Syntactical analysis of context-free languages taking into account order of application of productions

V V Kishkan\textsuperscript{1}, K V Safonov\textsuperscript{1}, R Yu Tsarev\textsuperscript{2}

\textsuperscript{1}Reshetnev Siberian State University of Science and Technology, 31, pr. Krasnoyarsky Rabochy, Krasnoyarsk, 660037, Russia
\textsuperscript{2}Siberian Federal University, 79, Svobodny pr., Krasnoyarsk, 660041, Russia

E-mail: safonovkv@rambler.ru

Abstract. The article sets the task of syntactical analysis of monomials of context-free languages taking into account the order of application of productions in the process of outputting monomials. An attribute model of the process, which defines the description of the process being modeled as business process models, correlates with the level of context-free languages. The business process, presented in the form of a structural model, uses the alphabet and syntax of a specific modeling language, which allows the process to be described independently of the subject area. The problem of syntactical analysis is supplemented as follows. Developing a deadlock algorithm to determine whether makes it possible to obtain a monomial from the initial symbol using the productions of a given context-free language. Determining the choice of productions and number of times was used to derive this monomial and establish the order of using these productions. The article proposes the completed method of monomial labels which allows establishing the order of application of productions. Based on this method, research technologies of programming language that are of interest to the software industry can be developed.

1. Introduction

Context-free grammars, along with regular expressions, are actively used to solve problems associated with the elaboration of formal languages and syntactic analyzers [1–3]. One of the main advantages of context-free grammars is the ability to specify a wide class of languages while maintaining the relative compactness of the representation [4, 5].

The attribute model of the process, which defines the description of the process being modeled as business process models, correlates with the context-free language level. The transition to context-free languages is performed by modeling the business process using the tools of a structural or object-oriented approach. A business process presented in the form of a structural or object model uses the alphabet and syntax of a specific modeling language, which allows the process to be described independently of the subject area [6].

In order to formulate the problem of syntactical analysis of context-free languages, we need some definitions.

In the theory of languages and grammars, the original object is the alphabet, that is, the set of symbols $z_1, \ldots, z_m$, $x_1, \ldots, x_m$. A noncommutative operation of multiplication (concatenation) and a commutative
operation of formal sum are defined over the alphabet symbols, which allows considering the alphabet together with these operations as a semiring [7, 8].

Initially, the alphabet characters are divided into two subsets. So, the symbols $x_1, \ldots, x_m$ are called terminal symbols and are understood as a language dictionary. The symbols $z_1, \ldots, z_n$ are called nonterminal; they are needed to specify the set of grammatical rules that generate the language. The combination of these rules is called the grammar of the language and defines the correct monomials from terminal symbols $x_1, \ldots, x_m$, which are considered as correct sentences of the language [7, 8].

A language is considered as the formal sum of all regular monomials (sentences), that is, as a formal power series (FPS).

Practically all programming languages belong to the important class of context-free languages (CF-languages). The grammar of a CF-language is a set of substitution rules (productions):

$$z_j \rightarrow q_{jk}(z, x), \ j = 1, \ldots, n, \ k = 1, \ldots, p_j,$$

where $q_{jk}(z, x)$ – given monomials for symbols $z = (z_1, \ldots, z_n)$, $x = (x_1, \ldots, x_m)$. Thus, a context-free grammar (CF-grammar) is characterized by the fact that there is the only one nonterminal symbol in the left part of each production [7–9].

Monomials of a CF-language are obtained as follows. Productions (1) are applied to the initial symbol $z_1$, and then to other nonterminal symbols $z_j$, contained in monomials, in any order an unlimited number of times. The CF-language is an FPS whose members are derived monomials for terminal symbols.

It is known that the problem of syntactical analysis of monomials of a CF-language is to solve the inverse problem. Developing a deadlock algorithm to determine whether it is possible to obtain the monomial from the initial symbol using the productions of the given CF-language is needed (syntax control or recognition). After determining the variant of productions and the number of times it was used in the derivation of this monomial [8, 9].

Traditionally it is believed that the order of use of products does not matter [8, 9].

Let us note that for an arbitrary CF-grammar the syntactic analysis algorithms are rather complicated, so there is no deadlock algorithm [8].

However, without knowledge of the order of application of the productions, it is impossible to obtain the desired monomial, since, as is easy to see, applying two productions to a monomial in a different order can lead to different monomials.

In fact, let us consider productions $z_1 \rightarrow z_1z_2$, $z_2 \rightarrow z_2z_1$ and monomial $z_1z_2$. Applying the first production to it, and then the second one, we get a monomial:

$$z_1z_2z_1z_2z_1z_2z_1.$$

If we apply the second production first, and then the first, we get another monomial:

$$z_1z_2z_1z_2z_1z_2.$$
First, let us note that the FPS, representing the CF-language, has the following property [7, 10]. Let us consider a system of polynomial equations with noncommutative variables:

\[ z_j = Q_j(z, x) = q_{j1}(z, x) + \cdots + q_{jp_j}(z, x), \quad j = 1, \ldots, n, \]

which is called the Chomsky–Schützenberger system of equations [7, 10] and can be considered as the mathematical model of a CF-language.

It is known that this system has a unique solution \((z_1(x), \ldots, z_n(x))\) in the form of FPS, and the CF-language generated by the CF-grammar (1) is equal to \(z_1(x)\).

In order to include in the monomials the information about the productions used in its derivation, it is proposed the method of monomial labels, which allows solving the traditional problem of syntactical analysis of monomials of a CF-language [11].

The method of monomial labels is as follows [11]. First, replace the grammar (1) by the extended grammar:

\[ z_j \rightarrow t_{jk} q_{jk}(z, x), \quad j = 1, \ldots, n, \quad k = 1, \ldots, p_j, \]

where \(t_{jk}\) are monomial labels, which are symbols from an extended alphabet.

Then the corresponding Chomsky–Schützenberger system of equations is considered for new substitution rules:

\[ z_j = Q^*_j(z, x, t) = t_{j1} q_{j1}(z, x) + \cdots + t_{jp_j} q_{jp_j}(z, x), \quad j = 1, \ldots, n. \]  

Further, the solution of this system can be obtained by the method of successive approximations:

\[ z^{(k+1)}(x, t) = Q^*(z^{(k)}(x, t), x, t); \quad k = 0, 1, \ldots; z^{(0)} = 0. \]

As a result, the solution is obtained in the form of FPS:

\[ z_j = z^*_j(x, t) = \sum_{i=0}^{\infty} \langle z^*_j, w_i \rangle w_i; \quad j = 1, \ldots, n, \]

where \(w_i\) are monomial for symbols \(x_1, \ldots, x_m, t_{11}, t_{12}, \ldots, t_{np_m}\).

Iterations of the method of successive approximations for the system of equations (3) give polynomials of increasing degree with respect to symbols \(x_1, \ldots, x_m, t_{11}, t_{12}, \ldots, t_{np_m}\), at the same time monomials of degree are not high \(\deg_v(v)\) for symbols \(x_1, \ldots, x_m\), stabilize after a finite number of iterations, not changing with subsequent iterations.

Thus, it is possible to obtain the initial terms of the solution of system (3) in the form of FSR (4) up to any, arbitrarily high degree, including members of the FSR representing the first component of this solution:

\[ z_i = z^*_i(x, t) = \sum_{j=0}^{\infty} \langle z^*_j, w_i \rangle w_i. \]
Finally, the syntactical analysis of the monomial of a CF-language can be done as follows. By reading the monomials of the power \( \text{deg}_x(v) \) of the series (5) with respect to symbols \( x_1, \ldots, x_m \) and passing symbols \( t_{jk} \), one can establish whether there is the monomial \( v \), among them, and therefore, whether it can be derived using productions (1). Each monomial label \( t_{jk} \), contained in such a monomial indicates that in its derivation the rule is used:

\[
z_j \rightarrow t_{jk} \ q_{jk}(z, x).
\]

Indeed, from the system of equations (3) and the method of successive approximations, it is easy to see that, applying this derivation rule to a monomial, we multiply it from the left by symbol \( t_{jk} \).

Hence, the monomial labels of a monomial solve the problem of its syntactical analysis, showing the choice of productions of the CF-language and the number of times it was used to derive this monomial up to the order of their application. Thus, the method of monomial labels allows for a finite number of steps to achieve a deadlock-free syntactical analysis of any monomial of the CF-language given by grammar (1) [11].

3. Completed monomial labels method
Since the method of monomial labels does not provide information on the order of using productions in the derivation of a monomial, we will use marked brackets.

Namely, we complete the extended grammar (2) as follows:

\[
z_j \rightarrow [ t_{jk} q_{jk}(z, x) ]_{jk}, \quad j = 1, \ldots, n, \quad k = 1, \ldots, p_j,
\]

where \( ]_{jk} \) – closed marked bracket, and \( [ \) – opening bracket, pair to bracket \( ]_{jk} \). We do not mark the opening bracket because for any closing bracket it is always possible to define the paired opening bracket for it.

Let us show how marked brackets allow determining the order of application of productions by the example.

**Example 1.** Let us consider productions \( z_i \rightarrow z_i z_i^3, \ z_i \rightarrow z_i z_i^3, \) and write them in the form of grammar (9):

\[
z_i \rightarrow [ t_{11} z_i z_i^3 ]_{11}, \ z_i \rightarrow [ t_{12} z_i z_i^3 ]_{12}.
\]

Applying to the initial symbol the first production and then the second, we obtain monomial:

\[
[ t_{11} [ t_{12} z_i z_i^3 ]_{12} z_i^3 ]_{11}.
\]

Now we can see what is the order of application of the productions. Namely, the exterior brackets indicate that production with monomial label \( t_{11} \) was used first and the interior brackets indicate that production with monomial label \( t_{12} \) was used then.

Thus, the syntactical analysis (taking into account the order of application of the productions) of monomial \( v \) of a CF-language is carried out by the completed method of monomial labels. Let us consider the completed extended Chomsky–Schützenberger system of equations:
Further, as in the method of monomial labels, the solution of this system can be obtained by the method of successive approximations:

\[ z^{(k+1)}(x,t) = Q^{\text{st}}(z^{(k)}(x,t),x,t); k = 0,1,\ldots; z^{(0)} = 0. \]

Now iterations of this method for the system of equations (10) give stabilizing polynomials of increasing degree for symbols \( x, t \).

4. Main result

By reading all monomials of degree \( \text{deg}_x(v) \) of series \( z_1(x) \) with respect to the symbols \( x_1, \ldots, x_m \), while skipping symbols \( t_{jk} \) one can determine if there is monomial \( v \) among them, and which productions (1) generate it and also the marked brackets indicate the order of application.

Finally, the following main result (theorem) were obtained.

**Theorem 1.** The completed method of monomial labels based on the completed extended Chomsky–Schützenberger system of equations (10) allows performing deadlock-free syntactical analysis of any monomial generated by CF-grammar (1) in a finite number of steps taking into account the order of application of the productions.

In fact, the transition from grammar (1) to completed and extended grammar (9) allows to include in the process of monomial derivation information about productions and the order of their application. However, the order of application of productions cannot always be determined. Let us consider the hierarchy of marked brackets in the following situation.

**Example 2.** Let productions \( z_1 \rightarrow z_1 z_2 \), \( z_1 \rightarrow x_1 \), \( z_2 \rightarrow x_2 \) and monomial \( x_1 x_2 \) be given. We write the completed extended grammar with monomial labels and marked brackets:

\[ z_1 \rightarrow [t_{11} z_1 z_2]_{11}, z_1 \rightarrow [t_{12} x_1]_{12}, z_2 \rightarrow [t_{21} x_2]_{21}, \]

and note that monomial \( x_1 x_2 \) is obtained by sequentially applying productions with monomial labels \( t_{11}, t_{12}, t_{21} \), or productions with monomial labels \( t_{11}, t_{21}, t_{12} \).

In both cases, there is the derivation in the completed extended grammar:

\[ z_1 \rightarrow [t_{11} z_1 z_2]_{11} \rightarrow [t_{11} [t_{12} x_1]_{12} [t_{21} x_2]_{21} ]_{11}. \]

The last monomial shows that production corresponding to exterior brackets is used first, and productions that correspond to interior brackets are used later, but they are not subordinate to each other and can be used in any order.

5. Conclusion

The problem of syntactical analysis of monomials of a CF-language is supplemented as follows. Developing a deadlock algorithm to determine whether it is possible to obtain a monomial from the initial symbol using the productions of a given CF-language is needed. Afterwards, determining the choice of productions and number of times we derived this monomial and established the order of using these productions.
The syntactical analysis (taking into account the order of application of the productions) of the monomial of a CF-language is carried out by the completed method of monomial labels with marked braces studied above.

Finally, a next task would be to estimate the number of operations for syntactical analysis (taking into account the order of application of the productions) of the monomial.

Acknowledgement
The reported study was funded by Russian Foundation for Basic Research, Government of Krasnoyarsk Territory, Krasnoyarsk Region Science and Technology Support Fund to the research project no. 17-47-240318.

References
[1] Brachthäuser J, Rendel T and Ostermann K 2016 Parsing with first-class derivatives Proceedings of the Conference on Object-Oriented Programming Systems, Languages and Applications (OOPSLA 2016) 588–606
[2] Cereda P and Neto J 2017 Instrumenting a context-free language recognizer Proceedings of the 19th International Conference on Enterprise Information Systems (ICEIS 2017) 2 203–210
[3] Moraes S, Godbole A and Gharpure P 2018 Affinity analysis for context-free grammars IEEE International Conference on Power, Control, Signals and Instrumentation Engineering (ICPCSI 2017) 2471–2474
[4] Badano M and Vaggione D 2017 Characterization of context-free languages Theoretical Computer Science 676 92–96
[5] Engelfriet J, Maletti A and Maneth S 2018 Multiple context-free tree grammars: Lexicalization and characterization Theoretical Computer Science 728 29–99
[6] Stachtiari E and Katsaros P 2018 Compositional execution semantics for business process verification Journal of Systems and Software 137 217–238
[7] Salomaa A and Soittola M 1978 Automata-Theoretic Aspects of Formal Power Series (New-York: Springer–Verlag)
[8] Glushkov V, Cejtlin G and Yushchenko E 1974 Algebra, yazyki, programmirovanie (Kiev: Naukova dumka) (in Russian)
[9] Aho A, Lam M, Sethi R and Ullman J 2007 Compilers: Principles, Techniques, and Tools. 2nd Edition (New-York: Pearson Education)
[10] Egorushkin O, Kolbasina I and Safonov K 2016 On solvability of systems of symbolic polynomial equations Journal of Siberian Federal University. Mathematics & Physics 9(2) 166–172
[11] Egorushkin O, Kolbasina I and Safonov K 2017 On application of multidimensional complex analysis in formal language and grammar theory. Prikladnaya Diskretnaya Matematika 37 76–89