数理解析研究所講究録 1366 巻 2004 年 165-176

On systems of language equations with Boolean operations

Alexander Okhotin*

Abstract

This paper reports preliminary results on language equations with explicit operations of concatenation, union, intersection and complement. It turns out that even defining the language denoted by such a system is a nontrivial task, and four different methods of associating a single language with a system are consequently developed in this paper. A grammar representation for these systems, named *Boolean grammars*, is introduced. Many important decision problems (such as the compliance to each of the mentioned semantics) are shown to be undecidable, but still it is argued that, under some restrictions, it could nevertheless be possible to develop a computationally feasible theory of such systems.

1 Introduction

The characterization of context-free grammars with systems of language equations containing the operations of union and concatenation is well-known [1]. It is based upon the algebraic representation of formal languages as a semiring, with union and concatenation acting as addition and multiplication respectively, with zero $\emptyset$ and identity $\{\epsilon\}$; by the general results on the algebraic properties of systems of equations of this kind one can prove that such a system always has a least solution, and establish the correspondence between the least solutions of these systems and the derivability in context-free grammars.

In a recent paper [5], more general systems of language equations that allow the use of an additional intersection operation were investigated, and it was proved that such systems have basically the same properties as the classical systems of language equations, and that these systems characterize conjunctive grammars [4], an extension of context-free grammars with an intersection operation, in the same way as the classical systems of language equations characterize standard context-free grammars.

This paper reports on a preliminary investigation of the properties of systems of language equations that, in addition to the concatenation, union and intersection, allow the use of complement. The motivation for the study of these systems of equations is given by the old question of developing a sufficiently potent and at the same time computationally efficient tool for denoting formal languages (see the discussion and bibliography in [2]). The proposed explicit negation operation is, perhaps, the next most natural thing to add

*School of Computing, Queen's University, Kingston, Ontario, Canada K7L 3N6. E-mail: okhotin@cs.queensu.ca

to the formalism of context-free rules after the conjunction that came with conjunctive grammars [4]. However, the explicit negation significantly complicates the state of affairs in comparison with context-free and conjunctive grammars, because complement is not a monotonous function, and thus the familiar fixed point approach is no longer applicable. Some systems of this kind turn out to have multiple pairwise incomparable solutions and some have no solutions at all; even worse, as proved in this paper, there is no algorithm to determine the type to which a given system belongs.

In Section 2 the systems of language equations with Boolean operations are introduced, and their basic properties, such as the issues of existence and uniqueness of solutions, are investigated. In Section 3 the systems of equations are considered as a means of specifying languages; it turns out that the question of associating a language with a system is quite nontrivial, and four different semantics for them are proposed in this paper. Finally, in Section 4 an attempt is made to reformulate the systems of equations as grammars, named *Boolean grammars*. A normal form for these grammars is suggested and the questions of transforming a grammar into this normal form are discussed.

2 Systems of equations

2.1 Formulae and equations

Definition 1 (A formula). *Let Σ be a finite nonempty alphabet and let $X = (X_1, \dots X_n)$ $(n \geqslant 1)$ be a vector of language variables. A formula over Σ in variables X is defined inductively on its structure: the empty string ϵ is a formula; any symbol from Σ is a formula; any variable from X is a formula; if φ and ψ are formulae, then $(\varphi\psi)$, $(\varphi \vee \psi)$, $(\varphi \& \psi)$ and $(\neg\varphi)$ are formulae as well.*

For the convenience of notation, the parentheses will be omitted whenever possible: concatenation, conjunction and disjunction are considered associative, and the default precedence of operations is first concatenation, then negation, then conjunction and finally disjunction.

Definition 2 (Value of a formula). *Let φ be a formula over the alphabet Σ in variables $X = (X_1, \dots, X_n)$. Let $L = (L_1, \dots, L_n)$ be a vector of languages over Σ.*

The value of the formula φ on the vector of languages L, denoted as $\varphi(L)$, is defined inductively on the structure of φ:

- $\epsilon(L) = \{\epsilon\}$.
- $a(L) = \{a\}$ *for every* $a \in \Sigma$.
- $X_i(L) = L_i$ *for every* i $(1 \leqslant i \leqslant n)$.
- $(\psi\xi)(L) = \psi(L) \cdot \xi(L)$.
- $(\psi \vee \xi)(L) = \psi(L) \cup \xi(L)$.
- $(\psi \& \xi)(L) = \psi(L) \cap \xi(L)$.
- $(\neg\psi)(L) = \Sigma^* \setminus \psi(L)$.

The value of a vector of formulae $\overline{\varphi} = (\varphi_1, \ldots, \varphi_m)$ *on the vector of languages* L *is a vector of languages* $\overline{\varphi}(L) = (\varphi_1(L), \ldots, \varphi_m(L))$.

Let us define the systems of equations we are going to study.

Definition 3 (Resolved system of equations). *Let* Σ *be an alphabet. Let* $n \geqslant 1$. *Let* $X = (X_1, \ldots, X_n)$ *be a set of language variables. Let* $\varphi = (\varphi_1, \ldots, \varphi_n)$ *be vectors of formulae in variables* X *over the alphabet* Σ. *Then*

$$X_i = \varphi_i(X_1, \ldots, X_n) \quad \textit{(for all } i\textit{)} \tag{1}$$

is called a resolved system of equations over Σ *in variables* X.

Definition 4 (Solution of a resolved system). $L = (L_1, \ldots, L_n)$ *is said to be a solution of a system (1), if for every* i $(1 \leqslant i \leqslant n)$ *it holds that* $L_i = \varphi_i(L)$.

Example 1. *The following system of equations over the alphabet* $\Sigma = \{a, b\}$

$$\begin{aligned} X_1 &= \neg X_2 X_3 \& \neg X_3 X_2 \& X_4 \\ X_2 &= (a \vee b) X_2 (a \vee b) \vee a \\ X_3 &= (a \vee b) X_3 (a \vee b) \vee b \\ X_4 &= (aa \vee ab \vee ba \vee bb) X_4 \vee \epsilon \end{aligned} \tag{2}$$

has the unique solution $(\{ww \mid w \in \{a,b\}^*\}, \{xay \mid x, y \in \{a,b\}^*\}, \{xby \mid x, y \in \{a,b\}^*\}, \{u \mid u \in \{a,b\}^{2n}, n \geqslant 0\})$.

It we use the first variable as a kind of "start symbol", then the grammar given in Example 1 can be said to denote the language $\{ww \mid w \in \{a,b\}^*\}$. This abstract language is often given as an example that captures the notion of "reduplication", which is viewed as an essential property of natural languages. The language $\{ww \mid w \in \{a,b\}^*\}$ is co-context-free, but not context-free; additionally, it is known not to be representable as a finite intersection of context-free languages. Although conjunctive grammars can denote a very similar language $\{wcw \mid w \in \{a,b\}^*\}$ [4], it is not known whether $\{ww \mid w \in \{a,b\}^*\}$ is a conjunctive language (or, equivalently, whether it can be denoted using a system of language equations containing concatenation, union and intersection, but not negation).

However, as we have just seen, it can be easily denoted using systems of language equations with Boolean operations. Generally, being able to apply negation whenever it is needed is a practically useful property in itself. But, as we shall see in the following, this increase in expressive power has a substantial cost in terms in the vital theoretical properties of the model: even *defining* the language denoted by such a system is a quite nontrivial task that admits many different solutions, none of which seems perfect.

2.2 On the existence and uniqueness of solutions

Unlike the systems of language equations without negation, which always have at least one solution, the newly introduced systems with arbitrary Boolean operations are not guaranteed to have them:

Proposition 1. *A system of equations does not necessarily have solutions.*

$X_1 = \neg X_1$ is an example of a system of language equations without solutions.

Since the systems without negation and conjunction already can have multiple solutions, so can the systems with the full set of logical connectives:

Proposition 2. *A system of equations that has solutions may have one or multiple solutions.*

The system $X_1 = X_1$ has continuum of solutions (in fact, every language is a solution of this system), the system $X_1 = X_1 \& \epsilon$ has two solutions ($\emptyset$ and $\{\epsilon\}$), while the system $X_1 = aX_1b \vee \epsilon$ has the unique solution $\{a^n b^n \mid n \geqslant 0\}$.

In the case of context-free and conjunctive grammars, where the righthand sides of the corresponding systems of equations are always monotonous, every system is guaranteed to have the least solution under the partial order of componentwise inclusion. This does not hold in the nonmonotonous case:

Proposition 3. *A system of equations that has solutions does not necessarily have the least solution.*

Proof. Consider the system $\{X_1 = \neg X_2,\ X_2 = X_2\}$, which has the set of solutions $\{(\Sigma^* \setminus L,\ L) \mid L \subseteq \Sigma^*\}$, and all these solutions are pairwise incomparable. □

The mentioned properties of systems of language equations with Boolean operations are not only all nontrivial, but also turn out to be all undecidable.

Theorem 1. *The following properties of systems of equations are undecidable: existence of solution, uniqueness of solution, existence of minimal solution.*

Proof. We reduce Post correspondence problem to each of the three mentioned problems. Let $\{(u_i, v_i)\}_{i=1}^k$, where $u_i, v_i \in \Sigma^*$, be an instance of Post correspondence problem.

1. Existence of solution. Let $b_1, \ldots, b_k$ be symbols not in Σ and consider the system

$$\begin{aligned} X_1 &= \neg X_1 \& X_2 \& X_3 \\ X_2 &= b_1 X_2 u_1 \vee \ldots \vee b_k X_2 u_k \vee b_1 u_1 \vee \ldots b_k u_k \\ X_3 &= b_1 X_3 v_1 \vee \ldots \vee b_k X_3 v_k \vee b_1 v_1 \vee \ldots b_k v_k \end{aligned} \tag{3}$$

   Let L_2 and L_3 be the languages uniquely defined by the equations for X_2 and X_3. Every solution of the system (3) must be of the form (L, L_2, L_3) for some $L \subseteq \Sigma^*$.

   If Post correspondence problem has a solution, then the language $L_2 \cap L_3$ is nonempty, i.e., there exists a string $w \in L_2 \cap L_3$. If there exists a solution (L, L_2, L_3) of the system (3), then, by the first equation of the system, $w \in L$ if and only if $w \notin L$, which is a contradiction.

   If Post correspondence problem does not have solutions, then $L_2 \cap L_3 = \emptyset$ and the triple $(\emptyset, L_2, L_3)$ is the unique solution of the system (3).

2. Uniqueness of solution.

$$\begin{aligned} X_1 &= X_1 \& X_2 \& X_3 \\ X_2 &= b_1 X_2 u_1 \vee \ldots \vee b_k X_2 u_k \vee b_1 u_1 \vee \ldots b_k u_k \\ X_3 &= b_1 X_3 v_1 \vee \ldots \vee b_k X_3 v_k \vee b_1 v_1 \vee \ldots b_k v_k \end{aligned} \tag{4}$$

(L, L_2, L_3) is a solution of the system (4) if and only if $L = L \cap (L_2 \cap L_3)$, which is true if and only if $L \subseteq L_2 \cap L_3$.

If Post correspondence problem has a solution, then $L_2 \cap L_3 \neq \emptyset$ and there exist more than one language L, such that (L, L_2, L_3) is a solution of the system (4). On the other hand, if Post correspondence problem has no solutions, then the only solution of the system (4) is $(\emptyset, L_2, L_3)$.

3. Existence of a least solution. Let us augment the system (4) with two additional equations.

$$\begin{aligned} X_1 &= X_1 \& X_2 \& X_3 \\ X_2 &= b_1 X_2 u_1 \vee \ldots \vee b_k X_2 u_k \vee b_1 u_1 \vee \ldots b_k u_k \\ X_3 &= b_1 X_3 v_1 \vee \ldots \vee b_k X_3 v_k \vee b_1 v_1 \vee \ldots b_k v_k \\ X_4 &= X_4 \\ X_5 &= \neg X_1 \end{aligned} \tag{5}$$

Now every quintuple $(L, L_2, L_3, L', \Sigma^* \setminus L)$, such that $L \subseteq L_2 \cap L_3$, is a solution of the system (5).

If Post correspondence problem has a solution, then there exist multiple solutions of the system (5) and none of these solutions is the least. If Post correspondence problem has no solutions, then the set of solutions of the system consists of all quintuples $(\emptyset, L_2, L_3, L', \Sigma^*)$, where $L' \subseteq \Sigma^*$, and obviously $(\emptyset, L_2, L_3, \emptyset, \Sigma^*)$ is the least among these solutions. □

2.3 Solutions modulo some language

Let us call two languages $L_1, L_2 \subseteq \Sigma^*$ equal modulo the third language $M \subseteq \Sigma^*$ (denoted $L_1 = L_2 \pmod M$), if $L_1 \cap M = L_2 \cap M$. This relation can be extended to the vectors of languages by saying that $L' = (L'_1, \ldots, L'_n)$ equals $L'' = (L''_1, \ldots, L''_n)$ modulo M if $L'_i = L''_i \pmod M$ for all i.

Lemma 1. *Let $M \subseteq \Sigma^*$ be an arbitrary language closed under substring, i.e., such that for every string $w \in M$ all substrings of w are also in M. Let $\varphi(X_1, \ldots, X_n)$ be a formula.*

Then, if two vectors of languages, $(L'_1, \ldots, L'_n)$ and $(L''_1, \ldots, L''_n)$, are equal modulo M, then $\varphi(L'_1, \ldots, L'_n)$ and $\varphi(L''_1, \ldots, L''_n)$ are also equal modulo M.

The proof is a straightforward induction on the structure of φ.

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

Lemma 2. *Let $M' \subseteq M'' \subseteq \Sigma^*$ be two languages, each closed under substring. Let $L = (L_1, \ldots, L_n)$ be a solution of a system of equations $X = \overline{\varphi}(X)$ modulo M''. Then $L' = (L_1 \cap M', \ldots, L_n \cap M')$ is a solution of the same system modulo M'.*

Proof. Since $L' = L \pmod{M'}$, then, by Lemma 1,

$$\varphi_i(L') = \varphi_i(L) \pmod{M'} \tag{6}$$

Since L is a solution modulo M'', for every i it holds that $\varphi_i(L) = L_i \pmod{M''}$, which, by the inclusion $M' \subseteq M''$, implies that

$$\varphi_i(L) = L_i \pmod{M'} \tag{7}$$

It is left to combine (6) and (7) to obtain

$$\varphi_i(L') = \varphi_i(L) \pmod{M'}, \tag{8}$$

which, due to the arbitrary choice of i, means $\overline{\varphi}(L') = \overline{\varphi}(L) \pmod{M'}$. □

Lemma 3. *If a system of equations $X = \overline{\varphi}(X)$ over Σ has a unique solution modulo substrings(w) for any string $w \in \Sigma^*$, then it has a unique solution modulo every language closed under substring.*

3 Defining the semantics

In this section we consider systems of equations as a means of specifying languages. A semantics for systems of equations is a certain law that associates a definite vector of languages with every system of equations from a certain class of systems valid under this semantics; if some variable is then designated as the start variable, this will give a single language specified by the system.

3.1 Semantics of unique solution

The most obvious thing to do is to confine our consideration to the systems of equations that have unique solution and then associate this solution with a system: with every system of equations $X = \overline{\varphi}(X)$ over an alphabet Σ in variables $X = (X_1, \ldots, X_n)$ that has a unique solution $L = (L_1, \ldots, L_n)$, this vector L is the vector specified by the system under the semantics of unique solution.

Under this semantics, the decision problem we are interested in solving can be formulated as "given a system of equations, a variable and a string, determine, whether the component of the unique solution of the system corresponding to the given variable contains the given string". However, it turns out that this problem is quite difficult to solve. Consider two instances of Post correspondence problem, P' and P'', and the following system of equations:

$$\begin{aligned}
X_1 &= \neg X_2 \& a \\
X_2 &= \neg X_1 \& a \\
X_3 &= \neg X_3 \& a X_5 \& X_1 X_5 \\
X_4 &= \neg X_4 \& a X_6 \& X_2 X_6 \\
X_5 &= \text{language of PCP } P' \\
X_6 &= \text{language of PCP } P'' \\
&\ldots \text{(variables and equations used to simulate PCPs)}
\end{aligned} \tag{9}$$

Under the assumption that exactly one of the Post correspondence problems P' and P'' is solvable, the system (9) has a unique solution; that is $(\emptyset, \{a\}, \emptyset, \emptyset, \ldots)$ if P' is solvable while P'' is not, and $(\{a\}, \emptyset, \emptyset, \emptyset, \ldots)$ if P'' is solvable while P' is not. If both PCPs will turn to be solvable, then the system (9) will have no solutions, and if neither P' nor P'' is solvable, then the system (9) will have multiple solutions.

In any case the system (9) has two solutions modulo $\{\epsilon, a\}$, and in order to check the membership of the string a in the first component of the unique solution of the system it is necessary to check the membership of strings of unbounded length in other components of the system; if the solution is not unique, then this search will not terminate.

3.2 Semantics of unique solution in the strong sense

Let us impose an additional restriction upon the systems of equations that will make them a more sensible generative device. Instead of the requirement that the system just has a unique solution, we now require that the system has a unique solution *modulo every language closed under substring.* This makes the membership of shorter strings in the solution independent of the membership of the longer strings.

The following statement follows from Lemma 2:

Proposition 4. *Let $X = \overline{\varphi}(X)$ be a system of equations that has unique solution modulo every language closed under substring. Let M be an arbitrary finite language closed under substring, let $u \in \Sigma^*$ be a string not in M, such that all of its substrings are in M, and let $L = (L_1, \ldots, L_n)$ be the unique solution of the system modulo M.*

Then the unique solution of the system modulo $M \cup \{u\}$ can be determined by trying all 2^n vectors of languages $L' = (L'_1, \ldots, L'_n)$ ($L'_i \subseteq \{u\}$ for all i), substituting $L'' = (L_1 \cup L'_1, \ldots, L_n \cup L'_n)$ into the system and checking it for being the solution of the system modulo $M \cup \{u\}$.

By letting $M_2 = \Sigma^*$ in Lemma 2, it can be determined that in order to check the membership of a given string w in the unique solution of the system, it now suffices to check its membership in the unique solution modulo the set of substrings of w, which can be determined using the technique given in Proposition 4.

However, the problem of whether a given system complies to the semantics of unique solution in the strong sense remains undecidable, as PCP could be reduced to it by constructing the system

$$\begin{aligned} X_1 &= \neg X_1 \& X_2 \& X_3 \\ X_2 &= b_1 X_2 u_1 \vee \ldots \vee b_k X_2 u_k \vee b_1 u_1 \vee \ldots b_k u_k \\ X_3 &= b_1 X_3 v_1 \vee \ldots \vee b_k X_3 v_k \vee b_1 v_1 \vee \ldots b_k v_k \end{aligned} \tag{10}$$

which has unique solution modulo every language iff PCP has no solutions.

3.3 Semantics of naturally reachable solution

Semantics of context-free grammars is the semantics of the least fixed point, which well corresponds to the notion of derivability; semantics of conjunctive grammars [4, 5] continues with this tradition.

Things change once the nonmonotonous operation of negation is introduced: here the possible absense of the least solution of a system well corresponds to the quite predictable difficulties with developing a set of derivation rules equivalent to any denotational semantics, such as the semantics of the unique solution of the system.

However, in order to maintain the connection with conjunctive and context-free grammars, it would be highly desirable to find a way to choose, if possible, the most natural among multiple solutions of some system, which would be the least solution in the monotonous case and something else in the general case. For that purpose we slightly modify the iterative approach of Proposition 4 to define *the semantics of naturally reachable solution*, which has some traits of operational semantics:

Definition 6 (Naturally reachable solution). *Let $X = \overline{\varphi}(X)$ be a system of equations. Naturally reachable solution modulo finite languages closed under substring is defined inductively on the cardinality of the modulus.*

Basis. *Naturally reachable solution modulo $\emptyset$ is $(\emptyset, \ldots, \emptyset)$.*

Induction step. *Let M be an arbitrary finite language closed under substring, let $u \in \Sigma^*$ be a string not in M, such that all of its substrings are in M, and let $L = (L_1, \ldots, L_n)$ be naturally reachable solution of the system modulo M. Consider an arbitrary finite sequence of vectors of languages*

$$\{L^{(i)} = (L_1^{(i)}, \ldots, L_n^{(i)})\}_{i=0}^{k}, \tag{11}$$

such that (i) $L^{(0)} = L$, (ii) every next vector $L^{(i+1)}$ in the sequence is obtained from the previous vector $L^{(i)}$ by substituting some j-th component with $\varphi_j(L^{(i)}) \cap (M \cup \{u\})$, and (iii) the last element of the sequence is a solution of the system modulo $M \cup \{u\}$. If there exists at least one such sequence, and all these sequences converge to a single vector, then this vector is called naturally reachable solution of the system modulo $M \cup \{u\}$; otherwise, the whole system of equations is said to be noncompliant to this semantics.

While this semantics does not have the same theoretical justifications as the semantics of the unique solution does, the semantics of naturally reachable solution can be shown to define the least fixed point of any system which does not make use of negation, and thus ensures "backward compatibility" with conjunctive and context-free grammars.

The undecidability of compliance to this semantics is shown by the same construction (10) as in the previous case.

3.4 Semantics of convergence

In this section we propose one more method of defining the semantics of language equations with negation inspired by the fixed point semantics of the case of systems with monotonous operations. We use notion of convergence of language sequences given in [3], which says that $\{L_n\} \to L$ iff $\{2^{-\min_{x \in L_n \Delta L} |x|}\} \to 0$, where Δ means symmetric difference. This notion of convergence is easily (componentwise) extended to sequences of vectors of languages.

Let $\varphi = (\varphi_1, \ldots, \varphi_n)$ be a vector of formulae over an alphabet Σ and in variables $X = (X_1, \ldots, X_n)$. If all these formulae do not contain negation, then, by the results of [5], the system of language equations $X = \varphi(X)$ has a least solution given by

$$L = (L_1, \ldots, L_n) = \lim_{n \to \infty} \varphi^n(\emptyset, \ldots, \emptyset), \tag{12}$$

This is basically proved as follows: since, without negation, φ is monotonous, the sequence $\{\varphi^n(\emptyset,\dots,\emptyset)\}_{n=1}^{\infty}$ is a monotonous sequence, i.e.,

$$(\emptyset,\dots,\emptyset) \preccurlyeq \varphi(\emptyset,\dots,\emptyset) \preccurlyeq \varphi(\varphi(\emptyset,\dots,\emptyset)) \preccurlyeq \dots \preccurlyeq \varphi^n(\emptyset,\dots,\emptyset) \preccurlyeq \dots, \tag{13}$$

which can be proved by a straightforward induction on n, and hence has a limit L.

On the other hand, since φ is continuous, then the sequence $\{\varphi(\varphi^n(\emptyset,\dots,\emptyset))\}_{n=1}^{\infty}$ necessarily converges to $\varphi(L)$. However, as $\{\varphi(\varphi^n(\emptyset,\dots,\emptyset))\}$ and $\{\varphi^n(\emptyset,\dots,\emptyset)\}$ is in fact the same sequence, they necessarily have the same limit, and therefore $L = \varphi(L)$.

This proves that the limit (12) is a solution of the system. Using again the monotonicity of φ, one can prove that for any L', such that $\varphi(L') = L'$, it holds that $\varphi^n(\emptyset,\dots,\emptyset) \preccurlyeq L'$ for any number n (using a simple induction on n), and then, using some properties of a limit, conclude that $L \preccurlyeq L'$, which means that L is the least solution.

In the case when negation is allowed, φ may be a nonmonotonous function, and therefore this kind of proof is not seen to be possible. Already the first step, where the convergence of the sequence (12) is established, can easily fail – for instance, consider the system $X = \neg X$, for which this kind of sequence assumes the form

$$\emptyset, \Sigma^*, \emptyset, \Sigma^*, \dots, \emptyset, \Sigma^*, \dots, \tag{14}$$

and therefore diverges.

However, φ can be proved to remain a continuous function regardless of the use of negation. Therefore, *if* for a certain system $X = \varphi(X)$ the sequence (12) happens to converge, then it converges to a solution of the system. This fact can be used to define the following semantics:

Definition 7. *A system of language equations $X = \overline{\varphi}(X)$ is said to be compliant to the semantics of convergence if the sequence $\{\varphi^n(\emptyset,\dots,\emptyset)\}$ has a limit. This limit is then taken as the semantical value of the system under the semantics of convergence.*

This definition is accompanied with a customary undecidability result:

Theorem 2. *It is undecidable whether a given system complies to the semantics of convergence.*

Proof. As before, we reduce Post correspondence problem to the question of whether the system of language equations (10). Denote the n-th term of the sequence $\{\varphi^n(\emptyset,\dots,\emptyset)\}$ as $L^{(n)} = (L_1^{(n)}, L_2^{(n)}, L_3^{(n)})$. The second and the third components of the terms of this sequence are computed by a monotonous function, and therefore we know that the sequences $\{L_2^n\}_{n=0}^{\infty}$ and $\{L_3^n\}_{n=0}^{\infty}$ are monotonous and converge to some (context-free) languages L_2 and L_3 respectively.

If Post correspondence problem does not have a solution, then $L_2 \cap L_3 = \emptyset$ and therefore $L_2^{(n)} \cap L_3^{(n)} = \emptyset$ for all n. Accordingly, the sequence $\{\varphi^n(\emptyset,\dots,\emptyset)\}$ peacefully converges to $(\emptyset, L_2, L_3)$.

However, if Post correspondence problem does have a solution, then $L_2 \cap L_3 \neq \emptyset$ and therefore there exists a string $w \in \Sigma^*$ and a number $n_0 \geqslant 1$, such that $w \notin L_2^{(n_0-1)} \cap L_3^{(n_0-1)}$, but $w \in L_2^{(n)} \cap L_3^{(n)}$ for all $n \geqslant n_0$. As a result,

$$w \notin L_1^{(n_0)},\ w \in L_1^{(n_0+1)}, \dots,\ w \notin L_1^{(n_0+2k)},\ w \in L_1^{(n_0+2k+1)}, \dots \tag{15}$$

and consequently the sequence $\{\varphi^n(\emptyset,\dots,\emptyset)\}_{n=1}^{\infty}$ diverges. □

4 Grammar representation

Now let us represent the systems of equations in the terms of grammars; two formalisms, *negative* and *Boolean grammars*, are being proposed.

Definition 8. *A negative grammar is a quadruple $G = (\Sigma, N, P, S)$, where*

- *Σ and N are disjoint finite nonempty sets of terminal and nonterminal symbols respectively.*
- *$P \subseteq N \times (\Sigma \cup N)^* \times \mathbb{B}$ is the set of rules. Formally, each rule is a triple (A, α, σ), in which A is a nonterminal, the string α is the body of the rule, while σ is a Boolean variable that determines whether the rule is positive ($\sigma = 1$) or negative ($\sigma = 0$). The rules $(A, \alpha, 1)$ and $(A, \alpha, 0)$ are notated as*

$$A \to \alpha \quad \textit{and} \tag{16a}$$
$$A \to \neg\alpha \tag{16b}$$

  respectively.
- *$S \in N$ is the start symbol of the grammar.*

The right side of every rule is a formula, and a grammar is interpreted as a system of equations over Σ in variables N, having form

$$A = \bigvee_{A \to \varphi \in P} \varphi \quad (\textit{for all } A \in N) \tag{17}$$

The language generated by a grammar is then defined using one of the semantics described in Section 3.

Every context-free grammar is a negative grammar, in which all rules are of the form (16a). Since de Morgan laws are immediately applicable to the semantics of unique solution and the semantics of unique solution in the strong sense, for these semantics there is no loss of generality in restricting the systems to be of the form (17) for some negative grammar G. For the other two semantics the question of generality of negative grammars ought to be studied.

If we combine the formalism of conjunctive grammars with that of negative grammars, we obtain a class of *Boolean grammars*:

Definition 9. *A Boolean grammar is a quadruple $G = (\Sigma, N, P, S)$, where*

- *Σ and N are disjoint finite nonempty sets of terminal and nonterminal symbols respectively.*
- *P is the set of rules, each of the form*

$$A \to \alpha_1 \& \ldots \& \alpha_m \& \neg\beta_1 \& \ldots \& \neg\beta_n \quad (m + n \geqslant 1,\ \alpha_i, \beta_i \in (\Sigma \cup N)^*) \tag{18}$$

  Objects of the form $A \to \alpha_i$ and $A \to \neg\beta_j$ are called conjuncts, positive and negative respectively.

- $S \in N$ *is the start symbol of the grammar.*

Boolean grammars are interpreted as systems of equations in the same way as negative grammars are (17).

Every conjunctive grammar is a Boolean grammar, in which every rule (18) contains only positive conjuncts, i.e., $m \geqslant 1$ and $n = 0$. Every negative grammar is a Boolean grammar, in which every rule (18) contains a single conjunct, either positive ($m = 1$ and $n = 0$), or negative ($m = 0$, $n = 1$). Every context-free grammar in a Boolean grammar, in which every rule (18) contains a single positive conjunct ($m = 1$, $n = 0$).

Consider the system of equations from Example 1, and let us use its general idea to produce a Boolean grammar that will generate the language $\{ww \mid w \in \{a,b\}^*\}$ under all the semantics defined in this paper:

Example 2. *Let* $G = (\{a,b\}, \{S,A,B,C,X\}, P, S)$ *be a Boolean grammar, where* P *consists of the following rules:*

$$\begin{array}{ll} S \to \neg AB \& \neg BA \& C & \\ A \to XAX & C \to XXC \\ A \to a & C \to \epsilon \\ B \to XBX & X \to a \\ B \to b & X \to b \end{array} \tag{19}$$

Now let us generalize context-free Chomsky normal form for the case of Boolean grammars:

Definition 10. *A Boolean grammar* $G = (\Sigma, N, P, S)$ *is said to be in the binary normal form if every rule in* P *is of the form*

$$A \to B_1C_1 \& \ldots \& B_mC_m \& \neg D_1E_1 \& \ldots \& \neg D_nE_n \& \neg\epsilon \quad (m+n \geqslant 0) \tag{20a}$$

$$A \to a \tag{20b}$$

$$S \to \epsilon \quad \text{(only if } S \text{ does not appear in right parts of rules)} \tag{20c}$$

It should not be hard to prove that every system in the binary normal form complies to all the semantics defined in this paper and defines the same language under all given types of semantics. The membership of strings in that language can be checked by a variation of Cocke–Kasami–Younger algorithm that further generalizes the extended version of this algorithm given in [4].

However, this positive result has an immediate negative implication: there cannot exist an algorithm to transform a given Boolean grammar to binary normal form, because as every grammar in this form is always semantically correct, this would require to test the semantical correctness of the input grammar, which is known to be an undecidable problem for all types of semantics considered in this paper.

Is there a way to overcome all these difficulties and to turn Boolean grammars into a practical device for defining formal languages? The positive result obtained by the author is that for the semantics of unique solution in the strong sense and for the semantics of naturally reachable solution there exists a *noneffective transformation procedure* (i.e., a procedure that involves solving undecidable problems in course of the transformation)

to convert any given Boolean grammar that is compliant to one of the two mentioned semantics to binary normal form. This procedure is basically a generalization of the similar procedure for conjunctive grammars [4], which in turn extends the case of context-free grammars, and consists of three stages:

- Removal of *positive epsilon conjuncts* and introduction of *negative epsilon conjunct*, i.e., eliminating rules that contain conjuncts $A \to \epsilon$ and adding a conjunct $A \to \neg\epsilon$ to every rule. Once it is known which components of the solution defined by the system under the given semantics contain the empty string, this step can be effectively performed using a variation of the corresponding context-free technique.
- Removal of both *positive and negative unit conjuncts*, i.e., getting rid of rules that contain conjuncts $A \to B$ or $A \to \neg B$. This step involves solving undecidable problems.
- Shortening of long conjuncts and bringing the grammar to the final conformity to (20). Once there are neither positive epsilon conjuncts nor unit conjuncts in the rules of the grammar, this step is straightforward.

The existence of such a noneffective procedure implies that systems in the binary normal form define the same class of languages as the whole family of systems compliant to the two semantics mentioned above. This also means that by imposing some further restrictions on systems of language equations one can obtain a sufficiently large subclass, for which the transformation to the binary normal form is effective and computationally feasible.

This leaves hope that, despite the omnipresent undecidability results, the new generative device might still have practical value. However, only a rigorous investigation of their properties could determine whether this conjecture is true.

References

[1] J. Autebert, J. Berstel and L. Boasson, "Context-Free Languages and Pushdown Automata", *Handbook of Formal Languages*, Vol. 1, 111–174, Springer-Verlag, Berlin, 1997.

[2] J. Dassow, G. Păun, A. Salomaa, "Grammars with Controlled Derivations", in *Handbook of Formal Languages*, Vol. 2, 101–154, Springer-Verlag, Berlin, 1997.

[3] A. Ehrenfeucht, H. J. Hoogeboom, G. Rozenberg, N. van Vugt, "Sequences of languages in forbidding–enforcing families", *Soft Computing*, 5:2 (2001), 121–125.

[4] A. Okhotin, "Conjunctive grammars", *Journal of Automata, Languages and Combinatorics*, 6:4 (2001), 519–535.

[5] A. Okhotin, "Conjunctive grammars and systems of language equations", in Russian, *Programmirovanie* 28:5 (2002).

- English translation: *Programming and Computer Software*, 28:5 (2002), 243–249.