Complexity of language equations with one-sided concatenation and all Boolean operations

Franz Baader[1] and Alexander Okhotin[2,3]*

[1] Theoretical Computer Science, Technical University of Dresden, Germany,
baader@tcs.inf.tu-dresden.de
[2] Department of Mathematics, University of Turku, Finland
[3] Research Group on Mathematical Linguistics, Rovira i Virgili University, Spain
alexander.okhotin@utu.fi

Abstract. Language equations are equations where both the constants occurring in the equations and the solutions are formal languages. They have first been introduced in formal language theory, but are now also considered in other areas of computer science. In particular, they can be seen as unification problems in the algebra of languages whose operations are the Boolean operations and concatenation. They are also closely related to monadic set constraints. In the present paper, we restrict the attention to language equations with one-sided concatenation, but in contrast to previous work on these equations, we allow not just union but all Boolean operations to be used when formulating them. In addition, we are not just interested in deciding solvability of such equations, but also in deciding other properties of the set of solutions, like its cardinality (finite, infinite, uncountable) and whether it contains least/greatest solutions. We show that all these decision problems are ExpTime-complete.

1 Introduction

Unification in equational theories [5] can be seen as solving equations in the free algebra with countably many generators induced by the theory in question. In some cases, however, one also considers unification in arbitrary algebras, not just free ones (see, e.g., [8,16]). In the present paper, the algebra over which we want to solve equations consists of all languages (i.e., sets of words) over a fixed finite alphabet, and the unification problems are built using Boolean operations and (one-sided) concatenation. In unification theory, given a class of unification problems, one is either interested in computing complete sets of unifiers or in deciding solvability. For our language equations, we consider not just solvability (i.e., whether the set of solutions is empty or not), but also more general questions regarding the solution set: is it finite/countable or not; does it contain least/greatest solutions?

* Supported by the Academy of Finland under grant 206039.

In formal language theory, equations with formal languages as constant parameters and unknowns have been studied since the 1960s, when two basic concepts of the theory of computation, finite automata and context-free grammars, were respectively represented as systems of equations with union and one-sided concatenation [7] and with union and unrestricted concatenation [12]. This topic was further studied in the monographs on algebraic automata theory by Salomaa [23] and Conway [11]. For example, it is well-known that the equation $X = AX \cup B$, where A, B are fixed formal languages, has A^*B as a solution. If the empty word does not belong to A, then this is the only solution. Otherwise, A^*B is the least solution (w.r.t. inclusion), and all solutions are of the form A^*C for $C \supseteq B$. Depending on A, B and the available alphabet, the equation may thus have finitely many, countably infinitely many, or even uncountably many solutions. The above equation is an equation with one-sided concatenation since concatenation occurs only on one side of the variable. In contrast, the equation $X = aXb \cup XX \cup \varepsilon$ is not one-sided. Its least solution is the Dyck language of balanced parentheses generated by the context-free grammar $S \to aSb \mid SS \mid \varepsilon$, whereas its greatest solution is $\{a, b\}^*$.

Both examples are *resolved* equations in the sense that their left-hand sides consist of a single variable. If only monotonic operations (in the examples: union and concatenation) are used, then such resolved equations always have a least and greatest solution due to the Tarski-Knaster fixpoint theorem [26]. Once the resolved form of equations is no longer required or non-monotonic operations (like complement) are used, a given language equation need no longer have solutions, and thus the problem of deciding solvability of such an equation becomes non-trivial. The same is true for other decision problems, like asking for the existence of a least/greatest solution or determining the cardinality of the set of solutions.

In the case of *language equations with unrestricted concatenation*, the solvability problem becomes undecidable since the intersection emptiness problem of context-free languages can easily be encoded [10]. A systematic study of the hardness of decision problems for language equations with unrestricted concatenation (i.e., the position of these problems in the arithmetic hierarchy) was carried out by Okhotin [19,20,21], who also characterized recursive and recursively enumerable sets by solutions of language equations. A surprising proof of the computational universality of very simple language equations of the form $LX = XL$, where L is a finite constant language, has recently been given by Kunc [14]. Though such equations are syntactically close to word equations (i.e., unification problems modulo associativity) [15], like the equation $aX = Xa$, there is no strong relationship between the two types of equations since the unknowns stand for different mathematical objects: a single word in the case of word equations versus a set of words in the case of language equations. In principle, the relationship between word equations and languages equations is similar to the relationship between syntactic unification problems and set constraints [1], where instead of terms one considers sets of terms, and also allows (certain) Boolean operations to occur in the equations.

Language equations with one-sided concatenation usually do not have undecidable decision problems. In fact, many properties of the solution sets of such equations, such as existence and uniqueness of their solutions, can be expressed in Rabin's monadic second-order logic on infinite trees [22]. This implies the decidability of these problems, but only yields a non-elementary complexity upper-bound [25]. Language equations with one-sided concatenation can also be regarded as a particular case *set constraints*, which received significant attention in the literature [1,9,13] since they can, e.g., be used in program analysis. In fact, language equations with one-sided concatenation correspond to monadic set constraints, where all function symbols are unary. Thus, decidability results for set constraints also yield decidability results for the corresponding language equations. However, since set constraints are in general more complex than monadic set constraints, this does not necessarily yield optimal complexity bounds. Language equations with one-sided concatenation *and union* have been studied in the context of unification problems in description logics: Baader and Narendran [3] show that the existence of a finite solution (i.e., a solution where all unknowns are replaced by finite languages) is an ExpTime-complete problem; Baader and Küsters [2] show the same for the existence of an arbitrary (possibly infinite) solution. In the latter work, it is also shown that a solvable equation always has a greatest solution, and that this solution is regular (i.e., consists of regular languages).

The present paper extends the results of [2] in two directions.[4] On the one hand, we consider language equations with one-sided concatenation and *all Boolean operations*, and on the other hand we consider *additional decision problems*, like determining the existence of least/greatest solutions and the cardinality of the solution set. All these problems turn out to be ExpTime-complete for language equations with one-sided concatenation and any set of available Boolean operations between $\{\cup\}$ and $\{\cup, \cap, \neg\}$. After a preliminary section in which we give the relevant definitions, we first concentrate in Section 3 on showing the ExpTime upper-bounds for the mentioned decision problems in the case of the most general type of one-sided equations where all Boolean operations are available. This is done by translating language equations into a special kind of looping tree automata, showing a 1–1-relationship between the solutions of the equation and the runs of the corresponding automaton, and then characterizing the relevant properties of solution sets by decidable properties of the automaton. Thus, we have a uniform approach for solving all decision problems by one automaton construction. The decision procedures for the respective problems only differ in what property of the constructed automaton must be decided. In Section 4, we then show the ExpTime lower-bounds for the mentioned decision problems in the case of one-sided language equations with union: the reduction is from the intersection emptiness problem for deterministic looping tree automata, whose ExpTime-completeness easily follows from the ExpTime-completeness of the same problem for deterministic top-down tree automata on

[4] Detailed proofs of our new results are given in [4].

finite trees [24,2]. Again, the hardness proofs are uniform: one reduction shows hardness of all decision problems under consideration.

2 Preliminaries

In this section, we first introduce the language equations investigated in this paper, and show that they can be transformed into a simpler normal form. Then, we introduce some notions regarding automata working on infinite trees.

2.1 Language equations with one-sided concatenation

For a fixed finite alphabet Σ, we consider systems of equations of the following general form:

$$\begin{cases} \psi_1(X_1, \ldots, X_n) = \xi_1(X_1, \ldots, X_n) \\ \qquad\qquad \vdots \\ \psi_m(X_1, \ldots, X_n) = \xi_m(X_1, \ldots, X_n) \end{cases} \tag{1}$$

where the form of the expressions ψ_i and ξ_i is defined inductively:

- any variable X_i is an expression;
- any regular language $L \subseteq \Sigma^*$ is an expression;
- a concatenation φL of an expression φ and a regular language $L \subseteq \Sigma^*$ is an expression;
- if φ, φ' are expressions, then so are $(\varphi \cup \varphi')$, $(\varphi \cap \varphi')$ and $(\sim\varphi)$.

We assume that the regular languages in expressions are given by nondeterministic finite automata. An effective description of a system (1) would contain transition tables and accepting states of these automata, and thus the number of their states and transitions adds to the size of the description.

If the expressions in such a system contain neither intersection nor complement, then we call it a system of language equations with one-sided concatenation and *union*.

A *solution* of a general system (1) is a vector of languages $(L_1, \ldots, L_n)$ such that a substitution of L_j for X_j for all j turns each instantiated equation into an equality. Solutions can be compared w.r.t. inclusion of their components: we define $(L_1, \ldots, L_n) \preceq (L'_1, \ldots, L'_n)$ iff $L_i \subseteq L'_i$ holds for $i = 1, \ldots, n$. In addition to the problem of deciding whether a system has a solution or not, we consider additional decision problems that look more closely at properties of the set of solutions: its cardinality (is there a unique solution, are there finitely or infinitely many solutions, are there countably or uncountably many solutions) and whether it contains least/greatest elements w.r.t. $\preceq$.

In order to design algorithms for solving these decision problems, it is more convenient to consider language equations in the following normal form: a single equation

$$\varphi(Z_1, \ldots, Z_k) = \varnothing, \tag{2}$$

in the unknowns $Z_1, \ldots, Z_k$, where the constant regular languages occurring in φ are singleton languages $\{\varepsilon\}$ and $\{a\}$ for $a \in \Sigma$, which we simply write as ε and a.

The next lemma implies that w.r.t. all decision problems concerned with the cardinality of the set of solutions (including the existence of a solution), the restriction to equations of form (2) is without loss of generality.

Lemma 1. *For every system (1) in the unknowns $X_1, \ldots, X_n$ we can construct in polynomial time an equation (2) in the unknowns $X_1, \ldots, X_n, Y_1, \ldots, Y_\ell$ for some $\ell \geqslant 0$ such that the set of solutions of (2) is*

$$\big\{\big(L_1, \ldots, L_n, \eta_1(L_1, \ldots, L_n), \ldots, \eta_\ell(L_1, \ldots, L_n)\big) \;\big|\; (L_1, \ldots, L_n) \text{ solves (1)}\big\}$$

for some functions $\eta_1, \ldots \eta_\ell : (2^{\Sigma^*})^n \to 2^{\Sigma^*}$. The size of the resulting equation is linear in the size of the original system.

Proof sketch: Regular languages in (1) can be expressed by employing resolved equations for additional variables $Y_1, \ldots, Y_\ell$. For example, the expression $(\sim X)a^*b$ can be replaced by Y_2 if we add the resolved equations $Y_2 = Y_1 b$ and $Y_1 = Y_1 a \cup \sim X$. Since resolved equations of this form have a unique solution, any value for X yields unique values for Y_1, Y_2. The total size of equations added is proportional to the number of transitions in an NFA, and hence the growth is linear.

Every equation $\psi_i = \xi_i$ has the same solutions as $(\psi_i \cap \sim\xi_i) \cup (\xi \cap \sim\psi_i) = \varnothing$, and the system $\varphi_1 = \varnothing, \varphi_2 = \varnothing$ has the same solutions as $\varphi_1 \cup \varphi_2 = \varnothing$. □

Regarding the existence of least/greatest solutions, we must be more careful. For example, when representing $(\sim X)a^*b$ by Y_2 and the equations $Y_2 = Y_1 b, Y_1 = Y_1 a \cup \sim X$, a larger value for X yields smaller values for Y_1, Y_2. Thus, even if the original system has a least/greatest solution, the new one need not have one. The solution to this problem will be that when defining the relation $\preceq$ on solutions, we do not necessarily compare solutions w.r.t. all components, but only w.r.t. the components corresponding to a set of *focus variables*.[5] In this case, the constructed system (2) with unknowns $X_1, \ldots, X_n, Y_1, \ldots, Y_\ell$ has a least/greatest solution w.r.t. the focus variables $X_1, \ldots, X_n$ iff the original system (1) has a least/greatest solution.

2.2 Automata on infinite trees

Given a ranked alphabet Γ where every symbol has a rank > 0, infinite trees over Γ are defined in the usual way, i.e., every node in the tree is labeled with an element $f \in \Gamma$ and has rank of f many successor nodes. A *looping tree automaton* $\mathcal{A} = (Q, \Gamma, Q_0, \Delta)$ consists of a finite set of states Q, a ranked alphabet Γ, a set of initial states $Q_0 \subseteq Q$, and a transition function $\Delta : Q \times \Gamma \to 2^{Q^*}$ that maps each pair (q, f) to a subset of Q^k where k is the rank of f. This automaton is *deterministic* if $|Q_0| = 1$ and $|\Delta(q, f)| \leqslant 1$ for all pairs (q, f). A *run* r of $\mathcal{A}$ on a

[5] Note that $\preceq$ is then no longer a partial order but only a preorder.

tree t labels the nodes of t with elements of Q such that the root is labeled with $q_0 \in Q_0$, and the labels respect the transition function, i.e., if node v has label $t(v)$ in t and label $r(v)$ in r, then the tuple $(q_1, \ldots, q_k)$ labeling the successors of v in r must belong to $\Delta(r(v), t(v))$. The tree t is *accepted* by $\mathcal{A}$ if there is a run of $\mathcal{A}$ on t. The *language accepted by* $\mathcal{A}$ is defined as

$$L(\mathcal{A}) := \{t \mid t \text{ is an infinite tree over } \Gamma \text{ that is accepted by } \mathcal{A}\}.$$

It is well-known that the emptiness problem for looping tree automata, i.e., the question whether the accepted language is non-empty, is decidable in linear time (see, e.g., [6]). However, the intersection emptiness problem, i.e., given looping tree automata $\mathcal{A}_1, \ldots, \mathcal{A}_k$, is $L(\mathcal{A}_1) \cap \ldots \cap L(\mathcal{A}_k)$ empty or not, is ExpTime-complete even for deterministic automata [24,2]. This result will be used to show the complexity lower-bounds in Section 4.

When showing the complexity upper-bounds in Section 3, we actually employ a very restricted form of looping automata. First, we restrict the attention to a ranked alphabet Γ containing a single symbol γ of some fixed rank $k > 0$. Thus, there is only one infinite tree, and the labeling of its nodes by γ can be ignored. Given an arbitrary finite alphabet $\Sigma := \{a_1, \ldots, a_k\}$ of cardinality k, every node in this tree can uniquely be represented by a word $w \in \Sigma^*$, where a_i corresponds to the ith successor. Second, we consider not arbitrary looping tree automata working on this tree, but tree automata induced by word automata. A nondeterministic finite automaton (NFA) $A = (Q, \Sigma, Q_0, \delta)$ without accepting states working on words over Σ induces a looping tree automaton $\mathcal{A} = (Q, \Gamma, Q_0, \Delta)$ working on the infinite tree over Γ as follows:

$$\Delta(q, \gamma) := \{(q_1, \ldots, q_k) \mid q_i \in \delta(q, a_i) \text{ for } i = 1, \ldots, k\}.$$

We call such an automaton *looping tree automaton with independent transitions (ILTA)* since in every component the successor states can be chosen independently from what is chosen in another component. In the following, we do not distinguish between the NFA and the ILTA it represents. For example, we will talk about runs of the NFA, but mean the runs of the corresponding ILTA. The runs of the NFA $A = (Q, \Sigma, Q_0, \delta)$ can thus be represented as functions $r : \Sigma^* \to Q$ such that $r(\varepsilon) \in Q_0$ and $r(wa) \in \delta(r(w), a)$ for all $w \in \Sigma^*$ and $a \in \Sigma$. In addition, when defining an ILTA, we will usually introduce just the corresponding NFA, and call it ILTA. In the next section, we are not interested in the tree language accepted by an ILTA (which is either empty or a singleton set); instead, we are interested in the runs themselves.

Following the definition of looping tree automata, an ILTA is called *deterministic* if $|\delta(q, a)| \leqslant 1$ for all $q \in Q$ and $a \in \Sigma$, that is, if the underlying NFA is a partial DFA. Note that a deterministic ILTA has at most one run; furthermore, having a completely defined function δ is a sufficient condition of having exactly one run.

We call an NFA $A = (Q, \Sigma, Q_0, \delta)$ and the ILTA it represents *trim* if every state is reachable from an initial state, and $\delta(q, a) \neq \varnothing$ for all $q \in Q$ and $a \in \Sigma$. It is easy to see that every NFA can be transformed in polynomial time into

a trim NFA having the same runs. In such a trim NFA, every finite or infinite path can be completed to a run containing it. In addition, it has a run iff Q is non-empty.

3 The complexity upper-bounds

In this section we show that all the decision problems for language equations with one-sided concatenation introduced above can be solved within deterministic exponential time. To this purpose, we show how to translate a given language equation in normal form $\varphi = \varnothing$ into an ILTA such that there is a 1–1-correspondence between the solutions of the equation and the runs of the corresponding ILTA.

3.1 Translating language equations into ILTA

Let $\Sigma = \{a_1, \ldots, a_m\}$, and $\varphi(X_1, \ldots, X_n)$ be an expression. In the following, we assume that φ is fixed, and denote the set of its subexpressions by Φ. We assume that $\varepsilon, X_1, \ldots, X_n \in \Phi$ (otherwise, we simply add them). Let $\Phi_0 = \{\psi a \mid a \in \Sigma, \psi a \in \Phi\} \cup \{\varepsilon\}$ and $\Phi_1 = \Phi_0 \cup \{X_1, \ldots, X_n\}$. We define two elementary operations on subsets of Φ. The first of them, *select*, maps a set $q_0 \subseteq \Phi_0$ to a collection of subsets of Φ_1:

$$select(q_0) = \{q \subseteq \Phi_1 \mid q \setminus \{X_1, \ldots, X_n\} = q_0\}$$

Note that $|select(q_0)| = 2^n$, and the elements of $select(q_0)$ correspond to different choices of a set of variables.

The other operation, *closure*, completes a subset $q \subseteq \Phi_1$ by computing all applicable Boolean operations over these subexpressions. In order to define the set $closure(q) \subseteq \Phi$, we specify for every expression $\xi \in \Phi$ whether $\xi \in closure(q)$ or not by induction on the structure of ξ:

Base case: For each $\xi \in \{\varepsilon, X_1, \ldots, X_n\}$, let $\xi \in closure(q)$ iff $\xi \in q$.
Induction step: Consider $\xi \in \Phi \setminus \{\varepsilon, X_1, \ldots, X_n\}$ and assume that the membership of all proper subexpressions of ξ in $closure(q)$ has already been defined. There are four cases depending on the top operation of ξ:

- If ξ is of the form ψc, then $\xi \in closure(q)$ iff $\xi \in q$.
- If $\xi = \psi \cup \eta$, then $\xi \in closure(q)$ iff at least one of ψ, η is in $closure(q)$.
- If $\xi = \psi \cap \eta$, then $\xi \in closure(q)$ iff both ψ and η are in $closure(q)$.
- If $\xi = \sim\psi$, then $\xi \in closure(q)$ iff ψ is not in $closure(q)$.

Definition 1. *The* ILTA $A = (\Sigma, Q, Q_0, \delta)$ induced by the expression φ *is defined as* $Q := 2^{\Phi}$, $Q_0 := \{closure(q) \mid q \in select(\{\varepsilon\})\}$, *and* $\delta(q, a) := \{closure(q') \mid q' \in select(\{\psi a \in \Phi \mid \psi \in q\})\}$.

Note that $|Q_0| = 2^n$ and $|\delta(q, a)| = 2^n$ for all $q \in Q$ and $a \in \Sigma$. Intuitively, the non-determinism is used to "guess" the values of the variables.

There exists a one-to-one correspondence between the runs of A and n-tuples of languages over Σ. First, we show how to associate a run with every vector of

languages. The run $r_L : \Sigma^* \to Q$ corresponding to $L = (L_1, \ldots, L_n)$ is defined inductively as:

$$r_L(\varepsilon) = closure(\{\varepsilon\} \cup \{X_i \mid \varepsilon \in L_i\}) \tag{3a}$$

$$r_L(wa) = closure(\{\psi a \in \Phi \mid \psi \in r_L(w)\} \cup \{X_i \mid wa \in L_i\}) \tag{3b}$$

It is easy to see that r_L is indeed a run of A.

Conversely, a given run $r : \Sigma^* \to Q$ induces the vector of languages $L^r := (L_1^r, \ldots, L_n^r)$, where $L_i^r := \{w \mid X_i \in r(w)\}$.

Lemma 2. *The mapping of runs to vectors of languages introduced above is a bijection, and the mapping of vectors of languages to runs is its inverse.*

For each run r_L, the set of subexpressions in a state $r_L(w)$ (for each string $w \in \Sigma^*$) contains exactly those subexpressions that produce this string when replacing $X_1, \ldots, X_n$ by $L_1, \ldots, L_n$:

Lemma 3. *Let $L = (L_1, \ldots, L_n)$ be a vector of languages and r_L be the corresponding run. Then, for every $w \in \Sigma^*$ and $\xi \in \Phi$, we have $w \in \xi(L)$ iff $\xi \in r_L(w)$.*

Since the vector $L = (L_1, \ldots, L_n)$ is a solution of $\varphi(X_1, \ldots, X_n) = \varnothing$ iff $w \notin \varphi(L)$ for all $w \in \Sigma^*$, this lemma implies the following characterization of the runs corresponding to solutions:

Proposition 1. *The vector $L = (L_1, \ldots, L_n)$ is a solution of the equation $\varphi(X_1, \ldots, X_n) = \varnothing$ iff $\varphi \notin r_L(w)$ for every $w \in \Sigma^*$.*

Consequently, if we remove from A all states containing φ, then we obtain an automaton whose runs are in a 1–1-correspondence with the solutions of $\varphi(X_1, \ldots, X_n) = \varnothing$. In addition, we can make this automaton trim without losing any runs/solutions. Let us call the resulting ILTA A_φ. Obviously, the size of A_φ is exponential in the size of φ, and this automaton can be constructed in exponential time.

Proposition 2. *For every language equation $\varphi(X_1, \ldots, X_n) = \varnothing$ of the form (2) one can construct in exponential time a trim ILTA A_φ whose states are subsets of the set of strict subexpressions of φ such that the mapping $r \mapsto L^r = (L_1^r, \ldots, L_n^r)$ is a bijection between the runs of A_φ and the solutions of $\varphi(X_1, \ldots, X_n) = \varnothing$.*

3.2 Counting the number of solutions

As an immediate consequence of Proposition 2, (unique) solvability of a language equation can be characterized as follows:

Proposition 3. *A language equation $\varphi = \varnothing$ with one-sided concatenation has*

- *at least one solution iff the corresponding ILTA A_φ is non-empty.*

– *exactly one solution iff the corresponding ILTA A_φ is non-empty and deterministic.*

Before we can characterize finitely many solutions, we must introduce some notation.

Definition 2. *Let $A = (\Sigma, Q, Q_0, \delta)$ be an ILTA. A state $q \in Q$ is* cyclic *if $q \in \delta(q, w)$ for some $w \in \Sigma^+$, and it is* branching *if $|\delta(q, a)| > 1$ for some $a \in \Sigma$.*

Lemma 4. *A trim ILTA $A = (\Sigma, Q, Q_0, \delta)$ has finitely many runs iff no branching state is reachable from any cyclic state.*

The condition in this lemma can obviously be tested in time polynomial in the size of the ILTA since it is basically a reachability problem. The conditions in the previous proposition can trivially be tested in time polynomial in the size of A_φ. Since the size of A_φ is exponential in the size of φ, we thus obtain the following complexity upper-bounds:

Theorem 1. *The problems of testing whether a language equation with one-sided concatenation has a solution, a unique solution, or finitely many solutions are decidable in deterministic exponential time.*

Note that an ExpTime decision procedure for the solvability problem was already sketched in [1]. The other two results are new. Regarding the cardinality of the solution set, it remains to show how we can decide whether an equation has countably or uncountably many solutions. For this purpose, we adapt Niwiński's condition for countability of the language accepted by a Rabin tree automaton [17] to our situation of counting runs of ILTAs.[6] If A is an ILTA and q one of its states, then a *q-run* is defined like a run, with the only exception that instead of requiring that the root is labeled with an initial state we require that it is labeled with q. Two q-runs r_1, r_2 are called *essentially different* if there are words v_1, v_2, w such that

– $r_1(v_1) = q = r_2(v_2)$ and v_1, v_2 are not the empty word,
– $r_1(w) \neq r_2(w)$ and w has neither v_1 nor v_2 as prefix.

Proposition 4 (Niwiński). *An ILTA has uncountably many runs iff it has a state q such that there are two essentially different q-runs.*

In contrast to the previous conditions, it is not immediately clear how this condition can be decided in time polynomial in the size of the ILTA. In [4] we show this by reducing the problem to the emptiness problem for Büchi tree automata. To compare, Niwiński proves for his condition for Rabin automata only an elementary upper bound.

Theorem 2. *The problem of testing whether a language equation with one-sided concatenation has countably many solutions is decidable in exponential time.*

[6] Actually, we never use that the automaton has independent transitions, and thus the results stated below also hold for arbitrary looping tree automata.

3.3 Least and greatest solutions

As pointed out before, we must compare solution vectors not on all components, but only on those components corresponding to a set of focus variables. Let $\varphi(X_1, \ldots, X_n, Y_1, \ldots, Y_\ell) = \varnothing$ be a language equation with one-sided concatenation, and $X_1, \ldots, X_n$ be the set of focus variables. Given vectors of languages $L = (L_1, \ldots, L_n, L_{n+1}, \ldots, L_{n+\ell}), L = (L'_1, \ldots, L'_n, L'_{n+1}, \ldots, L'_{n+\ell})$ we define $L \preceq L'$ iff $L_i \subseteq L'_i$ for all $i = 1, \ldots, n$.

Let $A_\varphi = (\Sigma, Q, Q_0, \delta)$ be the ILTA corresponding to the above language equation with focus variables $X_1, \ldots, X_n$. We define a preorder on its set of states Q as follows:

$$q \preccurlyeq q' \text{ iff } q \cap \{X_1, \ldots, X_n\} \subseteq q' \cap \{X_1, \ldots, X_n\}.$$

This preorder on states defines the following preorder on runs of A: for any $r, r' : \Sigma^* \to Q$ we say that $r \preccurlyeq r'$ if $r(w) \preccurlyeq r'(w)$ for all $w \in \Sigma^*$. As an easy consequence of the definition of the mapping $L \mapsto r_L$ we obtain that it is a preorder isomorphism:

Lemma 5. *Let L, L' be vectors of languages. Then $L \preceq L'$ iff $r_L \preccurlyeq r_{L'}$.*

Consequently, to decide whether the equation $\varphi = \varnothing$ has a least/greatest solution w.r.t. $\preceq$, it is enough to decide whether A_φ has a least/greatest run w.r.t. $\preccurlyeq$. To show that this is decidable in polynomial time, we introduce another preorder $\sqsubseteq$ on Q as follows: $q \sqsubseteq q'$ iff there exists a run r with root label q such that, for every run r' with root label q', we have $r \preccurlyeq r'$.

Lemma 6. *For every trim ILTA $A = (\Sigma, Q, Q_0, \delta)$ and for every polynomial time decidable preorder $\preccurlyeq$ on Q, the corresponding preorder $\sqsubseteq$ on Q can be constructed in time polynomial in $|Q|$. In addition, A has a least run with respect to the preorder $\preccurlyeq$ on Q iff Q_0 has a least element with respect to $\sqsubseteq$.*

Since the size of A_φ is exponential in the size of φ, we thus obtain the following complexity upper bound for deciding the existence of a least solution. (Greatest solutions can be treated analogously.)

Theorem 3. *The problem of testing whether a language equation with one-sided concatenation has a least (greatest) solution is decidable in* ExpTime.

4 The complexity lower-bounds

We show that the decision problems for language equations introduced in Section 2 are ExpTime-hard already for language equations with one-sided concatenation *and union*. For solvability, this was already shown in [2]. Since it was also shown there that such an equation has a solution iff it has a greatest solution, ExpTime-hardness of the existence of a greatest solution follows from this result as well. In the following we will concentrate on the remaining decision problems. Similarly to [2], we show ExpTime-hardness by a reduction from the intersection emptiness problem for deterministic looping tree automata. First, we show how trees can be represented as languages.

4.1 Representing infinite trees by languages

Given a ranked alphabet Γ, we use the alphabet $\Sigma_\Gamma := \{f^{[i]} \mid f \in \Gamma, 1 \leqslant i \leqslant \mathit{rank}\, f\}$ as the alphabet underlying our language equations. For every infinite tree t over Γ, we define a representation of t as a string language over Σ_Γ:

$$S(t) = \{f_1^{[i_1]} \dots f_\ell^{[i_\ell]} \mid \ell \geqslant 0,\ t \text{ has a path with label } f_1, \dots, f_\ell, f_{\ell+1}, \text{ in which } f_1 \text{ labels the root of } t, \text{ and each } f_{j+1} \text{ labels the } i_j\text{-th successor of the node with label } f_j\} \tag{4}$$

The strings in $S(t)$ unambiguously encode finite prefixes of paths in t. Obviously, for every infinite tree $f(t_1, \dots, t_k)$, the following holds:

$$S(f(t_1, \dots, t_k)) = \{\varepsilon\} \cup \bigcup_{i=1}^{k} \{f^{[i]} u \mid u \in S(t_i)\}$$

The following lemma characterizes the languages of the form $S(t)$:

Lemma 7. *A language $L \subseteq \Sigma_\Gamma^*$ is of the form $L = S(t)$ for some infinite tree t iff*

1. *$\varepsilon \in L$;*
2. *for every $w \in L$ there exists a unique symbol $f \in \Gamma$, such that $wf^{[1]} \in L$;*
3. *if $wf^{[i]} \in L$, then $wf^{[j]} \in L$ for every j ($1 \leqslant j \leqslant \mathit{rank}\, f$);*
4. *for every $w \in \Sigma_\Gamma^*$ and $f^{[i]} \in \Sigma_\Gamma$, $wf^{[i]} \in L$ implies $w \in L$.*

The mapping S is extended in the obvious way to sets of trees: $S(T) := \bigcup_{t \in T} S(t)$. We also consider the "inverse" operation $S^{-1}(L) := \{t \mid S(t) \subseteq L\}$.

Lemma 8. *For every set of trees T, $T \subseteq S^{-1}(S(T))$ and $S(S^{-1}(S(T))) = S(T)$.*

4.2 Representing looping tree automata by language equations

Let $\mathcal{A} = (Q, \Gamma, \{q_0\}, \Delta)$ be a *deterministic* looping tree automaton over Γ, where Δ is represented as a partial function from $Q \times \Gamma$ to Q^*. We introduce another partial function $\mathfrak{q} : \Sigma_\Gamma^* \to Q$ that simulates the operation of $\mathcal{A}$ on a finite prefix of a single path encoded as in (4). Define $\mathfrak{q}(w)$ inductively on the length of w: $\mathfrak{q}(\varepsilon) = q_0$, while $\mathfrak{q}(uf^{[i]})$ is defined as the i-th component of $\Delta(\mathfrak{q}(u), f)$ if this transition is defined, and undefined otherwise. Basically, if $\mathfrak{q}(u)$ is defined, then it gives the unique label of the node corresponding to u in a run of $\mathcal{A}$ on a tree containing the path encoded by u.

Now define a system of language equations (5) over the alphabet $\Sigma_\Gamma \cup Q$, which simulates the computation of the automaton $\mathcal{A}$. The set of variables of this system is $\{X_{q,f} \mid \Delta(q, f) \text{ is defined}\} \cup \{X_0\}$, and the system consists of the

two equations

$$\bigcup_{\Delta(q,\,f)\text{ is defined}} X_{q,f} \cdot \{q\} = \{q_0\} \cup \bigcup_{\Delta(q,f)=(q_1,\ldots,q_k)} X_{q,f} \cdot \{f^{[1]}q_1, \ldots, f^{[k]}q_k\} \tag{5a}$$

$$X_0 = \bigcup_{\Delta(q,\,f)\text{ is defined}} X_{q,f} \tag{5b}$$

The following lemma establishes some basic properties of solutions of this system.

Lemma 9. *For every solution $(\ldots, L_{q,f}, \ldots, L_0)$ of (5),*

1. *$w \in L_{q,f}$ iff $\mathfrak{q}(w) = q$ and $wf^{[i]} \in L_0$ for all i ($1 \leqslant i \leqslant \operatorname{rank} f$).*
2. *If $w \in L_{q,f}$ for some $q \in Q$, then there exists an infinite tree t such that $\{wf^{[1]}, \ldots, wf^{[\operatorname{rank} f]}\} \subseteq S(t) \subseteq L_0$.*

Based on this lemma and the properties of the mapping S mentioned above, the following characterization of solutions of (5) is shown in [4].

Proposition 5. *A vector of languages $(\ldots, L_{q,f}, \ldots, L_0)$ is a solution of (5) iff*

$$\begin{aligned} &\varnothing \subset S^{-1}(L_0) \subseteq L(\mathcal{A}),\\ &L_{q,f} = \{w \mid \mathfrak{q}(w) = q,\ wf^{[i]} \in L_0 \text{ for all } i\} \quad (\Delta(q,f) \text{ is defined}), \end{aligned} \tag{6}$$

and there exists a set of trees T such that $L_0 = S(T)$.

This shows that the language L_0 substituted for X_0 determines the whole solution.

4.3 Complexity of the decision problems

The next theorem summarizes the main results of this paper.

Theorem 4. *The problems of testing, for a given system of language equations with one-sided concatenation and any set of Boolean operations containing union, whether*

1. *it has a solution,*
2. *it has a unique solution,*
3. *it has finitely many solutions,*
4. *it has countably many solutions,*
5. *it has a least (greatest) solution with respect to componentwise inclusion*

are all ExpTime*-complete.*

Given the results shown in Section 3 and in [1,2], it is enough to prove that testing whether a system of language equations with one-sided concatenation *and union* has a unique solution, finitely many solutions, countably many solutions, and a least solution, respectively, are ExpTime-hard problems.

All four cases are proved by a single reduction from the ExpTime-complete intersection emptiness problem for deterministic looping tree automata [24,2]. Let $\mathcal{A}_1, \ldots, \mathcal{A}_n$ be deterministic looping tree automata over a common ranked alphabet Γ, and assume without loss of generality that their sets of states $Q_1, \ldots, Q_n$ are pairwise disjoint and that the initial state $q_0^{(i)}$ of every $\mathcal{A}_i$ is not reachable, i.e., it never occurs on the right-hand side of a transition.

We augment Γ with a new unary symbol f_{triv}, and transform each automaton $\mathcal{A}_i$ into an automaton $\mathcal{A}'_i$ over the alphabet $\Gamma' = \Gamma \cup \{f_{triv}\}$ by adding the extra transition $(q_0^{(i)}, f_{triv}) \to q_0^{(i)}$. The set of trees accepted by $\mathcal{A}'_i$ equals $\{f_{triv}^{\ell}(t) \mid \ell \geqslant 0, t \in L(\mathcal{A}_i)\} \cup \{t_{triv}\}$, where t_{triv} denotes an infinite branch with all vertices labeled by f_{triv}. Consequently, the intersection $\bigcap_{i=1}^{n} L(\mathcal{A}'_i)$ is equal to $\{f_{triv}^{\ell}(t) \mid \ell \geqslant 0, t \in \bigcap_{i=1}^{n} L(\mathcal{A}_i)\} \cup \{t_{triv}\}$.

For each automaton $\mathcal{A}'_i$, construct two language equations of the form (5), and consider the resulting system of $2n$ equations, which share a common variable X_0. It is easy to show that the vector of languages $L_{triv} := (\ldots, L_{q,f}^{(i)}, \ldots, L_0)$ defined by

$$L_0 := S(t_{triv}) \quad \text{and} \quad L_{q,f}^{(i)} \text{ determined by } L_0 \text{ and } \mathcal{A}'_i \text{ according to (6)}$$

is always a solution of the system. Whether the system has any other solutions depends on whether $\bigcap_{i=1}^{n} L(\mathcal{A}_i)$ is empty or not.

- If $\bigcap_{i=1}^{n} L(\mathcal{A}_i) = \varnothing$, then $\bigcap_{i=1}^{n} L(\mathcal{A}'_i) = \{t_{triv}\}$. We can prove that the system of language equations then has the unique solution L_{triv}.
- If $\bigcap_{i=1}^{n} L(\mathcal{A}_i) \neq \varnothing$, then there exists a tree $t_0 \in \bigcap_{i=1}^{n} L(\mathcal{A}_i)$, and $f_{triv}^{\ell}(t_0) \in \bigcap_{i=1}^{n} L(\mathcal{A}'_i)$ for all $\ell \geqslant 0$. Let us construct uncountably many solutions of the system. For every nonempty set of integers $N \subseteq \mathbb{N}$, define the set of trees

$$T_N = \{f_{triv}^{\ell}(t_0) \mid \ell \in N\}.$$

  The vector of languages $(\ldots, L_{q,f,N}^{(i)}, \ldots, L_{0,N})$ determined by $L_{0,N} := S(T_N)$ according to (6) can be shown to be a solution of the system.

Since the constructed system of language equations has either exactly one or uncountably many solutions, we can conclude that it has a unique solution (finitely many solutions, countably many solutions) iff the intersection of the languages recognized by the n given deterministic looping tree automata is empty. Similarly, we can show that there is a least solution iff the given automata have an empty intersection. In fact, if the intersection is nonempty, then we can construct a pair of incomparable minimal solutions as $L_0 := S(t_{triv})$ and $L'_0 := S(T_{\{0\}}) = S(t_0)$, where t_0 and $T_{\{0\}}$ are defined as in the previous case.

This completes the proof of Theorem 4.

5 Conclusion

We have shown that several interesting decision problems for language equations with one-sided concatenation are ExpTime-complete. The decision procedures based on the construction of an ILTA have been implemented. This implementation does not just answer yes or no; in case there is a (least, greatest) solution, its DFA is constructed [4].

Acknowledgment.
We thank Thomas Wilke for alerting us to the work of Niwiński, and Moshe Vardi for suggesting the name "looping tree automata with *independent* transitions".

References

1. A. Aiken, D. Kozen, M. Y. Vardi, E. L. Wimmers, "The complexity of set constraints", *Computer Science Logic* (CSL 1993, Swansea, UK, September 13–17, 1993), LNCS 832, 1–17.
2. F. Baader, R. Küsters, "Unification in a description logic with transitive closure of roles", *Logic for Programming, Artificial Intelligence, and Reasoning* (LPAR 2001, Havana, Cuba, December 3–7, 2001), LNCS 2250, 217–232.
3. F. Baader, P. Narendran, "Unification of concept terms in description logic", *Journal of Symbolic Computation*, 31 (2001), 277–305.
4. F. Baader, A. Okhotin, *On Language Equations with One-Sided Concatenation*, LTCS-Report 06-01, Institute for Theoretical Computer Science, Dresden University of Technology, 2006. See `http://lat.inf.tu-dresden.de/research/reports.html`
5. F. Baader and W. Snyder. Unification theory. In *Handbook of Automated Reasoning*, volume I. Elsevier, 2001.
6. F. Baader, S. Tobies, "The inverse method implements the automata approach for modal satisfiability", In *Proc. IJCAR'01*, Springer LNCS 2083, 2001.
7. V. G. Bondarchuk, "Sistemy uravnenii v algebre sobytii" (Systems of equations in the event algebra), in Russian, *Zhurnal vychislitel'noi matematiki i matematicheskoi fiziki* (Journal of Computational Mathematics and Mathematical Physics), 3:6 (1963), 1077–1088.
8. W. Büttner. Unification in finite algebras is unitary(?). In *Proc. CADE-9*, Springer LNCS 310, 1988.
9. W. Charatonik, L. Pacholski, "Set constraints with projections are in NEXPTIME", In *Proc. FOCS'94*, IEEE Press, 1994.
10. W. Charatonik, "Set constraints in some equational theories", *Information and Computation*, 142 (1998), 40–75.
11. J. H. Conway, *Regular Algebra and Finite Machines*, Chapman and Hall, 1971.
12. S. Ginsburg, H. G. Rice, "Two families of languages related to ALGOL", *Journal of the ACM*, 9 (1962), 350–371.
13. R. Gilleron, S. Tison, M. Tommasi, "Set constraints and automata", *Information and Computation*, 149:1 (1999), 1–41.
14. M. Kunc, "The power of commuting with finite sets of words", In *Proc. STACS'05*, Springer LNCS 3404, 2005.
15. G. S. Makanin, The problem of solvability of equations in a free semigroup. *Math. Sbornik* 103:147–236. English translation in Math. USSR Sbornik 32, 1977.

16. T. Nipkow. Unification in primal algebras, their powers and their varieties. *J. of the ACM*, 37(1):742–776, 1990.
17. D. Niwiński, "On the cardinality of sets of infinite trees recognizable by finite automata", *Mathematical Foundations of Computer Science* (MFCS 1991, Kazimierz Dolny, Poland, September 9–13, 1991), LNCS 520, 1991, 367–376.
18. A. Okhotin, "Conjunctive grammars and systems of language equations", *Programming and Computer Software*, 28:5 (2002), 243–249.
19. A. Okhotin, "Decision problems for language equations with Boolean operations", *Automata, Languages and Programming* (Proceedings of ICALP 2003, Eindhoven, The Netherlands, June 30–July 4, 2003), LNCS 2719, 239–251.
20. A. Okhotin, "Unresolved systems of language equations: expressive power and decision problems", *Theoretical Computer Science*, 349:3 (2005), 283–308.
21. A. Okhotin, "Strict language inequalities and their decision problems", *Mathematical Foundations of Computer Science* (MFCS 2005, Gdansk, Poland, August 29–September 2, 2005), LNCS 3618, 708–719.
22. M. O. Rabin, "Decidability of second-order theories and automata on infinite trees", *Transactions of the American Mathematical Society*, 141 (1969), 1–35.
23. A. Salomaa, *Theory of Automata*, Pergamon Press, Oxford, 1969.
24. H. Seidl, "Haskell overloading is DEXPTIME-complete", *Information Processing Letters*, 52(2):57–60, 1994.
25. L. R. Stockmeyer, *The complexity of decision problems in automata theory and logic*, Ph.D. thesis, Dept. of Electrical Engineering, MIT, 1974.
26. A. Tarski, "A Lattice-Theoretical Fixpoint Theorem and Its Applications", *Pacific Journal of Mathematics*, 5:285–309, 1955.