Journal of Computer and System Sciences 76 (2010) 251–266

Contents lists available at ScienceDirect

Journal of Computer and System Sciences

www.elsevier.com/locate/jcss

Decision problems for language equations ☆

Alexander Okhotin [a,b,*]

[a] *Academy of Finland, Finland*
[b] *Department of Mathematics, University of Turku, Turku FIN-20014, Finland*

ARTICLE INFO

Article history:
Received 13 February 2004
Received in revised form 5 September 2008
Available online 21 August 2009

Keywords:
Language equations
Boolean operations
Computability

ABSTRACT

Equations with formal languages as unknowns using all Boolean operations and concatenation are studied. Their main properties, such as solution existence and uniqueness, are characterized by first-order formulae. It is shown that testing solution existence is Π_1-complete, while solution uniqueness and existence of a least and of a greatest solution are all Π_2-complete problems. The families of languages defined by components of unique, least and greatest solutions of such systems are shown to coincide with the classes of recursive, recursively enumerable and co-recursively enumerable sets, respectively.

1. Introduction

Equations, in which variables assume values of formal languages over a finite alphabet, the constants are formal languages as well, and the operations used are the language-theoretic operations (such as concatenation, Boolean operations, Kleene star, etc.) are known as *language equations*. Being a mathematical abstraction for reasoning about sets of strings, language equations naturally arise in different areas of computer science, and problems that can be formally described by language equations can be found in all kinds of applications.

Language equations were first used in connection to their most natural application: the description of syntax. The basic and the most well-known model of syntax is a context-free grammar, and Ginsburg and Rice [9] defined the semantics of these grammars by systems of equations of the form

$$\begin{cases} X_1 = \varphi_1(X_1, \ldots, X_n), \\ \vdots \\ X_n = \varphi_n(X_1, \ldots, X_n), \end{cases} \qquad (*)$$

where each variable X_i represents an unknown language (a "nonterminal symbol" of a grammar), while each expression φ_i is a union of concatenations of variables and singleton constants. This semantics is in many senses preferable to the Chomskian definition based on derivation: as rightfully observed by Autebert et al. [2], a specification of the form "an instruction is …" is more natural than "a symbol for instruction derives …". Furthermore, the approach of Ginsburg and Rice [9] can be generalized to obtain more powerful models of syntax based upon language equations, such as *conjunctive grammars* [17,18], which extend context-free grammars with an explicit conjunction operation interpreted by intersection in $(*)$, and *Boolean grammars* [19] that allow the use of all propositional connectives represented by the corresponding Boolean operations on languages.

☆ A preliminary version of this paper was presented at the ICALP 2003 conference held in Eindhoven, the Netherlands, June 30–July 4, 2003. This research was done during the author's studies at the School of Computing, Queen's University (Kingston, Ontario, Canada).

* Address for correspondence: Department of Mathematics, University of Turku, Turku FIN-20014, Finland.
E-mail address: alexander.okhotin@utu.fi.

0022-0000/$ – see front matter
doi:10.1016/j.jcss.2009.08.002

Another recurring application of language equations is representing various properties of computation. Systems of the form $(*)$ were used to represent finite automata in an early monograph by Salomaa [21], and Brzozowski and Leiss [5] generalized these systems to define alternation in finite automata. Later, more general equations over sets of terms, known as *set constraints*, were used to represent some properties of programs, and efficient algorithms for their analysis were obtained by Aiken et al. [1]. Connections between set constraints and language equations of a more general form were investigated by Charatonik [6], who is notable for obtaining the first undecidability result for language equations.

Language equations have been adopted as a model in several applied areas of computer science. For instance, language equations with special operations on strings were used by Daley, Ibarra and Kari [8] to model recombination of genes in DNA, Yevtushenko et al. [22] applied language equations to represent component design of complex systems, and more recently Kari and Konstantinidis [13] used language equations of another form to analyze error-detection properties of a communication channel. In applied logic, Baader and Narendran [3] used finite solutions of language equations to represent unification and matching in *description logic* FL_0, while Zhang [23] characterized an extension of *propositional domain logic* by fixed points of language equations.

Besides the research motivated by applications, some purely theoretical work in the area has been done as well. Conway [7] was the first to study a large class of systems with regular solutions, as well as to raise important questions on more general equations. Later Kari and Thierrin [14] studied equations of a simple form using generalized string operations, Leiss [16] constructed the first example of a language equation over a one-letter alphabet with a nonperiodic solution, Karhumäki and Petre [12] investigated the equation $XL = LX$ proposed by Conway [7], while Karhumäki and Lisovik [11] were the first to consider infinite systems of language equations and to show undecidability of their basic properties.

Since language equations naturally occur whenever sets of strings are being considered, a general theory of these equations would be helpful for many applications. However, as sharply noted by Karhumäki and Petre [12] in 2003, the present knowledge on language equations amounts to "very little, or in fact almost nothing". Though some interesting results on particular kinds of equations have been obtained, no general methods of reasoning about language equations have been developed. The goal of this paper is to introduce such methods and to develop basic formal properties of equations with Boolean operations and concatenation.

As the main model, this paper adopts system of the form $(*)$, in which the right-hand sides φ_i may contain any Boolean operations and concatenation. As shown in Section 2, these systems are as powerful as more general systems involving arbitrary equalities $\varphi(X) = \psi(X)$ and inequalities $\varphi(X) \subseteq \psi(X)$, and hence almost all language equations ever studied fall under this case. The next Section 3 presents technical results on encoding computations of Turing machines using language equations, which are subsequently used in all hardness arguments.

The main tool for the analysis of language equations introduced in this paper is the notion of a *solution modulo a language* defined in Section 4. The idea is to take a system of equations and to consider only finitely many strings, thus turning the system into a finitely manageable object. Then, applying quantification over the finite set of strings under consideration (the *modulus*), one can represent various properties of the system. This is used in the following Sections 5 and 6 to characterize the conditions of having a solution and having a unique solution by first-order formulae, and determine the exact undecidability levels of the corresponding decision problems. A similar study of least and greatest solutions is conducted in Section 7.

Finally, the families of languages defined by unique, least and greatest solutions of systems of language equations are considered in Section 8, where it is proved that these are exactly the recursive, the recursively enumerable (r.e.) and the co-recursively enumerable sets.

2. Language equations

Let us begin with the basic notation of formal language theory used in the paper. An *alphabet* Σ is a finite nonempty set, its elements are called *symbols* or *letters*. A *string* over Σ is a finite sequence $w = a_1 \dots a_\ell$ with $\ell \geqslant 0$ and $a_i \in \Sigma$, where the number ℓ is the *length* of the string, denoted $|w|$. The unique *empty string* of length 0 is denoted by ε. The set of all strings is denoted Σ^*, and any subset of this set is called a *language*. The *concatenation* of two strings $u, v \in \Sigma^*$ is the string $uv \in \Sigma^*$, and the concatenation of two languages $K, L \subseteq \Sigma^*$ is the language $KL = \{uv \mid u \in K,\ v \in L\}$. The concatenation of k copies of the same string or language is denoted by w^k and L^k, respectively. The *Kleene star* of a language $L \subseteq \Sigma^*$ is the language $L^* = \bigcup_{k \geqslant 0}^{\infty} L^k$. Define $L^+ = \bigcup_{k \geqslant 1}^{\infty} L^k$.

The language equations considered in this paper may use concatenation and all Boolean operations, and are constructed of expressions of the following form.

Definition 1 *(Language expressions).* Let Σ be an alphabet, and let $X = (X_1, \dots, X_n)$ with $n \geqslant 1$ be a vector of language variables. The set of *expressions* over Σ in variables X is defined inductively as follows:

- any constant language $L_0 \subseteq \Sigma^*$ is an expression;
- any variable from X is an expression;
- if φ and ψ are expressions, then so are $(\varphi\psi)$, $(\varphi \cap \psi)$, $(\varphi \cup \psi)$ and $\overline{\varphi}$.

The concatenation in an expression is said to be *linear*, if for every subformula $(\varphi\psi)$, either φ or ψ is a constant language.

It will be assumed that concatenation has a higher precedence than Boolean operations. This, in addition to associativity of concatenation, union and intersection, allows omitting some of the parentheses in the expressions. For succinctness, when a singleton constant $\{w\}$ is concatenated to any expression, it will be denoted by w.

Definition 2 *(Resolved system of equations).* Let Σ be an alphabet. Let $n \geqslant 1$. Let $X = (X_1, \ldots, X_n)$ be a vector of language variables. Let $\varphi = (\varphi_1, \ldots, \varphi_n)$ be a vector of expressions over the alphabet Σ and in variables X. Then

$$\begin{cases} X_1 = \varphi_1(X_1, \ldots, X_n), \\ \vdots \\ X_n = \varphi_n(X_1, \ldots, X_n) \end{cases}$$

or $X = \varphi(X)$ in vector form, is called a (*resolved*) *system of language equations* over Σ in variables X. A vector of languages $L = (L_1, \ldots, L_n)$ is said to be a *solution* of the system if the substitution $X_i = L_i$ for all i turns each j-th equation into an equality $L_j = \varphi_j(L_1, \ldots, L_n)$. In the vector form, this is denoted as $L = \varphi(L)$.

Besides the resolved systems defined above, one can consider *unresolved systems* formed of any equations $\varphi(X) = \psi(X)$, and well as the more general inequalities $\varphi(X) \subseteq \psi(X)$. However, it turns out that any such equations and inequalities are expressible using resolved equations. In order to impose a restriction $\varphi(X) \subseteq \psi(X)$, it suffices to add an auxiliary variable Y and an equation

$$Y = \overline{Y} \cap \varphi \cap \overline{\psi},$$

which is a contradiction unless the mentioned inclusion holds. Multiple inclusions $\varphi_i(X) \subseteq \psi_i(X)$ $(1 \leqslant i \leqslant m)$ can be specified in a similar way using a single auxiliary variable:

$$Y = \overline{Y} \cap \bigcup_{i=1}^{m} (\varphi_i \cap \overline{\psi_i}). \tag{1}$$

An equality $\varphi(X) = \psi(X)$ can be expressed by two inclusions. In particular, Conway's [7] *commutation equation* $XL = LX$, where L is a constant language, can be written down as the following resolved system:

$$\begin{cases} X = X, \\ Y = \overline{Y} \cap [(XL \cap \overline{LX}) \cup (\overline{XL} \cap LX)]. \end{cases}$$

This shows that unresolved systems of relations do not provide any additional expressive power in comparison to resolved systems of equations, and without loss of generality one can restrict attention to systems as in Definition 2.

Consider the following example of a resolved system:

Example 1. The system of language equations

$$\begin{cases} X_1 = \overline{X_2 X_3} \cap \overline{X_3 X_2} \cap X_4, \\ X_2 = \{a, b\} X_2 \{a, b\} \cup \{a\}, \\ X_3 = \{a, b\} X_3 \{a, b\} \cup \{b\}, \\ X_4 = \{aa, ab, ba, bb\} X_4 \cup \{\varepsilon\} \end{cases}$$

over the alphabet $\Sigma = \{a, b\}$ has the unique solution $X_1 = \{ww \mid w \in \{a, b\}^*\}$, $X_2 = \{xay \mid x, y \in \{a, b\}^*,\ |x| = |y|\}$, $X_3 = \{xby \mid x, y \in \{a, b\}^*,\ |x| = |y|\}$, $X_4 = \{u \mid u \in \{a, b\}^{2n},\ n \geqslant 0\}$.

If the first variable of the system is interpreted as the main variable (cf. *start symbol* of a context-free grammar), then the system in Example 1 specifies the language $\{ww \mid w \in \{a, b\}^*\}$, which is a well-known non-context-free language.

Judging by this example, language equations with Boolean operations appear to be an extension of context-free grammars enriched by intersection and complementation. Indeed, a large subset of these equations define the language inductively, with the membership of longer strings in the solution determined by the membership of the shorter strings. This subclass gives rise to the family of *Boolean grammars* [19], which inherit many important properties of the context-free grammars, in particular, efficient parsing algorithms.

However, the general case of these language equations turns out to have fundamentally different properties, and a much higher expressive power. The key distinction is that in general the definition of the language is not bound to be inductive, which is illustrated in the following example:

Example 2. Let $\Sigma = \{a\}$. The system of language equations

$$\begin{cases} X = X, \\ Y = \overline{Y} \cap aX \end{cases}$$

has the unique solution $X = \emptyset$, $Y = \emptyset$.

Note that the explicit definition of X tells absolutely nothing about X, as every language satisfies $X = X$. However, if $X \neq \emptyset$ and there is a string $a^\ell \in X$, then the equation for Y becomes a contradiction of the form "$a^{\ell+1} \in Y$ if and only if $a^{\ell+1} \notin Y$". So X may only be $\emptyset$, and then Y is $\emptyset$ as well.

3. Computations of Turing machines

An important language expressible by language equations in the inductive, "context-free" way is the language of valid accepting computations of every Turing machine T. It is defined as the set of all strings of the form $w\natural C_T(w)$, where w is a string accepted by T, $\natural$ is a separator symbol, while $C_T(w)$ is a certain encoding of the computation of T on w, which is basically a concatenation of consecutive configurations of T in its computation on w.

This language was discovered by Hartmanis [10], who proved that it is an intersection of two context-free languages, while its complement is context-free, and inferred many undecidability results for context-free grammars from this fact. The construction was later refined by Baker and Book [4] by using linear context-free grammars. This result is now standard and can formulated as follows:

Proposition 1. *For every Turing machine T with an input alphabet Σ there exists an alphabet Γ disjoint with Σ and a mapping $C_T : L(T) \to \Gamma^+$, such that the language*

$$\mathrm{VALC}(T) = \left\{ w \cdot \natural \cdot C_T(w) \;\middle|\; T \text{ halts on } w \text{ and accepts} \right\} \subseteq \Sigma^* \natural \Gamma^*, \tag{2}$$

where $\natural \notin \Sigma \cup \Gamma$, is an intersection of two linear context-free languages $L_1, L_2 \subseteq \Sigma^* \natural \Gamma^*$. Given T, two such linear context-free grammars can be effectively constructed.

Sketch of a proof. Let T be a Turing machine with the work alphabet V and with the set of states Q, which contains initial state q_0, accepting state q_{acc} and rejecting state q_{rej}. Assume that $\Sigma \cap V = \emptyset$ and $\{a' \mid a \in \Sigma\} \subseteq V$, that is, the input symbols written on the tape are distinguished. Also assume, without loss of generality, that whenever T halts, it halts after an even number of steps. Denote its instantaneous descriptions by strings from $V^* Q V^+$; $x \vdash_T y$ means that there is a transition of T from instantaneous description x to y. Let $\# \notin V \cup Q$. Let $\Gamma = V \cup Q \cup \{\#\}$. For every string $w = a_1 \dots a_\ell$ accepted by T, define

$$\begin{aligned} C_T(a_1 \dots a_\ell) = \big\{ & x_1 \# x_3 \# \cdots \# x_{2k-1} \# x_{2k}^R \# x_{2k-2}^R \# \cdots \# x_2^R \# x_0^R \mid \\ & x_0 = q_0 a_1' \dots a_\ell',\ x_i \vdash_T x_{i+1} \text{ for all } i\ (0 \leqslant i \leqslant 2k-1),\ x_{2k} \in \Gamma^* q_{acc} \Gamma^* \big\}. \end{aligned}$$

Then the language (2) is an intersection of the following two languages over $\Gamma \cup \{\natural\}$:

$$\begin{aligned} L_1 = \big\{ & a_1 \dots a_\ell \natural x_1 \# x_3 \# \cdots \# x_{2k-1} \# x_{2k}^R \# x_{2k-2}^R \# \cdots \# x_2^R \# x_0^R \mid \\ & \ell \geqslant 0,\ a_j \in \Sigma,\ x_0 = q_0 a_1' \dots a_\ell',\ x_{2i-1} \vdash_T x_{2i} \text{ for all } i\ (1 \leqslant i \leqslant k) \big\}, \\ L_2 = \big\{ & a_1 \dots a_\ell \natural x_1 \# x_3 \# \cdots \# x_{2k-1} \# x_{2k}^R \# x_{2k-2}^R \# \cdots \# x_2^R \# x_0^R \mid \\ & \ell \geqslant 0,\ a_j \in \Sigma,\ x_{2i} \vdash_T x_{2i+1} \text{ for all } i\ (0 \leqslant i < k),\ x_{2k} \in \Gamma^* q_{acc} \Gamma^* \big\}. \end{aligned}$$

Note that each of these languages specifies only nested dependencies between x_i, and hence both L_1 and L_2 are linear context-free. The intersection $L_1 \cap L_2$ checks both sets of nested dependencies and thus equals (2). □

The above construction requires adding new symbols to the alphabet Σ. In order to obtain more precise characterizations in the later parts of this paper, it is important to establish this statement without adding any extra symbols. Fortunately, an improved result may be inferred from Proposition 1 by encoding $\mathrm{VALC}(T)$ as follows:

Lemma 1. *Let T be a Turing machine over an alphabet Σ with $|\Sigma| \geqslant 2$, let a, b be two distinct symbols in Σ, and let Γ, $\natural$, C_T and $L_1, L_2 \subseteq \Sigma^* \natural \Gamma^*$ be as in Proposition 1. Define the homomorphism $h : (\Sigma \cup \{\natural\} \cup \Gamma)^* \to \Sigma^*$ as a block code with respect to $\Gamma \cup \{\natural\}$ (with $h(s) \neq h(t)$ and $|h(s)| = |h(t)|$ for all $s, t \in \Gamma \cup \{\natural\}$, $s \neq t$), and as identity on Σ (with $h(c) = c$ for all $c \in \Sigma$). Then the language*

$$h\big(\mathrm{VALC}(T)\big) = \left\{ w \cdot h(\natural) \cdot h\big(C_T(w)\big) \;\middle|\; w \in L(T) \right\} \subseteq \Sigma^* \tag{3}$$

is an intersection of two linear context-free languages $L_1', L_2' \subseteq \Sigma^* \cdot h(\natural) \cdot h(\Gamma^*)$. Given T, two such linear context-free grammars can be effectively constructed.

Proof. Let L_1, L_2 be the linear context-free languages given by Proposition 1, with $\mathrm{VALC}(T) = L_1 \cap L_2$. It is claimed that

$$h(L_1 \cap L_2) = h(L_1) \cap h(L_2). \tag{4}$$

It is known that every injective mapping (in particular, a code) *respects intersection* in the above sense. Though h is not a code, it is injective on $\Sigma^* \natural \Gamma^*$, that is, $h(u \natural x) = h(v \natural y)$ for $u, v \in \Sigma^*$ and $x, y \in \Gamma^*$ implies $u = v$ and $x = y$. Indeed, if $x \neq y$, then the strings $h(\natural x)$ and $h(\natural y)$ are suffix-incomparable because h is a block code on $(\Gamma \cup \{\natural\})^*$, and hence $h(u \natural x)$ cannot be equal to $h(v \natural y)$. Since $L_1, L_2 \subseteq \Sigma^* \natural \Gamma^*$, the statement (4) follows.

It is left to define $L'_1 = h(L_1)$ and $L'_2 = h(L_2)$. These languages are linear context-free by the closure of this language family under homomorphisms, and the corresponding grammars can be constructed from the grammars given by Proposition 1. Then the required representation is $h(\mathrm{VALC}(T)) = L'_1 \cap L'_2$. □

Remark 1. An obvious modification in the proof of Proposition 1 allows representing the following language of terminating and rejecting computations of Turing machines:

$$h\big(\mathrm{VALC}_{\mathrm{rej}}(T)\big) = \big\{ w \cdot h(\natural) \cdot h\big(C_T(w)\big) \;\big|\; T \text{ halts on } w \text{ and rejects} \big\} \subseteq \Sigma^*.$$

4. Solutions modulo a language

A vector of languages $(L_1, \ldots, L_n)$ is a solution of a system if the substitution $X_i = L_i$ turns every equation into an equality, that is, every string $w \in \Sigma^*$ belongs to its left-hand side if and only if it is in its right-hand side. If instead of all strings w, only strings belonging to a subset $M \subset \Sigma^*$ are considered, this defines a *solution modulo M*. This notion is formalized in the following definitions.

Definition 3. Two languages $K, L \subseteq \Sigma^*$ are called *equal modulo* a third language $M \subseteq \Sigma^*$ (denoted $K = L \pmod M$), if $K \cap M = L \cap M$. This relation is extended to vectors of languages by saying that $K = (K_1, \ldots, K_n)$ equals $L = (L_1, \ldots, L_n)$ modulo M if $K_i = L_i \pmod M$ for all i.

In particular, every two languages are equal modulo $\emptyset$. Equality modulo Σ^* means equality in the ordinary sense. Obviously, equality modulo M implies equality modulo every subset of M. For every fixed M, equality modulo M is an equivalence relation.

The moduli M used throughout this paper shall be languages of the following special form:

Definition 4. For every string $w \in \Sigma^*$, let $\mathrm{substrings}(w) = \{y \mid w = xyz \text{ for some } x, z \in \Sigma^*\}$. For every language $L \subseteq \Sigma^*$, define $\mathrm{substrings}(L) = \bigcup_{w \in L} \mathrm{substrings}(w)$. A language L is said to be *closed under substrings* (or *substring-closed*), if $\mathrm{substrings}(L) = L$, that is, all substrings of every string from L are also in L.

For instance, the languages $\emptyset$, $\Sigma^{\leqslant \ell}$ with $\ell \geqslant 0$, and Σ^* are substring-closed.

The reason for using only substring-closed moduli is that the equations may contain concatenation, and the membership of a string in the concatenation of some languages depends on the membership of its substrings in those languages. Thus the closure of the modulus under substrings is essential for the following basic property to hold:

Lemma 2. *Let $\varphi(X_1, \ldots, X_n)$ be an expression on languages over Σ. Let $M \subseteq \Sigma^*$ be any substring-closed language. Then, if two vectors of languages, $L = (L_1, \ldots, L_n)$ and $L' = (L'_1, \ldots, L'_n)$, are equal modulo M, then $\varphi(L_1, \ldots, L_n)$ and $\varphi(L'_1, \ldots, L'_n)$ are also equal modulo M.*

Proof. By the symmetry, it is enough to prove that if $w \in M$ is in $\varphi(L_1, \ldots, L_n)$, then w is in $\varphi(L'_1, \ldots, L'_n)$ as well. The proof is a straightforward induction on the structure of φ.

- If $\varphi = C \subseteq \Sigma^*$ is a constant language, the statement is clear.
- Let $\varphi = X_i$. Since $L_i = L'_i \pmod M$ by assumption, the result holds.
- Let $\varphi = \psi \xi$. If $w \in \varphi(L)$, then there exists a factorization $w = uv$, with $u \in \psi(L)$ and $v \in \xi(L)$. Since $u, v \in M$ by the closure of M under substrings, by the induction hypothesis, $u \in \psi(L')$ and $v \in \xi(L')$. Therefore, $uv \in \varphi(L')$.
- The cases of Boolean operations are proved analogously, without using the closure of M under substrings. □

Now the notion of a solution modulo a substring-closed language can be defined:

Definition 5. Let $X = \varphi(X)$ be a system of equations and let M be a substring-closed language. A vector $L = (L_1, \ldots, L_n)$ is said to be a *solution modulo M* of the system if $\varphi_i(L) = L_i \pmod M$ for every i.

An important fact concerning this notion is that a vector's $L = (L_1, \ldots, L_n)$ being a solution modulo M depends entirely on this vector taken modulo M, that is, on $(L_1 \cap M, \ldots, L_n \cap M)$. Therefore, there are only finitely many candidates for being solutions modulo M. This is proved as follows:

Lemma 3. *Let $X = \varphi(X)$ be a system, let M be a substring-closed language, let L and L' be two vectors of languages equal modulo M. Then L is a solution of the system modulo M if and only if L' is a solution of the system modulo M.*

Proof. Suppose that L is a solution modulo M. Then $L' = L = \varphi(L) = \varphi(L') \pmod{M}$, where the first equality is by assumption, the second one holds since L is a solution modulo M, and the third one follows by Lemma 2. Therefore, $L' = \varphi(L') \pmod{M}$, that is, L' is a solution modulo M. □

In view of this property, equality of solutions modulo some M shall always be considered in the sense of equality modulo M. This notion of equality will be used whenever a solution modulo M is said to be unique.

Let us state an obvious property of solutions modulo a language:

Proposition 2 *(On nested moduli). A solution of a system $X = \varphi(X)$ modulo some language M closed under substrings is its solution modulo every substring-closed subset of M. In particular, every solution in the ordinary sense (that is, modulo Σ^*) is a solution modulo every substring-closed language.*

Besides being substring-closed, the moduli considered in the following will typically be *finite*. There are countably many such moduli. The reason to consider solutions modulo finite substring-closed languages is that some properties of solutions of a system may be reformulated as statements on solutions modulo languages of this form, using quantification over moduli. Consider a trivial case of such reformulation:

Proposition 3. *If two languages (vectors of languages) K, L are equal modulo every finite substring-closed language M, then $K = L$. Equivalently, if two languages (vectors of languages) are not equal, then they are not equal modulo some finite substring-closed language.*

Indeed, if $K \neq L$, then the symmetric difference $K \triangle L$ contains some string w, and therefore K and L are not equal modulo substrings(w).

In the following, several such statements will be established for solutions of language equations. Note that directly stated properties of solutions, such as "the system $X = \varphi(X)$ has a unique solution" are second-order formulae, as the quantification is over *sets* of strings. On the other hand, their reformulations through solutions modulo finite languages will be first-order formulae by definition. These first-order characterizations will form the basis of the analysis of language equations.

5. Existence of a solution

For some families of language equations, the question of solution existence is trivial, as there is always a solution. This is the case, for instance, for the systems of Ginsburg and Rice [9], as well as for their generalization with intersection [18]. However, it is easy to see that a system of language equations with complementation does not necessarily have a solution: consider an equation $X = \overline{X}$. In this section, a necessary and sufficient condition of existence of solutions is developed, which is based upon solutions modulo finite languages.

Several useful results on the relationship between solutions modulo finite languages and solutions in the ordinary sense (which may be regarded in this context as solutions modulo Σ^*) have to be proved first.

Lemma 4 *(Finite refutation of a non-solution). If $L = (L_1, \ldots, L_n)$ is not a solution of a system $X = \varphi(X)$, then there exists a finite language M closed under substrings, such that L is not a solution of the system modulo M.*

Equivalently, if a vector of languages L is a solution of a system $X = \varphi(X)$ modulo every finite language M closed under substrings, then L is a solution of the system.

Proof. If L is not a solution of $X = \varphi(X)$, then $L \neq \varphi(L)$. By Proposition 3, there exists a modulus M closed under substrings, such that $L \neq \varphi(L) \pmod{M}$, which means that L is not a solution modulo M. □

In order to state the next result, some new terminology has to be introduced.

Definition 6 *(Extension and refutation of solutions modulo M).* Let $X = \varphi(X)$ be a system. Let $M \subseteq M'$ be two substring-closed languages. Let L be a solution modulo M. Then L is said to be *extendable to M'* if there exists a solution L' modulo M' with $L = L' \pmod{M}$; in this case L' is called *an extension of L to M'*. Otherwise, if there is no such L', then L is said to be *refuted modulo M'*.

A solution L modulo a finite substring-closed M is said to be *refutable*, if it is refuted modulo some finite substring-closed $M' \supseteq M$, and *unrefutable* otherwise.

Consider the system in Example 2 and the modulus $M = \{\varepsilon\}$. There are two solutions modulo M: $(\emptyset, \emptyset)$ and $(\{\varepsilon\}, \emptyset)$. The former is extendable to every M', as $(\emptyset, \emptyset)$ is a solution of the system. The other one, $(\{\varepsilon\}, \emptyset)$, is refuted modulo $M' = \{\varepsilon, a\}$, as $\varepsilon \in X$ turns the equation for Y into a contradiction of the form "$a \in Y$ if and only if $a \notin Y$". Therefore, all refutable solutions modulo M are refuted modulo this M'. The next lemma shows that such an M' always exists:

Lemma 5 *(Refutation of refutable solutions). Let $X = \varphi(X)$ be a system of language equations and let M be a finite language closed under substrings. Then there exists a finite language $M' \supseteq M$ closed under substrings, such that all refutable solutions modulo M are refuted modulo M'.*

Proof. Let $L^{(1)}, \ldots, L^{(m)}$ be all refutable solutions of the system modulo M. For all i with $1 \leqslant i \leqslant m$, let M_i be a finite substring-closed language modulo which $L^{(i)}$ is refuted. Define $M' = \bigcup_{i=1}^{k} M_i$. Then $L^{(1)}, \ldots, L^{(k)}$ are all refuted modulo M'. □

By definition, a solution modulo M is unrefutable if it can be extended to a solution modulo every finite superset of M. However, that solution modulo a superset can be refutable itself, and, in fact, one can imagine a hypothetical situation that a solution L modulo M might be extendable to every $M' \supseteq M$, but every such extension would be refutable. The following stronger claim rules out this possibility:

Lemma 6 *(Finite extension of an unrefutable solution). Let $X = \varphi(X)$ be a system, let M be a finite substring-closed language, let L be an unrefutable solution modulo M. Then, for every finite $M' \supseteq M$ closed under substrings, L can be extended to an unrefutable solution modulo M'.*

In other words, for every such M' there exists an unrefutable solution modulo M' that coincides with L modulo M.

Proof. Let

$$L^{[1]}, L^{[2]}, \ldots, L^{[m]} \tag{5}$$

be all solutions modulo M' that coincide with L modulo M. Let us prove that at least one of these solutions modulo M' must be unrefutable. Suppose the contrary, that is, that each $L^{[i]}$ is refutable. Then, by Lemma 5, all (5) are refuted modulo some language $M'' \supseteq M'$.

Since L is an unrefutable solution modulo M, it is not refuted modulo M'', that is, there exists a solution L'' modulo M'', which coincides with L modulo M. Define L' as the restriction of L'' modulo M'. By the construction of the collection (5), L' must be among $\{L^{[i]}\}_{i=1}^{m}$ and thus be refuted modulo M''. However, L'' is a witness to the contrary.

The contradiction obtained proves that one of the solutions (5) modulo M' must be unrefutable. Since all (5) are extensions of L to M', it has been proved that L can be extended to an unrefutable solution modulo M'. □

The next step is to apply Lemma 6 for larger and larger languages M', and to take the limit of this process. This will lead to an extension to a solution in the ordinary sense.

Lemma 7 *(Infinite extension of an unrefutable solution). Let $X = \varphi(X)$ be a system, let M be a finite substring-closed language, let L_M be an unrefutable solution modulo M. Then L_M can be extended to a solution L of the system.*

The statement of the lemma can be reformulated without using the notion of unrefutable solution as follows: if for every finite language $M' \supseteq M$ closed under substrings the system has a solution modulo M', which coincides with L_M modulo M, then the system has a solution that also coincides with L_M modulo M.

Proof. Consider any ascending sequence of nested finite moduli (each closed under substrings)

$$M = M_0 \subset M_1 \subset M_2 \subset \cdots \subset M_k \subset \cdots$$

that converges to Σ^* in the sense that $\bigcup_{k=0}^{\infty} M_k = \Sigma^*$. Let us show that there exists a corresponding sequence of vectors of finite languages

$$L_M = L^{(0)}, L^{(1)}, L^{(2)}, \ldots, L^{(k)}, \ldots, \tag{6}$$

where every $L^{(k)}$ is an unrefutable solution modulo the corresponding M_k, and which is componentwise increasing in the sense that $L_i^{(k)} \subseteq L_i^{(k+1)}$. The proof is not constructive; the existence of consecutive terms of this sequence is inductively shown.

Basis. $L^{(0)} = L_M$ is an unrefutable solution modulo M by the assumption.

Induction step. Let $L^{(k)}$ be an unrefutable solution modulo M_k. By Lemma 6, it can be extended to an unrefutable solution $L^{(k+1)}$ modulo M_{k+1}. Next, by the definition of extension, $L^{(k)} = L^{(k+1)} \pmod{M_k}$. Then $L_i^{(k)} \subseteq L_i^{(k+1)}$ for each i.

Having obtained the increasing sequence (6), consider its limit

$$L = \left(\bigcup_{k=0}^{\infty} L_1^{(k)}, \ldots, \bigcup_{k=0}^{\infty} L_n^{(k)} \right).$$

Clearly, $L = L^{(k)} \pmod{M_k}$ for every k, and therefore L is a solution modulo every M_k. It remains to show that L is a solution modulo every finite language M' closed under substrings. Since the sequence $\{M_k\}_{k=0}^{\infty}$ is ascending and $\bigcup_{k=0}^{\infty} M_k = \Sigma^*$, there exists k, such that $M' \subseteq M_k$. Because L is a solution modulo M_k, it is a solution modulo M' by Proposition 2. Therefore, L is a solution of the whole system by Lemma 4. □

Using Lemma 7, the following characterization of systems of equations that have solutions can be obtained:

Theorem 1 *(Criterion of solution existence). A system has a solution if and only if it has a solution modulo every finite substring-closed language.*

Proof. $\Rightarrow$ If $L = (L_1, \ldots, L_n)$ is a solution, then it is a solution modulo every finite language closed under substrings by Proposition 2.

$\Leftarrow$ Let $M = \emptyset$ and consider that $L_M = (\emptyset, \ldots, \emptyset)$ is, trivially, a solution modulo M. Assume the system has a solution modulo every finite substring-closed language $M' \subseteq \Sigma^*$. Then L_M is extendable to every such M', that is, L_M is an unrefutable solution modulo M. Therefore, by Lemma 7, the system has a solution. □

The condition given by Theorem 1 is actually a first-order formula with one universal quantifier over a countable set. Hence, the set of systems that have at least one solution is co-recursively enumerable. The problem is hard for this class as well (that is, undecidable), which was first proved by Charatonik [6] in a different context.

Theorem 2. *The set of systems of language equations with Boolean operations, linear concatenation and singleton constants that have solutions is co-r.e.-complete. It remains co-r.e. for unrestricted concatenation and any recursive constants.*

Proof. *Membership in co-r.e.* (*unrestricted concatenation, recursive constants*). The complement of the problem is accepted by a Turing machine that considers all finite moduli and accepts if the given system has no solutions modulo any M. If no such modulus is found, the machine does not terminate. Then, according to Theorem 1, the machine accepts if and only if the system has no solutions.

Co-r.e.-hardness (*linear concatenation, singleton constants*). Reduction from the co-r.e.-complete Turing machine emptiness problem. Let T be any given Turing machine and construct a system of language equations $X_i = \varphi_i(X_1, \ldots, X_n)$ representing the language $\mathrm{VALC}(T)$. This is done according to Proposition 1: linear context-free grammars are transcribed as equations, and then intersection is used in the equation for X_1 to obtain $X_1 = \mathrm{VALC}(T)$ in the unique solution. Then consider the system of equations

$$\left.\begin{aligned} Y &= \overline{Y} \cap X_1, \\ X_1 &= \varphi_1(X_1, \ldots, X_n), \\ &\vdots \\ X_n &= \varphi_n(X_1, \ldots, X_n) \end{aligned}\right\} \begin{array}{l} \text{a system for the language} \\ \mathrm{VALC}(T) = \{w \natural C_T(w) \mid T \text{ halts on } w \text{ and accepts}\}, \\ \text{where } w \in \Sigma^* \text{ and } C_T(w) \in \Gamma^*. \end{array}$$

The equation for Y is a contradiction unless $X_1 = \emptyset$, and therefore the system has a solution if and only if $\mathrm{VALC}(T) = \emptyset$, which holds if and only if $L(T) = \emptyset$. This completes the reduction. □

6. Uniqueness of a solution

In Section 5 it was proved that a system has solutions if and only if it has solutions modulo every language closed under substrings. However, it turns out that the same property does not hold with respect to the uniqueness of solution, and a system can have multiple solutions modulo every finite language, but still a unique solution.

This is demonstrated by the system in Example 2. It has the unique solution $(\emptyset, \emptyset)$. However, for every finite nonempty $M \subset a^*$ closed under substrings, which is of the form $a^{\leqslant \ell} = \{\varepsilon, a, aa, \ldots, a^\ell\}$ for some $\ell \geqslant 0$, the system has exactly two solutions modulo every such M: $(\{a^\ell\}, \emptyset)$ and $(\emptyset, \emptyset)$. Of these, the former is refuted modulo $a^{\leqslant \ell+1}$, while the latter can be

extended to the modulus $a^{\leqslant \ell+1}$. Thus, in order to check the membership of a string of length ℓ in the components of the unique solution, one has to consider strings of length $\ell+1$.

This illustrates the following property of systems of language equations with a unique solution: the membership of *longer* strings in the solution may in fact determine the membership of *shorter* strings by refuting one of the alternative solutions modulo the smaller language. From Lemma 4 it is known that every "wrong" solution modulo a finite language (that is, one that is not extendable to a solution) has a refutation modulo some greater finite language, and thus if a system has a unique solution, then, for every finite M closed under substrings, all but one of the solutions modulo M should be refutable. This necessary condition of solution uniqueness is actually sufficient, and the following theorem provides a first-order characterization of systems with a unique solution similar to Theorem 1:

Theorem 3 *(Criterion of solution uniqueness). A system has a unique solution if and only if for every finite language M closed under substrings there exists a finite language $M' \supseteq M$ closed under substrings, such that the system has at least one solution modulo M', and all the solutions modulo M' are equal modulo M.*

Proof. $\Rightarrow$ Let a system $X = \varphi(X)$ have a unique solution L, and fix a finite substring-closed M. By Lemma 5, all refutable solutions modulo M are refuted modulo some finite superset of M; denote it by M'. L is a solution modulo M' by Proposition 2; it remains to argue that all solutions modulo M' must coincide modulo M.

Suppose the contrary, that there exist two solutions L and L' modulo M', which are different modulo M. Let $L_M \neq L'_M$ be these solutions taken modulo M. They are not refuted modulo M', and therefore, by the choice of M' and by Lemma 5, they are unrefutable. Hence, by Lemma 7, they can be extended to distinct solutions of the whole system, which contradicts the uniqueness of solution and proves the necessity claim.

$\Leftarrow$ Let a system $X = \varphi(X)$ be such that for every finite modulus M closed under substrings there exists a finite modulus $M' \supseteq M$ closed under substrings, such that all solutions of the system modulo M' are equal modulo M.

Suppose that the system has at least two distinct solutions, $L = (L_1, \ldots, L_n)$ and $L' = (L'_1, \ldots, L'_n)$. Then $L \neq L'$ implies that $L \neq L' \pmod{M}$ for some finite substring-closed modulus M. By assumption, for this particular M there exists a finite modulus $M' \supseteq M$ closed under substrings, such that all solutions modulo M' are equal modulo M. By Proposition 2, L and L' are solutions of the system modulo M', and therefore must coincide modulo M, which yields a contradiction. □

The necessary and sufficient condition of solution uniqueness given by Theorem 3 specifies the set of systems that have a unique solution by a first-order formula with one universal quantifier and one existential quantifier over a countable set. Therefore, the problem is in the second level of the arithmetical hierarchy, namely in Π_2. The next theorem shows that it is complete for this class:

Theorem 4. *The set of systems of language equations with Boolean operations, linear concatenation and singleton constants that have exactly one solution is Π_2-complete. The similar set for systems for unrestricted concatenation and any recursive constants remains in Π_2.*

Proof. *Membership in Π_2 (unrestricted concatenation).* According to Theorem 3, the uniqueness of a solution is expressed by the following first-order formula

$$\phi(w) = \forall x\, \exists y\; R(x, y, w), \tag{7}$$

where R is a recursive predicate that evaluates to true on a triple (x, y, w) if and only if

(i) w is a syntactically valid description of an alphabet Σ and of a system of language equations over Σ,
(ii) x and y describe two finite languages $M_x \subseteq M_y \subset \Sigma^*$, each closed under substrings,
(iii) the system specified by w has solutions modulo the language given by y, and all of these solutions coincide modulo the language given by x.

The correctness of this representation is given by Theorem 3, while first-order formulae of the form (7) are precisely those that form the class Π_2.

Π_2-hardness (linear concatenation, singleton constants). Reduction from the Turing machine universality problem, which is stated as "Given a Turing machine T over an alphabet Σ, determine whether $L(T) = \Sigma^*$" and is known to be complete for Π_2 [20, §14.8].

Fix T, a Turing machine over an alphabet Σ. Let $X_i = \varphi_i(X_1, \ldots, X_n)$ with $i \in \{1, \ldots, n\}$ be a system of language equations with a unique solution $(L_1, \ldots, L_n)$, in which L_1 is the language $\mathrm{VALC}(T)$ of valid accepting computations of T. Such a system exists and can be effectively constructed by Proposition 1. Add four more variables, Y, Z_1, Z_2 and T, and construct the following system:

$$Y = Y, \tag{8a}$$

$$Z_1 = Y\natural \cup \bigcup_{s\in\Gamma} Z_1 s, \tag{8b}$$

$$Z_2 = \{\varepsilon\} \cup \bigcup_{a\in\Sigma} aZ_2, \tag{8c}$$

$$T = \overline{T} \cap \big((X_1 \cap \overline{Z_1}) \cup (Y \cap \overline{Z_2})\big), \tag{8d}$$

$$\left.\begin{array}{l} X_1 = \varphi_1(X_1,\ldots,X_n), \\ \vdots \\ X_n = \varphi_n(X_1,\ldots,X_n) \end{array}\right\} \begin{array}{l}\text{a system for the language} \\ \text{VALC}(T) = \{w\natural C_T(w) \mid T \text{ halts on } w \text{ and accepts}\}, \\ \text{where } w \in \Sigma^* \text{ and } C_T(w) \in \Gamma^*.\end{array} \tag{8e}$$

Here the equation (8b) specifies $Z_1 = Y\natural\Gamma^*$, while the equation (8c) represents $Z_2 = \Sigma^*$. Hence the equation for T implements the following two inclusions using the method (1):

$$\text{VALC}(T) \subseteq Y\natural\Gamma^*, \tag{9a}$$

$$Y \subseteq \Sigma^*. \tag{9b}$$

The inclusion (9a) states that for every string $w \in \Sigma^*$ accepted by the Turing machine, the corresponding computation history $w\natural C_T(w)$ must be in $Y\natural\Gamma^*$. This implies $w \in Y$, that is, every string accepted by T must be in Y. The second constraint (9b) restricts Y to subsets of Σ^*. Therefore, the set of solutions of the system (8) is

$$\left\{ \big(L, L\natural\Gamma^*, \Sigma^*, \emptyset, L_1, \ldots, L_n\big) \;\middle|\; \underbrace{L(T) \subseteq L \subseteq \Sigma^*}_{(10^*)} \right\}. \tag{10}$$

Clearly, the solution of (8) is unique if and only if the bounds (10*) are tight, that is, if $L(T) = \Sigma^*$.

This completes the reduction from the Turing machine universality problem. Since the latter is Π_2-complete, the Π_2-hardness of the solution uniqueness problem for systems of language equations is established. □

7. Least and greatest solutions

Every system of language equations of the kind defined by Ginsburg and Rice [9], possibly with intersection [18], is known to have two special solutions: the least and the greatest one, which are the componentwise intersection and the componentwise union, respectively, of all solutions of the system. As the right-hand sides of such systems are monotone and continuous functions, these solutions always exist and can be obtained by fixpoint iteration starting from the vectors $(\emptyset,\ldots,\emptyset)$ and $(\Sigma^*,\ldots,\Sigma^*)$. However, once the use of complementation in equations is allowed, this property is lost. For instance, the system

$$\begin{cases} X = \overline{Y}, \\ Y = Y \end{cases}$$

has the set of solutions $\{(\overline{L}, L) \mid L \subseteq \Sigma^*\}$, and these solutions are pairwise incomparable. Accordingly, having a least (greatest) solution is a nontrivial property of a system, which should be studied in the same way as solution uniqueness.

Let us formally introduce comparison of n-tuples of languages:

Definition 7. A partial order "$\sqsubseteq$" on the set of languages over an alphabet Σ is defined as $K \sqsubseteq L$ if $K \subseteq L$. For each $n \geqslant 1$, this order is extended to vectors of n languages as $(K_1,\ldots,K_n) \sqsubseteq (L_1,\ldots,L_n)$ if $K_i \sqsubseteq L_i$ for all i. Languages (vectors) K and L are said to be *incomparable*, if $K \not\sqsubseteq L$ and $K \not\sqsupseteq L$.

Definition 8. Let $X = \varphi(X)$ be a system of language equations. A vector L is said to be the *least* (*greatest*) *solution* of the system if it is a solution and for every solution L' it holds that $L \sqsubseteq L'$ ($L \sqsupseteq L'$, respectively).

Comparison of languages can also be done modulo a language M, similarly to equality modulo languages.

Definition 9. A language $K \subseteq \Sigma^*$ is said to be *less than or equal to* $L \subseteq \Sigma^*$ *modulo* $M \subseteq \Sigma^*$, denoted $K \sqsubseteq L \pmod M$, if $K \cap M \subseteq L \cap M$. This relation is extended to vectors of languages so that $(K_1,\ldots,K_n) \sqsubseteq (L_1,\ldots,L_n) \pmod M$ if $K_i \sqsubseteq L_i \pmod M$ for all i. Languages (vectors) K and L are said to be *incomparable modulo* M, if $K \not\sqsubseteq L \pmod M$ and $L \not\sqsubseteq K \pmod M$.

It is easy to note that $K \sqsubseteq L \pmod M$ implies $K \sqsubseteq L \pmod{M_0}$ for all $M_0 \subseteq M$. If K, L are incomparable modulo some M, they are incomparable modulo every superset of M.

The following analogue of Proposition 3 can be stated:

Proposition 4. *If for two languages (vectors of languages) K, L it holds that $K \sqsubseteq L \pmod{M}$ for every finite M closed under substrings, then $K \sqsubseteq L$. Equivalently, if for two languages (vectors of languages) K, L it holds that $K \not\sqsubseteq L$, then $K \not\sqsubseteq L \pmod{M}$ for some finite substring-closed M.*

If two languages (vectors of languages) are incomparable, then they are incomparable modulo some finite substring-closed language.

Proof. The proof of the first part is immediate: if $K \not\sqsubseteq L$, then there exists a string $w \in K \setminus L$, and hence $K \not\sqsubseteq L \pmod{\text{substrings}(w)}$.

If K, L are incomparable, then $K \not\sqsubseteq L$ and $K \not\sqsupseteq L$. According to the first part, there exist finite M_1, M_2 closed under substrings, such that $K \not\sqsubseteq L \pmod{M_1}$ and $K \not\sqsupseteq L \pmod{M_2}$. Then K and L are incomparable modulo $M_1 \cup M_2$. □

Similarly to the case of uniqueness of solution, the existence of a least (greatest) solution cannot be reduced to having a least (greatest) solution modulo every finite M. Consider the following variant of Example 2:

Example 3. The system

$$\begin{cases} X = X, \\ Y = \overline{Y} \cap ((X \cap aX) \cup a^2 X) \end{cases}$$

has a unique solution $(\emptyset, \emptyset)$, which is trivially the least and the greatest.

For each substring-closed language $M = a^{\leqslant \ell}$, there are three solutions modulo M: $L = (\emptyset, \emptyset)$, $L' = (\{a^{\ell-1}\}, \emptyset)$ and $L'' = (\{a^{\ell}\}, \emptyset)$. Of these, L is unrefutable, L' is refuted modulo $a^{\leqslant \ell+1}$ and L'' is refuted modulo $a^{\leqslant \ell+2}$.

In this example, there is a least solution and it is the least modulo every M. On the other hand, though there is a greatest solution, there is no greatest solution modulo any M (except for $M = \emptyset$ and $M = \{\varepsilon\}$), but if one considers only *unrefutable* solutions modulo M, then there is always the greatest among them.

The next lemma shows that such a property holds for every system with a least (greatest) solution. Furthermore, since all refutable solutions are known to be refuted modulo some finite language by Lemma 5, this fact can be stated as follows:

Lemma 8. *If a system has a least (greatest) solution L then for every finite substring-closed language $M \subset \Sigma^*$ there exists a finite substring-closed language $M' \supseteq M$, such that L is the least (greatest, respectively) solution modulo M extendable to M'.*

Proof. Fix an arbitrary finite language M closed under substrings. By Lemma 5, there should exist a finite substring-closed language, modulo which all refutable solutions modulo M are refuted. Denote it by M'.

Suppose L is not the least solution modulo M extendable to M', that is, there exists a solution $\widetilde{L}$ modulo M' with $L \not\sqsubseteq \widetilde{L} \pmod{M}$. By the choice of M', $\widetilde{L}$ is an unrefutable modulo M, and therefore, in accordance with Lemma 7, it can be extended to a solution $\widehat{L}$ of the whole system, which inherits the property $L \not\sqsubseteq \widehat{L}$. The latter contradicts the assumption that L is the least solution of the system. □

The condition in Lemma 8 can be developed into the following necessary and sufficient condition of having a least or a greatest solution:

Theorem 5 *(Criterion of least/greatest solutions). A system has a least (greatest) solution if and only if for every finite language M closed under substrings there exists a finite language $M' \supseteq M$ closed under substrings, such that there is the least (the greatest) among the solutions modulo M extendable to M'.*

Note that M' in Theorem 5 need not be the language modulo which all refutable solutions modulo M are refuted. In particular, looking at Example 3, for $M = a^{\leqslant \ell}$, one possible value of M' is $a^{\leqslant \ell+1}$: the solutions L and L'' modulo M are extendable to $a^{\leqslant \ell+1}$, while the solution L' modulo M is refuted modulo $a^{\leqslant \ell+1}$, and since $L \sqsubset L''$, the condition of Theorem 5 is met. The modulus $M' = a^{\leqslant \ell+2}$ satisfies the theorem as well, and L is the unique solution modulo M extendable to this M'. But even though not every modulus M' in the theorem gives reliable information about the least (greatest) solution modulo M, the criterion holds as stated.

Proof. The below proof applies to least solutions; the case of greatest solutions is proved identically.

⊜ Assuming that the system has a least solution, for every such M the corresponding M' is given by Lemma 8.

⊜ Let a system $X = \varphi(X)$ satisfy the condition written in the statement of the theorem. Consider an arbitrary ascending sequence

$$M_1 \subset M_2 \subset \cdots \subset M_n \subset \cdots \tag{11}$$

of finite substring-closed languages that converges to Σ^* in the sense that $\bigcup_{n=1}^{\infty} M_n = \Sigma^*$. For each $i \geqslant 1$, let $M'_i \supseteq M_i$ be the finite substring-closed language, modulo which all refutable solutions modulo M_i are refuted; it is known to exist by Lemma 5.

For every such M'_i, by assumption, there exists a finite M''_i closed under substrings, such that there is the least among the solutions modulo M'_i that are not refuted modulo M''_i. Denote this solution modulo M'_i by L'_i. Let L_i be L'_i taken modulo $M_i \subseteq M'_i$. Let us establish the properties of the sequence $\{L_i\}$.

Claim 1. *L_i is an unrefutable solution modulo M_i.*

Indeed, by its construction, L_i is not refuted modulo M'_i (this is witnessed by L'_i), while every refutable solution modulo M_i is (by the choice of M'_i).

Claim 2. *L_i is the least among unrefutable solutions modulo M_i.*

Suppose there is another unrefutable solution $\widetilde{L}_i$ modulo M_i, such that $L_i \not\sqsubseteq \widetilde{L}_i \pmod{M_i}$. By Lemma 6, there consequently exists an unrefutable solution $\widetilde{L}'_i$ modulo M'_i, with $\widetilde{L}_i = \widetilde{L}'_i \pmod{M_i}$. This implies $L'_i = L_i \not\sqsubseteq \widetilde{L}_i = \widetilde{L}'_i \pmod{M_i}$, and hence $L'_i \not\sqsubseteq \widetilde{L}'_i \pmod{M'_i}$. As both L'_i and $\widetilde{L}'_i$ are solutions modulo M'_i that are not refuted modulo M''_i, this means that L'_i is not the least among such solutions, which contradicts its construction.

Claim 3. *$L_i \sqsubseteq L_{i+1}$ for all i.*

Since L_i is the least among unrefutable solutions modulo M_i, while L_{i+1} is an unrefutable solution modulo M_i, it follows that $L_i \sqsubseteq L_{i+1} \pmod{M_i}$. On the other hand, L_i, by definition, is taken modulo M_i, and thus none of its components contain strings outside of M_i, that is, $\emptyset = L_i \sqsubseteq L_{i+1} \pmod{\overline{M_i}}$. Therefore, $L_i \sqsubseteq L_{i+1}$.

Claim 4. *$L_i = L_{i+1} \pmod{M_i}$ for all i.*

The unrefutable solution L_i modulo M_i is extendable to some unrefutable solution $\widetilde{L}_{i+1}$ modulo M_{i+1} by Lemma 6. Since L_{i+1} is the least among unrefutable solutions modulo M_{i+1} by Claim 2, $L_{i+1} \sqsubseteq \widetilde{L}_{i+1}$. This implies

$$L_{i+1} \sqsubseteq \widetilde{L}_{i+1} = L_i \pmod{M_i},$$

and since the converse inequality $L_i \sqsubseteq L_{i+1} \pmod{M_i}$ is known from Claim 3, the statement is proved.

Thus an increasing sequence of solutions modulo the languages (11) has been obtained:

$$L_1 \sqsubseteq L_2 \sqsubseteq \cdots \sqsubseteq L_n \sqsubseteq \cdots, \tag{12}$$

where each L_i is the least among unrefutable solutions modulo M_i. As a monotone sequence, (12) converges to a certain vector L, such that $L = L_i \pmod{M_i}$ for all $i \geqslant 1$. It is left to prove that L is the least solution of the system.

As in the proof of Lemma 7, it can be inferred that L is a solution. Suppose the existence of some other solution $\widetilde{L}$, with $L \not\sqsubseteq \widetilde{L}$. Then, by Proposition 4, there exists a finite language M closed under substrings, for which $L \not\sqsubseteq \widetilde{L} \pmod{M}$. Let i be a number, such that $M \subseteq M_i$. Then $L \not\sqsubseteq \widetilde{L} \pmod{M_i}$, and therefore for L_i (which coincides with L taken modulo M_i by Claim 4) it holds that $L_i \not\sqsubseteq \widetilde{L} \pmod{M_i}$, where $\widetilde{L}$ taken modulo M_i is an unrefutable solution modulo this language. Hence, L_i is not the least among unrefutable solutions modulo M_i, which contradicts Claim 2. □

As in the case of solution uniqueness, the statement of Theorem 5 is again a Π_2-formula, which leads to the following computational characterization:

Theorem 6. *The set of systems of language equations with Boolean operations, linear concatenation and singleton constants that have a least (greatest) solution is Π_2-complete. It remains Π_2-complete if concatenation is unrestricted and any recursive constants are allowed.*

Proof. *Membership in Π_2 (unrestricted concatenation, recursive constants).* As in the proof of Theorem 4, the property of having a least (greatest) solution is represented by the following first-order formula:

$$\phi(w) = \forall x\, \exists y\; R(x, y, w),$$

where R is true on (x, y, w) if and only if

(i) w describes an alphabet Σ and a system of language equations over Σ,
(ii) x and y describe finite substring-closed languages $M_x \subseteq M_y \subset \Sigma^*$,

(iii) the system specified by w has solutions modulo the language represented by y, and among them there is the least (the greatest, resp.) modulo the language given by x.

The correctness of this Π_2 representation is given by Theorem 5.

Π_2-hardness (*linear concatenation, singleton constants*). Reduction from the Turing machine universality problem. Given a Turing machine, consider the system (8) augmented by an additional equation

$$Y' = \overline{Y}. \tag{13}$$

Let us show that the resulting system (13), (8) in variables $(Y', Y, Z_1, Z_2, T, X_1, \ldots, X_n)$ has both a least and a greatest solution if $L(T) = \Sigma^*$, and neither a least nor a greatest solution if $L(T) \neq \Sigma^*$.

Indeed, if $L(T) = \Sigma^*$, then (8) has the unique solution

$$\big(\Sigma^*, \Sigma^* \natural \Gamma^*, \Sigma^*, \emptyset, L_1, \ldots, L_n\big),$$

and therefore (13, 8) has the unique solution

$$\big(\emptyset, \Sigma^*, \Sigma^* \natural \Gamma^*, \Sigma^*, \emptyset, L_1, \ldots, L_n\big),$$

which is at the same time the least and the greatest.

If $L(T) \neq \Sigma^*$, then the set of solutions of (13), (8) is

$$\left\{ \big(\bar{L}, L, L \natural \Gamma^*, \Sigma^*, \emptyset, L_1, \ldots, L_n\big) \;\middle|\; \underbrace{L(T) \subseteq L \subseteq \Sigma^*}_{(14^*)} \right\}, \tag{14}$$

which consists of multiple pairwise incomparable vectors of languages. This shows the correctness of the reduction and proves Π_2-hardness. □

8. Families of languages

Consider systems of language equations that have a unique, least or greatest solution. Such systems can be regarded as specifications of the components of these solutions, and every class of language equations accordingly defines a family of formal languages. The family defined by equations with union and concatenation is the family of context-free languages [9]. Equations with union, intersection and concatenation correspond to *conjunctive grammars*, an extension of the context-free grammars [17,18]. Normal form theorems for these grammars show that unique, least and greatest solutions yield the same family of languages. On the other hand, for language equations with all Boolean operations and concatenation, the expressive power of unique, least and greatest solutions turns out to be different, and it will now be determined.

Theorem 7. *For every alphabet Σ with $|\Sigma| \geqslant 2$, the family of languages defined by components of unique solutions of systems of language equations with Boolean operations, linear concatenation and singleton constants is exactly the class of recursive sets over Σ. The same result holds for unrestricted concatenation and any recursive constants.*

Proof. The first claim is that if a system $X = \varphi(X)$ has a unique solution, then each of its components is recursive. This is given by the following decision procedure that determines the membership of strings in the first component:

Given $w \in \Sigma^*$, let $M = \text{substrings}(w)$.
For all finite moduli $M' \supseteq M$ closed under substrings:
  If all solutions of $X = \varphi(X)$ modulo M' coincide modulo M.
    Let $(L_1, \ldots, L_n)$ be the common part modulo M of solutions modulo M'.
    Accept if $w \in L_1$, reject if $w \notin L_1$.

The loop *for all finite moduli* considers all finite substring-closed languages in any order. There are countably many of them. Since $X = \varphi(X)$ has a unique solution, by Theorem 3, the modulus sought in the `if` statement will eventually be found, and therefore this algorithm always terminates. What it computes is the unique solution modulo M, which must be the unique solution of the system taken modulo M. This shows that the membership of w is determined correctly.

Now consider an arbitrary recursive set $L \subseteq \Sigma^*$. The task is to construct a system of language equations with a unique solution, whose first component is L. Let T be a Turing machine over Σ that halts on every input and recognizes the language L. Let $a, b \in \Sigma$ $(a \neq b)$ be a pair of distinct symbols in Σ. By Lemma 1 and Remark 1, there exist and can be effectively constructed systems of language equations $X = \varphi(X)$ and $Y = \psi(Y)$ over Σ, such that the first components of their unique solutions define, respectively, the languages $h(\text{VALC}(T))$ and $h(\text{VALC}_{\text{rej}}(T))$ of all *accepting* and *rejecting* computations of T, encoded according to Lemma 1. Construct the following system of equations:

$$Z = Z, \tag{15a}$$

$$U = Zh(\natural) \cup \bigcup_{s \in \Gamma} Uh(s), \tag{15b}$$

$$V = \overline{Z}h(\natural) \cup \bigcup_{s \in \Gamma} Vh(s), \tag{15c}$$

$$T = \overline{T} \cap \big((X_1 \cap \overline{U}) \cup (Y_1 \cap \overline{V})\big), \tag{15d}$$

$$\left.\begin{array}{l} X_1 = \varphi_1(X_1, \ldots, X_m), \\ \vdots \\ X_m = \varphi_m(X_1, \ldots, X_m) \end{array}\right\} \text{the system for the language } h(\mathrm{VALC}(T)) = \{w \cdot h(\natural) \cdot h(C_T(w)) \mid T \text{ halts on } w \text{ and accepts}\}, \tag{15e}$$

$$\left.\begin{array}{l} Y_1 = \psi_1(Y_1, \ldots, Y_n), \\ \vdots \\ Y_n = \psi_n(Y_1, \ldots, Y_n) \end{array}\right\} \text{the system for the language } h(\mathrm{VALC}_{\mathrm{rej}}(T)) = \{w \cdot h(\natural) \cdot h(C_T(w)) \mid T \text{ halts on } w \text{ and rejects}\}. \tag{15f}$$

The equations (15b) and (15c) specify $Z \cdot h(\natural) \cdot h(\Gamma^*)$ and $\overline{Z} \cdot h(\natural) \cdot h(\Gamma^*)$, respectively. The equation for T implements the following two inclusions using the method (1):

$$X_1 \subseteq Zh(\natural)h(\Gamma)^*, \tag{16a}$$

$$Y_1 \subseteq \overline{Z}h(\natural)h(\Gamma)^*. \tag{16b}$$

Since X_1 specifies the language $h(\mathrm{VALC}(T))$, (16a) means that every string accepted by T must be in Z. Similarly, Y_1 represents $h(\mathrm{VALC}_{\mathrm{rej}}(T))$, and hence (16b) ensures that every string rejected by T in finitely many steps is not in Z. Because T halts on every input, this completely defines the membership of every $w \in \Sigma$ in Z. That is, the system has the unique solution

$$\big(L(T),\ L(T) \cdot h(\natural) \cdot h(\Gamma)^*,\ \overline{L(T)} \cdot h(\natural) \cdot h(\Gamma)^*,\ \emptyset,\ L'_1,\ \ldots,\ L'_m,\ L''_1,\ \ldots,\ L''_n\big), \tag{17}$$

where the first component is the given arbitrary recursive language. □

Theorem 8. *For every alphabet Σ with $|\Sigma| \geqslant 2$, the family of languages defined by components of least (greatest) solutions of systems of language equations with Boolean operations, linear concatenation and singleton constants is exactly the family of recursively enumerable (co-recursively enumerable, respectively) sets over Σ. The same result holds for equations with unrestricted concatenation and any recursive constants.*

Proof. Let a system $X = \varphi(X)$ has a least solution. It is sufficient to prove that the first component of this solution is an r.e. language. This is given by the following semi-algorithm that tests the membership of strings in this first component:

Given $w \in \Sigma^*$, let $M = \mathrm{substrings}(w)$.
For all finite moduli $M' \supseteq M$ closed under substrings:
    If every solution $L = (L_1, \ldots, L_n)$ modulo M' has $w \in L_1$, then
        Accept.

Let $\widehat{L} = (\widehat{L}_1, \ldots, \widehat{L}_n)$ be the least solution of the system. If $w \notin \widehat{L}_1$, then, since $\widehat{L}$ is a solution modulo every M', the condition in the *if* statement will never be true, and the algorithm does not terminate. Assume $w \in \widehat{L}_1$ and consider the language $M = \mathrm{substrings}(w)$. By Lemma 8, there exists $M' \supseteq M$, such that $L \sqsubseteq L'$ (mod M) for every solution L' modulo M'. This implies, in particular, that every solution modulo M' contains w in its X_1-component. As this M' will eventually be considered by the algorithm, the condition in the *if* statement will eventually be satisfied, and w will be accepted. Therefore, the given semi-algorithm recognizes the language $\widehat{L}_1$.

Similarly, for any system $X = \varphi(X)$ with a greatest solution $\widetilde{L} = (\widetilde{L}_1, \ldots, \widetilde{L}_n)$, the language $\widetilde{L}_1$ is co-r.e., because its complement is recognized by the following semi-algorithm:

Given $w \in \Sigma^*$, let $M = \mathrm{substrings}(w)$.
For all finite moduli $M' \supseteq M$ closed under substrings:
    If every solution $L = (L_1, \ldots, L_n)$ modulo M' has $w \notin L_1$, then
        Accept.

The proof is correctness is analogous to the case of a least solution.

Now consider an arbitrary Turing machine T over an alphabet Σ, with $a, b \in \Sigma$ and $a \neq b$. By Lemma 1, there exists and can be effectively constructed a system of language equations $X = \varphi(X)$ over Σ, such that the first component of its unique

solution is the language $h(\mathrm{VALC}(T))$ of all accepting computations of T, encoded over Σ using h. Construct the following new system:

$$Z = Z, \tag{18a}$$

$$Z' = Z', \tag{18b}$$

$$U = Zh(\natural) \cup \bigcup_{s\in\Gamma} Uh(s), \tag{18c}$$

$$V = \overline{Z'}h(\natural) \cup \bigcup_{s\in\Gamma} Vh(s), \tag{18d}$$

$$T = \overline{T} \cap (X_1 \cap \overline{U} \cup X_1 \cap \overline{V}), \tag{18e}$$

$$\left.\begin{aligned} X_1 &= \varphi_1(X_1, \ldots, X_m), \\ &\vdots \\ X_m &= \varphi_n(X_1, \ldots, X_m) \end{aligned}\right\} \text{the system for the language } h(\mathrm{VALC}(T)) = \{w \cdot h(\natural) \cdot h(C_T(w)) \mid T \text{ halts on } w \text{ and accepts}\}. \tag{18f}$$

Here the equations (18c) and (18d) specify $Zh(\natural)h(\Gamma)^*$ and $\overline{Z}h(\natural)h(\Gamma)^*$, while the equation for T ensures that the following two inclusions hold:

$$X_1 \subseteq Zh(\natural)h(\Gamma)^*, \tag{19a}$$

$$X_1 \subseteq \overline{Z'}h(\natural)h(\Gamma)^*. \tag{19b}$$

Since the variable X_1 specifies the language $h(\mathrm{VALC}(T))$, the inclusion (19a) forces every string accepted by T to be in Z, and similarly (19b) ensures that no string accepted by T could be in Z'. Unlike the conditions (16) in the previous theorem, these conditions do not completely specify Z and Z': indeed, nothing is said about the strings *not* in $L(T)$ and whether should they be in Z, Z' or not. Due to (19), the set of solutions of (18) is

$$\left\{ \left(L,\ L',\ L \cdot h(\natural) \cdot h(\Gamma)^*,\ \overline{L'} \cdot h(\natural) \cdot h(\Gamma)^*,\ \emptyset,\ L_1,\ \ldots,\ L_m\right) \;\middle|\; \underbrace{L(T) \subseteq L}_{(20^*)},\ \underbrace{L' \subseteq \overline{L(T)}}_{(20^{**})} \right\}. \tag{20}$$

Among these, the solutions

$$\left(L(T),\ \emptyset,\ L(T) \cdot h(\natural) \cdot h(\Gamma)^*,\ \emptyset,\ \emptyset,\ L_1,\ \ldots,\ L_m\right), \tag{21a}$$

$$\left(\Sigma^*,\ \overline{L(T)},\ \Sigma^* \cdot h(\natural) \cdot h(\Gamma)^*,\ L(T) \cdot h(\natural) \cdot h(\Gamma)^*,\ \emptyset,\ L_1,\ \ldots,\ L_m\right) \tag{21b}$$

are the least and the greatest, respectively. Indeed, the first component $L(T)$ in (21a) cannot be reduced due to (20^*), while there is no room for reduction in the second component $\emptyset$. Similarly, (21b) is the greatest solution because Σ^* cannot be increased and due to (20^{**}). Since the Turing machine T was chosen arbitrarily, this construction represents any recursively enumerable set as the first component of the least solution, and the complement of every recursively enumerable set as the second component of the greatest solution. □

The representability results of Theorems 7 and 8 require an alphabet of at least two letters. These results can be applied to unary languages if the alphabet is formally extended to binary; in this case the second symbol will be used in the equations for the languages of computations of Turing machines, while the first component will only contain strings over the original unary alphabet. However, if $|\Sigma| = 1$ and no auxiliary symbols are allowed, then there seems to be no obvious way to reproduce the constructions (15) and (18). The exact expressive power of language equations over a one-letter alphabet is left as an open problem.

9. Conclusion

The basic tools for the analysis of language equations introduced in this paper have proved to be useful in the study of their computational properties. These new methods gave some upper bounds on the computational hardness of language equations; matching lower bounds were obtained by encoding the standard constructs of formal language theory into equations. The exact complexity of basic decision problems and complete characterizations of representable languages were thus obtained.

The research on language equations has much progressed over the last few years, while this paper was under review. Following this paper, their computational properties have been in the focus of attention. In particular, the major problem of the commutation of languages [7] was finally solved by encoding a universal computation in its greatest solution. For this result and for an overview of the new developments, the reader is referred to a recent survey by Kunc [15].

References

[1] A. Aiken, D. Kozen, M.Y. Vardi, E.L. Wimmers, The complexity of set constraints, in: Computer Science Logic, CSL 1993, Swansea, United Kingdom, September 13–17, 1993, in: Lecture Notes in Comput. Sci., vol. 832, 1994, pp. 1–17.
[2] J. Autebert, J. Berstel, L. Boasson, Context-free languages and pushdown automata, in: G. Rozenberg, A. Salomaa (Eds.), Handbook of Formal Languages, vol. 1, Springer, 1997, pp. 111–174.
[3] F. Baader, P. Narendran, Unification of concept terms in description logic, J. Symbolic Comput. 31 (3) (2001) 277–305.
[4] B.S. Baker, R.V. Book, Reversal-bounded multipushdown machines, J. Comput. System Sci. 8 (1974) 315–332.
[5] J.A. Brzozowski, E.L. Leiss, On equations for regular languages, finite automata, and sequential networks, Theoret. Comput. Sci. 10 (1980) 19–35.
[6] W. Charatonik, Set constraints in some equational theories, Inform. and Comput. 142 (1) (1998) 40–75.
[7] J.H. Conway, Regular Algebra and Finite Machines, Chapman and Hall, 1971.
[8] M. Daley, O.H. Ibarra, L. Kari, Closure and decidability properties of some language classes with respect to ciliate bio-operations, Theoret. Comput. Sci. 306 (2003) 19–38.
[9] S. Ginsburg, H.G. Rice, Two families of languages related to ALGOL, J. ACM 9 (1962) 350–371.
[10] J. Hartmanis, Context-free languages and Turing machine computations, in: Proc. Sympos. Appl. Math., vol. 19, AMS, 1967, pp. 42–51.
[11] J. Karhumäki, L. Lisovik, The equivalence problem of finite substitutions on ab^*c, with applications, Internat. J. Found. Comput. Sci. 14 (2003) 699–710.
[12] J. Karhumäki, I. Petre, Conway's problem for three-word sets, Theoret. Comput. Sci. 289 (2002) 705–725.
[13] L. Kari, S. Konstantinidis, Language equations, maximality and error-detection, J. Comput. System Sci. 70 (2005) 157–178.
[14] L. Kari, G. Thierrin, Maximal and minimal solutions to language equations, J. Comput. System Sci. 53 (3) (1996) 487–496.
[15] M. Kunc, What do we know about language equations? in: Developments in Language Theory, DLT 2007, Turku, Finland, July 3–6, 2007, in: Lecture Notes in Comput. Sci., vol. 4588, 2007, pp. 23–27.
[16] E.L. Leiss, Unrestricted complementation in language equations over a one-letter alphabet, Theoret. Comput. Sci. 132 (1994) 71–93.
[17] A. Okhotin, Conjunctive grammars, J. Autom. Lang. Comb. 6 (4) (2001) 519–535.
[18] A. Okhotin, Conjunctive grammars and systems of language equations, Program. Comput. Software 28 (5) (2002) 243–249.
[19] A. Okhotin, Boolean grammars, Inform. and Comput. 194 (1) (2004) 19–48.
[20] H. Rogers Jr., Theory of Recursive Functions and Effective Computability, McGraw–Hill, 1967.
[21] A. Salomaa, Theory of Automata, Pergamon Press, 1969.
[22] N. Yevtushenko, T. Villa, R.K. Brayton, A. Petrenko, A.L. Sangiovanni-Vincentelli, Solution of parallel language equations for logic synthesis, in: International Conference on Computer-Aided Design, ICCAD 2001, San Jose, CA, USA, November 4–8, 2001, ACM, 2001, pp. 103–110.
[23] G.-Q. Zhang, Domain mu-calculus, RAIRO Inf. Théor. Appl. 37 (2003) 337–364.