Journal of Computer Languages 51 (2019) 48–56

Contents lists available at ScienceDirect

Journal of Computer Languages

journal homepage: www.editorialmanager.com/cola/default.aspx

On solving cycle-free context-free grammar equivalence problem using numerical analysis☆,☆☆

José João Almeida*,[a], Eliana Grande[b], Georgi Smirnov[a]

[a] Universidade do Minho, Campus de Gualtar, 4710-057 Braga, Portugal
[b] Federal Institute Goiano, Rua 88, nº310, Setor Sul, Goiania, 74.085-010, Brazil

HIGHLIGHTS

- Context free grammars comparison from the mathematical analysis point of view.
- Algorithms to decide if two context free grammars are equivalent or not.
- Definition of a grammar to numeric domain transform.
- Mathematical elegant method with classical flavor.

ARTICLE INFO

Keywords:
Formal languages
Context-free grammars
Automatic assessment

ABSTRACT

In this paper we consider the problem of cycle-free context-free grammars equivalence. To every context-free grammar there corresponds a system of formal equations. Formally applying the iteration method to this system we obtain the grammar axiom in the form of a formal power series composed of the words generated by the grammar "multiplied" by the respective ambiguities.

We define a transform that attributes a matrix meaning to the system of formal equations and to formal power series: terminal symbols are substituted by matrices and formal sum and product are substituted by the matrix ones. In order to effectively compute the sum of a matrix series we numerically solve the system of matrix equations. We prove distinguishability theorems showing that if two formal power series generated by cycle-free context-free grammars are different, then there exists a matrix substitution such that the sums of the respective matrix series are different. Based on this result, we suggest a procedure that can resolve the problem of equivalence of cycle-free context-free grammars in many practical cases.

The results obtained in this paper form a theoretical basis for algorithms oriented to automatic assessment of students' answers in computer science. We present the respective algorithms. Then we compare our approach with a simple heuristic method based on CYK algorithm and discuss the limitations of our method.

1. Introduction

The capability of writing correct grammars is an essential task in computer science (used, for example, in the creation of programming languages, compilers, etc.). Assessment of students' answers in computer science is a very hard and time-consuming activity. Computer-assisted assessment is a natural way to reduce the time spent by the teachers in their assessment task (see, e.g., [1,2]). This paper deals with assessment in the theory of cycle-free context-free grammars. Its main objective is to create a theoretical basis for algorithms allowing one to decide if two cycle-free context-free grammars are equivalent or not. It is well known that the equivalence of two context-free grammars is an undecidable problem [3]. The problem of context-free grammar equivalence was an object of intensive studies [4,5]. For example, it was solved when the equivalence is understood in structural sense [6], and some practical algorithms for grammars equivalence checking were developed (see [7,8], and the references therein).

We shall consider context-free grammars G. We denote by V_N the

☆ On a 2012, i5 Linux machine with 4 Gbytes RAM.
☆☆ This article was originally submitted to the Journal of Visual Languages and Computing, however during the process of review the journal underwent a name change and is now titled Journal of Computer Languages.

* Corresponding author.
E-mail addresses: jj@di.uminho.pt (J.J. Almeida), eliana.tiba@ifgoiano.edu.br (E. Grande), smirnov@math.uminho.pt (G. Smirnov).

https://doi.org/10.1016/j.cola.2019.02.005
Received 11 October 2018; Received in revised form 11 January 2019; Accepted 19 February 2019
Available online 09 April 2019
1045-926X/

∀n ∃K and matrices $A, B \in \mathbb{Q}^{k\times k}$: $\{A,B\}^k$ has size 2^k

nonterminal alphabet, by V_T the terminal alphabet, and by S the axiom. By $W(V)$ we shall denote the set of words over the alphabet V, and by $\mathbb{Z}_+$ the set of nonnegative integers. A map $\phi: W(V_T) \to \mathbb{Z}_+$ defines a formal power series (see [3,9])

$$S = \sum_{P \in W(V_T)} \phi(P) P. \tag{1}$$

It is well known [3,9] that to any context-free grammar there corresponds a system of nonlinear equations that allows one to obtain the respective formal power series via successive iterations. The terms of the series are the words of the respective language. The function ϕ is the ambiguity function, i.e., $\phi(P)$ is the number of distinct leftmost derivations of the word P. We shall use the notation ϕ_G to indicate that the ambiguity function corresponds to the grammar G. A grammar whose corresponding ambiguity function takes only values 0 or 1 is called unambiguous (otherwise the grammar is said to be ambiguous). The solution of an exercise submitted by a computer science student is supposed to be an unambiguous grammar. So it is very important from the point of view of automatic assessment to distinguish between an ambiguous and an unambiguous grammars. For this reason we understand the equivalence of two grammars in the following sense. We say that two grammars are equivalent if they generate the same language with the same ambiguity function.

In this paper, the attention is paid to cycle-free grammars. Recall that a context-free grammar is said to be cycle-free if for each $A \in V_N$, $A \overset{+}{\Longrightarrow} A$ is impossible.

In order to present the methodology adopted in this paper, let us consider the following simple example. Let the language L be $\{c, ab, acb, accb, acccb, ...\}$. It can be generated by the grammar $S \to aAb|c$; $A \to cA|\epsilon$. The following system of formal equations corresponds to this grammar

$$S = aAb + c, \tag{2}$$

$$A = cA + \epsilon. \tag{3}$$

Formally applying the iteration method to this system we obtain the formal power series

$$S = c + ab + acb + accb + acccb + ... \tag{4}$$

corresponding to this grammar [9].

Below we define a transform that attributes a matrix meaning to formal power series (4). Namely,

- any terminal letter a, b, c, is substituted by an $(\mathfrak{N} \times \mathfrak{N})$-matrix μ_a, μ_b, μ_c, where $\mathfrak{N}$ is a positive integer;
- the nonterminal symbols S and A are substituted by $(\mathfrak{N} \times \mathfrak{N})$-matrix variables $S(\mu_a, \mu_b, \mu_c)$ and $A(\mu_a, \mu_b, \mu_c)$;
- the formal sum and product are substituted by the matrix ones;
- the empty word ϵ is substituted by the $(\mathfrak{N} \times \mathfrak{N})$ identity matrix I.

Then a matrix $S(\mu_a, \mu_b, \mu_c)$ calculated as the sum of the matrix series

$$S(\mu_a, \mu_b, \mu_c) = \mu_c + \mu_a\mu_b + \mu_a\mu_c\mu_b + \mu_a\mu_c\mu_c\mu_b + \mu_a\mu_c\mu_c\mu_c\mu_b + ...,$$

corresponds to S. In order to effectively compute this sum we numerically solve the system of matrix equations

$$\begin{aligned} S(\mu_a, \mu_b, \mu_c) &= \mu_a A(\mu_a, \mu_b, \mu_c)\mu_b + \mu_c \\ A(\mu_a, \mu_b, \mu_c) &= \mu_c A(\mu_a, \mu_b, \mu_c) + I \end{aligned}$$

obtained applying the transform to formal system (2) and (3). The system is solved using the method of iterations starting at zero initial point. This gives us the matrix $S(\mu_a, \mu_b, \mu_c)$.

For example, substituting randomly generated matrices

$$\mu_a = \begin{pmatrix} 0.12715 & 0.10244 \\ 0.15314 & 0.21512 \end{pmatrix}, \quad \mu_b = \begin{pmatrix} 0.12978 & 0.19281 \\ 0.08857 & 0.06573 \end{pmatrix},$$

and

$$\mu_c = \begin{pmatrix} 0.05885 & 0.17198 \\ 0.12595 & 0.12162 \end{pmatrix},$$

and numerically solving system (2) and (3) we get

$$S(\mu_a, \mu_b, \mu_c) = \begin{pmatrix} 0.09195 & 0.21148 \\ 0.17718 & 0.17901 \end{pmatrix}.$$

In the case of the grammar $S \to bAa \,|\, c$; $A \to cA \,|\, \epsilon$, the same random matrix substitution gives

$$S(\mu_a, \mu_b, \mu_c) = \begin{pmatrix} 0.11946 & 0.24404 \\ 0.15422 & 0.15318 \end{pmatrix}.$$

This implies that the two grammars specified so far either generate different languages, or the same language but with a different ambiguity function. Note that it is impossible to establish the non-equivalence of these two grammars using scalar ((1×1)-matrix) substitutions, because the multiplication of scalars is commutative. Instead, $(\mathfrak{N} \times \mathfrak{N})$-matrix substitutions with $\mathfrak{N} > 1$ allow us to detect the difference in the order of symbols in the words.

In the same way, in general case of a grammar with the terminal alphabet $V_T = \{a_1, ..., a_n\}$, one can calculate the matrix $S(\mu_{a_1}, ..., \mu_{a_n})$.

The main result proved in this paper (Distinguishability Theorem I) shows that if two cycle-free context-free grammars G_1 and G_2 are not equivalent, then there exist a positive integer $\mathfrak{N}$ and an $(\mathfrak{N} \times \mathfrak{N})$-matrix substitution $\mu = (\mu_{a_1}, ..., \mu_{a_n})$ such that

$$S_1(\mu) = S_1(\mu_{a_1}, ..., \mu_{a_n}) \neq S_2(\mu_{a_1}, ..., \mu_{a_n}) = S_2(\mu).$$

The distinguishability theorem allows us to construct tools for comparison of cycle-free context-free grammars. Namely, we calculate $S_1(\mu)$ and $S_2(\mu)$ for a sufficiently large number of matrix substitutions and if for all substitutions the equality $S_1(\mu) = S_2(\mu)$ is satisfied, then we conclude that the grammars are equivalent. Indeed, if two grammars are equivalent in the sense considered here, then the corresponding formal power series S_1 and S_2 coincide, and we have $S_1(\mu) = S_2(\mu)$ for any matrix substitution. Thus, if there exists a matrix substitution such that $S_1(\mu) \neq S_2(\mu)$, then the grammars are not equivalent. Next, if two formal power series can be distinguished using $(\mathfrak{N} \times \mathfrak{N})$-matrix substitutions, then the probability to get $S_1(\mu) = S_2(\mu)$ using random $(\mathfrak{N} \times \mathfrak{N})$-matrix substitution is zero, since the Lebesgue measure of the zero set of a real analytic function is zero (see [10, Lemma 5.22] or [11]). Of course, since the analytic function is implemented using floating-point arithmetic, the Lebesque measure of the zero set might no longer be zero. In fact, it is zero if and only if no point from the zero set can be represented by a floating-point number. Otherwise the probability is different from zero but, usually is very low. Let us consider a trivial example of two nonequivalent grammars $S \to a$ and $S \to b$. It is possible to imagine that the randomly generated $(\mathfrak{N} \times \mathfrak{N})$-matrix substitution is $a = \mu = b$, and this says us that the grammars are equivalent. Such an event is possible but the respective probability is about $2^{-64 \times \mathfrak{N}^2}$. We would like to note that according to our experience one (2×2) or (3×3) random matrix substitution is enough to distinguish between two nonequivalent cycle-free context-free grammars.

Let us outline the main steps of the grammar comparison method presented in this work.

1. Transformation of grammars into systems of formal equations.
2. Substitution of terminal symbols in the systems of formal equations by randomly generated matrices.
3. Numerical solution of the systems of matrix equations.
4. Comparison of the matrix values of the axioms.
5. If the matrix values of axioms are different we conclude that the grammars are not equivalent, i.e., that they generate different languages or the same language but have different ambiguities. If the matrix values of axioms are equal we conclude that the grammars are equivalent, i.e., generate the same language and have the same

ambiguities.

This paper continues the research started in [12] where we considered distinguishability based on (2 × 2)-matrices. Due to negligence, the distinguishability theorem was not clearly formulated. We remedy this situation below (Theorem 2).

Note also that the idea to use matrices to study formal power series is not new (cf. [13]), but the approach presented in [13] is completely different from the one discussed here.

The paper is organized as follows. In Section 2 we prove the distinguishability theorems. Section 3 contains convergence analysis of the iteration method for nonlinear matrix equations. The respective algorithms are described in Section 4. A comparison with a simple heuristic method is presented in Section 5. The limitations of the method are discussed in Section 6. Section 7 contains a brief conclusion.

2. Distinguishability theorems

In the sequel, we denote the length of a word $P \in W(V_N \cup V_T)$ by $|P|$. The cardinality of an alphabet V is denoted by $|V|$. According to [14,Theorem 12.2.1], in a cycle-free context-free grammar, the length of a derivation is linear in the length of the generated word. Therefore the ambiguity function of each cycle-free context-free grammar grows at most exponentially, i.e.,

$$0 \le \phi_G(P) \le Cq^{|P|}, \tag{5}$$

where C and q are positive constants (see [15] for a detailed classification of ambiguity functions).

Let μ be a map from V_T to the set $\mathbb{R}^{\mathfrak{N}\times\mathfrak{N}}$ of $(\mathfrak{N} \times \mathfrak{N})$-matrices with real elements. By $P(\mu)$ we will denote the matrix obtained substituting the letters a_i in the word P by the matrices μ_i and calculating the respective matrix product, i.e., if $P = a_{i_1}...a_{i_n}$, then $P(\mu) = \mu_{i_1}...\mu_{i_n}$. The euclidean norm of a vector $x \in \mathbb{R}^{\mathfrak{N}}$ is denoted by $\|x\|$. The norm of a matrix $\mathfrak{M}$ is denoted by $\|\mathfrak{M}\| = \max_{x\in\mathbb{R}^{\mathfrak{N}},x\neq 0}\|\mathfrak{M}x\|/\|x\|$. If the series

$$S(\mu) = \sum_{P\in W(V_T)} \phi_G(P)P(\mu) \tag{6}$$

converges, its sum is an $(\mathfrak{N} \times \mathfrak{N})$-matrix. Observe that the matrix series (6) corresponding to a cycle-free context-free grammar converges, if $M = \max_{1\le i\le|V_T|}\|\mu_i\|$ satisfies

$$M < 1/(q|V_T|). \tag{7}$$

Indeed, since the number of words of length N over the alphabet V_T does not exceed $|V_T|^N$, we have

$$\|S(\mu)\| \le \sum_{P\in W(V_T)} \phi_G(P)\|P(\mu)\| \le \sum_{N\ge 0} Cq^N|V_T|^N M^N = \frac{C}{1-q|V_T|}, \tag{8}$$

whenever M satisfies (7). We shall show that matrix substitutions permit to distinguish between two nonequivalent cycle-free context-free grammars. The following *distinguishability* theorems form a theoretical basis for assessment algorithms.

2.1. General distinguishability theorems

Theorem 1 (*distinguishability I*). *Let S_1 and S_2 be two different formal power series corresponding to two cycle-free context-free grammars. Then there exist a positive integer $\mathfrak{N}$ and a matrix substitution μ: $V_T \to \mathbb{R}^{\mathfrak{N}\times\mathfrak{N}}$ such that $S_1(\mu) \neq S_2(\mu)$.*

To prove the theorem we need an auxiliary lemma.

Lemma 1. *Let U and V be two different finite formal series composed of words of length N. Then there exist a positive integer $\mathfrak{N}$ and a matrix substitution μ: $V_T \to \mathbb{R}^{\mathfrak{N}\times\mathfrak{N}}$ such that $U(\mu) \neq V(\mu)$.*

Proof. Let $a_{j_1}...a_{j_k}a_{i_1}...a_{i_l}$, where $1 \le l \le N$ and $k + l = N$, be a word appearing in U or V. We say that $a_{i_1}...a_{i_l}$ is a suffix. (Note that the empty string is not considered a suffix.) Denote the set of suffixes of the words from U and V by $\mathcal{S}$. By $\mathfrak{N}$ we denote the cardinality of $\mathcal{S}$ plus one. Let us consider the set of unit orthogonal vectors $\{e_0\} \cup \{e_{i_1...i_l} | a_{i_1}...a_{i_l} \in \mathcal{S}\}$ in an $\mathfrak{N}$-dimensional space. We define the linear operators μ_i, $i = 1,...,|V_T|$, by:

$$\mu_i e_0 = \begin{cases} e_i, & a_i \in \mathcal{S} \\ 0, & \text{otherwise} \end{cases}$$

$$\mu_i e_{i_1...i_l} = \begin{cases} e_{ii_1...i_l}, & a_i a_{i_1}...a_{i_l} \in \mathcal{S} \\ 0, & \text{otherwise} \end{cases}$$

Let $a_{i_1}...a_{i_N}$ be a word from U or V. The corresponding linear operator has the form $\mu_{i_1}...\mu_{i_N} \in \mathbb{R}^{\mathfrak{N}\times\mathfrak{N}}$. Obviously we have $\mu_{i_1}...\mu_{i_N}e_0 = e_{i_1...i_N}$. Hence we get

$$U(\mu)e_0 = \sum_{a_{i_1}...a_{i_N}\in U} e_{i_1...i_N} \neq \sum_{a_{i_1}...a_{i_N}\in V} e_{i_1...i_N} = V(\mu)e_0,$$

Indeed, the sets of orthogonal vectors in the two sums are different. Therefore $U(\mu) \neq V(\mu)$. □

Consider a simple example that may help the reader to better understand the construction used in the proof of Lemma 1. Let $U = a_1a_2$ and $V = a_2a_1$. In this case $\mathcal{S} = \{a_1, a_2, a_1a_2, a_2a_1\}$ and $\mathfrak{N} = 5$. Set

$$e_0 = (1, 0, 0, 0, 0),$$
$$e_1 = (0, 1, 0, 0, 0),$$
$$e_2 = (0, 0, 1, 0, 0),$$
$$e_{12} = (0, 0, 0, 1, 0),$$
$$e_{21} = (0, 0, 0, 0, 1),$$

then

$$\mu_1e_0 = e_1, \quad \mu_1e_1 = 0, \quad \mu_1e_2 = e_{12}, \quad \mu_1e_{12} = 0, \quad \mu_1e_{21} = 0,$$
$$\mu_2e_0 = e_2, \quad \mu_2e_1 = e_{21}, \quad \mu_2e_2 = 0, \quad \mu_2e_{12} = 0, \quad \mu_2e_{21} = 0,$$

and we have

$$\mu_1\mu_2e_0 = \mu_1e_2 = e_{12} \neq e_{21} = \mu_2e_1 = \mu_2\mu_1e_0.$$

Hence $\mu_1\mu_2 \neq \mu_2\mu_1$. Obviously, in this example it is possible to distinguish between U and V with the help of (2 × 2)-matrices. Lemma 1 establishes only the possibility of distinguishability without calculating the minimal needed dimension $\mathfrak{N}$.

Proof of Theorem 1. Let N be the least integer such that the coefficients at some word of length N in the two series are different. Then we can write $S_1 = S_0 + U + R_1$ and $S_2 = S_0 + V + R_2$ where S_0 is the part of coinciding words of length less than or equal to N, U and V are different parts composed of words with length equal to N, and R_1 and R_2 contain terms with words of length greater than N. By Lemma 1 there exist a positive integer $\mathfrak{N}$ and a matrix substitution μ: $V_T \to \mathbb{R}^{\mathfrak{N}\times\mathfrak{N}}$ such that $U(\mu) \neq V(\mu)$. Let $t > 0$. Then we have

$$\Delta(t) = S_1(t\mu) - S_2(t\mu) = t^N(U(\mu) - V(\mu)) + (R_1(t\mu) - R_2(t\mu)). \tag{9}$$

The norms of the matrices μ_i, $i = 1,...,|V_T|$, constructed in Lemma 1, do not exceed some $\sigma > 0$. Since the grammars are cycle-free, we obtain from (5)

$$\|R_1(t\mu)\| \le C_1((q_1\sigma t)^{N+1} + (q_1\sigma t)^{N+2} + ...)$$

and

$$\|R_2(t\mu)\| \le C_2((q_2\sigma t)^{N+1} + (q_2\sigma t)^{N+2} + ...).$$

Let $C = \max\{C_1, C_2\}$, $q = \max\{q_1, q_2\}$ and $t < 1/(q\sigma)$. Then the series $R_1(t\mu)$ and $R_2(t\mu)$ are convergent and we obtain

$$\|R_1(t\mu) - R_2(t\mu)\| \le 2C((q\sigma t)^{N+1} + (q\sigma t)^{N+2} + ...) = 2C(q\sigma)^{N+1}\frac{t^{N+1}}{1-q\sigma t}.$$

From this and (9) we see that $\Delta(t) \neq 0$ whenever $t > 0$ is sufficiently small. □

In many situations it suffices to consider matrix substitutions $\mu: V_T \to \mathbb{R}^{2\times 2}$. We associate with the symbols $a_i \in V_T$, $i = 1,...,|V_T|$, pairs of variables x_i and y_i. Let J and N be positive integers and let

$$U = \sum_{j=1}^{J} P_j \quad \text{and} \quad V = \sum_{j=1}^{J} Q_j$$

be two formal sums of words of length N, given by the functions $f, g: \{1, 2,...,J\} \times \{1, 2,...,N\} \to \{1, 2, ...,|V_T|\}$ as follows:

$$P_j = a_{f(j,N)} a_{f(j,N-1)} ... a_{f(j,1)},$$
$$Q_j = a_{g(j,N)} a_{g(j,N-1)} ... a_{g(j,1)},$$

for $j = 1, 2, ...,J$. To U and V we assign two sets of monomials

$$\mathcal{M}_U = \left\{ \left(\prod_{l=k+1}^{N} x_{f(j,l)} \right) y_{f(j,k)} \,|\, 1 \le j \le J,\ 1 \le k \le N \right\},$$
$$\mathcal{M}_V = \left\{ \left(\prod_{l=k+1}^{N} x_{g(j,l)} \right) y_{g(j,k)} \,|\, 1 \le j \le J,\ 1 \le k \le N \right\},$$

and say that U and V satisfy condition $(\mathcal{M})$ if $\mathcal{M}_U \neq \mathcal{M}_V$.

Lemma 2. *Assume that U and V satisfy condition* $(\mathcal{M})$, *then there exists a matrix substitution* $\mu: V_T \to \mathbb{R}^{2\times 2}$ *such that* $U(\mu) \neq V(\mu)$.

Proof. For $1 \le i \le |V_T|$, let T_i be the 2×2 upper-triangular matrix

$$T_i = \begin{pmatrix} x_i & y_i \\ 0 & 1 \end{pmatrix}$$

depending on real variables x_i and y_i. For each $j \in \{1, 2, ...,J\}$, easy induction on n shows that

$$T_{f(j,n)} T_{f(j,n-1)} ... T_{f(j,1)} = \begin{pmatrix} \prod_{k=1}^{n} x_{f(j,k)} & \sum_{k=1}^{n} \left(\prod_{l=k+1}^{n} x_{f(j,l)} \right) y_{f(j,k)} \\ 0 & 1 \end{pmatrix}, \tag{10}$$

$$T_{g(j,n)} T_{g(j,n-1)} ... T_{g(j,1)} = \begin{pmatrix} \prod_{k=1}^{n} x_{g(j,k)} & \sum_{k=1}^{n} \left(\prod_{l=k+1}^{n} x_{g(j,l)} \right) y_{g(j,k)} \\ 0 & 1 \end{pmatrix}, \tag{11}$$

for $n \in \{1, 2, ...,N\}$. Define a matrix substitution ν by setting $\nu_{a_i} = T_i$ for $i = 1, 2,...,|V_T|$. Then from (10) and (11) we have

$$\begin{aligned} U(\nu) &= \sum_{j=1}^{J} \nu_{a_{f(j,N)}} \nu_{a_{f(j,N-1)}} ... \nu_{a_{f(j,1)}} \\ &= \sum_{j=1}^{J} T_{f(j,N)} T_{f(j,N-1)} ... T_{f(j,1)} \\ &= \begin{pmatrix} \sum_{j=1}^{J} \prod_{k=1}^{N} x_{f(j,k)} & \sum_{j=1}^{J} \sum_{k=1}^{N} \left(\prod_{l=k+1}^{N} x_{f(j,l)} \right) y_{f(j,k)} \\ 0 & J \end{pmatrix}, \\ V(\nu) &= \sum_{j=1}^{J} \nu_{a_{g(j,N)}} \nu_{a_{g(j,N-1)}} ... \nu_{a_{g(j,1)}} \\ &= \sum_{j=1}^{J} T_{g(j,N)} T_{g(j,N-1)} ... T_{g(j,1)} \\ &= \begin{pmatrix} \sum_{j=1}^{J} \prod_{k=1}^{N} x_{g(j,k)} & \sum_{j=1}^{J} \sum_{k=1}^{N} \left(\prod_{l=k+1}^{N} x_{g(j,l)} \right) y_{g(j,k)} \\ 0 & J \end{pmatrix}. \end{aligned}$$

Clearly, $\mathcal{M}_U$ is the set of all monomials that appear in the polynomial $U(\nu)_{1,2}$ and $\mathcal{M}_V$ is the set of all monomials that appear in the polynomial $V(\nu)_{1,2}$. By condition $(\mathcal{M})$ we have

$$U(\nu)_{1,2}\left(x_1,...,x_{|V_T|}, y_1,...,y_{|V_T|}\right) \neq V(\nu)_{1,2}\left(x_1,...,x_{|V_T|}, y_1,...,y_{|V_T|}\right)$$

for some $(x_1,...,x_{|V_T|}, y_1,...,y_{|V_T|}) = (\xi_1,...,\xi_{|V_T|}, \eta_1,...,\eta_{|V_T|})$. Let T_i' be the matrix obtained from T_i by substituting ξ_i for x_i and η_i for y_i. Then the matrix substitution $\mu: V_T \to \mathbb{R}^{2\times 2}$ defined by $\mu_{a_i} = T_i'$ for $i = 1, 2,...,|V_T|$, satisfies $U(\mu) \neq V(\mu)$. □

Theorem 2 (*distinguishability II*). *Let* S_1 *and* S_2 *be formal power series corresponding to two cycle-free context-free grammars. Assume that the series admit the following representation:* $S_1 = S_0 + U + R_1$ *and* $S_2 = S_0 + V + R_2$, *where* S_0 *is the part of coinciding words of length less than or equal to N, U and V are different parts composed of words with length equal to N, and* R_1 *and* R_2 *contain terms with words of length greater than N. If U and V satisfy condition* $(\mathcal{M})$, *then there exists a matrix substitution* $\mu: V_T \to \mathbb{R}^{2\times 2}$ *such that* $S_1(\mu) \neq S_2(\mu)$.

Proof. Using Lemma 2 and following the proof of Theorem 1 we obtain the result. □

Examples:. Let $U = aab + bab$ and $V = aba + bba$. In this case condition $(\mathcal{M})$ is satisfied since $u_1u_1v_2 \in \mathcal{M}_U$ and $u_1u_1v_2 \notin \mathcal{M}_V$. On the other hand, there exist languages/grammars that cannot be distinguished with the help of (2×2)-matrices. For example, using Maxima computer algebra system it is easy to show that for any choice of (2×2)-matrices μ_1 and μ_2 we have

$$\begin{aligned} &\mu_1\mu_1\mu_2\mu_2\mu_1 + \mu_1\mu_2\mu_1\mu_1\mu_2 + \mu_2\mu_1\mu_2\mu_1\mu_1 \\ &\quad = \mu_1\mu_1\mu_2\mu_1\mu_2 + \mu_1\mu_2\mu_2\mu_1\mu_1 + \mu_2\mu_1\mu_1\mu_2\mu_1. \end{aligned}$$

Therefore the languages

$$\{aabba, abaab, babaa\} \text{ and } \{aabab, abbaa, baaba\} \tag{12}$$

cannot be distinguished using (2×2)-matrices. However substituting (3×3)-matrices it is easy show that the languages (12) are different.

2.2. Comparison of short words

In many situations the non-equivalence of two grammars can be detected comparing short words through substitution of (2×2)-matrices. We fulfilled about $47 \cdot 10^6$ tests to analyze all finite languages over terminal alphabet $\{a, b, c\}$ containing no more than three words of length less than or equal to five. We found that only the following languages cannot be distinguished using (2×2)-matrix substitutions: the pair of the languages

$$S_1 \to aabca|abaac|bacaa \quad \text{and} \quad S_2 \to aabac|abcaa|baaca \tag{13}$$

and other pairs obtained as the result of permutation of the letters $\{a, b, c\}$ and/or replacement of c by a or b. We shall denote this set of pairs of languages by L.

Theorem 3 (*distinguishability III*). *Let* S_1 *and* S_2 *be formal power series corresponding to two cycle-free context-free grammars over the terminal alphabet* $\{a, b, c\}$ *and let* $N \le 5$. *Assume that the series admit the following representation:* $S_1 = S_0 + U + R_1$ *and* $S_2 = S_0 + V + R_2$, *where* S_0 *is the part of coinciding words of length less than or equal to N, U and V are different parts composed of no more than three words with length equal to N, and* R_1 *and* R_2 *contain terms with words of length greater than N. If the pair of formal sums U and V does not coincide with one of the pairs of formal sums corresponding to pairs of languages from L, then there exists a matrix substitution* $\mu: V_T \to \mathbb{R}^{2\times 2}$ *such that* $S_1(\mu) \neq S_2(\mu)$.

Proof. Since the pair U and V does not coincide with one of the pairs from L, there exists a matrix substitution $\mu: V_T \to \mathbb{R}^{2\times 2}$ such that $U(\mu) \neq V(\mu)$. Then, following the proof of Theorem 1 we obtain the result. □

Note that the pair of languages (13) can be used to construct examples of infinite languages that cannot be distinguished using (2×2)-

matrices. One of such examples is the pair

$S \to abAa|abaA|baAa \;\; ; \;\; A \to aA|b$

and

$S \to abaA|abAa|baAa \;\; ; \;\; A \to aA|b$

Note also that in all the grammars of several programming languages and of educational exercises that we analyzed, the distinguishability was always possible using (2×2)-matrices.

3. Systems of nonlinear matrix equations

The correspondence between formal power series and systems of nonlinear equations makes it possible to effectively compute the sums of the series for any $(\mathfrak{N} \times \mathfrak{N})$-matrix substitution. Let X_i, $i = 0,...,m$, be the nonterminals of a context-free grammar G and let P_j^i, $i = 0,...,m$, $j = 1,...,J_i$, be the words that appear in the right-hand sides of productions with the left-hand sides X_i. Let X_1 be the axiom of the grammar. The system of equations corresponding to the grammar has the form

$$\begin{aligned} X_1 &= P_1^1 + ... + P_{J_1}^1, \\ &\vdots \\ X_m &= P_1^m + ... + P_{J_m}^m. \end{aligned} \tag{14}$$

Formally applying the iteration method to this system we obtain the formal power series

$$X_1 = \sum_{P \in W(V_T)} \phi_G(P) P. \tag{15}$$

Substituting the symbols of the terminal alphabet $a_l \in P_j^i$ by matrices μ_l, we obtain a system of nonlinear matrix equations with unknowns X_i. This system, $X = F(X)$, can be solved using the iterative process $X^{k+1} = F(X^k)$, $X^0 = 0$. The convergence of the method of successive iterations can be guaranteed for any cycle-free grammar, if the norms of the matrices μ_l are sufficiently small. Note that for regular grammars the system (14) is linear.

Theorem 4 (*Convergence*). *Assume that the system of nonlinear Eqs. (14) corresponds to a cycle-free context-free grammar G. Then substituting the terminal symbols a_l by the matrices μ_l with a sufficiently small norm, we can guarantee the convergence of the sequence X^k to a solution of the matrix system $X = F(X)$. The sequence X_1^k converges to the sum of the series*

$$X_1(\mu) = \sum_{P \in W(V_T)} \phi_G(P) P(\mu) \tag{16}$$

obtained from (15) substituting the terminal symbols a_l by the matrices μ_l.

Proof. Since the grammar G is cycle-free the ambiguity function satisfies the condition

$0 \le \phi_G(P) \le C q^{|P|}$,

where C and q are positive constants. Consider a matrix substitution $\mu = (\mu_1, ...,\mu_{|V_T|})$ satisfying

$\|\mu_l\| \le M < 1/(q|V_T|)$.

The successive iterations X_1^k are the partial sums of the series (16). From the inequality

$$\sum_{P \in W(V_T)} \phi_G(P) \|P(\mu)\| \le \sum_{N \ge 0} C q^N |V_T|^N M^N < \infty$$

(see (8)) and the Cauchy convergence test we see that series (16) converges and its sum coincides with the limit of the sequence X_1^k, $k = 1, 2, ...$ □

Notice that in general the system of nonlinear equations has many solutions, but applying the method of iterations starting from the zero initial point we get exactly the solution which is the sum (16). This fact is crucial for the grammar comparison method considered here.

Input: $g1, g2$: grammar
Output: result of comparison
begin
  $P \leftarrow$ ... calculate configuration parameter *eta* $(g1, g2)$
    // $\eta = 1/(\mathfrak{N}|V_T||V_N|)$, the norm of randomly generated matrices
  for $t \in g1.T$ **do**
    // randomly generate matrices for terminal symbols
    $M\_T[t] \leftarrow TMatrixGen(t, P.eta)$
  $s1 \leftarrow solverGram(s1, M\_T)$
  $s2 \leftarrow solverGram(s2, M\_T)$
  $v \leftarrow matrixCompare(s1, s2)$ // norm $(s1 - s2) < \varepsilon$
  return v

Algorithm 1. grammequiv($g1$, $g2$).

```
Input: g: grammar
Input: M_T: matrices associated to terminal symbols
Output: matrix
begin
    M ← M_T  // M: matrices associated to terminal and
                // nonterminal symbols
    for nt ∈ g.nt do      // for each nonterminal symbol of g
        M[nt] ← 0                                   // initiate its value with the zero matrix
    repeat
        for nt ∈ g.nt do
            s ← 0                       // sum
            for rhs ∈ g[nt].rhss do
                p ← I          // I is the identity matrix
                for y ∈ rhs do
                    p ← p × M[y]
                s ← s + p
            M[nt] ← s
    until the matrix corresponding to the axiom stabilizes
    return M[g.axiom]
```

Algorithm 2. solverGram(*g*, *M_T*).

```
Input: A, B: matrices
Output: conc: boolean
begin
    if A or B contains 'nan' or 'inf' then                    // ''not a number'' or ∞
        return 'Error'
    // Sum of relative errors for each matrix element
    d = |A11 − B11|/(|A11| + |B11|)
        +|A12 − B12|/(|A12| + |B12|)
        +|A21 − B21|/(|A21| + |B21|)
        +|A22 − B22|/(|A22| + |B22|)
    if d < ε then                                             // ε is the admissible error
        conc ← 1
    else
        conc ← 0
    return conc
```

Algorithm 3. matrixCompare(*A*, *B*):.

Examples:

Let us consider the grammar

$$S \to SaA|A \;\; ; \;\; A \to cSd|b \tag{17}$$

which is a correct solution of an exercise. (This example is taken from [12].) The corresponding system of equations reads:

$$\begin{aligned} S &= SaA + A, \\ A &= cSd + b. \end{aligned} \tag{18}$$

Below we present three other possible answers.

Alternative correct solution. The following grammar is different but equivalent:

$$S \to AaS|A \;\; ; \;\; A \to cSd|b \tag{19}$$

The corresponding system of equations is

$$\begin{aligned} S &= AaS + A, \\ A &= cSd + b. \end{aligned} \tag{20}$$

Wrong answer. The following grammar does not generate the same language (does not generate the word *cbabd*):

$$S \to SaA|A \;\; ; \;\; A \to cAd|b \tag{21}$$

The corresponding system is

$$\begin{aligned} S &= SaA + A, \\ A &= cAd + b. \end{aligned} \tag{22}$$

Ambiguous grammar. The following grammar generates the same language but is ambiguous (the word *babab* can be generated in different ways):

$$S \to SaS|A \;\; ; \;\; A \to cSd|b \tag{23}$$

The corresponding system has the form

$$\begin{aligned} S &= SaS + A, \\ A &= cSd + b. \end{aligned} \tag{24}$$

Replacing the symbols *a, b, c*, and *d* by (2×2)-matrices with sufficiently small norm we get a system that can be solved using the iteration method. Starting the iterative process with $S = A = 0$ and solving systems (18), (20), (22), and (24) we see that the difference between S-components of solution of systems (18) and (20) is zero, while for the pair (18) and (22) or (18) and (24) the difference is not zero. This allows one to clearly distinguish between right and wrong answers.

Note that the interval where the elements of the matrices are generated must be (a) sufficiently small in order to guarantee the convergence of the iterative process, (b) big enough to distinguish between two different languages.

4. Algorithms

In this section we present the main algorithms implemented to test the method of matrix equations. The main function is `gramequiv`. It calls the functions `solverGram` which calculates the matrix corresponding to the axiom, and the function `matrixCompare` which compares the matrix values of axioms. Algorithm 1The function `solverGram` is an implementation of the method of iterations applied to the system of matrix equations corresponding to a cycle-free context-free grammar. Algorithm 2

This algorithm calculates the solution to the system $X = F(X)$ (system (14) after substitution of terminal symbols by matrices) with X_1 component given by (16).

Rounding errors can lead to a wrong comparison result, and rather

delicate methods are needed to distinguish between two matrices. For example, consider the matrices

$$\begin{pmatrix} 1 & A \\ 1\times 10^{-30} & B \end{pmatrix} \text{ and } \begin{pmatrix} 1+10^{-13} & A \\ 1.3\times 10^{-30} & B \end{pmatrix},$$

and put $\varepsilon = 10^{-13}$. Any usual norm of the difference of two matrices is less than or equal to 10^{-13}. But the difference between 1×10^{-30} and 1.3×10^{-30} is significant (30%), and indicates that the grammars are not equivalent. This problem can be solved using a kind of relative error. Below we describe it for the case of (2×2)-matrices. Algorithm 3

5. Comparison with a simple heuristic method

One natural heuristic method to compare two context-free grammars is:

- Explicitly check with a parsing algorithm (for example, the classic CYK algorithm [7] or its modification [16]) all words up to length n;
- If difference is found, output "grammars are different";
- If no difference is found, output "grammars could be the same".

Such an approach does not allow us to distinguish between two grammars with different ambiguities and cannot be applied to grammars with distinguishing words of big length.

For example, let us consider two grammars:

$$S \to \varepsilon|I\ I\ S|A\ \ ;\ \ I \to a|b|c|d\ \ ;\ \ A \to C\ C\ C\ a\ a\ a\ \ ;\ \ C \to a\ a\ a\ a$$

and

$$S \to \varepsilon|I\ I\ S|B\ \ ;\ \ I \to a|b|c|d\ \ ;\ \ B \to C\ C\ C\ b\ b\ b\ \ ;\ \ C \to b\ b\ b\ b$$

The productions $S \to \varepsilon|I\,I\,S$ and $I \to a|b|c|d$ generate all the words of even length containing *a, b, c* and *d*, while A generates the word a^{15} and B generates the word b^{15}.

Distinguishing words exist, but have length 15 or more. This means that any generation and testing based algorithm must test about $4^{15} \approx 10^9$ words. This would be a very time-consuming procedure.

The algorithm presented here can easily distinguish between these two grammars within a fraction of a second.

6. Limitations of the method

In some cases the presented method cannot distinguish between two different grammars. This happens when we consider grammars with very long words from $W(V_T)$. The point is that the computer precision is not sufficient to correctly compute products of many small numbers. Consider the following grammar

$$S \to AS|BS|B\ \ ;\ \ A \to a_1a_2...a_n\ \ ;\ \ B \to a_1|a_2|...|a_n. \tag{25}$$

The corresponding system of matrix equations reads

$$\begin{aligned} S &= AS + BS + B, \\ A &= a_1a_2...a_n, \\ B &= a_1 + a_2{+}...{+}a_n. \end{aligned} \tag{26}$$

Let the second grammar be

$$S \to AS|BS|B\ \ ;\ \ A \to a_2a_1...a_n\ \ ;\ \ B \to a_1|a_2|...|a_n. \tag{27}$$

with the corresponding system of equations

$$\begin{aligned} S &= AS + BS + B, \\ A &= a_2a_1...a_n, \\ B &= a_1 + a_2{+}...{+}a_n. \end{aligned} \tag{28}$$

Systems (26) and (28) are equivalent to the equations

$$S = a_1a_2...a_nS + (a_1 + a_2{+}...{+}a_n)S + (a_1 + a_2{+}...{+}a_n)$$

and

$$S = a_2a_1...a_nS + (a_1 + a_2{+}...{+}a_n)S + (a_1 + a_2{+}...{+}a_n),$$

respectively. To guarantee the convergence of iterations we have to impose the restriction $\|a_1\| + \|a_2\|{+}...{+}\|a_n\| = \alpha < 1$. From the inequality of arithmetic and geometric means we get

$$\sqrt[n]{\|a_1\|...\|a_n\|} \le (\|a_1\|{+}...{+}\|a_n\|)/n.$$

Hence for sufficiently large n we have $\|a_1\|...\|a_n\| \le \alpha^n/n^n < \delta$, where $\delta > 0$ is the computer precision. Thus the computer interprets the iterative processes

$$S_{k+1} = a_1a_2...a_nS_k + (a_1 + a_2{+}...{+}a_n)S_k + (a_1 + a_2{+}...{+}a_n)$$

and

$$S_{k+1} = a_2a_1...a_nS_k + (a_1 + a_2{+}...{+}a_n)S_k + (a_1 + a_2{+}...{+}a_n)$$

as the same process

$$S_{k+1} = (a_1 + a_2{+}...{+}a_n)S_k + (a_1 + a_2{+}...{+}a_n)$$

and, therefore, the method does not allow to distinguish between the two grammars.

However, the method can be applied to *real size grammars*, for example, simplified C with 44 NT symbols and 104 production rules, with a satisfactory result. The comparison of such grammars takes about 81 ms of CPU-time

7. Conclusion

In this paper we addressed the problem of cycle-free context-free grammars equivalence. A substitution of terminal symbols by matrices allows us to develop a numerical procedure that can resolve the problem of equivalence of cycle-free context-free grammars in many practical cases. Beside the elegance of the process, this method constitutes a solid base for construction of algorithms and tools for testing the equivalence of cycle-free context-free grammars: the distinguishability theorems show that if two formal power series generated by cycle-free context-free grammars are different, then there exists a matrix substitution such that the sums of the respective matrix series are different. Our experiments with a built prototype show that the use of this method with (2×2) and (3×3)-matrices in problems appearing in e-learning framework and even in cases of large grammars is very efficient. At least it is much more efficient than the simple heuristic method based on CYK algorithm. In all our experiments one (2×2) or (3×3) random matrix substitution was enough to distinguish between two nonequivalent cycle-free context-free grammars. The limitations of our approach caused by rounding errors can be an obstacle only in very exotic examples.

Acknowledgments

We are very grateful to the reviewers for their valuable suggestions and bibliographical support. Our special thanks to the reviewer who suggested convenient index notation in the proof of Lemma 2.

This work has been supported by FCT – Fundação para a Ciência e Tecnologia within the Project Scope: UID/CEC/00319/2013.

References

[1] M. Almeida, N. Moreira, R. Reis, Finite automata minimization algorithms, in: J. Wang (Ed.), Handbook of Finite State Based Models and Applications, Discrete Mathematics and Its Applications, Chapman and Hall/CRC Press, 2012, pp. 145–170.
[2] M. Almeida, N. Moreira, R. Reis, Testing equivalence of regular languages, J. Automata Lang. Combinatorics 15 (1/2) (2010) 7–25.
[3] A. Salomaa, Formal Languages, Acad. Press, 1973.
[4] P. Cousot, R. Cousot, Grammar analysis, and parsing by abstract interpretation, in: T. Reps, M. Sagiv, J. Bauer (Eds.), Program Analysis and Compilation, Theory and Practice: Essays dedicated to Reinhard Wilhelm, LNCS 4444, Springer-Verlag, 2006, pp. 178–203.
[5] P. Cousot, R. Cousot, Grammar semantics, analysis and parsing by abstract

interpretation, Theor. Comput. Sci. 412 (44) (2011) 6135–6192.
[6] M.C. Paull, S.H. Unger, Structural equivalence of context-free grammars, J. Comput. Syst. Sci. 2 (4) (1968) 427–463.
[7] R. Madhavan, M. Mayer, S. Gulwani, V. Kuncak, Automating grammar comparison, in: J. Aldrich, P. Eugster (Eds.), Proceedings of the 2015 ACM SIGPLAN International Conference on Object-Oriented Programming, Systems, Languages, and Applications, OOPSLA 2015, part of SPLASH 2015, Pittsburgh, PA, USA, October 25–30, 2015, ACM, 2015, pp. 183–200.
[8] R. Madhavan, M. Mayer, S. Gulwani, V. Kuncak, Towards automating grammar equivalence checking, Technical Report. 206921, EPFL, 2015.
[9] A. Salomaa, M. Soittola, Automata-Theoretic Aspects of Formal Power Series, Springer, 1978.
[10] P. Kuchment, An overview of periodic elliptic operators, Bull. Amer. Math. Soc. 53 (2016) 343–414.
[11] B. Mityagin, The zero set of a real analytic function, (2015) arXiv preprint arXiv: 1512.07276.
[12] J. Almeida, E. Grande, G. Smirnov, Context-free grammars: Exercise generation and probabilistic assessment, 5th Symposium on Languages, Applications and Technologies (SLATE'16), volume 51 of *OpenAccess Series in Informatics (OASIcs)*, Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2016, pp. 1–8. URL http://drops.dagstuhl.de/opus/volltexte/2016/6015 .
[13] M.P. Schützenberger, On a theorem of R. Jungen, Proc. Am. Math. Soc. 13 (6) (1962) 885–890. URL http://www.jstor.org/stable/2034080 .
[14] M.A. Harrison, Introduction to Formal Language Theory, Addison-Wesley, Reading, 1978.
[15] K. Wich, Ambiguity Functions of Context-Free Grammars and Languages, University of Stuttgart, 2005 Ph.D. thesis. URL http://elib.uni-stuttgart.de/opus/volltexte/2005/2282/ .
[16] M. Lange, H. Leiß, To CNF or not to CNF? an efficient yet presentable version of the CYK algorithm, Informatica Didactica 8 (2009) 1–21.