Context-Free Grammars: Exercise Generation and Probabilistic Assessment

José João Almeida*[1], Eliana Grande†[2], and Georgi Smirnov[3]

1 Departamento de Informática, Universidade do Minho, Braga, Portugal
jj@di.uminho.pt

2 Departamento de Informática, Universidade do Minho, Braga, Portugal
eliana.tiba@ifgoiano.edu.br

3 Departamento de Matemática, Universidade do Minho, Braga, Portugal
smirnov@math.uminho.pt

Abstract

In this paper we present a metagrammar based algorithm for exercise generation in the domain of context-free grammars. We also propose a probabilistic assessment algorithm based on a new identity theorem for formal series, a matrix version of the well-known identity theorem from the theory of analytic functions.

1998 ACM Subject Classification F.4.2 Grammars and Other Rewriting Systems, F.4.3 Formal Languages, Classes defined by grammars or automata, F.2.1 Numerical Algorithms and Problems

Keywords and phrases Exercise generation, context-free grammars, assessment

Digital Object Identifier 10.4230/OASIcs.SLATE.2016.10

1 Introduction

This paper deals with (1) exercise generation and (2) assessment in the theory of context-free grammars. We describe a process of exercise generation based on metagrammars and an algorithm of assessment allowing one to decide, with high probability, if two context-free grammars are equivalent or not. It is well-known that the equivalence of two context-free grammars is an undecidable problem [9]. This is true if we consider the problem as an algebraic one. In this paper we show that the context-free grammars equivalence problem admits a solution if considered as a problem of analysis. Such a situation is not new. For example the fundamental theorem of algebra [7] has no algebraic proves but admits a simple proof using methods of analysis. The main idea of the assessment algorithm is the following. We associate to any context-free grammar a system of nonlinear equations. The system admits a solution in the form of formal series [10]. We substitute the symbols of terminal alphabet by 2 × 2-matrices and numerically solve the system of nonlinear matrix equations. This amounts to calculating of the value of the corresponding matrix series. We prove a new identity theorem. This theorem claims that if the sums of two formal power series coincide for all possible substitutions of 2 × 2-matrices, then the series are identical. From the practical point of view this implies that it suffices to check the equality between the sums of two series for a sufficiently big number of different substitutions, in order to decide if the grammars are equivalent or not. This leads us to a probabilistic assessment algorithm.

[*] The work of J.João Almeida was partially supported by COMPETE: POCI-01-0145-FEDER-007043 and FCT – Fundação para a Ciência e Tecnologia within the Project Scope: UID/CEC/00319/2013.

[†] The work of Eliana Grande was supported by PIQ IF Goiano through the fellowship no. 02/2013.

5th Symposium on Languages, Applications and Technologies (SLATE'16).
Editors: Marjan Mernik, José Paulo Leal, and Hugo Gonçalo Oliveira; Article No. 10; pp. 10:1–10:8
Open Access Series in Informatics
OASICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

This algorithm allows one not only to distinguish between two different languages but also to distinguish between grammars with different ambiguity.

2 Exercise generation and assessment

The exercise generation in the field of formal grammars can be based on several methods:

- Completely hand written,
- Example based methods (sets of positive and negative examples)
- Automata or regular expression-based generation,
- Grammar based generation.

As a result one pretends to obtain the statement of the problem, its solution and, if possible, an assessment method.

In this paper we consider a method based on a generative grammars.

2.1 Metagrammars for exercise generation

We define an attribute grammar which generates an exercise statement and the respective solution (a grammar). We refer to this attribute grammar as *metagrammar* [12, 11, 6]. The right-hand side (rhs) of any production-rule of the metagrammar is composed of two parts:

- a traditional grammar rhs to describe a grammar;
- a template to build an exercise statement, using subparts returned by the elements of the first part (attributes).

Schematically, an exercise generating metagrammar production has the following form:

$$\begin{array}{lll} \boxed{\text{NonTerminal}} & \rightarrow & (''GrammarComponent'' \mid \boxed{\text{NonTerminal}})^{*} \\ & & \{TemplateWithAttributes\} \end{array} \tag{1}$$

2.2 Example

Consider the following metagrammar example:

$$\begin{array}{lll} \boxed{\text{Mg}} & \rightarrow & ''S \rightarrow '' \boxed{\text{B}} '' \mid A\,;\ A \rightarrow '' \boxed{\text{C}} ''|Int;'' \\ & & \{Give\,a\,context - free\,unambiguous\,grammar\,for\,the\,language\,of\,the \\ & & arithmetic\,expressions\,including\,integers,\ \$2\,operator\,and\,\$4\,.\} \\ \boxed{\text{B}} & \rightarrow & ''S - A''\{infix\,subtraction\} \\ & \mid & ''S + A''\{infix\,addition\} \\ & \mid & ''S * A''\{infix\,multiplication\} \\ & \mid & ''S/A''\{infix\,division\} \\ \boxed{\text{C}} & \rightarrow & ''(S)''\{round\,brackets\} \\ & \mid & ''f(S)''\{prefix\,unary\,functions\} \end{array} \tag{2}$$

In the templates \$2 and \$4 stand for attribute expansions following the Yacc[1], Bash, Perl syntax.

An example of generated exercise is:

▶ **Example 1.** Give a context-free unambiguous grammar for the language of the arithmetic expressions including integers, infix subtraction operator and round brackets.

The generated correct grammar for this exercise is the following:

$$\begin{array}{lll} S & \rightarrow & S - A \\ & \mid & A \\ A & \rightarrow & (S) \\ & \mid & Int \end{array} \tag{3}$$

2.3 Assessment

The comparison of languages may take different flavor depending on the situation:

- If we are dealing with regular languages, we can use minimal automata comparison [3, 2];
- It is possible to use a set of tests (positive and negative sentences) to verify if two grammars generate the same language;
- In some situations it is possible to directly compare the grammars.

But these approaches are only applicable to special cases or do not completely solve the problem. Below we propose a new assessment method. It consists of the following steps:

1. The transformation of a context-free grammar into a system of formal nonlinear equations [9, 10];
2. The substitution of the terminal alphabet letters by randomly generated 2×2-matrices;
3. Creation of a numerical solution of the system of nonlinear matrix equations;
4. Iterating the previous steps k times.

The identity theorem proved in the next section says that if the sums of two formal power series coincide for all possible substitutions of 2×2-matrices, then the series are identical. Motivated by this result we use a probabilistic approach to compare two grammars. Namely, if the solutions of the respective systems of matrix equations coincide for k successive random matrix substitutions, we conclude that the grammars generate the same language.

Note that the use of 2×2-matrices is a key point of the approach. The multiplication of matrices is not commutative, and this fact allows one to distinguish between two words composed of the same terminal symbols collocated in different order. Obviously the use of numbers (instead of matrices) would not solve this problem.

3 Context-free grammars, formal power series, and nonlinear matrix equations

Any language can be defined in terms of a formal powers series in associative but noncommutative variables [9, 10]. Let V_T be a terminal alphabet, $W(V_T)$ be the set of words over V_T, and Z_+ be the set of nonnegative integers. A map $\phi : W(V_T) \to Z_+$ defines a formal power series

$$s = \sum_{P \in W(V_T)} \phi(P)P \tag{4}$$

with nonnegative integer coefficients. Let μ be a map from V_T to the set $R^{2\times 2}$ of 2×2-matrices. By $\mu(P)$ we will denote the matrix obtained substituting the letters $a_i \in P$ by the matrices $\mu(a_i)$ and calculating the respective matrix product. If the series

$$s(\mu) = \sum_{P \in W(V_T)} \phi(P)\mu(P) \tag{5}$$

converges, its sum is a 2×2-matrix. (If the series is divergent, we set $s(\mu) = \infty$.) The assessment of exercises is base on the following *identity* theorem

▶ **Theorem 2.** *Let s_1 and s_2 be two formal power series. If $s_1(\mu) = s_2(\mu)$ for all $\mu : V_T \to R^{2\times 2}$, then $s_1 = s_2$.*

Proof. It suffices to consider the class of matrices

$$\mu = \begin{pmatrix} u & v \\ 0 & 1 \end{pmatrix}.$$

By induction it is easy to prove the equalities

$$\prod_{i=1}^{n} \mu_i = \prod_{i=1}^{n} \begin{pmatrix} u_i & v_i \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} P & Q \\ 0 & 1 \end{pmatrix}, \tag{6}$$

where

$$P = \prod_{i=1}^{n} u_i \text{ and } Q = \sum_{m=2}^{n} \left(\prod_{i=1}^{m-1} u_i \right) v_m + v_1. \tag{7}$$

(It is allowed $u_i = u_j$, and $v_i = v_j$.) The polynomial P allows one to identify the letters appearing in the word $a_1 a_2 \ldots a_n$, while the polynomial Q makes it possible to uniquely reconstruct the order of the letters. ◀

▶ Remark. It suffices to check the equality of the series sums for matrices with a sufficiently small norm. This will guarantee the convergence of the series.

It is well-known [9, 10] that for any context-free grammar there exists a system of nonlinear equations allowing one to obtain, by an iterative procedure, the formal power series whose terms are the words of the respective language. This correspondence between the series and systems of equations makes it possible to effectively calculate the sums of the series for all possible substitutions of 2×2-matrices.

Let X_i, $i = \overline{0, m}$, be the nonterminals of a context-free grammar and let P^i_j, $i = \overline{0, m}$, $j = \overline{1, l_i}$, be the words appearing on the right-hand sides of the production with the left-hand side X_i. Then the system of formal equations corresponding to the grammar reads

$$\begin{aligned} X_1 &= P^1_1 + \ldots + P^1_{l_1}, \\ &\vdots \\ X_m &= P^m_1 + \ldots + P^m_{l_m}. \end{aligned} \tag{8}$$

Substituting the letters of the terminal alphabet, $a_i \in P^i_j$, by matrices $\mu(a_i)$, we get a system of nonlinear (matrix) equations. This system, $X = F(X)$, can be solved using an iterative procedure: $X^{k+1} = F(X^k)$, $X^0 = 0$, or using the Newton method. The latter is much more faster. Note that, in view of the Identity Theorem, it suffices to consider only matrices of the form

$$\mu = \eta \begin{pmatrix} u & v \\ 0 & 1 \end{pmatrix}$$

with $u, v \in [0, 1]$ and sufficiently small η. The convergence of the iterations can be guaranteed for grammars in Chomsky and Greibach normal forms. If we deal with regular languages, then system (8) is linear.

4 Example

Let us go back to the example considered in Sec. 2. Assume that we generate the following exercise:

▶ **Example 3.** Give a context-free unambiguous grammar for the language of the arithmetic expressions including integers, infix subtraction operator and round brackets.

The generated correct grammar for this problem is:

$$\begin{array}{lcl} S & \to & S - A \\ & | & A \\ A & \to & (S) \\ & | & Int \end{array} \tag{9}$$

To avoid confusion in notation we set $a = -$, $b = Int$, $c = ($, and $d =)$. Thus the right solution is given by

$$\begin{array}{lcl} S & \to & SaA \\ & | & A \\ A & \to & cSd \\ & | & b \end{array} \tag{10}$$

The corresponding system of matrix equations reads [9, 10]

$$\begin{array}{l} S = SaA + A, \\ A = cSd + b. \end{array} \tag{11}$$

Let us consider other three possible answers.

Alternative correct solution. The following grammar is different but generates the same language:

$$\begin{array}{lcl} S & \to & AaS \\ & | & A \\ A & \to & cSd \\ & | & b \end{array} \tag{12}$$

and the corresponding system of matrix equations is

$$\begin{array}{l} S = AaS + A, \\ A = cSd + b. \end{array} \tag{13}$$

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

$$\begin{array}{lcl} S & \to & SaA \\ & | & A \\ A & \to & cAd \\ & | & b \end{array} \tag{14}$$

The corresponding system of matrix equations is

$$\begin{array}{l} S = SaA + A, \\ A = cAd + b. \end{array} \tag{15}$$

Ambiguous solution. The following grammar generates the same set of sentences but is ambiguous (the sentence *babab* possesses more then one derivation):

$$\begin{array}{ccl} S & \to & SaS \\ & \mid & A \\ A & \to & cSd \\ & \mid & b \end{array} \tag{16}$$

and the corresponding system of matrix equations is

$$\begin{array}{l} S = SaS + A, \\ A = cSd + b. \end{array} \tag{17}$$

Generating random matrices a, b, c, and d with random elements uniformly distributed in the interval $[0, 0.1]$ and solving systems (11), (13), (15), and (17) (the iteration process starts at $S = A = 0$) we see that the difference between S components of solutions for systems (11) and (13) is zero, while for the pair (11) and (15) or (11) and (17) is of the order $10^{-4} \div 10^{-5}$. This allows one to clearly distinguish between the correct solution and the wrong one.

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

5 Probabilistic assessment

It is clear that to distinguish between the correct solution and a wrong solution it suffices to find a matrix substitution $\mu(a_i)$, $a_i \in V_T$, giving different solutions to systems (8) corresponding to two different grammars. But how many substitutions are needed to conclude that two (unambiguous) grammars generate the same language? The answer to this question can be given in a probabilistic form. Probabilistic approach is not new in program testing [4] or assessment of math exercises [8] and showed good results.

Consider the following thought statistical experiment. We randomly generate a pair of grammars with the same terminal alphabet (see Sec. 2) and solve, for an infinite sequence of matrix substitutions $\mu(a_i)$, $a_i \in V_T$, the corresponding systems (8) of matrix equations. The number of substitutions giving the same result for two systems of equations is a random variable ξ. It is natural to assume that ξ is distributed according with the Poisson law:

$$P\{\xi = k\} = \frac{\lambda^k e^{-\lambda}}{k!},$$

where λ is the average number of events. Our simulations show that $\lambda \ll 1$. Thus the probability to have k times equal results for two different languages (recall that we consider languages with different ambiguity as different languages) is

$$P\{\xi = k\} = \frac{\lambda^k e^{-\lambda}}{k!} < \frac{1}{k!}.$$

From the practical point of view this implies that if we have the same result for two systems of equations in, for example, $k = 10$ successive random matrix substitutions $\mu(a_i)$, $a_i \in V_T$, then the grammars generate the same language.

6 Application to real size grammars

In the context of exercise generation, the grammars are rather small, because the aim is to deal with one thing at a time. Typical examples of such grammars have half a dozen of nonterminal symbols, and a dozen of productions. In this situation we had no problems with the presented approach.

Comparing two real size grammars, we may fall in situations of non-convergence, or in situations where it is impossible to distinguish between two different grammars due to insufficient computer accuracy. In order to test our approach for real size grammars we used a C-like grammar [5] (44 nonterminals, 104 production rules). We compared this grammar with an equivalent one and with a slightly modified grammar generating a different language. Generating random matrices with components in the interval $[0, 0.01]$, we could correctly solve the respective systems of equations and to establish the coincidence of the languages generated by the equivalent grammars and to distinguish between the correct and the wrong grammars.

7 Conclusion and future research

In this paper we demonstrated that numerical methods can be an effective tool to compare context-free grammars.

We presented:

1. An algorithm for exercise generation in the domain of formal languages, namely context-free grammars. The algorithm makes use of metagrammars;
2. A probabilistic assessment algorithm allowing one to decide, with high probability, if two context-free grammars are equivalent or not.

The algorithms allow one to automatically generate and assess exercises involving context-free grammars.

In the future research we plan to address the following issues:

1. Determination of the range of random matrices guaranteeing the convergence of numerical methods used to solve the systems of nonlinear matrix equations;
2. A profound study of statistical properties of the random variable ξ defined in section 5.

Acknowledgments. The authors would like to thank the reviewers for their comments, insights and corrections.

References

1 Alfred V. Aho, Ravi Sethi, and Jeffrey D. Ullman. *Compilers: Principles, Techniques, and Tools*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1986.

2 Marco Almeida, Nelma Moreira, and Rogerio Reis. Testing equivalence of regular languages. *Journal of Automata, Languages and Combinatorics*, 15(1/2), 2010.

3 Marco Almeida, Nelma Moreira, and Rogerio Reis. Finite automata minimization algorithms. In Jiacun Wang, editor, *Handbook of Finite State Based Models and Applications, Discrete Mathematics and Its Applications*, pages pp.145–170. Chapman and Hall/CRC Press, 2012.

4 Richard Demillo and Richard Lipton. A probabilistic remark on algebraic program testing. *Information Processing Letters*, 7(4):pp. 193–195, 1978.

5 Robert Heckendorn. A grammar for the C- programming language. Technical Report Version S16, Department of Computer Science – University of Idaho, 2016. URL: `http://marvin.cs.uidaho.edu/Teaching/CS445/c-Grammar.pdf`.

6 Daan Leijen and Erik Meijer. Parsec: Direct style monadic parser combinators for the real world. Technical report, Department of Computer Science, University of Utrecht, 2001. URL: `http://research.microsoft.com/pubs/65201/parsec-paper-letter.pdf`.

7 Jerrold E. Marsden and Michael J. Hoffman. *Basic Complex Analysis*. W. H. Freeman, third edition, 1999.

8 Minh Luan Nguyen, Siu Cheung Hui, and Alvis C. M. Fong. Probabilistic equivalence verification approach for automatic mathematical solution assessment. In *23rd International Joint Conference on Artificial Intelligence (IJCAI2013)*, pages pp. 1352–1356, 2013.

9 Arto Salomaa. *Formal Languages*. Academic Press, 1973.

10 Arto Salomaa and Matti Soittola. *Automata-theoretic aspects of formal power series*. Springer, 1978.

11 S. Doaitse Swierstra. Combinator parsers: From toys to tools. *Electronic Notes in Theoretical Computer Science*, 41, 2001. `doi:10.1016/S1571-0661(05)80545-6`.

12 Harald Heinz Vogt, S. Doaitse Swierstra, and Matthijs F. Kuiper. Higher order attribute grammars. *SIGPLAN Not.*, 24(7):131–145, June 1989. `doi:10.1145/74818.74830`.