leq m$ and $1 \leq j \leq k$ we define the set $L_{i,j} \subseteq \mathsf{N_R}^*$ as follows:

$$L_{i,j} := \{w_2 \mid \text{the check in line 10 was invoked for } w_2 \text{ and succeeded}\}.$$

Clearly, $L_{i,j} \subseteq \widehat{L}_{i,j}$ and the sets $L_{i,j}$ are finite since line 10 is reached only finitely often during a successful run of the procedure. Using these sets, we define the substitution $\sigma(X_i) = \bigsqcap_{j=1}^{k} \forall L_{i,j}.A_j,\ 1 \leq i \leq m$. We use Lemma 19 to show that σ is a solution of the matching problem. Since the algorithm did not fail, we know that $C \sqsubseteq_{\mathcal{T}} E$. Hence, it remains to show that $E \sqcap \forall L_1.\sigma(X_1) \sqcap \ldots \sqcap \forall L_m.\sigma(X_m) \sqsubseteq_{\mathcal{T}} C$, i.e., $E \sqcap \forall L_1.\sigma(X_1) \sqcap \ldots \sqcap \forall L_m.\sigma(X_m) \sqsubseteq_{\mathcal{T}} \forall w_0.A_0$ for each $\forall w_0.A_0 \in \widehat{C}$. If $\forall w_0.A_0 \in \widehat{E}$, then this subsumption holds trivially.

If $\forall w_0.A_0 \in \widehat{C} \setminus \widehat{E}$, then let $S_0, \ldots, S_n$ be the sequence of sets S computed by the corresponding iteration of the while loop. Similarly to the proof of Lemma 29, induction can be used to show that $(\forall w.A, p) \in S_i$ implies $E \sqcap \forall L_1.\sigma(X_1) \sqcap \ldots \sqcap \forall L_m.\sigma(X_m) \sqsubseteq_{\mathcal{T}} \forall w.A$. Since $S_0 = \{(\forall w_0.A_0, w_0)\}$, this implies $E \sqcap \forall L_1.\sigma(X_1) \sqcap \ldots \sqcap \forall L_m.\sigma(X_m) \sqsubseteq_{\mathcal{T}} \forall w_0.A_0$. Thus, we can conclude that $E \sqcap \forall L_1.\sigma(X_1) \sqcap \ldots \sqcap \forall L_m.\sigma(X_m) \sqsubseteq_{\mathcal{T}} C$. □

Lemma 33 (Completeness)**.** *If $C \equiv^?_{\mathcal{T}} E \sqcap \forall L_1.X_1 \sqcap \ldots \sqcap \forall L_m.X_m$ has an $\mathcal{FL}_0$ matcher, then Algorithm 2 answers **yes**.*

Proof. By Lemma 19, we know that $C \sqsubseteq_{\mathcal{T}} E$ and that there are finite languages $L_{i,j} \subseteq \widehat{L}_{i,j}$ such that $E \sqcap \forall L_1.\sigma(X_1) \sqcap \ldots \sqcap \forall L_m.\sigma(X_m) \sqsubseteq_{\mathcal{T}} C$, where σ is defined as in (3). The first condition guarantees that the test in line 1 does not fail. From the second one, we know that $E \sqcap \forall L_1.\sigma(X_1) \sqcap \ldots \sqcap \forall L_m.\sigma(X_m) \sqsubseteq_{\mathcal{T}} \forall w_0.A_0$ for all $\forall w_0.A_0 \in \widehat{C}$. By applying distributivity of value restrictions over conjunction, $E \sqcap \forall L_1.\sigma(X_1) \sqcap \ldots \sqcap \forall L_m.\sigma(X_m)$ can be transformed into an equivalent concept description D that is a conjunction of value restrictions. Observe that $\forall w.A_j \in \widehat{D}$ iff either $\forall w.A_j \in \widehat{E}$, or there exist $1 \leq i \leq m$ and $w_1, w_2 \in \mathsf{N_R}^*$ such that $w = w_1 w_2$, $w_1 \in L_i$ and $w_2 \in L_{i,j} \subseteq \widehat{L}_{i,j}$. By Lemma 30, $D \sqsubseteq_{\mathcal{T}} \forall w_0.A_0$ implies that there is a successful run of Algorithm 1 for the corresponding subsumption query. Due to the above observation, this is also a successful run of Algorithm 2 for the matching problem $C \equiv^?_{\mathcal{T}} E \sqcap \forall L_1.X_1 \sqcap \ldots \sqcap \forall L_m.X_m$. □

Again, backward TBoxes can be treated similarly.

Theorem 34. *Matching in $\mathcal{FL}_0$ w.r.t. forward (backward) TBoxes is in* PSPACE.

6 Conclusion

We have shown in this paper that matching in $\mathcal{FL}_0$ w.r.t. TBoxes is in EXPTIME, thus complementing the positive results for matching w.r.t. TBoxes in $\mathcal{EL}$ [7]. This is the best possible complexity for matching in this setting since already the subsumption problem is EXPTIME-hard. We have also shown that the complexity of subsumption and matching can be lowered to PSPACE if restricted kinds of TBoxes are considered. Unfortunately, until now we could not show a matching PSPACE lower bound, but we believe that for forward TBoxes these problems are indeed PSPACE-complete.

The big open problem in this area is unification in $\mathcal{FL}_0$ w.r.t. TBoxes, for which nothing is known. We actually conjecture that this problem is undecidable. For $\mathcal{EL}$, decidability of unification w.r.t. TBoxes is also an open problem, but there are positive results for TBoxes satisfying certain restrictions on cyclic dependencies [1]. It would be interesting to see whether this restriction or the restrictions we imposed in Section 5 of the present paper can lead to positive results for unification in $\mathcal{FL}_0$ w.r.t. TBoxes.

References

[1] Franz Baader, Stefan Borgwardt, and Barbara Morawska. Extending unification in $\mathcal{EL}$ towards general TBoxes. In *Proc. of the 13th Int. Conf. on Principles of Knowledge Representation and Reasoning (KR 2012)*, pages 568–572. AAAI Press/The MIT Press, 2012.

[2] Franz Baader, Sebastian Brandt, and Carsten Lutz. Pushing the $\mathcal{EL}$ envelope. In Leslie Pack Kaelbling and Alessandro Saffiotti, editors, *Proc. of the 19th Int. Joint Conf. on Artificial Intelligence (IJCAI 2005)*, pages 364–369, Edinburgh (UK), 2005. Morgan Kaufmann, Los Altos.

[3] Franz Baader, Oliver Fernández Gil, and Maximilian Pensel. Standard and non-standard inferences in the description logic $\mathcal{FL}_0$ using tree automata. In Daniel Lee, Alexander Steen, and Toby Walsh, editors, *GCAI 2018, 4th Global Conference on Artificial Intelligence*, volume 55 of *EPiC Series in Computing*, pages 1–14. EasyChair, 2018.

[4] Franz Baader and Ralf Küsters. Matching in description logics with existential restrictions. In *Proc. of the 7th Int. Conf. on Principles of Knowledge Representation and Reasoning (KR 2000)*, pages 261–272, 2000.

[5] Franz Baader and Ralf Küsters. Unification in a description logic with transitive closure of roles. In *Proc. of the 8th Int. Conf. on Logic for Programming and Automated Reasoning (LPAR 2001)*, volume 2250 of *Lecture Notes in Computer Science*, pages 217–232. Springer, 2001.

[6] Franz Baader and Barbara Morawska. Unification in the description logic $\mathcal{EL}$. *Logical Methods in Computer Science*, 6(3), 2010.

[7] Franz Baader and Barbara Morawska. Matching with respect to general concept inclusions in the description logic $\mathcal{EL}$. In Carsten Lutz and Michael Thielscher, editors, *Proc. of the 37th German Annual Conf. on Artificial Intelligence (KI'14)*, volume 8736 of *Lecture Notes in Computer Science*, pages 135–146. Springer, 2014.

[8] Franz Baader and Paliath Narendran. Unification of Concept Terms in Description Logics. *J. Symb. Comput.*, 31(3):277–305, 2001.

[9] Alexander Borgida, Ronald J. Brachman, Deborah L. McGuinness, and Lori Alperin Resnick. CLASSIC: A structural data model for objects. In *Proc. of the ACM SIGMOD Int. Conf. on Management of Data*, pages 59–67, 1989.

[10] Alexander Borgida and Ralf Küsters. What's not in a name? Initial explorations of a structural approach to integrating large concept knowledge-bases. Technical Report DCS-TR-391, Rutgers University, 1999.

[11] Alexander Borgida and Deborah L. McGuinness. Asking queries about frames. In *Proc. of the 5th Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR'96)*, pages 340–349, 1996.

[12] David Harel and Rivi Sherman. Propositional dynamic logic of flowcharts. *Information and Control*, 64(1-3):119–135, 1985.

[13] Hector J. Levesque and Ron J. Brachman. Expressiveness and tractability in knowledge representation and reasoning. *Computational Intelligence*, 3:78–93, 1987.

[14] Maximilian Pensel. An automata based approach for subsumption w.r.t. general concept inclusions in the description logic $\mathcal{FL}_0$. Master's thesis, Chair for Automata Theory, TU Dresden, Germany. See http://lat.inf.tu-dresden.de/research/mas., 2015.

[15] V. R. Pratt. A near-optimal method for reasoning about action. *J. of Computer and System Sciences*, 20:231–255, 1980.

[16] Klaus Schild. A correspondence theory for terminological logics: Preliminary report. In *Proc. of the 12th Int. Joint Conf. on Artificial Intelligence (IJCAI'91)*, pages 466–471. Morgan Kaufmann, 1991.