INFORMATICA, 2000, Vol. 11, No. 4, 441–454
© 2000 *Institute of Mathematics and Informatics, Vilnius*

Some Grammatical Structures of Programming Languages as Simple Bracketed Languages

Boris MELNIKOV, Elena KASHLAKOVA
Simbirsk State Univ., Russia
e-mail: bormel@mail.ru, helen@arbt.ru

Received: March 2000

Abstract. We consider in this paper so called simple bracketed languages having special limitations. They are sometimes used for the definitions of some grammatical structures of programming languages. Generally speaking, these languages are context-free, but not deterministic context-free, i.e., they cannot be defined by deterministic push-down automata. For the simple bracketed languages having special limitations, the equivalence problem is decidable.

We obtain the sufficient conditions for the representation some language by special sequences of simple bracketed languages. We also consider the examples of grammatical structures as the simple bracketed languages. Therefore, we can decide equivalence problem for some grammatical structures of programming languages, and such structures define neither regular, nor deterministic context-free languages.

Key words: context-free languages, bracket languages, equivalence problem.

1. Introduction

There is well-known, that we cannot solve equivalence problem for the whole class of the context-free languages (see Aho and Ullman, 1972 etc.). Moreover, authors do not know subclasses of this class, used in the programming languages as grammatical structures, for which this problem is decidable (except regular languages).

Really, there are the papers (Romanovskiy, 1986; Meitus, 1989; Meitus, 1993; Senizergues, 1997) where the equivalence problem for deterministic push-down automata is decided. However:

- the authors of that papers either consider the subsets of the deterministic push-down automata, or use proofs being too difficult;
- the decidability of the equivalence problem for the whole class of the deterministic push-down automata contradicts the fact obtained in (Stanevichene, 1993);[1]
- some of languages considered below cannot be defined by deterministic push-down automata;
- all the proofs of the papers cited before cannot be used for the real programming (for compilers, parser generators etc.); e.g., some of them contain operations using some algebraic objects.

[1] Therefore at least one of the papers cited before contains an error.

Authors of this paper consider the two last reasons as the main ones, because they hope to use the decidability of the equivalence problem (for one of subclasses of the context-free languages class) in the real compilers. Thus, we shall use so called simple bracketed languages having special limitations.

For such languages, the equivalence problem is decidable (see Melnikov, 1995; Dubasova and Melnikov, 1995). In this paper, we shall use the grammatical constructions considered in (Melnikov, 1995; Dubasova and Melnikov, 1995) for defining some programming languages.

In this paper (Section 2), we shall define the notation of (Melnikov, 1995; Dubasova and Melnikov, 1995) and consider theorems obtained there. Also in (Dubasova and Melnikov, 1995), some examples of grammatical structures of computer languages were considered. Such structures (defined there as *simple bracketed languages*) satisfy the limitations of the Theorem 1 (see also Section 2). By the theorems obtained in (Melnikov, 1995; Dubasova and Melnikov, 1995), the equivalence problems for the structures having such limitations are decidable. Thus, for some grammatical structures of programming languages, the equivalence problem is decidable, and these structures are not regular.

In special cases, we can also consider sequences of such structures and decide equivalence problem for these sequences. In this paper, we consider some sufficient conditions for grammatical structures of computer languages. Therefore, the structures satisfying such conditions are the sequences of simple bracketed languages with the decidable equivalence problem (see Section 3). We shall also consider compound example (Section 4), where the language is represented by the sequence of simple bracketed languages.

And in the last Section 5, we shall consider some questions connected with the simple bracketed languages, which do not have required limitations (e.g., the prefix property), and formulate some problems for the future solution.

2. Preliminaries

We shall use in this paper notation and definitions of (Aho, 1972; Salomaa, 1979; Shamir, 1965). Remark that unlike (Aho, 1972; Salomaa, 1979; Shamir, 1965), the considered alphabet Σ may be infinite. The motivation for the assuming of this fact can be found in (Melnikov, 1995; Dubasova and Melnikov, 1995). In addition, we shall consider the following definitions.

DEFINITION 1. $\mathrm{pref}\,(u)$ is the set of all prefixes of the word u.

For example, the prefixes of the word $abcd$ are e, a, ab, abc and $abcd$.

DEFINITION 2. We shall say, that the set A has the prefix property, if

$$(\forall u, v \in A,\ u \neq v)\,(u \notin \mathrm{pref}\,(v)).$$

Denote this property as $\mathrm{Pr}\,(A)$.

Similarly, suff (u) is the set of all suffixes of the word u. Sf (A) denotes, that the set A has the suffix property.

DEFINITION 3. If the set A has both prefix and suffix properties, we shall write PrSf (A).

We shall consider below the sets of pairs of words over the alphabet Σ and denote these sets by μ, sometimes with subscripts.

We shall also consider the cases, when the sets Σ and/or μ are infinite (i.e., they contain infinitely many words or pairs of them). Remark that all the definitions connected with the grammars may be also used in such cases, dealing with infinite sets; see for details (Melnikov, 1995).

We shall also use the following notation:

$$\alpha(\mu) = \{u | (\exists v)\,((u,v) \in \mu)\}, \qquad \alpha(\mu, v) = \{u | (u,v) \in \mu\}.$$

Similarly,

$$\beta(\mu) = \{v | (\exists u)\,((u,v) \in \mu)\}, \qquad \beta(\mu, u) = \{v | (u,v) \in \mu\}.$$

If both conditions Pr $(\alpha(\mu))$ and Sf $(\beta(\mu))$ hold, we shall write PrSf (μ).

DEFINITION 4. For the given language L, define the *simple bracketed language* $\Psi^*(\mu, L)$ by the following way: $\Psi^0(\mu, L) = L$ and for each $j \geqslant 0$,

$$\Psi^{j+1}(\mu, L) = \{u\Psi^j(\mu, L)v | (u,v) \in \mu\},$$
$$\Psi^*(\mu, L) = \bigcup_{j \geqslant 0} \Psi^j(\mu, L), \quad \Psi^+(\mu, L) = \bigcup_{j > 0} \Psi^j(\mu, L).$$

Considering some language $L \subseteq \Sigma^*$ and two sets of pairs μ_1 and μ_2, we shall write

$$\Psi_i^* = \Psi^*(\mu_i, L), \quad \Psi_i^+ = \Psi^+(\mu_i, L), \quad \Psi_i^j = \Psi^j(\mu_i, L).$$

Consider now the equation

$$\Psi_1^* = \Psi_2^* \tag{1}$$

with the following limitations:

$$\text{PrSf}\,(L), \quad \text{PrSf}\,(\mu_i), \tag{2}$$
$$L \cap \Psi_i^+ = \emptyset, \tag{3}$$
$$\Psi_1^1 = \Psi_2^1, \quad \Psi_1^2 = \Psi_2^2$$

(remark that we have $\Psi_1^+ = \Psi_2^+$ by (1) and (3)). In (Dubasova and Melnikov, 1995), the following theorem was obtained.

Theorem 1. *Let the following conditions hold:*

$$\begin{aligned}&\mathrm{PrSf}(L), \mathrm{PrSf}(\mu_i),\\ &L \cap \Psi_i^+ = \emptyset,\\ &\Psi_1^1 = \Psi_2^1, \quad \Psi_1^2 = \Psi_2^2.\end{aligned}$$

Then the equation

$$\Psi^*(\mu_1, L) = \Psi^*(\mu_2, L)$$

is true if and only if at least one of following conditions holds:

- $\mu_1 = \mu_2$*;*
- *for:*
  - *some languages* $C, \widetilde{C} \subseteq \Sigma^*$*, such that* $\widetilde{C}L = LC$*;*
  - *and some numbers* $k_1, l_1, k_2, l_2 \geqslant 0$*, such that* $k_1 + l_1 = k_2 + l_2$*,*

  the following equations hold:

  $$\mu_1 = \widetilde{C}^{k_1} \times C^{l_1}\,, \qquad \mu_2 = \widetilde{C}^{k_2} \times C^{l_2}\,.$$

Some examples of using of this theorem were also considered in (Dubasova and Melnikov, 1995).

3. Some Grammatical Structures as the Sequences of Simple Bracketed Languages

In this section, we obtain some sufficient conditions for the representation some grammatical structure as simple bracketed language.

PROPOSITION 1. Let the set of productions P be defined by the following way:

$$\begin{aligned}&S \to L \,|\, ASB,\\ &A \to A_1 a,\\ &B \to b\, B_1,\\ &L \to uL_1 v,\end{aligned} \tag{4}$$

where the following limitations hold:

- the language of the nonterminal L_1[2] does not begin by u and does not end by v, i.e., $u \notin$ pref$\,(\mathcal{L}(L_1))$ and $v \notin$ suff$\,(\mathcal{L}(L_1))$; [3]

[2] I.e., the language defined by the grammar $G = \{N, \Sigma, P, L_1\}$.

[3] We use the notation pref$\,(\mathcal{L}(L))$, not pref$\,(L)$, because, corresponding to Aho (1972), the set $\mathcal{L}(L)$ designates the language of the nonterminal L.

- the language of the nonterminal A_1 does not end with a, i.e., $a \notin \text{suff}\,(\mathcal{L}(A_1))$;
- the language of the nonterminal B_1 does not begin by b, i.e., $b \notin \text{pref}\,(\mathcal{L}(B_1))$.

In these conditions, $A, B, L, A_1, B_1, L_1 \in N$ and the languages of these nonterminals are not empty.

Then the language L defined by the grammar $G = \{N, \Sigma, P, S\}$, is a simple bracketed language having the limitations (2). (We do not define here the productions for the languages of nonterminals L_1, A_1 and B_1 at all, because they are not important here.)

Proof. The language of the grammar G is $\Psi^*(\mu, L)$, where $\mu = A \times B$.

The language of the nonterminal L has both prefix and suffix properties, because all the words of $\mathcal{L}(L)$ begin by u and end by v. Similarly, for the nonterminals A and B, the following conditions hold:

$$\text{Pr}\,(\mathcal{L}(A)), \quad \text{Sf}\,(\mathcal{L}(B)).$$

All these facts prove the proposition.

EXAMPLE 1. This example of the grammatical structure constructed by Proposition 1 is the `@if`-function of the Quattro-Pro language:[4]

```
[@if-function] ::= @if ( [condition] .
                         [true-expression] .
                         [false-expression] )
[true-expression] ::= [expression]
[false-expression] ::= [expression]
[expression] ::= "[string]"
              | [function]
              | [numerical expression]
```

PROPOSITION 2. Let the given grammar $G = \{N, \Sigma, P, S\}$ be defined by the following set of productions:

$$S \to L \,|\, ASB,$$
$$A \to uX,$$
$$B \to Yv,$$
$$L \to uCv,$$

where $u, v \in \Sigma^*$, $A, X, B, Y, L, C \in N$, and the limitations of Theorem 1 do not hold, i.e., $\overline{\text{Pr}}\,(\mathcal{L}(A))$ or $\overline{\text{Sf}}\,(\mathcal{L}(B))$. Let also be $|u| \geqslant 2$ or $|v| \geqslant 2$.

Then there exists the equivalent grammar G', which set of productions satisfies the conditions of Theorem 1.

[4] This example is cited by Dubasova and Melnikov (1995).

Proof. Consider the grammar $G' = \{N', \Sigma, P', S'\}$ obtained from the given grammar G by the following way.

If $|u| \geqslant 2$ and $|v| \geqslant 2$, then

$$\begin{aligned}&(\exists u_1, u_2 \in \Sigma^*,\ u_1 \neq e,\ u_2 \neq e)\ (u = u_1 u_2),\\&(\exists v_1, v_2 \in \Sigma^*,\ u_1 \neq e,\ u_2 \neq e)\ (v = v_1 v_2).\end{aligned}$$

Then let N' be equal to $N \cup S'$, and the set of productions P' be the following one:

$$\begin{aligned}&S' \to u_1 S v_2,\\&S \to L \,|\, ASB,\\&L \to u_2 C v_1, \qquad (5)\\&A \to u_2 X u_1,\\&B \to v_2 Y v_1.\end{aligned}$$

(6)

The equation $\mathcal{L}(G) = \mathcal{L}(G')$ is evident. In the grammar G', nonterminals A and B satisfy the conditions Pr $(\mathcal{L}(A))$ and Sf $(\mathcal{L}(B))$, since all the productions for A begin by u_2 and all the productions for B end by v_1.

All these facts prove the proposition.

EXAMPLE 2. Consider grammatical structure of programming language Pascal, which defines compound statement[5]

```
[compound statement] ::= begin [sequence of statements] end
[sequence of statements] ::= [statement] '( ;[statement] )'
```

where

$L = \{$`gin [sequence of statements] e`$\}$.

Suppose that the language L is obtained without the productions for the nonterminal `[sequence of statements]`.

$A = \{$`gin '( [statement]; )' be`$\}$,
$B = \{$`nd '( ;[statement] )' e`$\}$.

Then the language defined by the construction `[compound statement]` is

`be` $\Psi^*(\mu, L)$ `nd`.

[5] Special brackets `'(` and `)'` will denote iterations.

There is evident, that all the needed conditions (i.e., PrSf (L), Pr (A) and Sf (B)) do hold.

The given context-free grammar with the set of productions P is called *sequential*,[6] if we can define some linear order (denoted by the symbol $\prec$) over the set N, having the following property. If $A \prec B$, then for each production $B \to \alpha$ belonging to P, the word $\alpha \in (N \cup \Sigma)^*$ does not contain the letter A.

Theorem 2. *The language L can be defined by the sequence of simple bracketed languages having limitations* (2)*, if there exists the sequential grammar G, such that $\mathcal{L}(G) = L$, which satisfies the following conditions.*

- *Denote the nonterminals of this grammar, which contain themselves in the right parts of their own productions, by $S_0, S_1, \ldots, S_k$;*[7]
- *denote other nonterminals by $S_{k+1}, \ldots, S_m$.*

Then for each $i \leqslant k$, the languages of the nonterminals S_i are defined by the grammar with the productions of the type (4) *or* (5)*, and there exists j, such that $j \leqslant i$ and the given language L is equal to $\mathcal{L}(S_j)$.*

Proof. Let for each $i \in 0 \ldots k$, the language of the nonterminal S_i be defined by the grammar $G_i = \{N_i, \Sigma_i, P_i, S_i\}$. If for each i, the set of productions P_i has the type (4) or (5), and there exists the only j, such that $L = \mathcal{L}(S_j)$, then the following equation holds:

$$\mathcal{L}(S_i) = \Psi^*(\mu_i, \mathcal{L}(S_j)).$$

By the conditions of the theorem, we can consider nonterminals in the order

$$S_0, S_1, \cdots, S_k,$$

such that for each $i \in 1, \cdots, k+1$, the following condition holds:

$$\mathcal{L}(S_{i-1}) = \Psi^*(\mu_i, \mathcal{L}(S_i)).$$

Then the sequence of languages is defined by

$$\mathcal{L}(S_0), \mathcal{L}(S_1), \cdots, \mathcal{L}(S_k), \cdots, \mathcal{L}(S_m).$$

The theorem is proved.

Let some language L be defined by the grammar G. We have proved, that for the representation L by the simple bracketed language or by the sequence of such languages, there is sufficient to find some grammar G', being equivalent to G, such that the set of productions of the grammar G satisfies the conditions of Propositions 1 or 2 or Theorem 2.

[6] For details, see (Shamir, 1965).

[7] I.e., for some S_i, there exists a production of the type $S_i \to \alpha S_i \beta$.

4. A Compound Example

We consider here the sequential context-free grammar, which is equivalent to the programming language PL/0 (see Wirth, 1979). We also consider the representation this language by the finite sequence

$$L_0,\ L_1,\ \ldots\ ,\ L_n\ ,$$

where for each $i \in \{\,1, 2, \ldots, n\,\}$, the following condition holds:

$$L_{i-1} = \Psi^*(\mu_i, L_i)\,.$$

(There is also possible the simplest representation for some languages L_i of this sequence. See detailed below.)

We shall use the following notation for the nonterminals of (Wirth, 1979; Weingarten, 1973) and other ones, used for the making sequential grammar:

P	program (initial nonterminal)
B	block
DCC	section of constants
DC	definition of the constant
DNN	section of variables
S and S′	statement
SS	sequence of statements
C	condition
E	arithmetical expression
I	identifier
N	number
TEMP	temporary nonterminal

We do not consider below the productions for the identifier and the number at all, because they are simple (and, certainly, sequential). Therefore the symbols I and N are used in the right parts of the productions only. Thus, all the words consisting of capital letters and figures only, are below nonterminals. We do not use below the empty word e (we write nothing instead, as in the papers and books with the descriptions of the real programming languages) and also the symbol $|$.

Remark that in the folowing text, considering grammars, we shall not usually write punctuation marks (commas etc.), because the same symbols could also belong to the set of terminals of this grammar. Thus, the initial grammar is as follows:

```
P     ::= B .
B     ::= const DCC ; var DNN ; DPP S
DPP   ::= procedure I ; const DCC ; var DNN ; DPP S ; DPP
      ::=
DCC   ::= DC
      ::= DC , DCC
DC    ::= I = N
DNN   ::= I
      ::= I , DNN
```

```
S    ::= I := E
     ::= call I
     ::= if C then S
     ::= while C do S
     ::= begin SS end
SS   ::= S′
     ::= S′ ; SS
     ::= TEMP S′
     ::= TEMP S′ ; SS
     ::= begin SS end
     ::= begin SS end ; SS
     ::= TEMP begin SS end
     ::= TEMP begin SS end ; SS
TEMP ::= if C then TEMP
     ::= while C do TEMP
     ::=
S′   ::= I := E
     ::= call I
     ::= if C then S′
     ::= while C do S′
C    ::= odd E
     ::= E = E
     ::= E < E
     ::= E <= E
E    ::= + E
     ::= - E
     ::= E + E
     ::= E - E
     ::= E * E
     ::= E / E
     ::= ( E )
     ::= I
     ::= N
```

(compare this grammar with one of (Wirth, 1979)). The order for nonterminals is the folowing one:

$$\mathrm{P} \prec \mathrm{B} \prec \mathrm{DPP} \prec \cdots \prec \mathrm{C} \prec \mathrm{E} \prec \mathrm{I} \prec \mathrm{N} \qquad (7)$$

(remark that we have already written their productions in the same order). Using this order and definitions of (Shamir, 1965), we obtain that this grammar is the sequential one.[8]

Now consider this language PL/0 as the finite sequence of languages

$$L_0,\ L_1,\ \ldots\ ,\ L_n\ ,$$

[8] Remark also that the following sequential context-free grammar was obtained by the authors by the special algorithm of transformation of the given one, being not sequential. However, we consider in this paper neither this algorithm, nor the sufficient conditions for its using; very likely, we will do this thing in the further publications. But this fact is not the barrier for the further reading, because we can simply prove the equivalence of the grammar of Wirth (1979) and the one of this section without such algorithm.

where for each $i \in \{ 1, 2, \ldots, n \}$:

- either the following condition holds:

$$L_{i-1} = \Psi^*(\mu_i, L_i)\,;$$

- or the productions for the nonterminal corresponding to the language L_{i-1} do not have this nonterminal (and, certainly, nonterminals which are less than L_{i-1} in the order (7)) in the right parts.

Remark that some of the sets of pairs μ_i have the limitations considered in Sections 2 and 3.

Thus, we do not consider below nonterminals which have themselves in right parts of their productions. Some of these nonterminals were already mentioned in the grammar, and other ones are temporary. The last ones are `TEMP1`, `TEMP2`, …, `TEMP10`, we shall write for them their productions only.

For each other nonterminal (let it be `A`) the following equation will hold:

$$\mathcal{L}(\mathtt{A}) = \Psi^*(\mu_{\mathtt{A}}, L\mathtt{A})$$

(where subscripts are the same as the considered nonterminal, therefore they may contain 2 or more letters). We shall also believe, by default, that

$$\mu_{\mathtt{A}} = \alpha_{\mathtt{A}} \times \beta_{\mathtt{A}}\,,$$

where $\alpha_{\mathtt{A}}$, $\beta_{\mathtt{A}}$ and $L\mathtt{A}$ will be defined for each nonterminal `A`. In evident cases, we shall not write the facts, whether or not the considered languages have the prefix and / or suffix properties.

Thus, consider the languages of nonterminals `DPP`, `DCC`, `DNN`, `S`, `SS`, `TEMP`, `S'` and `E`.

$$\alpha_{\mathtt{DPP}} = \mathcal{L}(\mathtt{TEMP1}) \qquad \beta_{\mathtt{DPP}} = L\mathtt{DPP} = \{e\}$$

where there exists the only production for nonterminal `TEMP1`:

```
TEMP1 ::= procedure I ; const DCC ; var DNN ; DPP S ;
```

$$\alpha_{\mathtt{DCC}} = \mathcal{L}(\mathtt{DC}), \qquad \beta_{\mathtt{DCC}} = \{e\} \qquad L\mathtt{DCC} = \mathcal{L}(\mathtt{DC})$$

Language $\alpha_{\mathtt{DCC}}$ has prefix property, because each its word ends by ,. We can also believe that the language $L\mathtt{DCC}$ has both prefix and suffix properties, because each its word can be considered as a lexeme returned by the scanner.

$$\alpha_{\mathtt{DNN}} = \mathcal{L}(\mathtt{I}), \qquad \beta_{\mathtt{DNN}} = \{e\} \qquad L\mathtt{DNN} = \mathcal{L}(\mathtt{I})$$

This language is similar to the previous one.

$$\alpha_{\mathtt{S}} = \mathcal{L}(\mathtt{TEMP2}) \qquad \beta_{\mathtt{S}} = \{e\} \qquad L\mathtt{S} = \mathcal{L}(\mathtt{TEMP3})$$

where the languages of nonterminals `TEMP2` and `TEMP3` are defined by the following productions:

```
TEMP2 ::= if C then
      ::= while C do
TEMP3 ::= I := E
```

```
      ::= call I
      ::= begin SS end
```

Remark that the language $\mathcal{L}(\mathtt{S})$ contains sublanguage (before nonterminal `TEMP3`) which has not prefix property,[9] however the language α_{S} has it (because no word of the language $\mathcal{L}(\mathtt{C})$ contains `then` and `do` as subwords). The similar fact holds also for other languages, considered for $\mathcal{L}(\mathtt{C})$.

The following language is more difficult.

$$\mu_{\mathrm{SS}} = \mathcal{L}(\mathtt{TEMP4}) \times \{e\} \cup \mathcal{L}(\mathtt{TEMP5}) \times \mathcal{L}(\mathtt{TEMP6})\,,$$

where languages of new nonterminals are defined by the following productions:

```
TEMP4 ::= TEMP7 S' TEMP8
TEMP5 ::= TEMP7 begin
TEMP6 ::= end
TEMP6 ::= end ; SS
TEMP7 ::=
TEMP7 ::= TEMP
TEMP8 ::=
TEMP8 ::= ;
```

Thus, we have defined the set of pairs μ_{SS} using $\mathcal{L}(\mathtt{SS})$, but as we have said before (see also (Melnikov, 1995)), we can use infinite sets of pairs. Unfortunately, the language α_{SS} has not prefix property (see about this fact the next section).

$$\alpha_{\mathrm{TEMP}} = \mathcal{L}(\mathtt{TEMP2}) \qquad \beta_{\mathrm{TEMP}} = \mathcal{L}(\mathtt{TEMP}) = \{e\}$$

$$\alpha_{\mathrm{S}'} = \mathcal{L}(\mathtt{TEMP2}) \qquad \beta_{\mathrm{S}'} = \{e\} \qquad L_{\mathrm{S}'} = \mathcal{L}(\mathtt{TEMP9})$$

where the language of nonterminal `TEMP2` was already defined, and `TEMP9` has following productions:

```
TEMP9 ::= I := E
      ::= call I
```

Also, the nonterminal `E` cannot be defined by simple bracketed language having limitations considered above. This fact holds because the language β_{E} must include e. This language must also include other words, therefore the condition Sf (β_{E}) cannot be true. However, see also the next section.

Thus,

$$\mu_{\mathrm{E}} = \mathcal{L}(\mathtt{TEMP10}) \times \{e\} \cup \{\, \mathtt{(} \} \times \{ \mathtt{)} \,\}\,,$$

where the language of nonterminal `TEMP10` is defined by the following productions:

[9] The simplest example for this thing is the following one. It contains constructions `if a=b then` and `if a=b then if c=d then`.

```
    TEMP10 ::= +
           ::= -
           ::= E +
           ::= E -
           ::= E *
           ::= E /
```

and

$$L_{\mathtt{E}} = \mathcal{L}(\mathtt{I}) \cup \mathcal{L}(\mathtt{N})\,.$$

Remark that we defined the language $\mathcal{L}(\mathtt{E})$ using itself. Exactly, $\mathcal{L}(\mathtt{E})$ is defined by $\mu_{\mathtt{E}}$, $\mu_{\mathtt{E}}$ by $\mathcal{L}(\mathtt{TEMP10})$, and $\mathcal{L}(\mathtt{TEMP10})$ by $\mathcal{L}(\mathtt{E})$ again.

5. Conclusion

Thus, we have obtained the sufficient conditions for the representation some grammatical structure as by the sequence of simple bracketed languages. These conditions are important, because they form a subclass of the context-free languages class with the decidable equivalence problem.

As we said before, for most of grammatical structures considered in this paper (and also pairs of such structures, that are supposed to define the same language), we can use theorems of the Sections 2 and 3. We cannot use these theorems for nonterminals SS and E only. The reason is the following one: corresponding languages $\alpha(\mu)$ and / or $\beta(\mu)$ do not have prefix (suffix) properties. However, autors hope, that the representation some programming languages by the sequences of the simple bracketed languages can be used in various tasks connected with the theory of parsing. The reasons are as follows.

- We have already solved the equivalence problem for some grammatical structures.
- Grammatical structures being similar to the considered in Section 4, are used in all real programming languages.
- We may use not necessary and sufficient conditions of equivalence of two grammatical constructions only, but also sufficient ones. Using them, we can sometimes know, that the two grammatical structures supposed to be equivalent, are not such ones.

We can also set the following problem for the future solution: to formulate some algorithms for searching grammatical structures (like to previous ones, with the decidable equivalence problem) in the given context-free grammar. The possible solution of this problem can be used for the translators and other automation systems.

References

Aho A., J. Ullman (1972). *The Theory of Parsing, Translation and Compiling*. Vol 1. Prentice-Hall, Englewood Cliffs, N.Y.

Dubasova O., B. Melnikov (1995). On the generalization of the context-free languages class. *Programmirovanie*, Russian Academy of Sciences Ed., **6**, 46–58 (in Russian).

Meitus Y. (1989). Equivalence problem for the strong real-time deterministic push-down automata. *Kybernetika*, Ukranian Academy of Sciences Ed., **5**, 14–25 (in Russian).

Meitus Y. (1993). Decidability of the equivalence problem for deterministic push-down automata. *Cybernetics and System Analysis*, **28**, 672–690.

Melnikov B. (1995). Some equivalence problems for free monoids and for subclasses of the CF-grammar class. *Number Theoretic and Algebraic Methods in Computer Science*, World Sci. Publ., 125–137.

Romanovskiy V. (1986). Equivalence problem for the real-time deterministic push-down automata. *Kybernetika*, Ukranian Academy of Sciences Ed., **2**, 12–23 (in Russian).

Salomaa A. (1979). *Jewels of the Formal Languages Theory*.

Senizergues G. (1997). The equivalence problem for deterministic pushdown automata is decidable. *Lecture Notes in Computer Science*, **1256**, 671–681.

Shamir E. (1965). On sequential languages. *Z. Phonetik, Sprach. Kommunikationsforsch.* **18**, 61–69.

Stanevichene L. (1993). An insoluble algorithmic problem for the deterministic push-down automata. *Dokladi Akademii Nauk*, Russian Academy of Sciences Ed., **342**(6), 744–746 (in Russian).

Weingarten F. (1973). *Translation of Computer Languages*, Holden-Day, Inc.

Wirth N. (1979). *Algorithmes + Data Structures = Programs*.

B.F. Melnikov. Since 1999: Professor of Simbirsk (Ulyanovsk) State University (Russia). 1995–1999: Associated Professor of Simbirsk State University. 1991–1995: Assistant Professor of Simbirsk Branch of Moscow State University. 1988–1991: Post-graduate student of Moscow State University. 1984–1988: Programmer in the Research Institute (ex-Leningrad, ex-USSR). Main research interests: heuristical algorithms (genetic, branch-and-bounds etc.); artificial intelligence; finite automata, regular languages, theory of semigroups; mathematical aspects of the object-oriented programming.

E. Kashlakova. Since 1999: Programmer in the Research Institute (Moscow). 1996–1999: Post-graduate student of Simbirsk State University. Main research interests: formal languages theory; theory of semigroups.

Programavimo kalbų gramatinės struktūros kaip paprastos kalbos su skliaustais

Boris Melnikov, Elena Kashlakova

Straipsnyje nagrinėjamos vadinamosios parastos kalbos su skliaustais. Šios kalbos gali būti vartojamos kai kurioms programavimo kalbų gramatinėms struktūroms specifikuoti. Bendruoju atveju šios kalbos yra nepriklausomos nuo konteksto, bet nebūtinai determinuotos. Kitaip tariant, jų negalima apibrėžti determinuotu automatu su dėklu. Parodyta, kad tam tikrai šitokių kalbų klasei kalbų ekvivalentumo problema yra išsprendžiama.