

Strict language inequalities and their decision problems

Alexander Okhotin*

Department of Mathematics, University of Turku, Turku FIN-20014, Finland

Abstract. Systems of language equations of the form $\{\varphi(X_1, \dots, X_n) = \emptyset, \psi(X_1, \dots, X_n) \neq \emptyset\}$ are studied, where φ, ψ may contain set-theoretic operations and concatenation; they can be equivalently represented as strict inequalities $\xi(X_1, \dots, X_n) \subset L_0$. It is proved that the problem whether such an inequality has a solution is Σ_2 -complete, the problem whether it has a unique solution is in $(\Sigma_3 \cap \Pi_3) \setminus (\Sigma_2 \cup \Pi_2)$, the existence of a regular solution is a Σ_1 -complete problem, while testing whether there are finitely many solutions is Σ_3 -complete. The class of languages representable by their unique solutions is exactly the class of recursive sets, though a decision procedure cannot be algorithmically constructed out of an inequality, even if a proof of solution uniqueness is attached.

1 Introduction

Language equations are equalities of the form $\varphi(X_1, \dots, X_n) = \psi(X_1, \dots, X_n)$, where the variables X_1, \dots, X_n assume values of languages, while the expressions φ and ψ use language-theoretic operations from some predefined set. In a more general sense, a language equation is any formally specified relationship between sets of strings that contains unknowns. This definition includes some particular variants, such as language inequalities $\varphi \subseteq \psi$ [8,10], inequations $\varphi \neq \psi$, proper inequalities $\varphi \subset \psi$, as well as mixed systems of equations of these four types.

The origins of language equations can be traced to a paper by Ginsburg and Rice [5] on the specification of context-free languages, and to the monographs on automata theory by Salomaa [18] and by Conway [4]. The automata-theoretic direction in the study of language equations led to an important concept of an alternating finite automaton [3]. The results of Ginsburg and Rice have been extended to conjunctive grammars [11], and finally led to a definition of Boolean grammars [13], which depart from the generative paradigm.

What is perhaps the most important about language equations, is that language-theoretic problems arising in different areas can be reduced to their decision problems. The important turn towards understanding this role of language equations was made by Baader and Narendran [1] and by Baader and Küsters [2], who reduced term unification in several description logics to certain decision problems for language equations with one-sided concatenation, and determined the complexity of those problems.

* Supported by the Academy of Finland under grant 206039.

The next step towards a computational theory of language equations was undertaken by the author [12], who began a systematic study of language equations with Boolean operations and unrestricted concatenation. An important result was their computational universality [12], which has subsequently been extended to more restricted cases of language equations [9,15]. One of these cases led to an unexpected negative solution, due to Kunc [9], to the long-standing Conway's problem on the commutation of languages [4,7].

This paper continues the study of the computational properties of language equations [12], investigating new problems and more general types of equations. Important decision problems, such as whether a system has a finite or a regular solution [1,2], are for the first time considered for a general type of language equations with unrestricted concatenation and all Boolean operations. Another novelty is a further extension of the model: strict inequalities and inequations make their first appearance in the systems. Let us start with defining the general form of language equations to be studied in the following.

2 Forms of equations

We shall study all four types of language equations mentioned in the introduction, which are:

- *equations* (in the narrow sense) $\varphi(X_1, \dots, X_n) = \psi(X_1, \dots, X_n)$,
- *inequalities* $\varphi(X_1, \dots, X_n) \subseteq \psi(X_1, \dots, X_n)$,
- *inequations* $\varphi(X_1, \dots, X_n) \neq \psi(X_1, \dots, X_n)$ and
- *strict inequalities* $\varphi(X_1, \dots, X_n) \subset \psi(X_1, \dots, X_n)$.

Both sides of each of these types of equations may contain concatenation, union, intersection and complement, as well as any regular constant languages over a fixed alphabet Σ , while the variables assume values of languages over Σ . Actually, the constants can be restricted to $\{\varepsilon\}$ and $\{a\}$ ($a \in \Sigma$) without decreasing the expressive power [12,15], or relaxed to arbitrary recursive languages without changing any of the constructions of this paper.

We shall consider mixed systems of language equations of all four types. A vector of languages $L = (L_1, \dots, L_n)$ is a solution of such a system, if a substitution of L_i for X_i in all equations yields a tautology in each. Such systems can be represented in two normal forms; the first of these forms is a system of an equation and an inequation, each with the empty set in the right-hand side:

$$\varphi(X_1, \dots, X_n) = \emptyset \tag{1a}$$

$$\psi(X_1, \dots, X_n) \neq \emptyset \tag{1b}$$

Any inequality $\xi \subseteq \eta$ can be equivalently rewritten as $\xi \setminus \eta = \emptyset$. Any equality $\xi = \eta$ holds if and only if $\xi \Delta \eta = \emptyset$ (where Δ denotes the symmetric difference of sets), and, similarly, an inequation $\xi \neq \eta$ can be rewritten as $\xi \Delta \eta \neq \emptyset$. A proper inequality $\xi \subset \eta$ is equivalent to a system $\{\xi \setminus \eta = \emptyset, \eta \setminus \xi \neq \emptyset\}$. Finally, a system with multiple equations of each type (1a,1b), $\{\varphi_1 = \emptyset, \dots, \varphi_m = \emptyset,$

$\psi_1 \neq \emptyset, \dots, \psi_n \neq \emptyset$ can be simplified to $\{\varphi_1 \cup \dots \cup \varphi_m = \emptyset, \psi_1 \cdot \dots \cdot \psi_n \neq \emptyset\}$. These transformations can be applied to convert any system to the form (1).

The other form serving as a universal representation is a single strict inequality $\xi(X_1, \dots, X_n) \subset L_0$, where L_0 is a regular constant language. It is easy to see that (1) holds if and only if $a\varphi \cup b\bar{\psi} \subset b\Sigma^*$. The form (1) will be used in this paper for all results on arbitrary systems of language equations, while the examples of systems will utilize all four types of equations.

Example 1. The following system over $\Sigma = \{a\}$

$$Y \subseteq aY \cup \varepsilon \tag{2a}$$

$$Y \neq a^* \tag{2b}$$

$$X \subseteq Y \tag{2c}$$

$$aX \setminus Y \neq \emptyset \tag{2d}$$

has the set of solutions $\{(L, L') \mid L \text{ is finite, } L' \text{ is the substring closure of } L\}$.

The first equation (2a) states that Y is suffix-closed, which means that it is either $a^{<n}$ for some $n \geq 0$, or a^* ; the latter possibility is ruled out by (2b). So Y must be a finite language of the form $a^{<n}$, while X is its subset by (2c). However, Y cannot be *any* superset of X : according to (2d), if a is appended to the longest string in X , the resulting string should not be in Y . This limits Y to the substring closure of X .

Example 2. The system from Example 1 can be equivalently rewritten as a system of an equation and an inequality

$$(Y \setminus (aY \cup \varepsilon)) \cup (X \setminus Y) = \emptyset \tag{3a}$$

$$(Y \Delta a^*)(aX \setminus Y) \neq \emptyset \tag{3b}$$

or as a proper inequality $a[(Y \setminus (aY \cup \varepsilon)) \cup (X \setminus Y)] \cup b\overline{(Y \Delta a^*)(aX \setminus Y)} \subset ba^*$.

The language of valid accepting computations of a Turing machine, $\text{VALC}(T)$ [6], has proved to be a very important tool in the study of language equations [12,15,16]. In short, for every TM T over an input alphabet Σ one can construct an alphabet Γ and an encoding of computations $C_T : \Sigma^* \rightarrow \Gamma^*$, such that

$$\text{VALC}(T) = \{w\#C_T(w) \mid C_T(w) \text{ is an accepting computation}\}, \tag{4}$$

is an intersection of two linear context-free languages, and hence can be specified by a system of language equations.

Example 3 ([15]). For every Turing machine T , there exists and can be effectively constructed a two-variable language equation $v_T(Y, Z) = \emptyset$ which uses all Boolean operations and linear concatenation, and has a unique solution of the form $(\text{VALC}(T), L')$, where L' is a certain auxiliary language.

Language equations with Boolean operations and concatenation are expressive enough to extract the language *recognized* by a Turing machine out of the language of its computations [12]. This makes such equations computationally universal and binds their study to the arithmetical hierarchy.

Let us recall the definition of this key notion of the classical recursion theory [17]. The arithmetical hierarchy consists of the classes Σ_k and Π_k (for all $k \geq 1$). A language L is said to be in Σ_k if it can be represented as $\{w \mid \exists x_1 \forall x_2 \dots Q_k x_k R(w, x_1, \dots, x_k)\}$ for some recursive predicate R , where $Q_k = \exists$ if k is odd, $Q_k = \forall$ if k is even. Similarly, L is in Π_k if its complement is in Σ_k , or, in other words, if it is of the form $\{w \mid \forall x_1 \exists x_2 \dots Q_k x_k R(w, x_1, \dots, x_k)\}$. There are complete sets in each Σ_k and Π_k ($k \geq 1$). It is easy to see that $\Sigma_1 = \text{RE}$ and $\Pi_1 = \text{co-RE}$, and their complete sets are the TM halting problem and its complement. For all k , the inclusions $\Sigma_k, \Pi_k \subset \Sigma_{k+1}$ and $\Sigma_k, \Pi_k \subset \Pi_{k+1}$ are known to be proper, while Σ_k and Π_k are incomparable. The intersection of Σ_k and Π_k is the class of languages decidable using an oracle for Σ_{k-1} .

3 Existence of finite and regular solutions

A vector (L_1, \dots, L_n) is said to be a finite (a regular) solution of a system of language equations if it is a solution and all languages L_1, \dots, L_n are finite (regular, respectively). These special types of solutions have an advantage of being effectively representable, and algorithms to compute finite [1] and regular solutions [2] for some restricted types of language equations have been constructed.

However, that turns out to be impossible in our more general case:

Theorem 1. *The set of systems of language equations $\{\varphi(X_1, \dots, X_n) = \emptyset, \psi(X_1, \dots, X_n) \neq \emptyset\}$ that have a finite solution (a regular solution) is RE-complete. Both problems remain RE-complete for individual equations $\varphi(X_1, \dots, X_n) = \emptyset$, in which the concatenation is restricted to linear.*

Proof. Membership in RE. It suffices to consider all vectors of n finite languages (L_1, \dots, L_n) (all vectors of n finite automata A_1, \dots, A_n in the case of regular solutions) substituting each into both equations and determining whether the system is satisfied. If the system has a finite (regular, resp.) solution, such a vector will eventually be found. If there are no finite (regular, resp.) solutions, the computation will never terminate.

RE-hardness. Reduction from the Post Correspondence Problem for non-empty strings, stated as “Given an alphabet Σ and a finite set of pairs $(u_1, v_1), \dots, (u_m, v_m)$, where $u_i, v_i \in \Sigma^+$, determine whether there exists a finite sequence of numbers i_1, \dots, i_n ($n \geq 1, 1 \leq i_j \leq m$), such that $u_{i_1} \dots u_{i_n} = v_{i_1} \dots v_{i_n}$ ”.

Let $\{x_1, \dots, x_m\}$ be a block code over Σ (i.e., $|x_1| = \dots = |x_m|$ and $x_i \neq x_j$ for all $i \neq j$). Define a set of variables $\{X, Y, Z, Y_1, \dots, Y_m, Z_1, \dots, Z_m\}$ and

construct the following system of language equations over $\Sigma \cup \{\#\}$ ($\# \notin \Sigma$):

$$X = Y \cap Z \cap \Sigma^* \# \Sigma^+ \quad (5a)$$

$$Y = x_1 Y_1 u_1 \cup \dots \cup x_m Y_m u_m \cup \# \quad (5b)$$

$$Y = Y_1 \cup \dots \cup Y_m \cup X \quad (5c)$$

$$Z = x_1 Z_1 v_1 \cup \dots \cup x_m Z_m v_m \cup \# \quad (5d)$$

$$Z = Z_1 \cup \dots \cup Z_m \cup X \quad (5e)$$

Note that if the nonterminals Y_i are replaced with Y in (5b), while (5c) is altogether removed, and the same is done with respect to Z , we obtain the usual encoding of PCP as an intersection of two linear context-free languages, in which Y and Z assume nonregular values regardless of the solvability of the PCP instance. In our case, each variable Y_i means the set of those strings from Y that are prolonged with x_i and u_i . The equation (5c) means: “every string w in Y , unless it is also in X , must be prolonged to $x_i w y_i$ at least for one i ”. If any strings get into X , this process can be stopped, which allows us to obtain a finite solution when PCP is solvable.

It is easy to see that if a vector L satisfies (5), then every $w \in Y(L)$ must be of the form $x_{i_k} \dots x_{i_1} \# u_{i_1} \dots u_{i_k}$, for some $k \geq 0$ and $1 \leq i_j \leq m$; similarly, every $w \in Z(L)$ is of the form $x_{i_k} \dots x_{i_1} \# v_{i_1} \dots v_{i_k}$. Accordingly, if $x_{i_1} \dots x_{i_n} \# w \in X(L)$, then PCP has a solution $u_{i_1} \dots u_{i_n} = v_{i_1} \dots v_{i_n} = w$.

Suppose the given instance of PCP is not solvable, and let us prove that all solutions of the system (5) are infinite. Suppose L is a finite solution and let w be the longest string in $Y(L)$. Since $X(L) = \emptyset$, by (5c), there exists i , such that $w \in Y_i(L)$. Then $u_i w x_i \in Y(L)$ by (5b), and hence w is not the longest string. The contradiction obtained proves that $Y(L)$ is not finite. Furthermore, using the pumping lemma it can be proved that $Y(L)$ is not regular.

Now let the instance of PCP be solvable, and let $u_{i_1} \dots u_{i_n} = v_{i_1} \dots v_{i_n} = w$ be the shortest string that meets its specification. Then (5) has the following finite solution: $(X = \{x_{i_n} \dots x_{i_1} \# w\}, Y = \{x_{i_k} \dots x_{i_1} \# u_{i_1} \dots u_{i_k} \mid 0 \leq k \leq n\}, Y_i = \{x_{i_k} \dots x_{i_1} \# u_{i_1} \dots u_{i_k} \mid 0 \leq k < n, i_{k+1} = i\}, Z = \{x_{i_k} \dots x_{i_1} \# v_{i_1} \dots v_{i_k} \mid 0 \leq k \leq n\}, Z_i = \{x_{i_k} \dots x_{i_1} \# v_{i_1} \dots v_{i_k} \mid 0 \leq k < n, i_{k+1} = i\})$. \square

4 Existence of a solution

Let us now study the question of the existence of solutions of an arbitrary form. Some further terminology is required.

Two languages, L' and L'' , are said to be *equal modulo* a third language M , which is substring-closed (i.e., contains all substrings of each of its strings), if $L' \cap M = L'' \cap M$ [12]. This definition is extended to vectors of languages, which are said to be equal modulo M , if their corresponding components are equal modulo M . These definitions are also naturally extended to inequalities, strict inequalities and inequations. Note that if two vectors of languages L', L'' are equal modulo a substring-closed M , then $\varphi(L') = \varphi(L'') \pmod{M}$.

A vector of languages is a *solution modulo* M of a system of language equations, if each equation holds modulo M under the substitution of these languages for variables. A given vector (L_1, \dots, L_n) can be tested for being a solution modulo M by substituting $X_i = L_i \cap M$ into the equations and computing, modulo M , the value of each subexpression.

Let $\varphi(X_1, \dots, X_n) = \emptyset$ be an equation. Its solution L_M modulo M is said to be *extendable to* M' , for a given $M' \supseteq M$, if there exists a solution modulo M' that coincides with L_M modulo M . The vector L_M is said to be *extendable to a solution*, if the equation has a solution that equals L_M modulo M .

Lemma 1 ([12]). *Let $\varphi(X_1, \dots, X_n) = \emptyset$ be an equation and let M be a finite substring-closed language. Then there exists a finite substring-closed language $M' \supseteq M$, such that all solutions modulo M extendable to M' are extendable to solutions.*

Theorem 2 ([12]). *A language equation $\varphi(X_1, \dots, X_n) = \emptyset$ has a solution if and only if for every finite substring-closed language M there exists a solution of $\varphi(X) = \emptyset$ modulo M . The decision problem is co-RE-complete.*

Let us now establish an analogous necessary and sufficient condition of solution existence for more general systems involving inequations.

Theorem 3. *A system $\{\varphi(X_1, \dots, X_n) = \emptyset, \psi(X_1, \dots, X_n) \neq \emptyset\}$ has a solution if and only if there exists a finite substring-closed language M_0 , such that for every finite substring-closed language $M \supseteq M_0$ there exists a solution of $\varphi = \emptyset$ modulo M that is a solution of $\psi \neq \emptyset$ modulo M_0 .*

Proof. \ominus Let $L = (L_1, \dots, L_n)$ be a solution of the system. Since $\psi(L_1, \dots, L_n) \neq \emptyset$, there exists a finite substring-closed language, such that this inequality is satisfied modulo that language. Denote it by M_0 and consider an arbitrary finite substring-closed superset $M \supseteq M_0$. The vector $(L_1 \cap M, \dots, L_n \cap M)$ satisfies $\varphi = \emptyset$ modulo M and $\psi \neq \emptyset$ modulo M_0 .

\ominus Given a finite substring-closed M_0 , apply Lemma 1 to the equation $\varphi(X_1, \dots, X_n) = \emptyset$ and the modulus M_0 , to obtain a greater finite substring-closed modulus $M \supseteq M_0$.

By assumption, for this M there exists a vector L_M , such that $\varphi(L_M) = \emptyset \pmod{M}$ and $\psi(L_M) \neq \emptyset \pmod{M_0}$. Let L_{M_0} be L_M taken modulo M_0 ; then we know that $\psi(L_{M_0}) \neq \emptyset \pmod{M_0}$ and that L_{M_0} is extendable to M . The latter, by the choice of M according to Lemma 1, implies that the equation $\varphi = \emptyset$ has a solution L that coincides with L_{M_0} modulo M_0 . Hence L also satisfies $\psi \neq \emptyset$, and therefore is a solution of the system. \square

Theorem 4. *The set of systems of language equations $\{\varphi(X_1, \dots, X_n) = \emptyset, \psi(X_1, \dots, X_n) \neq \emptyset\}$ that have solutions is Σ_2 -complete.*

Proof. A necessary and sufficient condition of having solutions given by Theorem 3 is of the form $\exists M_0 \forall M R(\varphi, \psi, M_0, M)$, where the quantifiers range over

countable sets and R is a recursive predicate. A set thus defined is in Σ_2 by definition.

In order to prove Σ_2 -hardness, let us use a reduction from the complement of the Π_2 -complete Turing machine universality problem. This Σ_2 -complete problem can be stated as “Given a TM T over Σ , determine whether $L(T) \neq \Sigma^*$ ”.

$$v_T(Y, Z) = \emptyset \quad (6a)$$

$$Y \subseteq X \# \Gamma^* \quad (6b)$$

$$X \subseteq \Sigma^* \quad (6c)$$

$$X \neq \Sigma^* \quad (6d)$$

The equation (6a) expresses $Y = \text{VALC}(T)$ and $Z = L'$, as in Example 3. The next equation (6b) specifies that every string that begins an accepting computation (that is, every string accepted by M) should be in X . Together with (6c), this means that

$$L(T) \subseteq X \subseteq \Sigma^* \quad (7)$$

If $L(T) = \Sigma^*$, then the bounds (7) are tight, and the only candidate for being a solution of (6) is $X = \Sigma^*$. However, it is ruled out by the inequation (6d), and hence there are no solutions.

If $L(T) \neq \Sigma^*$, then $X = L(T)$ fits into the bounds (7) and at the same time satisfies the inequation (6d), and, therefore, $(L(T), \text{VALC}(T), L')$ is a solution of the system (6). This proves the correctness of the reduction. \square

5 Uniqueness of a solution

In order to study the systems that have exactly one solution, let us first characterize the following property:

Theorem 5. *Let $k \geq 1$. A system $\{\varphi(X_1, \dots, X_n) = \emptyset, \psi(X_1, \dots, X_n) \neq \emptyset\}$ has at most k solutions if and only if for every finite substring-closed language M , there exists a finite substring-closed language $M' \supseteq M$, such that all solutions of $\varphi(X_1, \dots, X_n) = \emptyset$ modulo M' that are solutions of $\psi(X_1, \dots, X_n) \neq \emptyset$ modulo M have at most k distinct images modulo M .*

In the case $k = 1$, the statement reads: “... all solutions of $\varphi(X) = \emptyset$ modulo M' that are solutions of $\psi(X) \neq \emptyset$ modulo M coincide modulo M ”.

Proof. \ominus Fix an M , and let M' be the language defined for the equation $\varphi = \emptyset$ and the modulus M by Lemma 1. Suppose that there exist $k + 1$ vectors modulo M' , which are distinct modulo M , which satisfy $\varphi = \emptyset$ modulo M' , and which remain solutions of $\psi \neq \emptyset$ modulo M . Let $L^{(1)}, \dots, L^{(k+1)}$ be these vectors taken modulo M . According to Lemma 1, each of them can be extended to a solution of $\varphi = \emptyset$, which will at the same time remain a solution modulo $\psi \neq \emptyset$. Therefore, the system has at least $k + 1$ solutions, which yields a contradiction.

\ominus Suppose the vectors $L^{(1)}, \dots, L^{(k+1)}$ are pairwise distinct solutions of the system. Then there exists a finite M closed under substring, such that these

vectors are pairwise distinct modulo M , and each of them satisfies $\psi \neq \emptyset$ modulo M . By assumption, for this particular M there exists a finite substring-closed M' , such that all solutions of $\varphi = \emptyset$ modulo M' that are solutions of $\psi \neq \emptyset$ have at most k distinct images modulo M . Since each of $L^{(1)}, \dots, L^{(k+1)}$ fits this description, they are not pairwise distinct modulo M , which contradicts the choice of M . \square

Now the property of having a unique solution can be expressed as a conjunction of two conditions: the Σ_2 -condition of having at least one solution (Theorem 3), and the Π_2 -condition of having at most one solution (Theorem 5 with $k = 1$). This can be represented as a Σ_3 - or as a Π_3 -formula, which puts the decision problem to $\Sigma_3 \cap \Pi_3$. Actually, this is the optimal representation.

Theorem 6. *For any $k \geq 1$, the set of systems $\{\varphi(X_1, \dots, X_n) = \emptyset, \psi(X_1, \dots, X_n) \neq \emptyset\}$ that have exactly k solutions is Σ_2 -hard, Π_2 -hard and recursive in Σ_2 (i.e., it belongs to $\Sigma_3 \cap \Pi_3$).*

Proof. Σ_2 -hardness. Though the solution existence problem is Σ_2 -complete, Theorem 4 does not imply the Σ_2 -hardness of our case. Taking a look at its proof, it is easy to see that, unless $L(T) = \Sigma^* \setminus \{w\}$ for some $w \in \Sigma^*$, the solution of the system (6) is not unique. Hence, a different proof is needed.

Let us use a reduction from another Σ_2 -complete problem, the Turing machine finiteness, stated as ‘‘Given a TM T over a unary alphabet $\{a\}$, determine whether $L(T)$ is finite’’. It is claimed that the following system has a unique solution if and only if $L(T)$ is finite.

$$v_T(Y, Z) = \emptyset \tag{8a}$$

$$Y \subseteq X\#\Gamma^* \tag{8b}$$

$$X \subseteq aX \cup \varepsilon \tag{8c}$$

$$(aY \cup \#) \setminus X\#\Gamma^* \neq \emptyset \tag{8d}$$

As in the proof of Theorem 4, the equations (8a, 8b) specify that $Y = \text{VALC}(T)$, $Z = L'$ and $L(T) \subseteq X$. The inequality (8c), cf. (2a) in Example 1, requires that $X \subseteq a^*$ and that X is closed under suffix. Hence, the system (8a, 8b, 8c) has the set of solutions $\{(L, \text{VALC}(T), L') \mid L(T) \subseteq L \subseteq a^* \text{ and } L \text{ is suffix-closed}\}$.

If $L(T) = \emptyset$, then $\text{VALC}(T) = \emptyset$, and X must be \emptyset as well. Supposing the contrary, that X is a nonempty suffix-closed language $\{\varepsilon, a, aa, \dots, a^\ell\}$ ($\ell \geq 0$), the left-hand side of the inequation (8d) takes form $(\emptyset \cup \#) \setminus \{\varepsilon, \dots\}\#\Gamma^* = \emptyset$, and hence the inequation is not satisfied. It can similarly be proved that if $L(T)$ is a finite nonempty set and a^m is the longest string in it, then X must be equal to $a^{\leq m}$.

If $L(T)$ is infinite, then a^* is the only potential value of $X \supseteq L(T)$, since this is the only infinite suffix-closed language over $\{a\}$. But then $a\text{VALC}(T) \cup \{\#\} \subseteq X\#\Gamma^* = a^*\#\Gamma^*$, and therefore the inequation (8d) does not hold.

Π_2 -hardness. Follows from the Π_2 -completeness of the same problem for equations $\varphi(X_1, \dots, X_n) = \emptyset$ [12]. Its proof uses a reduction from the Turing machine universality problem, and uses exactly the system (6a, 6b 6c).

Recursiveness in Σ_2 . As noted above, the Σ_2 condition of solution existence (Theorem 3) and the Π_2 condition of having at most one solution (Theorem 5) have to be checked. Hence, the problem is Turing-reducible to Σ_2 . \square

Theorem 7. *Let $\{\varphi(X_1, \dots, X_n) = \emptyset, \psi(X_1, \dots, X_n) \neq \emptyset\}$ be a system of an equation and an inequality that has a unique solution (L_1, \dots, L_n) . Then each component L_i is recursive.*

Proof. Since $\psi(L_1, \dots, L_n) \neq \emptyset$, there exists a string $w_0 \in \psi(L_1, \dots, L_n)$. Using this string, construct the following algorithm:

Input: $w \in \Sigma$
 Let $M = \text{substrings}(w_0) \cup \text{substrings}(w)$
 For all finite substring-closed $M' \supseteq M$
 Let $L^{(1)}, \dots, L^{(k)}$ be all vectors (mod M')
 that satisfy $\varphi = \emptyset$ modulo M' and $\psi \neq \emptyset$ modulo M'
 If $L^{(1)} = \dots = L^{(k)}$ (mod M)
 Accept if $w \in L_i^{(1)}$, reject if $w \notin L_i^{(1)}$

By Theorem 5, for the language M used by the algorithm there exists a finite substring-closed language $M' \supseteq M$, such that all solutions of $\varphi = \emptyset$ modulo M' that satisfy $\psi \neq \emptyset$ modulo M' coincide modulo M . This M' will be eventually reached by the algorithm's loop, the condition in the *if* statement will become true, and the algorithm will terminate.

It remains to argue that the algorithm accepts w if and only if it is in L_i . The actual solution (L_1, \dots, L_n) satisfies $\varphi = \emptyset$ modulo M' and $\psi \neq \emptyset$ modulo M' , so this vector, taken modulo M' , must be among $L^{(1)}, \dots, L^{(k)}$. Therefore, $L^{(1)} = (L_1, \dots, L_n) \pmod{M}$ and $w \in L_i^{(1)}$ is equivalent to $w \in L_i$. \square

Since it is already known that every recursive language can be specified by a unique solution of a language equation [12,15], the following can be concluded:

Corollary 1. *The class of languages representable as components of unique solutions of systems of the form $\{\varphi(X_1, \dots, X_n) = \emptyset, \psi(X_1, \dots, X_n) \neq \emptyset\}$ is exactly the class of recursive languages.*

Given an individual language equation $\varphi(X_1, \dots, X_n) = \emptyset$ with a unique solution, the algorithm for determining the membership of strings in this solution can be effectively constructed [12]. This turns out to be different in our more general case involving inequations:

Theorem 8. *There is no algorithm that, given a system of language equations $\{\varphi(X_1, \dots, X_n) = \emptyset, \psi(X_1, \dots, X_n) \neq \emptyset\}$ with an attached proof that it has a unique solution, determines this unique solution modulo $\{\varepsilon\}$.*

Proof. Suppose such an algorithm exists. Consider an arbitrary Turing machine T , and use the language equation $v_T(Y, Z) = \emptyset$ from Example 3 to construct

the following system of language equations:

$$v_T(Y, Z) = \emptyset \quad (9a)$$

$$X \subseteq \varepsilon \quad (9b)$$

$$XY = \emptyset \quad (9c)$$

$$X \cup Y \neq \emptyset \quad (9d)$$

The equation (9a) requires that $Y = \text{VALC}(T)$ and $Z = L'$, where L' is a certain irrelevant auxiliary language (see Example 3). By (9b), X can assume one of the two possible values: \emptyset or $\{\varepsilon\}$. The next two equations (9c, 9d) specify that exactly one of the languages X and $\text{VALC}(T)$ is nonempty. Since $\text{VALC}(T)$ is nonempty if and only if $L(T)$ is nonempty, it can be concluded that if $L(T) = \emptyset$, then $X = \{\varepsilon\}$, and if $L(T) \neq \emptyset$, then $X = \emptyset$. The system has a unique solution in either case, it is either $(\{\varepsilon\}, \emptyset, L')$ or $(\emptyset, \text{VALC}(T), L')$.

This argument can be properly formalized and attached to the system (9) to produce an input for the supposed algorithm. The solution modulo $\{\varepsilon\}$ it computes will be $(\{\varepsilon\}, \emptyset, L' \cap \{\varepsilon\})$ if $L(T) = \emptyset$, or $(\emptyset, \emptyset, L' \cap \{\varepsilon\})$ if $L(T) \neq \emptyset$. Therefore, the supposed algorithm can be used to solve the Turing machine emptiness problem, which is known to be undecidable. \square

In particular, Theorem 8 implies that, though for every system with a unique solution there exists an algorithm for testing the membership of strings in the components of that solution, this algorithm cannot be algorithmically constructed. This reveals a principal difference between these systems and the earlier studied computationally universal types of language equations [9,12,15].

6 Equations with finitely many solutions

Let us consider the problem of testing whether a given system of language equations has finitely many solutions. This property can naturally be represented as “there exists a number $k \geq 0$, such that there are exactly k solutions”, and we shall now see that this representation is optimal with respect to the number of quantifiers.

Theorem 9. *The set of systems of language equations $\{\varphi(X_1, \dots, X_n) = \emptyset, \psi(X_1, \dots, X_n) \neq \emptyset\}$ that have finitely many solutions is Σ_3 -complete. It remains Σ_3 -complete for individual equations $\varphi(X_1, \dots, X_n) = \emptyset$.*

Proof. Membership in Σ_3 . A system has finitely many solutions if and only if there exists a number $k \geq 1$, such that the system has at most k solutions. According to Theorem 5, the condition of having at most k solutions is representable as a Π_2 formula $\forall M \exists M' R(k, M, M')$, where R is a certain recursive predicate stated by the theorem. This predicate R can be used to characterize our problem as follows: $\exists k \forall M \exists M' R(k, M, M')$. This is a Σ_3 -formula.

Σ_3 -**hardness.** Reduction from the co-finiteness problem for Turing machines, stated as “Given a TM T over Σ , determine whether $\Sigma^* \setminus L(T)$ is finite”, which is known to be Σ_3 -complete [17, Corollary 14-XVI].

Construct the system (6a, 6b, 6c), as in Theorem 4, which has the set of solutions

$$\{(L, \text{VALC}(T), L') \mid L(T) \subseteq L \subseteq \Sigma^*\} \quad (10)$$

If $L(T)$ is co-finite, then there are finitely many values L that fit within the bounds in (10), and hence the set of solutions has finitely many elements. If $\Sigma^* \setminus L(T)$ is infinite, then (10) is infinite as well. This proves the reduction. \square

We already know that if a system has a unique solution, then the components of this solution are recursive languages (see Theorem 7). Let us extend this recursiveness result to the case of systems with any finite number of solutions.

Theorem 10. *Let $\{\varphi(X_1, \dots, X_n) = \emptyset, \psi(X_1, \dots, X_n) \neq \emptyset\}$ be a system of an equation and an inequality that has finitely many solutions. Then all components of all these solutions are recursive.*

Proof. Let $L^{(1)}, \dots, L^{(k)}$ ($k \geq 0$) be all solutions of the system. If $k = 0$, the result trivially holds, and if $k = 1$, it holds by Theorem 7. Consider the case $k \geq 2$ and let us construct a new system that would have $L^{(1)}$ as its unique solution. Since $L^{(1)}$ is different from each $L^{(i)}$ ($2 \leq i \leq k$), for every such i there exists a variable X_{j_i} , such that $L_{j_i}^{(1)} \neq L_{j_i}^{(i)}$; let w_i be any string in their symmetric difference. Now construct the following system of language equations:

$$\varphi(X_1, \dots, X_n) = \emptyset \quad (11a)$$

$$\psi(X_1, \dots, X_n) \neq \emptyset \quad (11b)$$

$$\begin{cases} X_{j_i} \cap w_i \neq \emptyset, & \text{if } w_i \in L_{j_i}^{(1)} \\ X_{j_i} \cap w_i = \emptyset, & \text{if } w_i \notin L_{j_i}^{(1)} \end{cases} \quad (\text{for all } 2 \leq i \leq k) \quad (11c)$$

Every solution of the constructed system satisfies (11a, 11b), i.e., is a solution of the original system. Therefore, $L^{(1)}, \dots, L^{(k)}$ are the only candidates for being solutions of (11). While each $L^{(i)}$ ($i \geq 2$) does not satisfy the i -th equation (11c), $L^{(1)}$ satisfies all of them, which makes it the unique solution of the system (11). Therefore, by Theorem 7, all components of $L^{(1)}$ are recursive. \square

7 Conclusion

The complexity of main decision problems for language equations with equality only and for systems involving proper inequalities and inequations is shown in Table 1. The results are fairly disparate, but they have one thing in common: undecidability.

The close relation of language equations of the general form to the recursion theory and to logic has been noted before [12,16], and the extensions introduced in this paper (namely, inequations and strict inequalities) yield new interesting undecidabilities that manifest themselves in Theorem 8. This particular turn of the theory of language equations suggests further mathematical problems to study. For instance, how hard is the following decision problem: ‘‘Given a

	$\varphi(X_1, \dots, X_n) = \emptyset$	$\begin{cases} \varphi(X_1, \dots, X_n) = \emptyset \\ \psi(X_1, \dots, X_n) \neq \emptyset \end{cases}$
Does there exist a finite solution?	Σ_1	Σ_1
Does there exist a regular solution?	Σ_1	Σ_1
Do there exist any solutions?	Π_1 [12]	Σ_2
Does there exist a unique solution?	Π_2 [12]	$\Sigma_3 \cap \Pi_3$
Are there finitely many solutions?	Σ_3	Σ_3
Class of languages	recursive [12]	recursive

Table 1. Complexity of decision problems. Expressive power of unique solutions.

system of language equations of one of the two forms, determine whether its set of solutions is countable”? For systems with countably many solutions, what is the class of languages that can occur in those solutions?

References

1. F. Baader, P. Narendran, “Unification of concept terms in description logic”, *Journal of Symbolic Computation*, 31 (2001), 277–305.
2. F. Baader, R. Küsters, “Unification in a description logic with transitive closure of roles”, *LPAR 2001* (Havana, Cuba), LNCS 2250, 217–232.
3. J. A. Brzozowski, E. L. Leiss, “On equations for regular languages, finite automata, and sequential networks”, *Theoretical Computer Science*, 10 (1980), 19–35.
4. J. H. Conway, *Regular Algebra and Finite Machines*, Chapman and Hall, 1971.
5. S. Ginsburg, H. G. Rice, “Two families of languages related to ALGOL”, *Journal of the ACM*, 9 (1962), 350–371.
6. J. Hartmanis, “Context-free languages and Turing machine computations”, *Proceedings of Symposia in Applied Mathematics*, Vol. 19, AMS, 1967, 42–51.
7. J. Karhumäki, I. Petre, “Two problems on commutation of languages”, in: Gh. Păun, G. Rozenberg, A. Salomaa (Eds.), *Current Trends in Theoretical Computer Science: The Challenge of the New Century, vol. 2*, World Scientific, 2004, 477–494.
8. M. Kunc, “Regular solutions of language inequalities and well quasi-orders”, *ICALP 2004*, LNCS 3142, 870–881.
9. M. Kunc, “The power of commuting with finite sets of words”, *STACS 2005*.
10. M. Kunc, “Largest solutions of left-linear language inequalities”, *AFL 2005*.
11. A. Okhotin, “Conjunctive grammars and systems of language equations”, *Programming and Computer Software*, 28 (2002), 243–249.
12. A. Okhotin, “Decision problems for language equations with Boolean operations”, *ICALP 2003*, LNCS 2719, 239–251.
13. A. Okhotin, “Boolean grammars”, *Information and Computation*, 194:1 (2004), 19–48.
14. A. Okhotin, “The dual of concatenation”, *MFCS 2004*, LNCS 3153, 698–710.
15. A. Okhotin, “On computational universality in language equations”, *MCU 2004* (St. Petersburg, Russia), LNCS 3354, 292–303.
16. A. Okhotin, “A characterization of the arithmetical hierarchy by language equations”, *DCFS 2004* (London, Ontario, Canada), 225–237.
17. H. Rogers, Jr., *Theory of Recursive Functions and Effective Computability*, McGraw-Hill, 1967.
18. A. Salomaa, *Theory of Automata*, Pergamon Press, Oxford, 1969.