LTCS–Report

Restricted Unification in the DL $\mathcal{FL}_0$ (Extended Version)

Franz Baader Oliver Fernández Gil Maryam Rostamigiv

LTCS-Report 21-02

Restricted Unification in the DL $\mathcal{FL}_0$

Franz Baader[1], Oliver Fernández Gil[1], and Maryam Rostamigiv[2]

[1] Theoretical Computer Science, TU Dresden, Dresden, Germany
franz.baader@tu-dresden.de, oliver.fernandez@tu-dresden.de
[2] Département d'Informatique, Paul Sabatier University, Toulouse, France
Maryam.Rostamigiv@irit.fr

Abstract. Unification in the Description Logic (DL) $\mathcal{FL}_0$ is known to be ExpTime-complete, and of unification type zero. We investigate in this paper whether a lower complexity of the unification problem can be achieved by either syntactically restricting the role depth of concepts or semantically restricting the length of role paths in interpretations. We show that the answer to this question depends on whether the number formulating such a restriction is encoded in unary or binary: for unary coding, the complexity drops from ExpTime to PSpace. As an auxiliary result, which is however also of interest in its own right, we prove a PSpace-completeness result for a depth-restricted version of the intersection emptiness problem for deterministic root-to-frontier tree automata. Finally, we show that the unification type of $\mathcal{FL}_0$ improves from type zero to unitary (finitary) for unification without (with) constants in the restricted setting.

1 Introduction

Unification of concept patterns has been proposed as an inference service in Description Logics that can, for example, be used to detect redundancies in ontologies. For the DL $\mathcal{FL}_0$, which has the concept constructors conjunction ($\sqcap$), value restriction ($\forall r.C$), and top concept ($\top$), unification was investigated in detail in [5]. It was shown there that unification in $\mathcal{FL}_0$ corresponds to unification modulo the equational theory *ACUIh* since (modulo equivalence) conjunction is associative (A), commutative (C), idempotent (I) and has top as a unit (U), and value restrictions behave like homomorphisms for conjunction and top (h). For this equational theory, it had already been shown in [1] that it has unification type zero, which means that a solvable unification problem need not have a minimal complete set of unifiers, and thus in particular not a finite one. From the DL point of view, the decision problem is, however, more interesting than the unification type. Since *ACUIh* is a commutative/monoidal theory [1,14], solvability of *ACUIh* unification problems (and thus of unification problems in $\mathcal{FL}_0$) can be reduced to solvability of systems of linear equations in a certain semiring, which for the case of *ACUIh* consists of finite languages over a finite alphabet, with union as semiring addition and concatenation as semiring multiplication [5]. By a reduction to the emptiness problem for root-to-frontier tree automata (RFAs), it was then shown in [5] that solvability of the language equations corresponding to an $\mathcal{FL}_0$ unification problem can be decided in exponential time. In addition, ExpTime-hardness of this problem is proved in [5] by a reduction from the intersection emptiness problem for deterministic RFAs (DRFAs) [16].

In the present paper, we investigate two kinds of restrictions on unification in $\mathcal{FL}_0$. On the one hand, we *syntactically restrict the role depth* (i.e., the maximal nesting of value restrictions) in the concepts obtained by applying a unifier to be bounded by a natural number $k \geq 1$. This restriction was motivated by a similar restriction used in research on least common subsumers (lcs) [15], where imposing a bound on the role depth guarantees existence of the lcs also in the presence of a (possibly cyclic) terminology. Also note that such a restriction was used in [11] for the theory *ACh*, for which unification is known to be undecidable [13]. It is shown in [11] that the problem becomes decidable if a bound on the maximal nesting of applications of homomorphisms is imposed. On the other hand, we consider a *semantic restriction* where, when defining the semantics of concepts, only interpretations for which the length of role paths is bounded by a given number k are considered. A similar restriction (for $k = 1$) was employed in [8] to improve the unification type for the modal logic $\mathbf{K}$ from type zero [10] to unitary or finitary for $\mathbf{K} + \Box\Box\bot$.

In the present paper we show that both the syntactic and the semantic restriction ensures that the unification type of $\mathcal{FL}_0$ (and equivalently, of the theory *ACUIh*) improves from type zero to unitary for unification without constants and finitary for unification with constants. Regarding the decision problem, we can show that the complexity depends on whether the bound k is assumed to be encoded in unary or binary. For binary encoding of k, the complexity stays ExpTime, whereas for unary coding it drops from ExpTime to PSpace. This is again the case both for the syntactic and the semantic restriction. As an auxiliary result we prove that a depth-restricted variant of the intersection emptiness for DRFAs is PSpace-complete. Showing these results requires combining methods and results from knowledge representation, unification theory, and automata theory.

2 The DL $\mathcal{FL}_0$ and Restrictions

Starting with mutually disjoint countably infinite sets N_C and N_R of concept and role names, respectively, the set of $\mathcal{FL}_0$ concepts is inductively defined as follows:

- $\top$ (top concept) and every concept name $A \in N_C$ is an $\mathcal{FL}_0$ concept,
- if C, D are $\mathcal{FL}_0$ concepts and $r \in N_R$ is a role name, then $C \sqcap D$ (conjunction) and $\forall r.C$ (value restriction) are $\mathcal{FL}_0$ concepts.

The *semantics* of $\mathcal{FL}_0$ concepts is defined using first-order interpretations $\mathcal{I} = (dom^{\mathcal{I}}, \cdot^{\mathcal{I}})$ consisting of a non-empty domain $dom^{\mathcal{I}}$ and an interpretation function $\cdot^{\mathcal{I}}$ that assigns a set $A^{\mathcal{I}} \subseteq dom^{\mathcal{I}}$ to each concept name A, and a binary relation $r^{\mathcal{I}} \subseteq dom^{\mathcal{I}} \times dom^{\mathcal{I}}$ to each role name r. This function is extended to $\mathcal{FL}_0$ concepts as follows:

$$\top^{\mathcal{I}} = dom^{\mathcal{I}} \text{ and } (C \sqcap D)^{\mathcal{I}} = C^{\mathcal{I}} \cap D^{\mathcal{I}},$$
$$(\forall r.C)^{\mathcal{I}} = \{x \in dom^{\mathcal{I}} \mid \forall y \in dom^{\mathcal{I}} \colon (x, y) \in r^{\mathcal{I}} \Rightarrow y \in C^{\mathcal{I}}\}.$$

Given two $\mathcal{FL}_0$ concepts C and D, we say that C is subsumed by D (written $C \sqsubseteq D$) if $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ holds for all interpretations $\mathcal{I}$, and that C is equivalent to D (written $C \equiv D$) if $C \sqsubseteq D$ and $D \sqsubseteq C$. It is well known that subsumption (and thus also equivalence) of $\mathcal{FL}_0$ concepts can be decided in polynomial time [12].

Note that, up to equivalence, conjunction is associative, commutative, and idempotent, and has the unit element $\top$. In addition, the following equivalences hold for value restrictions: $\forall r.\top \equiv \top$ and $\forall r.(C \sqcap D) \equiv \forall r.C \sqcap \forall r.D$. As an easy consequence we obtain that all $\mathcal{FL}_0$ concepts $C \not\equiv \top$ are equivalent to an $\mathcal{FL}_0$ concept C' not containing any occurrences of $\top$. Thus, we can assume without loss of generality that $\mathcal{FL}_0$ concepts different from $\top$ do not contain any occurrences of $\top$. We will do this in the rest of the paper.

Due to the above equivalences, one can transform $\mathcal{FL}_0$ concepts into a normal form that uses formal languages over the alphabet of role names to represent value restrictions that end with the same concept name. In fact, using these equivalences as rewrite rules from left to right, every $\mathcal{FL}_0$ concept can be transformed into an equivalent one that is either $\top$ or a conjunction of concepts of the form $\forall r_1. \cdots \forall r_n.A$, where $r_1, \ldots, r_n$ are role names and A is a concept name. Such a concept can be abbreviated as $\forall w.A$, where $w = r_1 \ldots r_n$ is a word over the alphabet

N_R. Note that $n = 0$ means that w is the empty word ε, and thus $\forall\varepsilon.A$ corresponds to A. Furthermore, a conjunction of the form $\forall w_1.A \sqcap \ldots \sqcap \forall w_m.A$ can be written as $\forall L.A$ where $L \subseteq N_R^*$ is the finite language $\{w_1, \ldots, w_m\}$. We use the convention that $\forall\emptyset.A$ corresponds to the top concept $\top$. Thus, any two $\mathcal{FL}_0$ concepts C, D containing only the concept names $A_1, \ldots, A_\ell$ can be represented as

$$C \equiv \forall K_1.A_1 \sqcap \ldots \sqcap \forall K_\ell.A_\ell \quad \text{and} \quad D \equiv \forall L_1.A_1 \sqcap \ldots \sqcap \forall L_\ell.A_\ell, \tag{1}$$

where $K_1, L_1, \ldots, K_\ell, L_\ell$ are finite languages over the alphabet of role names N_R. We call this representation the *language normal form (LNF)* of C, D. If C, D have the LNF shown in (1), then $C \equiv D$ holds iff $L_1 = K_1, \ldots, L_\ell = K_\ell$ (see Lemma 4.2 of [5]).

2.1 Syntactically Restricting the Role Depth

The role depth of an $\mathcal{FL}_0$ concept is the maximal nesting of value restrictions in this concept. To be more precise, we define the role depth $rd(C)$ of an $\mathcal{FL}_0$ concept C by induction:

- $rd(\top) = rd(A) = 0$ for all $A \in N_C$,
- $rd(C \sqcap D) = \max(rd(C), rd(D))$ and $rd(\forall r.C) = 1 + rd(C)$.

The role depth of $\mathcal{FL}_0$ concepts is preserved under equivalence.

Lemma 1. *Let C, D be $\mathcal{FL}_0$ concepts equal $\top$ or not containing any occurrences of $\top$. Then $C \equiv D$ implies $rd(C) = rd(D)$.*

This is an immediate consequence of the LNF-based characterization of equivalence for $\mathcal{FL}_0$ concepts.

We are now ready to define our first restricted version of subsumption and equivalence in $\mathcal{FL}_0$. For an integer $k \geq 1$ and $\mathcal{FL}_0$ concepts C and D (satisfying the restrictions formulated in Lemma 1), we define subsumption and equivalence restricted to concepts of role depth $\leq k$ as follows:

- $C \sqsubseteq^k_{syn} D$ if $C \sqsubseteq D$ and $rd(C) \leq k$ as well as $rd(D) \leq k$,
- $C \equiv^k_{syn} D$ if $C \sqsubseteq^k_{syn} D$ and $D \sqsubseteq^k_{syn} C$.

The effect of this definition is that subsumption and equivalence can only hold for concepts that satisfy the restriction of the role depth by k. For concepts satisfying this syntactic restriction, the relations $\sqsubseteq^k_{syn}$ and $\equiv^k_{syn}$ coincide with the classical subsumption and equivalence relations on $\mathcal{FL}_0$ concepts. Using the language normal form of $\mathcal{FL}_0$ concepts, the equivalence $\equiv^k_{syn}$ can be characterized as follows: if C, D have the LNF shown in (1), then

$$C \equiv^k_{syn} D \;\text{ iff }\; L_1 = K_1 \subseteq N_R^{\leq k},\ \ldots,\ L_\ell = K_\ell \subseteq N_R^{\leq k},$$

where $N_R^{\leq k}$ denotes the set of words over N_R of length at most k.

2.2 Semantically Restricting the Length of Role Paths

For an integer $n \geq 1$ and a given interpretation $\mathcal{I} = (dom^{\mathcal{I}}, \cdot^{\mathcal{I}})$, a role path of length n is a sequence $d_0, r_1, d_1, \ldots, d_{n-1}, r_n, d_n$, where $d_0, \ldots, d_n$ are elements of $dom^{\mathcal{I}}$, $r_1, \ldots, r_n$ are role names, and $(d_{i-1}, d_i) \in r_i^{\mathcal{I}}$ holds for all $i = 1, \dots, n$. The interpretation $\mathcal{I}$ is called k-restricted if it does not admit any role paths of length $> k$.

For an integer $k \geq 1$ and $\mathcal{FL}_0$ concepts C and D, we define subsumption and equivalence restricted to interpretations with role paths of length $\leq k$ as follows:

- $C \sqsubseteq^k_{sem} D$ if $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ holds for all k-restricted interpretations $\mathcal{I}$,
- $C \equiv^k_{sem} D$ if $C \sqsubseteq^k_{sem} D$ and $D \sqsubseteq^k_{sem} C$.

The effect of this notion of equivalence is that all concepts occurring at a role depth $> k$ can be replaced by $\top$. To be more precise, we define the restriction of a concept C to role depth $n \geq 0$ by induction on n as follows:

- $A|_n = A$ for $A \in N_C \cup \{\top\}$ and $(C \sqcap D)|_n = C|_n \sqcap D|_n$ for all $n \geq 0$;
- $(\forall r.C)|_0 = \top$ and $(\forall r.C)|_n = \forall r.(C|_{n-1})$ for all $n \geq 1$.

For example, $(\forall r.\forall r.\forall r.A)|_4 = \forall r.\forall r.\forall r.A = (\forall r.\forall r.\forall r.A)|_3$ and $(\forall r.\forall r.\forall r.A)|_2 = \forall r.\forall r.\top \equiv \top$. In the language normal form, restricting to role depth n means that all words that are longer than n can simply be removed.

The following lemma is an easy consequence of the definition of k-restricted interpretations, the semantics of value restrictions, and bisimulation-invariance of $\mathcal{FL}_0$ concepts [3].

Lemma 2. *Let C, D be $\mathcal{FL}_0$ concept. Then*

1. *$C \equiv^k_{sem} C|_k$ and $D \equiv^k_{sem} D|_k$;*
2. *$C|_k \equiv^k_{sem} D|_k$ iff $C|_k \equiv D|_k$;*
3. *$C \equiv^k_{sem} D$ iff $C|_k \equiv D|_k$.*

The third statement in this lemma yields the following characterization of the equivalence $\equiv^k_{sem}$: if C, D have the LNF shown in (1), then

$$C \equiv^k_{sem} D \;\text{ iff }\; L_1 \cap N_R^{\leq k} = K_1 \cap N_R^{\leq k},\; \dots,\; L_\ell \cap N_R^{\leq k} = K_\ell \cap N_R^{\leq k}.$$

The corresponding equational theory It was shown in [5] that equivalence of $\mathcal{FL}_0$ concepts can be axiomatized by the equational theory

$$\begin{aligned} ACUIh := \{\ & (x \wedge y) \wedge z = x \wedge (y \wedge z),\, x \wedge y = y \wedge x,\, x \wedge x = x,\, x \wedge 1 = x \ \} \\ \cup \{\ & h_r(x \wedge y) = h_r(x) \wedge h_r(y),\, h_r(1) = 1 \mid r \in N_R \ \}, \end{aligned}$$

where $\wedge$, h_r, and 1 in the terms respectively correspond to $\sqcap$, $\forall r.$, and $\top$ in the concepts. These identities say that $\wedge$ is associative (A), commutative (C), and idempotent (I) with unit 1 (U), and that the unary function symbols behave like homomorphisms (h) for $\wedge$ and 1.

The equivalence $\equiv^k_{sem}$ can be axiomatized by adding identities that say that nesting of homomorphisms of depth $> k$ produces the unit. Given a word $u = r_1 r_2 \dots r_n \in N_R^*$, we denote a term of the form $h_{r_1}(h_{r_2}(\cdots h_{r_n}(t) \cdots))$ as $h_u(t)$. It is now easy to see that $\equiv^k_{sem}$ is axiomatized by

$$ACUIh^k := ACUIh \cup \{h_u(x) = 1 \mid u \in N_R^* \text{ with } |u| = k+1\}.$$

Note that, due to the identity $h_r(1) = 1$ in $ACUIh$, we have $h_u(x) =_{ACUIh^k} 1$ also for all $u \in N_R^*$ with $|u| > k+1$.

3 Unification in $\mathcal{FL}_0$

In unification, we consider concepts that may contain variables, which can be replaced by concepts. More formally, we introduce a countably infinite set N_V of concept variables, which is disjoint with N_C and N_R. An $\mathcal{FL}_0$ concept pattern is an $\mathcal{FL}_0$ concept that is constructed using $N_C \cup N_V$ as concept names. The semantics of concept patterns is defined as for concepts, i.e., concept variables are treated like concept names when defining the semantics. This way, the notions of subsumption and equivalence (both in the restricted and in the unrestricted setting) transfer from concepts to concept patterns in the obvious way.

A substitution σ is a mapping from N_X into the set of all $\mathcal{FL}_0$ concept patterns such that $dom(\sigma) := \{X \in N_V \mid \sigma(X) \neq X\}$ is finite. This mapping is extended to concept patterns in the obvious ways:

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

An $\mathcal{FL}_0$ *unification problem* is an equation of the form $C \overset{?}{\equiv} D$ where C, D are $\mathcal{FL}_0$ concept patterns. A unifier of this equation is a substitution σ such that $\sigma(C) \equiv \sigma(D)$.

Obviously, when looking for unifiers of a given equation $C \overset{?}{\equiv} D$, we can restrict the attention to substitutions whose domain contains only variables occurring in C or D. In addition, it is well-known in unification theory [7] that a unification problem has a unifier iff it has a ground unifier, i.e., a unifier σ such that $\sigma(X)$ does not contain concept variables for all $X \in dom(\sigma)$. More precisely, if one is interested only in whether a unifier exists or not, then one can restrict the attention to substitutions σ such that, for all $X \in dom(\sigma)$, we have that $\sigma(X)$ is a concept built using only the concept names and role names occurring in C, D (possibly extended with a new concept name A in case C, D do not contain any concept names).

It was shown in [5] that the question of whether a given $\mathcal{FL}_0$ unification problem has a unifier or not can be reduced to solving linear language equations, i.e., equations of the form

$$S_0 \cup S_1{\cdot}X_1 \cup \cdots \cup S_n{\cdot}X_n = T_0 \cup T_1{\cdot}X_1 \cup \cdots \cup T_n{\cdot}X_n, \tag{2}$$

where $S_0, \ldots, S_n, T_0, \ldots, T_n$ are finite languages of words over an alphabet $\Delta = \{1, \ldots, \rho\}$[3] and $X_1, \ldots, X_n$ are variables that can be replaced by finite languages over Δ. A solution of the equation (2) is an assignment θ of finite languages $\theta(X_i)$ to the variables X_i (for $i = 1, \ldots, n$) such that

$$S_0 \cup S_1{\cdot}\theta(X_1) \cup \cdots \cup S_n{\cdot}\theta(X_n) = T_0 \cup T_1{\cdot}\theta(X_1) \cup \cdots \cup T_n{\cdot}\theta(X_n), \tag{3}$$

where $\cup$ is interpreted as union and $\cdot$ as concatenation of languages. Strictly speaking, a given $\mathcal{FL}_0$ unification problem yields one such language equation for every concept name occurring in the problem. But since these equations do not share variables, they can be solved separately. Also note that solvability of language equations of the form (2) can in turn be reduced in polynomial time to $\mathcal{FL}_0$ unification.

A word $w = i_1 \ldots i_\ell$ occurring in a solution of the form (3) of the equation (2) corresponds to a conjunct $\forall r_{i_1}. \cdots \forall r_{i_\ell}.A$ in the unified concept $\sigma(C) \equiv \sigma(D)$. Thus, the length of the word w is equal to the role depth of the corresponding sequence of value restrictions.

Example 1. Consider the $\mathcal{FL}_0$ unification problem $\forall r_1.\forall r_1.A \sqcap \forall r_1.\forall r_1.X \overset{?}{\equiv} X \sqcap \forall r_1.\forall r_1.\forall r_1.Y$. The substitution σ with $\sigma(X) = \forall r_1.\forall r_1.A$ and $\sigma(Y) = \forall r_1.A$ is one of the unifiers of this

[3] Intuitively, ρ is the number of different role names occurring in the unification problem and each letter $i, 1 \leq i \leq \rho$, stands for a role name r_i.

problem. The language equation induced by this unification problem is $\{11\} \cup \{11\} \cdot X = \{\varepsilon\} \cdot X \cup \{111\} \cdot Y$. The unifier σ corresponds to the following solution θ of this problem: $\theta(X) = \{11\}$ and $\theta(Y) = \{1\}$.

Using the corresponding equational theory In [5] the reduction of $\mathcal{FL}_0$ unification to solving linear language equations is shown both by a direct reduction and by using the fact that $ACUIh$ is a commutative/monoidal theory [1,14,6]. It was shown in [1,14] that unification in such theories can be reduced to solving linear equations over a corresponding semiring. In the case of $ACUIh$, this semiring consists of finite languages over the alphabet $\Delta = \{1, \ldots, \rho\}$, where ρ is the number of different role names occurring in the unification problem under consideration. The semiring operations are union (as addition) and concatenation (as multiplication), with $\emptyset$ as additive unit and $\{\varepsilon\}$ as multiplicative unit. Linear equations over this semiring are then exactly the language equations of the form (2).

3.1 Syntactically Restricted Unification in $\mathcal{FL}_0$

For an integer $k \geq 1$, a syntactically k-restricted unification problem is an equation of the form $C \overset{?}{\equiv}{}^k_{syn} D$, where C, D are $\mathcal{FL}_0$ concept patterns (satisfying the restrictions formulated in Lemma 1). A unifier of this equation is a substitution σ such that $\sigma(C) \equiv^k_{syn} \sigma(D)$.

Due to the LNF-based characterization of $\equiv^k_{syn}$ and the correspondence between role depth and word length mentioned above, the question of whether a given syntactically k-restricted unification problem has a unifier or not can be reduced to checking whether language equations of the form (2) have solutions θ such that

$$S_0 \cup S_1 \cdot \theta(X_1) \cup \cdots \cup S_n \cdot \theta(X_n) = T_0 \cup T_1 \cdot \theta(X_1) \cup \cdots \cup T_n \cdot \theta(X_n) \subseteq \Delta^{\leq k}. \tag{4}$$

where $\Delta^{\leq k}$ denotes the set of words over Δ of length at most k.

The unifier σ of the $\mathcal{FL}_0$ unification problem in Example 1 is not a syntactically 3-restricted unifier of this problem since the unified concept $\sigma(\forall r_1.\forall r_1.A \sqcap \forall r_1.\forall r_1.X) = \forall r_1.\forall r_1.A \sqcap \forall r_1.\forall r_1.\forall r_1.\forall r_1.A = \sigma(X \sqcap \forall r_1.\forall r_1.\forall r_1.Y)$ has role depth 4. This is reflected on the language equation side by the fact that $\{11\} \cup \{11\} \cdot \{11\} = \{11, 1111\} = \{\varepsilon\} \cdot \{11\} \cup \{111\} \cdot \{1\} \not\subseteq \Delta^{\leq 3}$. In fact, it is easy to see that this problem does not have a syntactically 3-restricted unifier.

3.2 Semantically Restricted Unification in $\mathcal{FL}_0$

For an integer $k \geq 1$, a semantically k-restricted unification problem is an equation of the form $C \overset{?}{\equiv}{}^k_{sem} D$, where C, D are $\mathcal{FL}_0$ concept patterns. A unifier of this equation is a substitution σ such that $\sigma(C) \equiv^k_{sem} \sigma(D)$.

Whereas in the syntactically restricted case a sequence of value restrictions of depth $> k$ (a word of length $> k$) destroys the property of being a unifier (solution), in the semantically restricted case one can simply ignore such sequences (words). Thus, one can reduce the question of whether a given semantically k-restricted unification problem has a unifier or not to checking whether, for language equations of the form (2), there is an assignment θ such that

$$\begin{aligned}(S_0 \cup S_1 \cdot \theta(X_1) \cup \cdots \cup S_n \cdot \theta(X_n)) \cap \Delta^{\leq k} = \\ (T_0 \cup T_1 \cdot \theta(X_1) \cup \cdots \cup T_n \cdot \theta(X_n)) \cap \Delta^{\leq k}.\end{aligned} \tag{5}$$

Note that, in general, such an assignment need not satisfy (3), but clearly any solution θ of (2) satisfying (3) also satisfies (5).

Example 2. The $\mathcal{FL}_0$ unification problem $\forall r_1.A \sqcap \forall r_1.\forall r_1.X \overset{?}{\equiv} X$ induces the language equation $\{1\} \cup \{11\} \cdot X = \{\varepsilon\} \cdot X$. This language equation does not have a solution in the classical sense, but it has a semantically 3-restricted solution. In fact, for the assignment θ with $\theta(X) = \{1, 111\}$ we have $\{1\} \cup \{11\} \cdot \theta(X) = \{1, 111, 11111\}$ and $\{\varepsilon\} \cdot \theta(X) = \{1, 111\}$. Intersecting these two sets with $\Delta^{\leq 3}$ yields the same set $\{1, 111\}$. Thus, the above unification problem does not have a unifier, but it has a semantically 3-restricted unifier.

Using the corresponding equational theory Again, the reduction of unification to finding assignments θ satisfying (5) can also be obtained by considering the corresponding equational theory, i.e., the theory $ACUIh^k$. It is easy to see that $ACUIh^k$ satisfies the definition of a monoidal theory in [14], and thus is also commutative [6]. In addition, the corresponding semiring consists of all finite languages over Δ containing only words of length $\leq k$. The addition operation of this semiring is again union of languages, but its multiplication operation is the following restricted form of concatenation: for finite sets of words $L_1, L_2 \subseteq \Delta^{\leq k}$ we define

$$L_1 \cdot_k L_2 = (L_1 \cdot L_2) \cap \Delta^{\leq k}.$$

Linear equations of this semiring are thus also of the form (2), but a solution is now an assignment θ satisfying (5).

4 Root-to-Frontier Tree Automata

It was shown in [5] that checking solvability of linear language equations can be reduced to testing emptiness of tree automata. More precisely, the tree automata employed in [5] work on finite node-labelled trees, going from the root to the leaves. Such automata are called root-to-frontier tree automata (RFAs) in [5]. Basically, given a linear language equation, one can construct an RFA whose size is exponential in the size of the language equation, and which accepts some tree iff the language equation has a solution. Since the emptiness problem for RFAs is polynomial, this yields an ExpTime upper bound for solvability of linear language equations. The matching ExpTime lower bound was proved in [5] by a reduction from the intersection emptiness problem for deterministic RFAs (DRFAs). In this section, we formally introduce (D)RFAs and the trees they accept, and recall the ExpTime-completeness result for the intersection emptiness problem for DRFAs from [16]. We will then show that a restricted version of this problem is PSpace-complete.

We consider trees with labels in the ranked alphabet Σ, where the number of successors of a node is determined by the rank of its label. Obviously, such trees are simply representations of terms over the signature Σ.

Definition 1. *Let Σ be a finite alphabet, where each $f \in \Sigma$ is associated with a rank $rank(f) \geq 0$, and let ρ be the maximal rank of the elements of Σ. A (finite) Σ-tree is a mapping $t : dom(t) \to \Sigma$ such that $dom(t)$ is a finite subset of $\{1, \ldots, \rho\}^*$ such that*

- *the empty word ε belongs to $dom(t)$;*
- *for all $u \in \{1, \ldots, \rho\}^*$ and $i \in \{1, \ldots, \rho\}$, we have $ui \in dom(t)$ iff $u \in dom(t)$ and $i \leq rank(t(u))$.*

The elements of $dom(t)$ are the nodes of the tree t, and $t(u)$ is called the label of node u. The empty word ε is the root of t, and the nodes u such that $ui \notin dom(t)$ for all $i = 1, \ldots, \rho$ are the leaves of t. By the above definition, the leaves are the nodes labeled with a symbol of rank zero, i.e., $\mathit{rank}(t(u)) = 0$ iff u is a leaf of t. We denote the set of symbols of rank 0 by $\Sigma_0 := \{f \in \Sigma \mid \mathit{rank}(f) = 0\}$. We always assume $\Sigma_0 \neq \emptyset$ since otherwise there is no finite Σ-tree. The set of all leaves of the tree t is called the frontier of t. Nodes of t that are not in the frontier are called inner nodes. If $ui \in \mathit{dom}(t)$ then it is called the ith son of u in t. The *depth of a node* $u \in \mathit{dom}(t)$ is just the length of the word u. The *depth* $\mathit{depth}(t)$ *of the tree* t is the maximal depth of a node in $\mathit{dom}(t)$.

Definition 2. *A (non-deterministic)* root-to-frontier tree automaton (RFA) *that works on* Σ-*trees is a* 5*-tuple* $\mathcal{A} = (\Sigma, Q, I, T, F)$ *where*

- Σ *is a finite, ranked alphabet,*
- Q *is a finite set of states,*
- $I \subseteq Q$ *is the set of initial states,*
- T *assigns to each* $f \in \Sigma \setminus \Sigma_0$ *of rank* n *a transition relation* $T(f) \subseteq Q \times Q^n$,
- $F : \Sigma_0 \to 2^Q$ *assigns to each* $c \in \Sigma_0$ *a set of final states* $F(c) \subseteq Q$.

A run *of* $\mathcal{A}$ *on the tree* t *is a mapping* $r : \mathit{dom}(t) \to Q$ *such that*

- $(r(u), r(u1), \ldots, r(un)) \in T(t(u))$ *for all inner nodes* u *of rank* n.

The run r *is called* successful *if*

- $r(\varepsilon) \in I$ *(root condition),*
- $r(u) \in F(t(u))$ *for all leaves* u *(leaf condition).*

The tree language accepted *by* $\mathcal{A}$ *is defined as*

$$\mathcal{L}(\mathcal{A}) := \{t \mid \text{there exists a successful run of } \mathcal{A} \text{ on } t\}.$$

The emptiness problem *for* $\mathcal{A}$ *is the question whether* $\mathcal{L}(\mathcal{A}) = \emptyset$.

It is well-known that the emptiness problem for RFAs is decidable in polynomial time (see, e.g., [17]). It is also known that, if an RFA $\mathcal{A}$ accepts a tree, then it also accepts one of depth at most q, where q is the number of states of $\mathcal{A}$.

In contrast to the emptiness problem, the intersection emptiness problem is ExpTime-complete even for deterministic RFAs [16], which are known to be weaker than general RFAs.

Definition 3. *The RFA* $\mathcal{A} = (\Sigma, Q, I, T, F)$ *is a* deterministic root-to-frontier automaton (DRFA) *if*

- *the set* I *of initial states consists of a single initial state* q_0,
- *for all states* $q \in Q$ *and all symbols* f *of rank* $n > 0$ *there exists exactly one* n*-tuple* $(q_1, \ldots, q_n)$ *such that* $(q, q_1, \ldots, q_n) \in T(f)$.

For deterministic automata it is often more convenient to use a transition function δ *in place of the (functional) transition relations. This function is defined as* $\delta(q, f) := (q_1, \ldots, q_n)$, *where* $(q_1, \ldots, q_n)$ *is the unique tuple satisfying* $(q, q_1, \ldots, q_n) \in T(f)$.

Given a collection $\mathcal{A}_1, \ldots, \mathcal{A}_n$ *of DRFAs, the* intersection emptiness problem *asks whether* $L(\mathcal{A}_1) \cap \ldots \cap L(\mathcal{A}_n) = \emptyset$. *For a natural number* k, *the* k-restricted intersection emptiness problem *asks, for given DFRAs* $\mathcal{A}_1, \ldots, \mathcal{A}_n$, *whether there is a tree* t *with* $\mathit{depth}(t) \leq k$ *such that* $t \in L(\mathcal{A}_1) \cap \ldots \cap L(\mathcal{A}_n)$.

The complexity of the k-restricted intersection emptiness problem depends on the encoding of the number k.

Theorem 1. *The k-restricted intersection emptiness problem for DRFAs is ExpTime-complete if the number k is encoded in binary, and PSpace-complete if the number k is encoded in unary.*

Proof. First, consider the case of binary coding of k. To show that the problem is in ExpTime, we first build the product automaton $\mathcal{A}$ of the automata $\mathcal{A}_1, \ldots, \mathcal{A}_n$. This automaton accepts the intersection language $L(\mathcal{A}_1) \cap \ldots \cap L(\mathcal{A}_n)$. It has as set of states the direct product $Q := Q_1 \times \ldots \times Q_n$ of the state sets $Q_1, \ldots, Q_n$ of the automata $\mathcal{A}_1, \ldots, \mathcal{A}_n$, whose cardinality is thus exponentially bounded by the combined size of the input automata. To check whether $\mathcal{A}$ accepts a tree of depth at most k, we add a counter to the states of $\mathcal{A}$, i.e., consider the set of states $Q' := Q \times \{0, \ldots, k\}$. The transitions of the extended automaton $\mathcal{A}'$ are basically the ones of $\mathcal{A}$, but in each transition, the counter values associated with the states are decremented. No transitions are possible from states whose counter has value 0. The initial state of $\mathcal{A}'$ is $q_0' := (q_0, k)$ where q_0 is the initial state of $\mathcal{A}$, and the final assignment is $F'(c) := F(c) \times \{0, \ldots, k\}$ where F is the final assignment of $\mathcal{A}$. It is easy to see that $\mathcal{A}'$ accepts exactly the trees in $L(\mathcal{A})$ that have a depth $\leq k$. Thus, to solve the k-restricted intersection emptiness problem for $\mathcal{A}_1, \ldots, \mathcal{A}_n$, it is sufficient to solve the emptiness problem for $\mathcal{A}'$. Since the size of $\mathcal{A}'$ is exponential in the combined size of $\mathcal{A}_1, \ldots, \mathcal{A}_n$ and the size of the binary representation of k, this yields the desired ExpTime upper bound.

The ExpTime lower bound can be shown by a reduction from the unrestricted intersection emptiness problem for DRFAs. Given DRFAs $\mathcal{A}_1, \ldots, \mathcal{A}_n$, we know that $L(\mathcal{A}_1) \cap \ldots \cap L(\mathcal{A}_n) \neq \emptyset$ iff the product automaton $\mathcal{A}$ constructed from these automata accepts some tree, which is the case iff $\mathcal{A}$ accepts a tree of depth at most $k := |Q_1 \times \ldots \times Q_n|$. This yields a reduction from the unrestricted intersection emptiness to the k-restricted intersection emptiness problem. This reduction is polynomial since the size of the binary representation of k is polynomial in the combined size of $\mathcal{A}_1, \ldots, \mathcal{A}_n$.

To show the PSpace upper bound for the case of unary coding of k, we consider the automaton $\mathcal{A}'$ constructed in our proof of the ExpTime upper bound for the case of binary coding. But now this automaton is not constructed completely before running the emptiness test. Instead, we construct the relevant parts of it on-the-fly while non-deterministically trying to construct a successful run, starting from the root. Since the depth of the run is linearly bounded by the size of the unary representation of k, and each state of $\mathcal{A}'$ can be represented using only polynomial space, this needs only polynomial space because only one branch of the run (plus some backtracking information) needs to be represented at each point in time. The NPSpace algorithm obtained this way yields the desired PSpace upper bound since, due to Savitch's theorem, it can be turned into a PSpace algorithm.

PSpace-hardness can be shown by a reduction from QBF (see Proposition 1 below). □

To prove the PSpace lower bound, we show how to reduce QBF to the k-restricted intersection emptiness problem. Thus, let

$$Q_1\, p_1. \cdots Q_m\, p_m.\phi \quad \text{for } Q_1, \ldots, Q_m \in \{\forall, \exists\}$$

be a quantified Boolean formula, where we assume without loss of generality that ϕ contains only the propositional variables $p_1, \ldots, p_m$ and is in conjunctive normal form, i.e., $\phi = \psi_1 \wedge \ldots \wedge \psi_n$ for clauses ψ_j. The problem QBF asks whether such a formula is valid.

It is easy to see that validity of the above formula corresponds to the existence of a certain tree, whose branching depends on the quantifier prefix. Basically, such a tree describes a set of propositional assignments that need to satisfy the formula ϕ. For a universally quantified

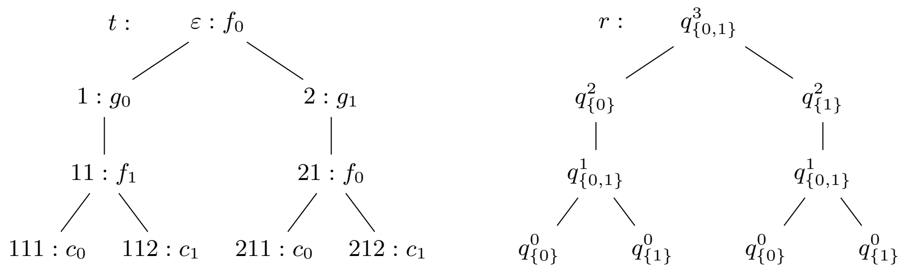


Fig. 1. A Σ-tree t conforming with the prefix $P = \forall p_1.\exists p_2.\forall p_3$, and a successful run of $\mathcal{A}_P$ on t.

variable, the corresponding node has two successors since both values must lead to satisfaction of ϕ, whereas for an existentially quantified variable one successor node is sufficient.

To represent such trees in the way introduced in Definition 1, we consider the alphabet $\Sigma = \{f_0, f_1, g_0, g_1, c_0, c_1\}$ where f_0, f_1 are binary, g_0, g_1 are unary, and c_0, c_1 are nullary. We now define what it means that a Σ-tree *conforms with* a quantifier-prefix $Q_1\, p_1. \cdots Q_m\, p_m$ by induction on the length of the prefix:

- Assume that $m = 1$, i.e., the prefix is $Q_1\, p_1$.
  - If $Q_1 = \forall$, then any tree t with $dom(t) = \{\varepsilon, 1, 2\}$, root label $t(\varepsilon) \in \{f_0, f_1\}$ and leaf labels $t(1) \in \{c_0\}$ and $t(2) \in \{c_1\}$ conforms with $\forall p_1$.
  - If $Q = \exists$, then any tree t with $dom(t) = \{\varepsilon, 1\}$, root label $t(\varepsilon) \in \{g_0, g_1\}$ and leaf label $t(1) \in \{c_0, c_1\}$ conforms with $\exists p_1$.
- Assume that $m > 1$, i.e., the prefix is $Q_1\, p_1.Q_2\, p_2. \cdots Q_m\, p_m$.
  - If $Q_1 = \forall$, then any tree t with root label $t(\varepsilon) \in \{f_0, f_1\}$ for which
    - the node 1 has label $t(1) \in \{f_0, g_0\}$ and is the root of a subtree that conforms with $Q_2\, p_2. \cdots Q_m\, p_m$,
    - the node 2 has label $t(2) \in \{f_1, g_1\}$ and is the root of a subtree that conforms with $Q_2\, p_2. \cdots Q_m\, p_m$,

    conforms with $Q_1\, p_1.Q_2\, p_2. \cdots Q_m\, p_m$.
  - If $Q_1 = \exists$, then any tree t with root label $t(\varepsilon) \in \{g_0, g_1\}$ for which
    - the node 1 has label $t(1) \in \{f_0, g_0, f_1, g_1\}$ and is the root of a subtree that conforms with $Q_2\, p_2. \cdots Q_m\, p_m$,

    conforms with $Q_1\, p_1.Q_2\, p_2. \cdots Q_m\, p_m$.

Note that any tree t conforming with $Q_1\, p_1. \cdots Q_m\, p_m$ has depth m since all leaves of such a tree have depth m. Every leaf of t determines a propositional valuation. For any symbol $h \in \Sigma$ we define its value $val(h)$ to be its index; e.g., $val(c_0) = 0$ and $val(f_1) = 1$. A leaf $u = i_1 \dots i_m \in \{1, 2\}^m$ of t determines the following valuation v_u:

$$v_u(p_j) := val(t(i_1 \dots i_j)) \quad \text{for } j = 1, \dots, m.$$

Example 3. The left-hand side of Figure 1 depicts a Σ-tree t conforming with the prefix $\forall p_1.\exists p_2.\forall p_3$. The leaves $111, 112, 211, 212$ of t determine the following propositional valuations:

$$\begin{aligned}
v_{111}(p_1) &= 0, & v_{111}(p_2) &= 1, & v_{111}(p_3) &= 0,\\
v_{112}(p_1) &= 0, & v_{112}(p_2) &= 1, & v_{112}(p_3) &= 1,\\
v_{211}(p_1) &= 1, & v_{211}(p_2) &= 0, & v_{211}(p_3) &= 0,\\
v_{212}(p_1) &= 1, & v_{212}(p_2) &= 0, & v_{212}(p_3) &= 1.
\end{aligned}$$

For example, the valuation v_{111} is determined by the indices $0, 1$, and 0 of $t(1) = g_0, t(11) = f_1$, and $t(111) = c_0$, respectively. The valuation v_{112} only differs from v_{111} in that the value of p_3 is determined by $t(112)$. Since $t(112) = c_1$, this yields $v_{112}(p_3) = 1$.

The following is an easy consequence of the semantics of quantified Boolean formulae.

Lemma 3. *Let $Q_1\, p_1. \cdots Q_m\, p_m.\phi$ be a quantified Boolean formula. Then this formula is valid iff there exists a Σ-tree conforming with $Q_1\, p_1. \cdots Q_m\, p_m$ such that v_u satisfies ϕ for every leaf u of t.*

The set of Σ-trees conforming with a given quantifier prefix can be accepted by a DRFA.

Lemma 4. *Given a quantifier-prefix $P = Q_1\, p_1. \cdots Q_m\, p_m$, we can construct in polynomial time a DRFA $\mathcal{A}_P$ that accepts exactly the Σ-trees that conform with P, and has a set of states whose cardinality is linear in m.*

Proof. We define $\mathcal{A}_P = (\Sigma, Q_P, q^m_{\{0,1\}}, \delta_P, F_P)$ where

- $Q_P := \{q^j_I \mid \emptyset \subset I \subseteq \{0,1\}$ and $j = 0, 1, \ldots, m\} \cup \{q_\bot\}$,
- for $j = 1, \ldots, m$ and $\emptyset \subset I \subseteq \{0,1\}$, the transitions issuing from q^j_I depend on the quantifier $Q_{(m-j)+1}$:
  - if $Q_{(m-j)+1} = \forall$, then
    - $\delta_P(q^j_I, f_i) := (q^{j-1}_{\{0\}}, q^{j-1}_{\{1\}})$ if $i \in I$,
    - $\delta_P(q^j_I, f_i) := (q_\bot, q_\bot)$ if $i \notin I$,
    - $\delta_P(q^j_I, h) := q_\bot$ for $h \in \{g_0, g_1\}$,
  - if $Q_{(m-j)+1} = \exists$, then
    - $\delta_P(q^j_I, g_i) := q^{j-1}_{\{0,1\}}$ if $i \in I$,
    - $\delta_P(q^j_I, g_i) := q_\bot$ if $i \notin I$,
    - $\delta_P(q^j_I, h) := (q_\bot, q_\bot)$ for $h \in \{f_0, f_1\}$,
- the states q^0_I with exponent 0 have only transitions to the state $q_\bot$, i.e., $\delta_P(q^0_I, h) := (q_\bot, q_\bot)$ for $h \in \{f_0, f_1\}$ and $\delta_P(q^0_I, h) := q_\bot$ for $h \in \{g_0, g_1\}$,
- the state $q_\bot$ reproduces itself, i.e., $\delta_P(q_\bot, h) := (q_\bot, q_\bot)$ for $h \in \{f_0, f_1\}$ and $\delta_P(q_\bot, h) := q_\bot$ for $h \in \{g_0, g_1\}$,
- the final assignment is defined as follows: $F_P(c_i) := \{q^0_I \mid i \in I\}$.

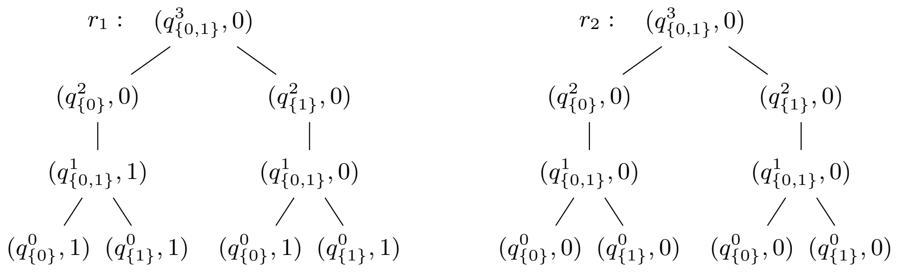
It is easy to see that $L(\mathcal{A}_P)$ indeed consists of exactly the Σ-trees that conform with $P = Q_1\, p_1. \cdots Q_m\, p_m$. □

For example, the tree r on the right-hand side of Figure 1 shows an accepting run of $\mathcal{A}_P$ on the Σ-tree t depicted on the left-hand side.

Due to Lemma 3, we are interested in those Σ-trees t conforming with $P = Q_1\, p_1. \cdots Q_m\, p_m$ for which the propositional valuations v_u for the leaves u of t all satisfy $\phi = \psi_1 \wedge \ldots \wedge \psi_n$. It would not be hard to construct a DRFA that accept exactly these trees, but this automaton would be of size exponential in the number of clauses n. Instead, we construct n automata $\mathcal{A}^{\psi_1}_P, \ldots, \mathcal{A}^{\psi_n}_P$, each ensuring that one of the clauses ψ_i is satisfied by the assignments v_u.

Lemma 5. *Let $P = Q_1\, p_1. \cdots Q_m\, p_m$ be a quantifier-prefix and ψ a clause containing only literals built using the variables $p_1, \ldots, p_m$. Then we can construct in polynomial time a DRFA $\mathcal{A}^\psi_P$ that accepts exactly the Σ-trees t such that t conforms with P and v_u satisfies ψ for every leaf u of t.*

Proof. The automaton $\mathcal{A}^\psi_P$ is constructed from the DRFA $\mathcal{A}_P$ by extending the states q^j_I with a second component, which is 0 or 1. Intuitively, if the partial assignment for the variables



r_1 : $(q^3_{\{0,1\}}, 0)$
$(q^2_{\{0\}}, 0)$ $(q^2_{\{1\}}, 0)$
$(q^1_{\{0,1\}}, 1)$ $(q^1_{\{0,1\}}, 0)$
$(q^0_{\{0\}}, 1)$ $(q^0_{\{1\}}, 1)$ $(q^0_{\{0\}}, 1)$ $(q^0_{\{1\}}, 1)$
r_2 : $(q^3_{\{0,1\}}, 0)$
$(q^2_{\{0\}}, 0)$ $(q^2_{\{1\}}, 0)$
$(q^1_{\{0,1\}}, 0)$ $(q^1_{\{0,1\}}, 0)$
$(q^0_{\{0\}}, 0)$ $(q^0_{\{1\}}, 0)$ $(q^0_{\{0\}}, 0)$ $(q^0_{\{1\}}, 0)$

Fig. 2. Successful runs of the automata $\mathcal{A}_P^{\psi_1}$ and $\mathcal{A}_P^{\psi_2}$.

$p_1, \dots, p_{m-j-1}$ seen so far on the path satisfies ψ, this second component is 1; and otherwise it is 0. Since for the states of the form q_I^m and q_I^{m-1} no assignment for a propositional variable has been determined yet, they receive the additional component 0. Only for states of the form q_I^j with $0 \le j \le m-2$ is there the option to obtain a 1 as second component.

To be more precise, assume that the unique run of $\mathcal{A}_P$ on a tree t is successful. The value for the variable p_1 in a path of the tree leading to the leaf u is decided at the node u_1 at depth 1 in this path, which is labeled in the run with a state of the form q_I^{m-1}. If $\mathit{val}(t(u_1)) = 0$ and $\neg p_1$ occurs in ψ or $\mathit{val}(t(u_1)) = 1$ and p_1 occurs in ψ, then the valuation v_u will satisfy ψ, independently of what symbols are encountered further down in the path. In this case, the successor states of q_I^{m-1}, which are of the form $q_{I'}^{m-2}$ in the run of $\mathcal{A}_P$, receive the additional label 1 in the corresponding run of $\mathcal{A}_P^{\psi}$. Otherwise, the additional label is 0. From then on, the successors states receive label 1 if the current state already has label 1, or if the value of the variable decided on this level makes ψ true (as described above for p_1); otherwise, the label 0 is kept.

The value of the variable p_m is decided at the leaf u. If ψ has already been satisfied by the assignment to the variables $p_1, \dots, p_{m-1}$, then this value is irrelevant. Otherwise, it determines whether the assignment v_u satisfies ψ. Thus, the final assignment is modified as follows:

$$F(c_i) := \{(q_I^0, 1) \mid i \in I\} \cup \{(q_I^0, 0) \mid i \in I \text{ and setting } p_m = i \text{ makes } \psi \text{ true}\}.$$

It is easy to see that this way the automaton $\mathcal{A}_P$ can indeed be transformed in polynomial time into a DRFA $\mathcal{A}_P^{\psi}$ satisfying the requirements stated in the formulation of the lemma. □

Example 4. Consider the clauses $\psi_1 = \neg p_1 \vee \neg p_2$ and $\psi_2 = p_3 \vee \neg p_3$. The trees r_1 and r_2 shown in Figure 2 correspond, respectively, to successful runs of the automata $\mathcal{A}_P^{\psi_1}$ and $\mathcal{A}_P^{\psi_2}$ on the tree t depicted on the left-hand side of Figure 1.

Both runs extend the run r of $\mathcal{A}_P$ on t from Figure 1. For ψ_1, for instance, the valuation v_{111} satisfies ψ_1 since $\mathit{val}(t(1)) = 0$ and $\neg p_1$ occurs in ψ_1. The run r_1 detects this at node 1, and the state $q^1_{\{0,1\}}$ labeling node 11 in r receives the additional label 1 to define $r_1(11) = (q^1_{\{0,1\}}, 1)$. The label 1 is then transferred to the leaves. In contrast, in the paths leading to the leaves 211 and 212, the node 21 receives the additional label 0, i.e., $r_1(21) = (q^1_{\{0,1\}}, 0)$, since these paths yield value 1 for the variable p_1. But since the variable p_2 receives value 0, the states at the leaves have again 1 as additional label.

Regarding ψ_2, since only the assignment to p_3 can lead to satisfaction of ψ_2, all states q_I^1 and q_I^0 labeling nodes in r receive the additional label 0 in r_2. Hence, the satisfaction of ψ_2 is always decided at the leaves of t, which in $\mathcal{A}_P^{\psi_2}$ is determined by the final assignment F. As both assignments to p_3 satisfy ψ_2, one can easily verify that $r_2(u) \in F(t(u))$ for all leaves u.

The following proposition is an easy consequence of Lemma 5.

Proposition 1. *Let $Q_1\, p_1. \cdots Q_m\, p_m.\phi$ be a quantified Boolean formula where $\phi = \psi_1 \wedge \ldots \wedge \psi_n$ for clauses ψ_j. Then we can construct in polynomial time DRFAs $\mathcal{A}_P^{\psi_1}, \ldots, \mathcal{A}_P^{\psi_n}$ such that $Q_1\, p_1. \cdots Q_m\, p_m.\phi$ is valid iff there is a tree $t \in L(\mathcal{A}_1^{\psi_1}) \cap \ldots \cap L(\mathcal{A}_n^{\psi_n})$ such that $depth(t) \leq m$.*

Since validity of quantified Boolean formulae is PSpace-complete [9], this proposition implies that the k-restricted intersection emptiness problem for DRFAs is PSpace-hard. This holds even for unary coding of k since, in the above proposition, the size of the unary representation of m is linearly bounded by the length of the quantifier-prefix.

5 Solving Linear Language Equations Using RFAs

As mentioned in Section 3, checking solvability of linear language equations was reduced in [5] to testing emptiness of RFAs. However, this approach cannot directly treat equations of the form (2). It needs equations where the variables X_i are in front of the coefficients S_i. Fortunately, such equations can easily be obtained from the ones of the form (2) by considering the mirror images of the involved languages. For a word $w = i_1 \ldots i_\ell \in \Delta^*$, its mirror image is defined as $w^{mi} := i_\ell \ldots i_1$, and for a finite set of words $L = \{w_1, \ldots, w_m\}$, its mirror image is $L^{mi} := \{w_1^{mi}, \ldots, w_m^{mi}\}$. Obviously, the assignment θ with $\theta(X_1) = L_1, \ldots, \theta(X_n) = L_n$ is a solution of (2) iff θ^{mi} with $\theta^{mi}(X_1) = L_1^{mi}, \ldots, \theta^{mi}(X_n) = L_n^{mi}$ is a solution of the corresponding mirrored equation

$$S_0^{mi} \cup X_1{\cdot}S_1^{mi} \cup \cdots \cup X_n{\cdot}S_n^{mi} = T_0^{mi} \cup X_1{\cdot}T_1^{mi} \cup \cdots \cup X_n{\cdot}T_n^{mi}. \qquad (6)$$

Finite languages over the alphabet $\Delta = \{1, \ldots, \rho\}$ can be represented by Σ-trees for the ranked alphabet $\Sigma = \{f_0, f_1, c_0, c_1\}$, where f_0, f_1 are ρ-ary and c_0, c_1 nullary symbols. A given Σ-tree t represents the finite language

$$L_t = \{u \in dom(t) \mid t(u) \in \{c_1, f_1\}\}.$$

Given an equation of the form (6), it is shown in [5] how to construct an RFA $\mathcal{A} = (\Sigma, Q, I, T, F)$ of size exponential in the size of the equation that satisfies the following property.

Lemma 6 (Lemma 6.3 in [5]). *For a Σ-tree t the following are equivalent:*

1. *The tree t is accepted by $\mathcal{A}$.*
2. *There are finite sets of words $\theta(X_1), \ldots, \theta(X_n)$ such that*

$$S_0^{mi} \cup \theta(X_1){\cdot}S_1^{mi} \cup \cdots \cup \theta(X_n){\cdot}S_n^{mi} = L_t = T_0^{mi} \cup \theta(X_1){\cdot}T_1^{mi} \cup \cdots \cup \theta(X_n){\cdot}T_n^{mi}.$$

Consequently, we have that (2) has a solution iff (6) has a solution iff the RFA $\mathcal{A}$ constructed from (6) accepts some tree. Since the size of $\mathcal{A} = (\Sigma, Q, I, T, F)$ is exponential in the size of (2), and the emptiness problem for RFAs is decidable in polynomial time, this yields an ExpTime decision procedure for solvability of language equations of the form (2), and thus for unifiability in $\mathcal{FL}_0$. As already mentioned in Section 4, it is also shown in [5], by a reduction from the intersection emptiness problem for DRFAs, that these problems are actually ExpTime-hard.

Theorem 2 ([5]). *Unifiability in $\mathcal{FL}_0$ as well as solvability of language equations of the forms (2) and (6) are ExpTime-complete problems.*

6 Solving Restricted Linear Language Equations Using RFAs

In the restricted setting, we consider equations of the form (2) and are looking for solutions θ satisfying (4) for the syntactically restricted setting or satisfying (5) for the semantically restricted setting. Since clearly $(\Delta^{\leq k})^{mi} = \Delta^{\leq k}$, the respective restrictions apply unchanged to the mirrored equation (6).

6.1 The Syntactically Restricted Case

In this case we are thus looking for solutions θ of (6) satisfying

$$\begin{aligned} &S_0^{mi} \cup \theta(X_1)\cdot S_1^{mi} \cup \cdots \cup \theta(X_n)\cdot S_n^{mi} = \\ &T_0^{mi} \cup \theta(X_1)\cdot T_1^{mi} \cup \cdots \cup \theta(X_n)\cdot T_n^{mi} \subseteq \Delta^{\leq k}. \end{aligned} \tag{7}$$

Intuitively, for the trees accepted by the automaton $\mathcal{A}$ this means that we want to check whether $\mathcal{A}$ accepts a tree of depth $\leq k$. Similarly to what we have done in the proof of Theorem 1, this can be achieved by adding a counter that is decremented whenever we go from a node in the tree to a successor node. As soon as the counter reaches 0, no more transitions are possible.

To be more precise, let $\mathcal{A} = (\Sigma, Q, I, T, F)$ be the RFA constructed from (6), as described in [5]. For an integer $k \geq 1$, we define the automaton $\mathcal{A}^k_{syn} = (\Sigma, Q^k_{syn}, I^k_{syn}, T^k_{syn}, F^k_{syn})$ as follows:

- $Q^k_{syn} = Q \times \{0, 1, \ldots, k\}$,
- $I^k_{syn} = I \times \{k\}$,
- $T^k_{syn}(f) = \{((q, i), (q_1, i-1), \ldots, (q_\rho, i-1)) \mid (q, q_1, \ldots, q_\rho) \in T(f) \text{ and } i \geq 1\}$ for $f \in \{f_0, f_1\}$,
- $F^k_{syn}(c) = F(c) \times \{0, 1, \ldots, k\}$ for $c \in \{c_0, c_1\}$.

Basically, $\mathcal{A}^k_{syn}$ works like $\mathcal{A}$, but once it has reached a node at depth k in the tree, it cannot make any transition. Thus, it accepts exactly the trees that have depth at most k and are accepted by $\mathcal{A}$. Since nodes at a depth i correspond to words of this length i, we obtain the following lemma.

Lemma 7. *The automaton $\mathcal{A}^k_{syn}$ accepts a tree t iff* (6) *has a solution θ that satisfies* (7).

Proof. If (6) has a solution θ that satisfies (7), then there is a tree t of depth at most k that represents this solution in the sense that it satisfies 2. of Lemma 6. The tree t then also satisfies 1. of Lemma 6, i.e., it is accepted by $\mathcal{A}$. Since t has depth at most k, it is then also accepted by $\mathcal{A}^k_{syn}$.

Conversely, if the tree t is accepted by $\mathcal{A}^k_{syn}$, then it is also accepted by $\mathcal{A}$ and has depth at most k. The former implies, by Lemma 6, that 2. of Lemma 6 holds, and the latter yields that $L(t) \subseteq \Delta^{\leq k}$. Thus, the sets $\theta(X_1), \ldots, \theta(X_n)$ provided by 2. of Lemma 6 satisfy (7). $\square$

As an easy consequence of this lemma, and the connection between syntactically k-restricted unification and the problem of finding solutions of (6) that satisfy (7), we obtain the following complexity results. Note that, as in the unrestricted case, solvability of language equations in the syntactically restricted sense of (6) can obviously be reduced to syntactically k-restricted unification.

Theorem 3. *Given an integer $k \geq 1$ and $\mathcal{FL}_0$ concepts C, D as input, the problem of deciding whether the syntactically k-restricted unification problem $C \overset{?}{\equiv}{}^{k}_{syn} D$ has a unifier or not is ExpTime-complete if the number k is assumed to be encoded in binary, and PSpace-complete if k is assumed to be encoded in unary.*

Proof. Let us first consider the case where the number k is encoded in binary. To show the complexity upper bound, first note that we can assume that the numbers $i \in \{0, \ldots, k\}$ used within the automaton $\mathcal{A}^k_{syn}$ are also encoded in binary. For this reason, a single state $(q, i) \in Q^k_{syn}$ requires only polynomial space for its representation since this is also the case for the states q of $\mathcal{A}$. However, in the size of the binary representation of k, the set $\{0, \ldots, k\}$ has an exponential cardinality. Nevertheless, since the cardinality of the set of states Q is exponential in the size of the concepts C, D, the overall cardinality of Q^k_{syn}, and thus of the automaton $\mathcal{A}^k_{syn}$, is still at most exponential in the size of the input. Since the emptiness problem for RFAs can be decided in polynomial time, this yields the desired ExpTime upper bound for the k-restricted unification problem.

We show the ExpTime lower bound by reducing the problem of testing whether an equation of the form (6) has a solution (which is known to be ExpTime-hard by Theorem 2) to the problem of whether this equation has a k-restricted solution for an appropriate number k. Given an equation of the form (6), we can compute the cardinality of the set of states of the corresponding automaton $\mathcal{A}$ without actually computing the automaton itself. Let k be the cardinality of this set. Then the binary representation of k has a size that is polynomial in the size of (6). We claim that (6) has a solution iff it has a k-restricted solution, i.e., one satisfying (7). In fact, we know that (6) has a solution iff $\mathcal{A}$ accepts some tree t. Since it is well-known that an RFA with set of states Q accepts a tree iff it accepts a tree of depth at most $|Q|$, the latter is equivalent to $\mathcal{A}^k_{syn}$ accepting a tree, which in turn is equivalent to (6) having a solution that satisfies (7).

For the case where the number k is represented in unary, the reduction described in the previous paragraph would no longer be polynomial. Similarly to the approach used to prove the PSpace upper bound in Theorem 1, we can show that there is a PSpace upper bound for checking whether $\mathcal{A}^k_{syn}$ accepts some tree. Again, the main idea is that we can generate and explore a polynomially branching tree of polynomial depth using only polynomial space. To be more precise, we can guess (in NPSpace) a tree of depth at most k and a successful run of $\mathcal{A}^k_{syn}$ on this tree. For this, we can, of course, not construct the whole automaton $\mathcal{A}^k_{syn}$ beforehand (since its representation would need exponential space), but we can generate transitions of the automaton on-the-fly using only polynomial space. One way of making this more formal would be to transfer the results in [2] for automata that have a PSpace on-the-fly construction from looping tree automata on infinite trees to root-to-frontier tree automata working on finite trees.

PSpace-hardness can be shown by a reduction of the k-restricted intersection emptiness problem for DRFAs. As shown in [5], the unrestricted version of this problem can be reduced in polynomial time to (unrestricted) solvability of language equations of the form (2). A close look at this reduction reveals that the length of the words in a solution of the language equation is linearly bounded by the depth of the tree contained in the intersection. To be more precise, if the depth of the tree in the intersection is bounded by ℓ, then the equation has a solution satisfying (4) for $k = 2\ell + 2$.

To be more precise, given a tree t of depth ℓ, the approach used in Section 7 of [5] constructs a finite set of words $S(t)$ such that the length of the words in $S(t)$ is bounded by $2\ell + 1$. If t is a tree accepted by all automata in the given sequence of DRFAs, then there are solutions of the constructed language equations (i) of the form (7.1) in [5] such that the longest words occurring in these solutions belongs to $S(t)$ (see Lemma 7.4 and the proof of Theorem 7.6 in [5]). This yields a bound of $2\ell + 1$ for the length of words in the solutions, but since the equations (i) for the single automata are encoded into a single equation (0) (following the approach sketched below Theorem 4 below), another 1 needs to be added.

For this reason, the reduction in [5] also is a reduction of the ℓ-restricted intersection emptiness problem for DRFAs to the question of whether a language equation of the form (2) has a syntactically k-restricted solution, i.e., a solution satisfying (4), where $k = 2\ell+2$. By Theorem 1, the ℓ-restricted intersection emptiness problem is PSpace-hard for unary coding of numbers. □

6.2 The Semantically Restricted Case

In this case, to solve the mirrored equation (6), we are looking for assignments θ satisfying

$$\begin{aligned}(S_0^{mi} \cup \theta(X_1)\cdot S_1^{mi} \cup \cdots \cup \theta(X_n)\cdot S_n^{mi}) \cap \Delta^{\leq k} =\\ (T_0^{mi} \cup \theta(X_1)\cdot T_1^{mi} \cup \cdots \cup \theta(X_n)\cdot T_n^{mi}) \cap \Delta^{\leq k}.\end{aligned} \tag{8}$$

We cannot directly apply Lemma 6 here since the assignment θ need not be a solution of (6). However, it is easy to see that a result similar to this lemma also holds for partial runs of the automaton $\mathcal{A} = (\Sigma, Q, I, T, F)$ constructed from (6). Given a Σ-tree t, we define $dom^n(t) := dom(t) \cap \Delta^{\leq n}$. A partial run of depth k of $\mathcal{A}$ on t is a mapping $p : dom^{k+1}(t) \to Q$ such that $p(\varepsilon) \in I$ and the following holds for all $u \in dom^k(t)$:

– if u is a leaf, then $p(u) \in F(t(u))$,
– if u is not a leaf, them $(p(u), p(u0), p(u1)) \in T(t(u))$.

It is easy to see that the following generalization of Lemma 6 also holds.

Lemma 8. *Let $\mathcal{A}$ be the RFA constructed in [5] from* (6)*. For a Σ-tree t and an integer $k \geq 1$ the following are equivalent:*

1. *There is a partial run of depth k of $\mathcal{A}$ on t.*
2. *There are finite sets of words $\theta(X_1), \ldots, \theta(X_n)$ such that*

$$\begin{aligned}(S_0^{mi} \cup \theta(X_1)\cdot S_1^{mi} \cup \cdots \cup \theta(X_n)\cdot S_n^{mi}) \cap \Delta^{\leq k} = L_t \cap \Delta^{\leq k} =\\ (T_0^{mi} \cup \theta(X_1)\cdot T_1^{mi} \cup \cdots \cup \theta(X_n)\cdot T_n^{mi}) \cap \Delta^{\leq k}.\end{aligned}$$

Note that Lemma 6 is in fact a special case of this lemma, which is obtained by considering the setting where $dom(t) \subseteq \Delta^{\leq k}$.

To check whether there is a partial run of depth k of $\mathcal{A}$ on t, we again add a counter to the states of the automaton $\mathcal{A}$, but now basically accept as soon as the counter goes below the value 0. To be more precise, let $\mathcal{A} = (\Sigma, Q, I, T, F)$ be the RFA constructed from (6), as described in [5]. For an integer $k \geq 1$, we define the automaton $\mathcal{A}_{sem}^k = (\Sigma, Q_{sem}^k, I_{sem}^k, T_{sem}^k, F_{sem}^k)$ as follows:

– $Q_{sem}^k = Q \times \{-1, 0, 1, \ldots, k\}$,
– $I_{sem}^k = I \times \{k\}$,
– $T_{sem}^k(f) = \{((q, i), (q_1, i-1), \ldots, (q_\rho, i-1)) \mid (q, q_1, \ldots, q_\rho) \in T(f) \text{ and } i \geq 0\}$ for $f \in \{f_0, f_1\}$,
– $F_{sem}^k(c) = (F(c) \times \{0, 1, \ldots, k\}) \cup \{(q, -1) \mid q \in Q\}$ for $c \in \{c_0, c_1\}$.

Lemma 9. *The automaton $\mathcal{A}_{sem}^k$ accepts a tree t iff there is an assignment θ that satisfies* (8)*.*

Proof. Assume that the automaton $\mathcal{A}^k_{sem}$ accepts the tree t with a successful run $r : dom(t) \to Q^k_{sem}$. It is easy to see that we obtain a partial run p of depth k of $\mathcal{A}$ on t by forgetting about the counter values, i.e., by setting $p(u) = q$ if $r(u) = (q, i)$ for all $u \in dom(t)$.[4] By Lemma 8, this implies that there is an assignment θ that satisfies (8).

Conversely, assume that θ is an assignment satisfying (8). It is easy to see that there is a tree t with $dom(t) \subseteq \Delta^{\leq k+1}$ such that $L_t \cap \Delta^{\leq k} = (S_0^{mi} \cup \theta(X_1)\cdot S_1^{mi} \cup \cdots \cup \theta(X_n)\cdot S_n^{mi}) \cap \Delta^{\leq k}$. By Lemma 8, this implies that there is a partial run p of depth k of $\mathcal{A}$ on t. This partial run can be turned into a successful run r of $\mathcal{A}^k_{sem}$ on t by adding the appropriate counter values, i.e., if $p(u) = q$ for $u \in dom(t)$, then set $r(u) = (q, k - |u|)$. □

Based on this lemma, the following theorem can be shown analogously to the proof of the complexity upper bounds in Theorem 3.

Theorem 4. *Given an integer $k \geq 1$ and $\mathcal{FL}_0$ concepts C, D as input, the problem of deciding whether the semantically k-restricted unification problem $C \overset{?}{\equiv}{}^k_{sem} D$ has a unifier or not is in ExpTime if the number k is assumed to be encoded in binary, and in PSpace if k is assumed to be encoded in unary.*

For the case of unary coding of k, we can show a matching PSpace lower bound. This lower bound cannot be shown in the same way as for the syntactically restricted case since in the semantically restricted case we may have a solution although there is no unrestricted solution. Instead, we reduce the syntactically restricted case to the semantically restricted one. The main idea is that we can express inclusion of a solution in $\Delta^{\leq k}$ using additional language equations. Until now, we have considered only single language equations. It is sometimes more convenient to consider a system of such equations. An assignment solves such a system if it is a solution of all the equations occurring in such a system. Solvability of finite systems of language equations can be reduced to solvability of a single language equation. In fact, the solutions of $\{L_1 = R_1, \ldots, L_\ell = R_\ell\}$ coincide with the solutions of $\{1\}\cdot L_1 \cup \ldots \cup \{\ell\}\cdot L_\ell = \{1\}\cdot R_1 \cup \ldots \cup \{\ell\}\cdot R_\ell$. Since this reduction increases the length of the words in the solved equations only by the constant 1, this reduction also works for the restricted setting since one only needs to increase the bound k by 1.

First, note that we can be express the language $\Delta^{\leq k}$ by the following system of language equations:

- $Z_{j+1} = \{\varepsilon\} \cup \{1, \ldots, \rho\}\cdot Z_j$ for $j \in \{0, \ldots, k-1\}$,
- $Z_0 = \{\varepsilon\}$ and $Z = Z_k$.

It is easy to see that any assignment θ that solves this system satisfies $\theta(Z) = \Delta^{\leq k}$. Now, assume that θ solves this system in the semantically k'-restricted sense for some bound $k' \geq k$. Then $\theta(Z)$ may contain words that are longer than k. But then these words must also be longer than k'.

Lemma 10. *Let θ be an assignment such that*

- $\theta(Z_{j+1}) \cap \Delta^{\leq k'} = (\{\varepsilon\} \cup \{1, \ldots, \rho\}\cdot\theta(Z_j)) \cap \Delta^{\leq k'}$ *for* $j \in \{0, \ldots, k-1\}$,
- $\theta(Z_0) \cap \Delta^{\leq k'} = \{\varepsilon\} \cap \Delta^{\leq k'}$ *and* $\theta(Z) \cap \Delta^{\leq k'} = \theta(Z_k) \cap \Delta^{\leq k'}$.

If $w \in \theta(Z)$ is such that $|w| > k$, then $|w| > k'$.

[4] Note that the definition of $\mathcal{A}^k_{sem}$ implies that $dom(t) \subseteq \Delta^{\leq k+1}$.

Proof. First, we show by induction on j that the following holds: if $w \in \theta(Z_j)$ is such that $|w| > j$, then $|w| > k'$. For $j = 0$, this is trivially satisfied.

For the induction step ($j \to j+1$), assume that $w \in \theta(Z_{j+1})$ is such that $|w| > j+1$. If $w \notin \{1, \dots, \rho\} \cdot \theta(Z_j)$, then this clearly implies $|w| > k'$. Otherwise, there is an $i, 1 \leq i \leq \rho$, and a word $w' \in \theta(Z_j)$ such that $w = iw'$. But then $|w'| > j$, and thus induction yields $|w'| > k'$, which implies $|w| > k'$.

Now assume that $w \in \theta(Z)$ with $|w| > k$. If $w \notin \theta(Z_k)$, then this clearly implies $|w| > k'$. Otherwise, $w \in \theta(Z_k)$ implies $|w| > k'$ by what we have shown above. □

Now, let $L = R$ be a linear language equation, where $L = S_0 \cup S_1 \cdot X_1 \cup \dots \cup S_n \cdot X_n$ and $R = T_0 \cup T_1 \cdot X_1 \cup \dots \cup T_n \cdot X_n$. We want to reduce syntactically k-restricted solvability of $L = R$ to semantically k'-restricted solvability of an extended system of equations E. Since we are interested in syntactically k-restricted solvability of $L = R$, we can assume without loss of generality that $S_0 \cup T_0 \subseteq \Delta^{\leq k}$ since otherwise the equation would be trivially unsolvable. The extended system E of equations consists of

- $Z_{j+1} = \{\varepsilon\} \cup \{1, \dots, \rho\} \cdot Z_j$ for $j \in \{0, \dots, k-1\}$,
- $Z_0 = \{\varepsilon\}$ and $Z = Z_k$,
- $L = R$ and $L \cup Z = Z$.

Let ℓ_{min} and ℓ_{max} respectively be the minimal and the maximal length of a word in $\bigcup_{i=1}^{n} S_i \cup T_i$. We define $k' := \ell_{max} + \max(0, k - \ell_{min})$. Note that $k' \geq k$ since $\ell_{min} \leq \ell_{max}$.

Lemma 11. *The language equation $L = R$ has a syntactically k-restricted solution iff the system of language equations E has a semantically k'-restricted solution.*

Proof. If $L = R$ has a syntactically k-restricted solution, then it is easy to see that E has an unrestricted solution. Obviously, this solution is also a semantically k'-restricted solution of E.

Conversely, assume that θ is a semantically k'-restricted solution of E. Additionally, we can assume without loss of generality that this is a solution for which $|\theta(X_1)| + \ldots + |\theta(X_n)|$ is minimal, where $|\cdot|$ denotes set cardinality. If $\bigcup_{i=1}^{n} (S_i \cdot \theta(X_i) \cup T_i \cdot \theta(X_i)) \subseteq \Delta^{\leq k}$, then $k \leq k'$ and $S_0 \cup T_0 \subseteq \Delta^{\leq k}$ imply that θ is a solution of $L = R$ in the unrestricted sense, and thus (due to the assumed inclusions in $\Delta^{\leq k}$) also in the syntactically k-restricted sense.

Now, assume that there is an i and a word $w \in S_i \cdot \theta(X_i) \cup T_i \cdot \theta(X_i)$ such that $|w| > k$. Then, due to the presence of the equation $L \cup Z = Z$ in E, we have $|w| > k'$ or $w \in \theta(Z)$. By Lemma 10, the latter also implies $|w| > k'$. Since $w \in S_i \cdot \theta(X_i) \cup T_i \cdot \theta(X_i)$, there are words $u \in S_i \cup T_i$ and $v \in \theta(X_i)$ such that $w = uv$. Now, $|uv| > k'$ and $|u| \leq \ell_{max}$ yield $|v| > \max(0, k - \ell_{min})$. This implies that $|u'v| > k$ for all $u' \in S_i \cup T_i$. By employing the same argument used above for w, we can show that this actually implies $|u'v| > k'$. But then it is clear that removing v from $\theta(X_i)$ yields a semantically k' restricted solution θ' of E. This contradicts our assumption that θ is a solution for which $|\theta(X_1)| + \ldots + |\theta(X_n)|$ is minimal. Thus, the case where there is an i and a word $w \in S_i \cdot \theta(X_i) \cup T_i \cdot \theta(X_i)$ such that $|w| > k$ cannot occur. This shows that θ is a solution of $L = R$ in the syntactically k-restricted sense. □

If k is encoded in unary, then the system E and the number k' can be constructed from $L = R$ and k in polynomial time. Since syntactically k-restricted solvability is PSpace-hard by Theorem 3, Lemma 11 together with Theorem 4 yields the following complexity result.

Corollary 1. *Given an integer $k \geq 1$ and $\mathcal{FL}_0$ concepts C, D as input, the problem of deciding whether the semantically k-restricted unification problem $C \overset{?}{\equiv}{}^{k}_{sem} D$ has a unifier or not is PSpace-complete if k is assumed to be encoded in unary.*

If k is encoded in binary, then the reduction described above is no longer polynomial since the system E contains $k+1$ variables, and the value k is exponential in the size of the binary representation of k. It is an open problem whether, for the case of binary coding, the ExpTime upper bound in Theorem 4 is tight.

7 The Unification Type

Until now, we were mainly interested in the complexity of deciding solvability of unification problems. For this, it is sufficient to consider ground unifiers. Now, we want to investigate the question of whether all unifiers of a given unification problem can be represented as instances of a finite set of (non-ground) unifiers.

In the unrestricted setting, the instance relation between $\mathcal{FL}_0$ unifiers is defined as follows. Let $C \overset{?}{\equiv} D$ be an $\mathcal{FL}_0$ unification problem, V the set of concept variables occurring in C and D, and σ, θ two unifiers of this problem. We define

$$\sigma \mathrel{\dot{\leq}} \theta \ \text{ if } \ \text{there is a substitution } \lambda \text{ such that } \theta(X) \equiv \lambda(\sigma(X)) \text{ for all } X \in V.$$

If $\sigma \mathrel{\dot{\leq}} \theta$, then we say that θ is an *instance* of σ.

Definition 4. *Let $C \overset{?}{\equiv} D$ be an $\mathcal{FL}_0$ unification problem. The set of substitutions M is called a* complete set of unifiers *for $C \overset{?}{\equiv} D$ if it satisfies*

1. *every element of M is a unifier of $C \overset{?}{\equiv} D$;*
2. *if θ is a unifier of $C \overset{?}{\equiv} D$, then there exists a unifier $\sigma \in M$ such that $\sigma \mathrel{\dot{\leq}} \theta$.*

The set M is a minimal complete set of unifiers *for $C \overset{?}{\equiv} D$ if it additionally satisfies*

3. *if $\sigma, \theta \in M$, then $\sigma \mathrel{\dot{\leq}} \theta$ implies $\sigma = \theta$.*

The unification type of a given unification problem is determined by the existence and cardinality of such a minimal complete set.

Definition 5. *Let $C \overset{?}{\equiv} D$ be an $\mathcal{FL}_0$ unification problem. This problem has type* unitary *(*finitary, infinitary*) if it has a minimal complete set of unifiers of cardinality* 1 *(finite cardinality, infinite cardinality). If $C \overset{?}{\equiv} D$ does not have a minimal complete set of unifiers, then it is of type* zero*.*

The unification types can be ordered as follows:

$$\text{unitary} < \text{finitary} < \text{infinitary} < \text{type zero}.$$

Basically, the unification type of $\mathcal{FL}_0$ is the maximal type of an $\mathcal{FL}_0$ unification problem. However, in unification theory, one usually distinguishes between unification with and without constants [7]. In an $\mathcal{FL}_0$ unification problem with constants, no restrictions are put on the concepts C and D to be unified. In an $\mathcal{FL}_0$ unification problem without constants, C and D must not contain concept names from N_C. The unification type of $\mathcal{FL}_0$ for unification with (without) constants is the maximal type of an $\mathcal{FL}_0$ unification problem with (without) constants.

The unification type of an equational theory is defined analogously (see [7] for details). It was shown in [1] that the unification type of the theory *ACUIh* (called *AIMH* in [1]) is zero, even if one has only one homomorphism h and considers unification without constants. Thus, unification in $\mathcal{FL}_0$ is also of type zero for unification without constants, and thus also for unification with constants. We will show in this section that this is no longer the case if we consider restricted unification. For the semantically restricted case, this is an easy consequence of general results about commutative/monoidal theories [1,14].

7.1 The Semantically Restricted Case

In this case, the instance relation between unifiers, (minimal) complete sets of unifiers, and the unification type are defined as for the unrestricted case, but now using $\equiv^k_{sem}$ instead of $\equiv$ in these definitions.

According to [14], unification without constants in a monoidal theory E is unitary if the semiring S_E corresponding to E is finite. In [1], the same result is shown for commutative theories E under the assumption that the finitely generated E-free algebras are finite. It is easy to see that these two conditions actually coincide for commutative theories [6]. In addition to unification without constants, also unification with constants is considered in [1], and it is shown that, if the finitely generated E-free algebras are finite, then unification with constants in the commutative theory E is at most finitary (i.e., unitary or finitary). The following theorem is an easy consequence of these results.

Theorem 5. *Unification in $ACUIh^k$, and thus also semantically k-restricted unification in $\mathcal{FL}_0$, is unitary for unification without constants and finitary for unification with constants.*

Proof. It is easy to see that the semiring corresponding to $ACUIh^k$ is finite since its elements are all the subsets of the finite set $\Delta^{\leq k}$. Thus, the results in [1,14] yield that $ACUIh^k$ is unitary for unification without constants and at most finitary for unification with constants. The following example shows that the theory is not unitary for unification with constants: if x is a variable and a a constant, then the terms $x \wedge a$ and a have (restricted to x) exactly two $ACUIh^k$ unifiers $\{x \mapsto 1\}$ and $\{x \mapsto a\}$, which are not in any instance relationship. □

7.2 The Syntactically Restricted Case

For the syntactically restricted case, we can actually use equivalence $\equiv$ rather than $\equiv^k_{syn}$ when defining the instance relation between unifiers, (minimal) complete sets of unifiers, and the unification type. But of course the set of unifiers is usually smaller than in the unrestricted case.

To deal with the syntactically restricted case, the results on the unification type for commutative/monoidal theories cannot be applied directly, but we can show the same results as for the semantically restricted case, using the ideas underlying the proofs in [1,14]. We will formulate our proof using the syntax of $\mathcal{FL}_0$ rather than the equational theory variant.

Let C, D be $\mathcal{FL}_0$ concepts and σ a syntactically k-restricted unifier of C and D. Let $X_1, \ldots, X_n$ be the concept variables occurring in C, D and $A_1, \ldots, A_\ell$ the concept constants. First, note that we can assume that σ does not introduce new concept constants since otherwise one could get a more general unifier by replacing such a constant by a new variable. Let $Y_1, \ldots, Y_m$ be the concept variables in the range of σ, where we assume without loss of generality that they are different from the variables $X_1, \ldots, X_n$. For $i = 1, \ldots, n$, the LNF of the concept $\sigma(X_i)$ is of the form

$$\sigma(X_i) = K_i \sqcap \forall L_{i,1}.Y_1 \sqcap \ldots \sqcap \forall L_{i,m}.Y_m, \qquad (9)$$

where K_i is a concept of role depth $\leq k$ not containing concept variables and only concept constants in $\{A_1, \ldots, A_\ell\}$ and the $L_{i,j}$ are subsets of $\Delta^{\leq k}$. Recall that $\forall L_{i,j}.Y_j$ abbreviates the conjunction of the value restrictions $\forall w.Y_j$ for $w \in L_{i,j}$, which in turn is an abbreviation for $\forall r_1. \cdots \forall r_\nu.Y_j$ if $w = r_1 \ldots r_\nu$.

Now, consider for every variable $Y_j, 1 \leq j \leq m$, the tuple of languages $L(Y_j) = (L_{1,j}, \ldots, L_{n,j})$, and assume that there are indices $j \neq j'$ such that $L(Y_j) = L(Y_{j'})$. Let $\theta_{j'}$ be the substitution that replaces $Y_{j'}$ with $\top$ and leaves all other variable Y_μ unchanged. Then $\sigma_{j'} = \sigma\theta_{j'}$ is an instance of σ, which is still a syntactically k-restricted unifier of C and D, but introduces one variable less. Conversely, using the substitution $\lambda_{j,j'} = \{Y_j \mapsto Y_j \sqcap Y_{j'}\}$, we obtain σ as an instance of $\sigma_{j'}$ since $\sigma = \sigma_{j'}\lambda_{j,j'}$.

Let c denote the (finite) cardinality of $\Delta^{\leq k}$. Then there are at most $2^{c \cdot n}$ different n-tuples of subsets of $\Delta^{\leq k}$. Thus, if a syntactically k-restricted unifier of C and D introduces more than $2^{c \cdot n}$ variables, it is an instance of a syntactically k-restricted unifier of C and D that introduces at least one variable less. This observation can be used to show the following lemma.

Lemma 12. *There is a complete set of syntactically k-restricted unifiers of C and D that consists of unifiers whose range contains at most the variables $Y_1, \ldots, Y_m$ for $m = 2^{c \cdot n}$.*

Turning the argument around again, once we have restricted the unifiers in the complete set to ones using only finitely many variables, we know that there can be only finitely many unifiers in this set. In fact, if we consider (9), then we see that the K_i and $L_{i,j}$ range over finite sets. This proves the following theorem.

Theorem 6. *Syntactically k-restricted unification with constants in $\mathcal{FL}_0$ is finitary.*

To show that unification with constants is not unitary, we can use the same example as in the semantically restricted case. Unification without constants is again unitary.

Corollary 2. *Syntactically k-restricted unification without constants in $\mathcal{FL}_0$ is unitary.*

Proof. By the previous theorem, there is a finite complete set $\{\sigma_1, \ldots, \sigma_\kappa\}$ of syntactically k-restricted unifiers of C, D. Without loss of generality, we can assume that the variables occurring in the ranges of these unifiers are disjoint and that no concept constant occurs in the range. The latter assumption can be made since the unification problem itself does not contain such constants. Under these assumptions, the substitution σ defined as

$$\sigma(X_i) = \sigma_1(X_i) \sqcap \ldots \sqcap \sigma_\kappa(X_i) \text{ for } i = 1, \ldots, n$$

is also a syntactically k-restricted unifier of C, D, and it has the substitutions $\sigma_1, \ldots, \sigma_\kappa$ as instances. Thus, $\{\sigma\}$ is a complete set of unifiers.

To make this more precise, the first claim can be shown by proving by induction that $\sigma(E) \equiv \sigma_1(E) \sqcap \ldots \sqcap \sigma_\kappa(E)$ holds for all $\mathcal{FL}_0$ concepts E. This then implies that $\sigma(C) \equiv \sigma_1(C) \sqcap \ldots \sqcap \sigma_\kappa(C) \equiv \sigma_1(D) \sqcap \ldots \sqcap \sigma_\kappa(D) \equiv \sigma(D)$. These equivalences also imply that the role depth of $\sigma(C)$ is equal to the maximum of the role depths of the $\sigma_i(C)$ for $i = 1, \ldots, \kappa$, and the same holds for D. Thus, σ is indeed a syntactically k-restricted unifier of C, D.

Regarding the second claim, it is easy to see that σ_i can be obtained as an instance of σ using the substitution λ_i that replaces all variables in the range of σ not occurring in the range of σ_i with $\top$. Note that this only holds since we have assumed that the ranges of the substitutions σ_j do not contain concept constants and use disjoint sets of variables. $\square$

8 Conclusion

We have investigated both a semantically and a syntactically restricted variant of unification in $\mathcal{FL}_0$, where either the role depth of concepts or the length of role paths in interpretations is restricted by a natural number $k \geq 1$. These restrictions lead to a considerable improvement of the unification type from the worst possible type to unitary/finitary for unification without/with constants. For the complexity of the decision problem, we only obtain an improvement if k is assumed to be encoded in unary.

As future work, we will investigate whether the ExpTime upper bound in Theorem 4 for the case of binary coding of k is tight. In addition, we will consider similar restrictions for other DLs. For example, the unification type of the DL $\mathcal{EL}$ is also known to be zero, and the decision problem is NP-complete [4]. We conjecture that, for this DL, the restricted variants will not lead to an improvement of unification type or complexity.

In [11], a syntactically restricted version of unification in the theory ACh was shown to be decidable, but neither the unification type nor the complexity of the decision problem was determined. It would be interesting to investigate these problems and also consider a semantically restricted variant.

References

1. Franz Baader. Unification in commutative theories. *J. Symbolic Computation*, 8(5):479–497, 1989.
2. Franz Baader, Jan Hladik, and Rafael Peñaloza. Automata can show PSPACE results for description logics. *Information and Computation*, 206(9–10):1045–1056, 2008.
3. Franz Baader, Ian Horrocks, Carsten Lutz, and Ulrike Sattler. *An Introduction to Description Logic*. Cambridge University Press, 2017.
4. Franz Baader and Barbara Morawska. Unification in the description logic $\mathcal{EL}$. *Logical Methods in Computer Science*, 6(3), 2010.
5. Franz Baader and Paliath Narendran. Unification of concept terms in description logics. *J. Symbolic Computation*, 31(3):277–305, 2001.
6. Franz Baader and Werner Nutt. Combination problems for commutative/monoidal theories: How algebra can help in equational reasoning. *J. Applicable Algebra in Engineering, Communication and Computing*, 7(4):309–337, 1996.
7. Franz Baader and Wayne Snyder. Unification theory. In J.A. Robinson and A. Voronkov, editors, *Handbook of Automated Reasoning*, volume I, pages 447–533. Elsevier Science Publishers, 2001.
8. Philippe Balbiani, Cigdem Gencer, Maryam Rostamigiv, and Tinko Tinchev. About the unification type of $\mathbf{K}+\Box\Box\bot$. In *Proc. of the 34th International Workshop on Unification (UNIF 2020)*, pages 4:1–4:6. RISC-Linz, 2020.
9. Michael R. Garey and David S. Johnson. *Computers and Intractability — A guide to NP-completeness*. W. H. Freeman and Company, San Francisco (CA, USA), 1979.
10. Emil Jerabek. Blending margins: The modal logic K has nullary unification type. *J. Logic and Computation*, 25(5):1231–1240, 2015.
11. Ajay Kumar Eeralla and Christopher Lynch. Bounded ACh unification. *Math. Struct. Comput. Sci.*, 30(6):664–682, 2020.
12. Hector J. Levesque and Ron J. Brachman. Expressiveness and tractability in knowledge representation and reasoning. *Computational Intelligence*, 3:78–93, 1987.
13. Paliath Narendran. Solving linear equations over polynomial semirings. In *Proc. of the 11th Annual IEEE Symposium on Logic in Computer Science (LICS 1996)*, pages 466–472. IEEE Computer Society, 1996.
14. Werner Nutt. Unification in monoidal theories. In Mark E. Stickel, editor, *Proc. of the 10th Int. Conf. on Automated Deduction (CADE 1990)*, volume 449 of *Lecture Notes in Artificial Intelligence*, pages 618–632, Kaiserslautern, Germany, 1990. Springer.
15. Rafael Peñaloza and Anni-Yasmin Turhan. A practical approach for computing generalization inferences in $\mathcal{EL}$. In Grigoris Antoniou, Marko Grobelnik, Elena Paslaru Bontas Simperl, Bijan

Parsia, Dimitris Plexousakis, Pieter De Leenheer, and Jeff Z. Pan, editors, *Proc. of the 8th Extended Semantic Web Conference (ESWC 2011)*, volume 6643 of *Lecture Notes in Computer Science*, pages 410–423. Springer, 2011.

16. Helmut Seidl. Haskell overloading is DEXPTIME-complete. *Information Processing Letters*, 52:57–60, 1994.
17. Wolfgang Thomas. Automata on infinite objects. In *Handbook of Theoretical Computer Science*, volume B, chapter 4, pages 134–189. Elsevier Science Publishers (North-Holland), Amsterdam, 1990.