Cybernetics and Systems Analysis, Vol. 43, No. 2, 2007

EQUIVALENCE OF DETERMINISTIC PUSHDOWN AUTOMATA REVISITED

V. Yu. Meitus

UDC 510.53

This paper briefly analyzes main ideas underlying the comparison algorithm that made it possible to prove the equivalence of deterministic pushdown automata. An example of using this algorithm is presented. The relationship of this algorithm with other results in this area is shown. Moreover, the decidability of problems associated with some classes of formal grammars is established.

Keywords: *deterministic pushdown automata, equivalence problem, decidability of problems, deterministic CF language.*

INTRODUCTION

Deterministic context-free languages that form a widely used subclass of the class of context-free languages (CF languages) are considered in many works. Various variants of specification of languages used in programming computers, grammars representing these languages, program schemes, and language transducers and recognizers are connected in many respects with deterministic pushdown automata (DPDAs).

One of the most important problems of representation of formal languages is the equivalence problem formulated as the coincidence problem for languages represented in different forms. For an arbitrary CF grammar G, $L(G)$ is the language generated by this grammar. For CF languages, the inclusion problem, i.e., the determination of the inclusion $L(G_1) \subseteq L(G_2)$ for arbitrary CF grammars G_1 and G_2, and the equivalence problem, i.e., the determination of the coincidence of the languages $L(G_1)$ and $L(G_2)$ [1] for arbitrary context-free CF grammars G_1 and G_2, are undecidable.

It is well known that, for deterministic CF languages, the equivalence problem $L(G) = R$ is decidable, where $L(G)$ is an arbitrary deterministic CF language generated by a grammar G and R is a regular set, and, at the same time, the inclusion problem $L(G_1) \subseteq L(G_2)$ is undecidable. At the same time, the equivalence problem for deterministic CF languages or DPDAs that represent them remained open for a long time, as well as the equivalence problems for the grammars introduced by D. Knuth [2], strict deterministic grammars considered in [3], and other grammars generating the class of deterministic CF languages [14].

At the same time, numerous investigations of various subclasses of the class of DPDAs are pursued and positive results are obtained. For example, the decidability of the equivalence problem is proved for simple deterministic languages [4], finite-turn DPDAs [5], real-time DPDAs [6–8], $LL(k)$-grammars [9], and deterministic one-counter automata [10].

In 1992, a full-scale proof of decidability of the equivalence problem for DPDAs was presented in [11]. The developed comparison algorithm for checking the equivalence of DPDAs made it possible to solve in a finite number of steps whether the languages recognized by two DPDAs coincide or there is a word in which they differ. In 1997, the same result was announced by G. Senizergues [12] who presented his proof of decidability of this problem [13] in 2001 (but did not mention that the solution was first published in 1992).

In the present article, a brief analysis of main ideas underlying the construction of the algorithm developed in [11] for solution of the equivalence problem for DPDAs is given, and its connection with the work [13] of G. Senizergues that is devoted to the same problem is established. Moreover, a number of decidable problems are formulated that are related to concrete classes of grammars considered in [14].

International Scientific-Educational Center of Information Technologies and Systems, Kiev, Ukraine, *vmeitus@i.um.ua.* Translated from Kibernetika i Sistemnyi Analiz, No. 2, pp. 24-39, March–April 2007. Original article submitted April 18, 2006.

1. DPDAs AND SYSTEMS OF EQUATIONS

A deterministic pushdown automaton is understood to be the following ordered collection consisting of seven symbols: $\mathcal{A} = \{S, \Sigma, \Gamma, \delta, q_0, Z_0, q_F\}$, where S is a finite set of states of the automaton, Σ is its input (terminal) alphabet containing the symbol ε of the empty word by assumption, Γ is the alphabet of stack symbols, δ is the transition function of the automaton, q_0 is its initial state, Z_0 is the initial stack symbol, and q_F is the final state of the automaton. Hereafter, a pair (automaton state, top stack symbol) is called a configuration of the DPDA.

A function

$$\delta : \Sigma \times S \times \Gamma \to S \times \Gamma^* \tag{1}$$

assigns no more than one element from the set $S \times \Gamma^*$ to each collection $(a, p, A) \in \Sigma \times S \times \Gamma$, and the fact that $\delta(\varepsilon, p, A) \neq \varnothing$ implies the equality $\delta(a, p, A) = \varnothing$ for all $a \in \Sigma$, $a \neq \varepsilon$. If, for some $a \in \Sigma$, p, $q \in S$, $A \in \Gamma$, and $\gamma \in \Gamma^*$, the equality $\delta(a, p, A) = (q, \gamma)$ is true, then we say that a rule of the automaton $\mathcal{A}$ is specified according to which the automaton in a state p receives an input symbol a and the stack symbol A, passes to a state q, and replaces the symbol A by the string γ.

If the stack string γ is empty, $\gamma = \varepsilon$, then the symbol A is erased from the stack. If we have $\gamma \neq \varepsilon$, then, without any loss of generality, we can assume that $\gamma = A$ and, in this case, the automaton passes to a new state or that $\gamma = A\gamma'$ and, in this case, the automaton pushes the string γ' onto the stack over the symbol A.

Such a deterministic pushdown automaton is called normalized if the length of the string γ' does not exceed unity, i.e., only one new symbol can be pushed onto the stack over the symbol A. It is well known that any DPDA can be normalized.

A word $x = a_0 \ldots a_n \in \Sigma^*$ is acceptable by the automaton $\mathcal{A}$ if there is a sequence of states $q_0, \ldots, q_n, q_F$ that determines a sequence of rules $\delta(a_i, q_i, A_i) = (q_{i+1}, \gamma_i)\,(0 \le i \le n)$, $A_0 = Z_0$, $q_{n+1} = q_F$, $\gamma_n = \varepsilon$ according to which the automaton passes from the configuration (q_0, Z_0) to the configuration (q_F, ε) (at the final state the stack is empty) and each transition is determined by the function δ. We denote the acceptability relation by the symbol $\Rightarrow$.

The language $L_{\mathcal{A}}$ acceptable by the DPDA $\mathcal{A}$ is defined as the set of words in the alphabet Σ that are accepted by the automaton $\mathcal{A}$,

$$L_{\mathcal{A}} = \{x | x \in \Sigma^*,\ (x, q_0, Z_0) \Rightarrow (q_F, \varepsilon)\}. \tag{2}$$

Let two DPDAs $\mathcal{A}$ and $\mathcal{B}$ be specified over the same alphabet Σ. We say that the automata $\mathcal{A}$ and $\mathcal{B}$ are equivalent if the languages acceptable by them coincide as sets of words.

Using the rules of the automaton $\mathcal{A}$, one can construct a system of equations $E_{\mathcal{A}}$ whose minimal solution is represented by a finite collection of sets of words in the terminal alphabet of the automaton and each set corresponds to a variable of the system. One of sets of this collection coincides with the language $L_{\mathcal{A}}$ acceptable by the automaton $\mathcal{A}$. A possible algorithm of construction of such a system $E_{\mathcal{A}}$ for a DPDA is formulated in [8].

Let us consider a scheme of construction of the system $E_{\mathcal{A}}$. For the simplicity of its representation, we assume that the DPDA is normalized. Variables of the system are of the form X_{qb}^{pA}, where q and p are states of the automaton $\mathcal{A}$, $b \in \Sigma$, and A is a stack symbol.

We denote by V^{pQ} the set of all rules of the automaton $\mathcal{A}$ that are of the form

$$V^{pQ} = \{(a, p, Q) \to (q, Q) | \delta(a, p, Q) = (q, Q), a \in \Sigma\}$$

and according to which the automaton passes from one state to another without changing the stack symbol, by

$$U^{pQ} = \{(a, p, Q) \to (q, QR) | \delta(a, p, Q) = (q, QR), a \in \Sigma\}$$

the set of rules of the automaton that push a symbol R onto the stack during the transition from one state to another, and by

$$W^{Q} = \{(a, p, Q) \to (q, \varepsilon) | \delta(a, p, Q) = (q, \varepsilon), a \in \Sigma\}$$

the set of rules of the automaton according to which the top stack symbol Q is erased under the action of an input symbol during the transition from a state p to a state q. We recall that $\varepsilon \in \Sigma$. In what follows, the symbol $[A]$ denotes

$pA \xrightarrow{a} q\varepsilon \quad [A]=(q,a) \quad \mu[A]=q$

the pair (a state at which the symbol A is erased from the stack; a terminal symbol that erases A) and the symbol $\mu[A]$ denotes the new state to which the automaton passes.

The general form of the equation of the system $E_{\mathcal{A}}$ for an arbitrary variable X^{qA}_{pb} is as follows:

$$X^{qA}_{pb} = b\delta_{qp} \vee \bigvee_{\substack{\eta\in V^{qA}\\ \eta:(a,q,A)\to(t,A)}} aX^{tA}_{pb}$$

?

$$\vee \bigvee_{\substack{\theta\in U^{qA}\\ \theta:(a,q,A)\to(p,AB)}} \bigvee_{\substack{\xi\in W^{B}\\ \xi:([B],B)\to(\mu[B],\varepsilon)}} aX^{qB}_{[B]}\, X^{\mu[B]A}_{pb}. \qquad (3)$$

If the equation is written for the first variable of the system $X^{q_0Z_0}_{q_F}$, then one more disjunctive term of the form $\vee_{\eta:(a,q_0,Z_0)\to(q_F,\varepsilon)}\, a$ can be in it for the rules determining the direct transition from the initial state to the final state with erasing the stack symbol Z_0.

$qA \xrightarrow{*} \xrightarrow{b} p\varepsilon$

A set of words that switch the DPDA $\mathcal{A}$ from the state q with the top stack symbol A to the state p is associated with each variable X^{qA}_{pb} of the system $E_{\mathcal{A}}$, and the symbol A is not once erased from the stack. At the last step, the DPDA is in the state p and reads the input symbol b. We denote this set of words by the symbol $\hat{X}^{qA}_{pb}$. Then it is obvious that we have $\hat{X}^{q_0Z_0}_{q_F} = L_{\mathcal{A}}$ since, by definition, the words that switch the automaton from the state q_0 with the initial stack symbol Z_0 to the state q_F at which Z_0 is erased are words of the language acceptable by the DPDA.

The construction of the system for the DPDA begins with writing the equation for the variable $X^{q_0Z_0}_{q_F}$. Then similar equations are written for all the variables of the right side of the first equation. This process is repeated for the second and subsequent equations. By virtue of the finiteness of the number of possible variables, it comes to an end after a finite number of steps.

Note that the expression

$$\bigvee_{\substack{\theta\in U^{qA}\\ \theta:(a,q,A)\to(p,AB)}} \bigvee_{\substack{\xi\in W^{B}\\ \xi:([B],B)\to(\mu[B],\varepsilon)}} aX^{qB}_{[B]}\, X^{\mu[B]A}_{pb}$$

$qA \xrightarrow{a} pAB$

$*B \xrightarrow{*} *\varepsilon$

can be briefly written in the form

$$\bigvee_{\substack{\theta\in U^{qA}\\ \theta:(a,q,A)\to(p,AB)}} a\mathfrak{U}^{qAB}_{pb}$$

without writing it in terms of the corresponding variables $X \in \overline{X}$.

For any variable $X \in \overline{X}$, the set of words $\hat{X}$ can be represented in the form of a series over a Boolean semiring including two elements 0 and 1. The coefficients of the words that belong to $\hat{X}$ are unities and those of the other words are zeros [13]. The relation between such series and formal languages was investigated in [15].

A language $L \subseteq \Sigma^*$ possesses the following **prefix** property: $x \in L$ implies that there is not any nonempty word $y \in \Sigma^*$ such that the word $xy \in L$. Languages representable by DPDAs possess this property. Moreover, any language $\hat{X}^{qA}_{pb}$ associated with the variable X^{qA}_{pb} entering into system (3) also possesses it [3, 8]. The prefix property is one of the most important properties used in the algorithm of comparison of DPDAs.

2. DPDA COMPARISON ALGORITHM

The proof of the equivalence problem for DPDAs $\mathcal{A}$ and $\mathcal{B}$ (or for deterministic CF languages acceptable by such automata) is reduced to the proof of the recursiveness of some relation R over the set of collections determined by the automata $\mathcal{A}$ and $\mathcal{B}$ and operations of a special procedure Π. Moreover, it has been possible to construct an algorithm called the comparison algorithm that makes it possible to solve the equivalence problem in a finite number of steps, i.e., to construct a finite relation $\mathfrak{R}$ and to show that such automata are equivalent or that a word can be found that is acceptable by one automaton and is not acceptable by the other.

Let $E_{\mathcal{A}}(\overline{X})$ and $E_{\mathcal{B}}(\overline{Y})$ be systems of equations of the form (3) that are constructed from the DPDAs $\mathcal{A}$ and $\mathcal{B}$. The symbols $\overline{X}$ and $\overline{Y}$ denote, respectively, the alphabets of variables entering into these systems. The first element of the relation $\mathfrak{R}$ is specified by the pair $(X_{q_F}^{q_0 Z_0}, Y_{q_F}^{q_0 Z_0})$, where, by definition, q_0 and q_F are the initial and final states of the automata $\mathcal{A}$ and $\mathcal{B}$ and Z_0 is the initial stack symbol of these automata that is erased at the state q_F.

We associate the following collection with an arbitrary pair $(\mathfrak{U}, \mathfrak{B})$ belonging to the relation $\mathfrak{R}$:

$$[n, \mathfrak{U}, \mathfrak{B}, \delta] \tag{4}$$

in which the number n determines the nth level of application of the algorithm with the help of which the relation $\mathfrak{R}$ is constructed. The element δ has five values 0, 1, 2, 3, and 4 and is equal to zero if the pair $\mathfrak{U}, \mathfrak{B}$ is unlabeled, to unity if the pair is labeled and the algorithm is not applied to it any more, to two if both of elements of the pair are equal as words in the terminal alphabet, to three if the labeled pair has occurred at previous levels of application of the algorithm, and to four if the pair consists of unequal words in the terminal alphabet Σ.

According to this agreement, to the pair $\mathfrak{U}, \mathfrak{B}$ corresponds collection (4) that belongs to $\mathfrak{R}_V$, where the symbol $\mathfrak{R}_V$ denotes the set of collections generated by the comparison algorithm. Then the first collection of the relation $\mathfrak{R}_V$ is the collection

$$[0, X_{q_F}^{q_0 Z_0}, Y_{q_F}^{q_0 Z_0}, 0]. \tag{5}$$

Let us consider the operations used for construction of elements of the relation $\mathfrak{R}$. (In [13], such operations are called strategies.) The totality of these operations forms the procedure Π.

Before considering the operations of the procedure Π, we introduce the concept of the **representative** $\langle X \rangle$ of an arbitrary variable X and define it as the shortest word belonging to the language $\hat{X}$. If there are several such words in the language, we choose one of them. The concept of a representative is easily extended to products of variables or to disjunctive polynomials in variables.

Let collection (4) be given for which we have $\delta = 0$. In what follows, the operations of the procedure Π are applied only to such collections.

The first operation is **replacement**. Let $\mathfrak{U} = \vee X_i \mathfrak{U}_i$, let $\mathfrak{B} = \vee Y_j \mathfrak{B}_j$, and let $(n, \mathfrak{U}, \mathfrak{B}, 0) \in \mathfrak{R}_V$. We replace each variable X_i by the right side of the equation for this variable from the system $E_{\mathcal{A}}$ and each variable Y_j by the right side of the equation for this variable from the system $E_{\mathcal{B}}$. We determine the following new collection:

$$\left[n+1, \bigvee_{h=1}^{k} a_h X_h \mathfrak{U}_h, \bigvee_{g=1}^{k} a_g Y_g \mathfrak{B}_g, \delta = 0 \right] \tag{6}$$

and replace $\delta = 0$ by $\delta = 1$ in the initial collection entering into $\mathfrak{R}_V$.

The second operation is **complexation**. For each a_h $(1 \le h \le k)$ entering into collection (6), we determine the following new collection:

$$[n+2, a_h X_h \mathfrak{U}_h, a_h Y_h \mathfrak{B}_h, \delta = 0] \tag{7}$$

and replace $\delta = 0$ by $\delta = 1$ in collection (6) entering into $\mathfrak{R}_V$. Thus, we have obtained k new collections. It is this transformation which is called the complexation operation.

If all the disjunctive terms in the expression $\vee_{g=1}^{k} a_g Y_g \mathfrak{B}_g$ begin with nonempty terminal symbols and some h can be found such that there exists an expression $a_h X_h \mathfrak{U}_h$ and there is no expression beginning with the symbol a_h in $\vee_{g=1}^{k} a_g Y_g \mathfrak{B}_g$, then, based on the string from the first collection $[0, X_{q_F}^{q_0 Z_0}, Y_{q_F}^{q_0 Z_0}, 1]$ to collection (7), a word x can be constructed that makes it possible to pass from the configuration (q_0, Z_0) to the configuration associated with the expression $a_{i_0} X_{i_0} \mathfrak{U}_{i_0}$. At the same time, such a word is absent for the second automaton and, in this case, the automata are nonequivalent, or this discrepancy has arisen during the application of the comparison algorithm. This situation is discussed below in considering the concept of correctness of the operations of the procedure Π.

The third operation is **decomposition**. Assume that, in the set $\mathfrak{R}_V$, there is the pair of collections

$$[m, \mathfrak{U}, \mathfrak{B}, \delta],\ [n, \mathfrak{U}\mathfrak{U}', \mathfrak{B}\mathfrak{B}', 0] \tag{8}$$

or

$$[m, \mathfrak{U}, \mathfrak{B}, \delta],\ [n, \mathfrak{U}'\mathfrak{U}, \mathfrak{B}'\mathfrak{B}, 0]. \tag{9}$$

Then, for collections (8) or (9), we determine the new collection $[n+1, \mathfrak{U}', \mathfrak{B}', 0]$ and replace $\delta = 0$ by $\delta = 1$ in the second collection of pair (8).

A special case of decomposition is the passage from collection (7) to the pair of collections

$$[n+3, X_h\mathfrak{U}_h, Y_h\mathfrak{B}_h, \delta = 0],\ [n+3, a_h, a_h, 2] \tag{10}$$

and replacement of $\delta = 0$ by $\delta = 1$ in collection (7).

The operation inverse to decomposition is the fourth operation called **composition** that, based on the pair of collections

$$[m, \mathfrak{U}, \mathfrak{B}, \delta],\ [n, \mathfrak{U}', \mathfrak{B}', 0]\ (n \geq m), \tag{11}$$

constructs one collection

$$[n+1, \mathfrak{U}'\mathfrak{U}, \mathfrak{B}'\mathfrak{B}, 0] \tag{12}$$

or

$$[n+1, \mathfrak{U}\mathfrak{U}', \mathfrak{B}\mathfrak{B}', 0]. \tag{13}$$

The fifth operation is **splitting** that, based on the pair of collections

$$[m, \mathfrak{U}, \mathfrak{B}, \delta],\ [n, \mathfrak{U} \vee \mathfrak{U}', \mathfrak{B} \vee \mathfrak{B}', 0]\ (n \geq m), \tag{14}$$

determines the new collection $[n+1, \mathfrak{U}', \mathfrak{B}', 0]$ and replaces $\delta = 0$ by $\delta = 1$ in the second collection of pair (14).

The sixth operation is **substitution**. Let the following collection be given:

$$\left[n, \bigvee_{p=1}^{m} X_p\mathfrak{U}_p, \mathfrak{B}_s^{r\xi} \right] \tag{15}$$

that belongs to the set $\mathfrak{R}_V$. In the expression $\mathfrak{B}_s^{r\xi}$ entering into collection (15), the symbol ξ denotes a string of stack symbols and is called the characteristic of the expression $\mathfrak{B}_s^{r\xi}$.

For each variable X_p, we choose its representative $x_p = \langle X_p \rangle$ and substitute it into each component of the pair

$$\left[\bigvee_{p=1}^{m} X_p\mathfrak{U}_p, \mathfrak{B}_s^{r\xi} \right].$$

To substitute a word into the first expression of a pair means to substitute it into the DPDA that is in the state determined by the variable X_p with the corresponding stack symbol. Since x_p belongs only to $\hat{X}_p$ and does not belong to other languages $\hat{X}_t$ $(t \in [1, m], t \neq p)$, we obtain the expression $\mathfrak{U}_p$ after substitution of x_p. The substitution of a word x into an expression $\mathfrak{U}$ is denoted by the symbol $\int (x|\mathfrak{U})$. Then we have

$$\int (x_p | \bigvee_{p=1}^{m} X_p\mathfrak{U}_p) = \mathfrak{U}_p.$$

If we substitute x_p into the expression $\mathfrak{B}_s^{r\xi}$, then, in the general case, the automaton determining $\mathfrak{B}_s^{r\xi}$ first passes to the state r_p with erasing a part of the string ξ from the stack (only the initial part of the string $\xi - \xi_p$ remains) and then pushes a new string η_p onto the stack, i.e., we have $\int (x_p | \mathfrak{B}_s^{r\xi}) = \mathfrak{B}_s^{r_p\xi_p\eta_p}$. Thus, after substitution of all x_p $(1 \leq p \leq m)$ into the expressions entering into collection (15), we obtain a system consisting of m collections

$$[n+1, \mathfrak{U}_p, \mathfrak{B}_s^{r_p\xi_p\eta_p}, 0] \tag{16}$$

that supplement the set $\mathfrak{R}_V$. This system is called the direct system for the pair $(\bigvee_{p=1}^{m} X_p \mathfrak{U}_p, \mathfrak{B}_s^{r\xi})$ belonging to collection (15).

The construction of a direct system is only the first suboperation of the substitution operation, and its second suboperation is the construction of the inverse system. We consider a sequence of strings

$$\xi_1, \xi_2, \ldots, \xi_m \tag{17}$$

that belong to direct system (16), and assume that we have $t = \min|\xi_p|$, i.e., t is the length of the minimum string of set (17) in which the last symbol is a stack symbol, for example, A. Then each string belonging to set (17) can be represented in the form $\xi_p = \zeta A \zeta_p$ $(1 \le p \le m)$, where the length $|\zeta A| = t$.

We select the pairs of expressions that belong to the relation $\mathfrak{R}$ from the collections determining the direct system,

$$(\mathfrak{U}_p, \mathfrak{B}_s^{r_p \xi_p \eta_p}) = (\mathfrak{U}_p, \mathfrak{B}_s^{r_p \zeta A \zeta_p \eta_p})\ (1 \le p \le m). \tag{18}$$

(In [8], such pairs are called conditional equalities.) We multiply each pair from (18) by the variable X_p and construct their disjunctive union. Then we obtain

$$\left(\bigvee_{p=1}^{m} X_p \mathfrak{U}_p,\ \bigvee_{p=1}^{m} X_p \mathfrak{B}_s^{r_p \zeta A \zeta_p \eta_p} \right) = \left(\bigvee_{p=1}^{m} X_p \mathfrak{U}_p,\ \bigvee_{p=1}^{m} \bigvee_{[A]} X_p \mathfrak{B}_{[A]}^{r_p A \zeta_p \eta_p} \mathfrak{B}_s^{\mu[A]\zeta} \right). \tag{19}$$

Since, according to collection (15), $\vee_{p=1}^{m} X_p \mathfrak{U}_p$ is assumed to be coincident with $\mathfrak{B}_s^{r\xi}$, we replace it in union (19) by

$$\mathfrak{B}_s^{r\xi} = \bigvee_{[A]} \mathfrak{B}_{[A]}^{rA\tau} \mathfrak{B}_s^{\mu[A]\zeta} \tag{20}$$

because we have $\xi = \zeta A \tau$ since ζA is the initial subsequence of the sequence ξ.

We replace the first element in union (19) according to equality (20) and obtain the following pair:

$$\left(\bigvee_{[A]} \mathfrak{B}_{[A]}^{rA\tau} \mathfrak{B}_s^{\mu[A]\zeta},\ \bigvee_{[A]} \bigvee_{p=1}^{m} X_p \mathfrak{B}_{[A]}^{r_p A \zeta_p \eta_p} \mathfrak{B}_s^{\mu[A]\zeta} \right) \tag{21}$$

that can be transformed into a set consisting of l pairs determined by different symbols $[A]$,

$$\left(\mathfrak{B}_{[A]}^{rA\tau},\ \bigvee_{p=1}^{m} X_p \mathfrak{B}_{[A]}^{r_p A \zeta_p \eta_p} \right), \tag{22}$$

and each of these pairs determines a new collection

$$\left[n+1,\ \bigvee_{p=1}^{m} X_p \mathfrak{B}_{[A]}^{r_p A \zeta_p \eta_p},\ \mathfrak{B}_{[A]}^{rA\tau}, 0 \right]. \tag{23}$$

Collections (23) specify the system determined by the substitution operation and inverse to collection (19). We add collections (23) to $\mathfrak{R}_V$ and, in collection (19) that has been already added to $\mathfrak{R}_V$, replace $\delta = 0$ by $\delta = 1$ after execution of the substitution operation.

The inverse system (23) is determined under the condition that all $\mathfrak{B}_s^{\mu[A]\zeta}$ are different from one another, i.e., the intersection of the languages determined by them is empty. Otherwise, this fact can affect the further functioning of the comparison algorithm, which requires its elaboration (see Sec. 5). The definition of the substitution operation completes the specification of the operations of the procedure Π.

3. SPECIAL SYMBOLS AND SEQUENCES

The lengths of the characteristics of the expressions obtained during the execution of operations of the procedure Π can increase in comparison with those of the expressions that do not subject to such operations. (Recall that a characteristic ξ is a string of stack symbols entering into an expression $\mathfrak{u}^{p\xi}_{qa}$.)

The basic idea associated with the proof of decidability of the equivalence problem for DPDAs consists of the proof of impossibility of the infinite increase in the characteristics of the expressions entering into the relation $\mathfrak{R}$. In this case, the relation $\mathfrak{R}$ is finite and the comparison algorithm comes to an end after a finite number of steps.

The basic quantity with respect to which expressions with different characteristics are compared is the norm of an expression. By the norm of a variable X we understand the length of the shortest word belonging to the language $\hat{X}$ and denote it by the symbol $||X||$. The norm of the empty word equals zero. The concept of norm can be extended to disjunctive polynomials as follows:

$$\left\|\bigvee_i X_{i1}\ldots X_{ik_i}\right\| = \min_i ||X_{i1}\ldots X_{ik_i}|| = \min_i \sum_{j=1}^{k_i} ||X_{ij}||.$$

If a letter $a \neq \varepsilon$, then the norm of any variable $X^{p\xi}_{[A]a}$ is more than zero when $\hat{X}^{p\xi}_{[A]a} \neq \varnothing$ since all the words of the language $\hat{X}^{p\xi}_{[A]a}$ terminate in the letter a. Therefore, the norm of an expression $\mathfrak{u}^{p\xi}_{sa}$ is proportional to the length of its characteristic if this expression is distinct from zero and contains variables of the form $X^{pA}_{[A]a}$. For real-time DPDAs that are precisely those that satisfy this condition, the decidability of the equivalence problem is proved [6–8]. In particular, the proof of decidability in [8] is based on the analysis of the lengths of the characteristics of the expressions being considered. However, for arbitrary DPDAs, stack symbols can exist that are erased by the empty word. For example, if the rule $\delta(\varepsilon, p, A) = (p, \varepsilon)$ exists, then the representative of the variable $X^{pA}_{p\varepsilon}$ equals ε. We call such stack symbols special symbols. Then we have $||X^{pA}_{p\varepsilon}|| = 0$ and call a sequence consisting of special symbols a special sequence. Therefore, the norm of an expression whose characteristic includes special symbols is not proportional to the length of the characteristic.

It can be shown that the assumption of the possibility of unbounded lengthening of a special sequence is finally reduced to the assumption of the existence of a collection

$$[m, X, \mathfrak{u}^{r\theta\eta\theta'}_{\bullet}, 0], \tag{24}$$

where η is a special sequence that is transformed into the following new collection after the application of operations of the procedure Π:

$$[m', X, \mathfrak{u}^{r\theta\eta\eta_1\theta'}_{\bullet}, 0] \tag{25}$$

whose special sequence $\eta\eta_1$ is obtained by lengthening η owing to the addition of a new special sequence η_1 to it. Under the conditions of this assumption, the next cycle of operations could lead to the lengthening of the special sequence and, accordingly, the characteristic of the expression entering into the collection.

In actual fact, this is not the case. It follows from the definition of collections (24) and (25) that one can construct the following collection:

$$[m'', \mathfrak{u}^{r\theta\eta\theta'}_{\bullet}, \mathfrak{u}^{r\theta\eta\eta_1\theta'}_{\bullet}, 0] \tag{26}$$

by virtue of the coincidence of the second components in collections (24) and (25) and, applying the substitution operation, can pass from collection (26) to the collection

$$[m''+2, \mathfrak{u}^{s\eta}_{\cdot}, \mathfrak{u}^{s\eta\eta_1}_{\cdot}, 0] \tag{27}$$

whose characteristic includes special sequences. Hereafter, it suffices to consider only this collection without considering longer special sequences for this case.

Since the number of expressions of the form (24) that can be initial expressions for the formation of long special sequences is finite, the largest possible length of such sequences that can arise during the application of the comparison algorithm is also finite.

4. CORRECTNESS AND INCORRECTNESS OF OPERATIONS

Let a collection $[n, \mathfrak{U}, \mathfrak{B}, 0]$ be given, where $(\mathfrak{U}, \mathfrak{B}) \in \mathfrak{R}$. A set of words is associated with each element of this pair, and the comparison algorithm must determine whether the sets $\hat{\mathfrak{U}}$ and $\hat{\mathfrak{B}}$ coincide. The operations of the procedure Π somewhat transform these sets and, instead of checking the sets $\hat{\mathfrak{U}}$ and $\hat{\mathfrak{B}}$ for coincidence, other sets are considered and analyzed.

For example, the complexation operation replaces the analysis of initial sets by the analysis of their disjoint parts. At the same time, an attempt is made to find the shortest word that would belong to one language and do not belong to the other.

Since the comparison algorithm should find out whether the sets $\hat{\mathfrak{U}}$ and $\hat{\mathfrak{B}}$ coincide, it is important that, after the application of any operation, the sets $\hat{\mathfrak{U}}_k$ and $\hat{\mathfrak{B}}_k$ $(1 \le k \le m)$ obtained as a result of the operation satisfy the following conditions: (a) if $\hat{\mathfrak{U}} = \hat{\mathfrak{B}}$, then, for all k, we have $\hat{\mathfrak{U}}_k = \hat{\mathfrak{B}}_k$; (b) if $\hat{\mathfrak{U}} \ne \hat{\mathfrak{B}}$, then, at least for one $k = k_0$, we have $\hat{\mathfrak{U}}_{k_0} \ne \hat{\mathfrak{B}}_{k_0}$. In other words, any operation must not transform unequal sets into equal ones and equal sets into unequal ones. We call such operations correct.

If only the second condition is true for an operation. then this operation is called weakly correct. As a result of a weakly correct operation, a pair of unequal sets can be obtained though the initial sets are equal but a pair of equal sets cannot be obtained from unequal ones. If the algorithm containing weakly correct operations produces coincident pairs of sets, then this means that the initial pair consists of equal sets. (In [8, 11], weakly correct operations are called incorrect.)

As is shown in [8], the operations of replacement, complexation, composition, decomposition, and splitting are correct, whereas the substitution operation is weakly correct.

The weak correctness of the substitution operation is determined by the fact that, during the construction of an inverse system, the independence of expressions $\mathfrak{U}_s^{\mu[A]\zeta}$ is implicitly assumed. For example, the following collection:

$$\left[n, \mathfrak{G}_1 \mathfrak{U}_s^{\mu[A]_1\zeta} \vee \mathfrak{G}_2 \mathfrak{U}_s^{\mu[A]_2\zeta}, \mathfrak{G}_3 \mathfrak{U}_s^{\mu[A]_1\zeta} \vee \mathfrak{G}_4 \mathfrak{U}_s^{\mu[A]_2\zeta}, 0\right] \tag{28}$$

specifies the pair of collections $[n+1, \mathfrak{G}_1, \mathfrak{G}_3, 0]$, $[n+1, \mathfrak{G}_2, \mathfrak{G}_4, 0]$ of the inverse system. Assuming that there exists a dependence between $\mathfrak{U}_s^{\mu[A]_1\zeta}$ and $\hat{\mathfrak{U}}_s^{\mu[A\}_2\zeta}$ or their coincidence as a special case, we can obtain only one collection $[n+1, \mathfrak{G}_1 \vee \mathfrak{G}_2, \mathfrak{G}_3 \vee \mathfrak{G}_4, 0]$ from collection (28). In this case, one can easily construct an example in which we have $\mathfrak{G}_1 \vee \mathfrak{G}_2 = \mathfrak{G}_3 \vee \mathfrak{G}_4$ and, at the same time, $\mathfrak{G}_1 \ne \mathfrak{G}_3, \mathfrak{G}_2 \ne \mathfrak{G}_4$.

5. MAIN RESULTS

The algorithm of comparison of two DPDAs consists of sequential formation of labeled collections of the following form with the help of operations of the procedure Π:

$$[n, \mathfrak{U}, \mathfrak{B}, 0], \tag{29}$$

beginning with initial collection (5). This totality of collections belonging to the relation $\mathfrak{R}_V$ can be considered as a tree whose root is collection (5).

If an operation α of the procedure Π is applied to collection (29), then the label $\delta = 0$ in collection (29) is replaced by the label $\delta = 1$ and new collections formed as a result of execution of the operation α are labeled by $\delta = 0$ and increase the first component that becomes equal to $n+1$. Operations of the procedure Π are not applied to the collections labeled by $\delta > 0$.

Then all new collections $[n+1, \mathfrak{U}_i, \mathfrak{B}_i, 0]$ $(1 \le i \le m)$ are checked according to the following rules:

(1) if a collection $[n+1, \mathfrak{U}_i, \mathfrak{B}_i, 0]$ has already occurred earlier in the form $[k, \mathfrak{U}_i, \mathfrak{B}_i, 1]$, then it is replaced by the collection $[n+1, \mathfrak{U}_i, \mathfrak{B}_i, 3]$;

(2) if the components of a collection $\mathfrak{U}_i, \mathfrak{B}_i$ are represented by a word of the alphabet Σ and these words coincide, then we have $\delta = 2$;

(3) if these words do not coincide, then we have $\delta = 4$.

The comparison algorithm functions until there are collections with labels $\delta = 0$ or a collection with the label $\delta = 4$ is obtained.

In the second case, in traversing from a leaf $[n+1, x_0, y_0, 4]$ to the root of the tree, the pair of words $x = x' x_0 x''$, $y = y' y_0 y''$ can be constructed that must belong to $\hat{X}_{qF}^{q_0 Z_0}$ and $\hat{Y}_{qF}^{q_0 Z_0}$, respectively, by construction but are different. We substitute the obtained word x into $\mathcal{B}$ and word y into $\mathcal{A}$. If the word x is not accepted by the automaton $\mathcal{B}$ or y is not accepted by $\mathcal{A}$, then the automata are nonequivalent and a word is efficiently found in which they differ.

It is not unlikely that the automata accept both words and that a collection with the label $\delta = 4$ has occurred because of the weak correctness of the substitution operation applied at a previous step. In this case, as is shown in [8], the found word can be used for the direct determination of this dependence. Then the whole branch of the tree constructed from the moment of the application of the substitution operation to the moment of obtaining the collection $[n+1, x_0, y_0, 4]$ is eliminated, and the comparison algorithm is performed again from this point but already with collections determined by the established dependence.

We note that the most essential fact connected with this rollback is that the possible number of such rollbacks for each substitution operation is finite, and since a finite number of substitutions is used, their total number is also finite.

It is obvious that, in the tree of collections being constructed according to the relation $\mathfrak{R}_V$, each collection with a label $\delta > 0$ is a leaf of the tree. One can assume that, during the functioning of the algorithm, the tree of collections can potentially increase infinitely owing to the growth in the lengths of the characteristics of the expressions being compared.

We first note that, for any collection of the form (29), the norms of its components coincide. Otherwise, a word would exist that belongs to one associated language but does not belong to the other.

The length of the characteristic of expressions being compared is also associated with norm since the length of possible special sequences is bounded. Therefore, there is a numerical interval whose bounds cannot be exceeded by the length of any characteristic obtained during the functioning of the comparison algorithm. The reason is that the substitution operation for this interval always decreases the characteristic and, in the direct system, the norm has decreased by at least unity. When the characteristic of an expression is sufficiently long, a substitution changes only its small part. In constructing the inverse system, the characteristic decreases by the length of the sequence ζ entering into the expression $\mathfrak{B}_s^{\mu[A]\zeta}$. Therefore, as is shown in [11], the length of characteristics of all expressions occurring during the application of the comparison algorithm is bounded from above by the boundary of the chosen interval.

The finiteness of lengths of characteristics implies the finiteness of the total number of collections considered during the functioning of the comparison algorithm and, hence, at a step, all collections are labeled by numbers from unity to three and the algorithm comes to an end or a word is found that proves the nonequivalence of the automata being compared.

This result is formulated in the form of the theorem that is presented below and that was first proved in [11].

THEOREM 1. The equivalence problem is decidable for deterministic pushdown automata.

It follows from the definition of the comparison algorithm that the equivalence problem is solved for any pairs of expressions even if they are determined by one DPDA $\mathcal{A}$.

Let a word x switch the automaton $\mathcal{A}$ from the state q_0 with the stack symbol Z_0 to some state q and simultaneously pushes a string $\xi_0 A\xi$ onto the stack. Then an expression $\mathfrak{U}_{qF}^{q\xi_0 A\xi}$ associated with the set of words $\hat{\mathfrak{U}}_{qF}^{q\xi_0 A\xi}$ can be constructed and, in this case, the language $x\hat{\mathfrak{U}}_{qF}^{q\xi_0 A\xi} \subseteq L_{\mathcal{A}}$. The following equality is true:

$$\mathfrak{U}_{qF}^{q\xi_0 A\xi} = \bigvee_{[A]} \mathfrak{U}_{[A]}^{q_1 A\xi}\, \mathfrak{U}_{qF}^{\mu[A] Z_0 \xi_0}$$

in which the union is taken over all possible pairs $[A]$ in $\mathcal{A}$. Let us consider the expression $\mathfrak{U}_{[A]}^{q_1 A\xi}$. We assume that we have $[A] = (t, a)$. Then we have the expression $\mathfrak{U}_{t,a}^{q_1 A\xi} = \mathfrak{U}$. The language $\hat{\mathfrak{U}}$ is associated with it. We say that the expression $\mathfrak{U}$ is definable by the DPDA $\mathcal{A}$.

Similarly, assume that we can determine the expression $\mathfrak{B}_{sa}^{pB\eta} = \mathfrak{B}$ associated with the language $\hat{\mathfrak{B}}$ by substituting a word y that switches the automaton $\mathcal{A}$ to a state p with the stack string $Z_0\eta_0 B\eta$ into this automaton. At the state s, the symbol a erases the stack symbol B.

We consider that expressions $\mathfrak{U}$ and $\mathfrak{B}$ are equivalent if we have $\hat{\mathfrak{U}} = \hat{\mathfrak{B}}$. Then, with the help of the comparison algorithm, the equivalence of the expressions $\mathfrak{U}$ and $\mathfrak{B}$ can be determined since, otherwise, two DPDAs can be constructed for which the equivalence problem is undecidable, contrary to Theorem 1. Thus, the theorem formulated below is true.

THEOREM 2. The equivalence problem for expressions $\mathfrak{U}$ and $\mathfrak{B}$ definable by an arbitrary DPDA is decidable.

6. AN EXAMPLE OF USING THE COMPARISON ALGORITHM

Let us consider an example of application of the proposed comparison algorithm to the analysis of the equivalence problem for two DPDAs $\mathcal{A}$ and $\mathcal{B}$ each of which recognizes the following CF language:

$$L = \{a^n b^k a^n \mid k, n \geq 1\} \cup \{a^k b^n c^n \mid k,\ n \geq 1\}.$$

Each automaton specifies its system of equations ($E_{\mathcal{A}}$ and $E_{\mathcal{B}}$). The first system is as follows:

$$X_0 = aX_1 \vee aX_2,\ X_1 = aX_1 a \vee bX_3,\ X_2 = aX_2 \vee bX_4,\ X_3 = bX_3 \vee a,\ X_4 = bX_4 c \vee c \tag{30}$$

(Example 2.1 from [3]).

The second system $E_{\mathcal{B}}$ is constructed using of the equations specified by equalities (3). To simplify the notations, all the indices of the variables are eliminated and the variables are numbered. Moreover, if a variable equals a terminal symbol, then it is replaced by this symbol in the corresponding equation to decrease the total number of equations of the system,

$$Y_0 = aY_1 \vee aY_2 \vee aY_5,\ Y_1 = bY_3 \vee ba,\ Y_2 = aY_2 \vee bY_4 \vee bc,$$

$$Y_3 = bY_3 \vee ba,\ Y_4 = bY_4 c \vee bcc,\ Y_5 = aY_1 a \vee aY_5 a. \tag{31}$$

The root of the tree determined by the relation $\mathfrak{R}_V$ is specified by the collection

$$[0, X_0, Y_0, 0]. \tag{32}$$

Executing the replacement operation over collection (32), we obtain

$$[q, a(X_1 \vee X_2),\ a(Y_1 \vee Y_2 \vee Y_5), 0] \tag{33}$$

and replace $\delta = 0$ by unity in collection (32). Now, collection (32) is of the form $[0, X_0, Y_0, 1]$ in $\mathfrak{R}_V$.

At the following step of the algorithm, the decomposition operation is applied to collection (33) and reduces it by a from the left. We obtain the following two new collections of the second level:

$$[2, X_1 \vee X_2, Y_1 \vee Y_2 \vee X_5, 0] \tag{34}$$

$$[2, a, a, 2], \tag{35}$$

and replace $\delta = 0$ by $\delta = 1$ in collection (33).

Collection (34) is the unique collection with the label $\delta = 0$ in $\mathfrak{R}_V$. We apply the operation of replacement of variables to it and simultaneously replace $\delta = 0$ by $\delta = 1$. From collection (34), we obtain the following collection of the third level:

$$[3, aX_1 a \vee bX_3 \vee aX_2 \vee bX_4,\ bY_3 \vee ba \vee aY_2 \vee bY_4 \vee bc \vee aY_1 a \vee aY_5 a, 0]. \tag{36}$$

Applying the complexation operation to collection (36) and labeling collection (36) as a collection that has been already used ($\delta = 1$), we obtain

$$[4, a(X_1 a \vee X_2), a(Y_2 \vee Y_1 a \vee Y_5 a), 0], \tag{37}$$

$$[4, b(X_3 \vee X_4), b(Y_3 \vee a \vee Y_4 \vee c), 0]. \tag{38}$$

Thus, two new branches that must be individually analyzed have appeared in the tree determined by the relation $\mathfrak{R}_V$. Let us first consider the first branch. We apply the decomposition operation to collection (37) and reduce the components of the collection by a from the left. Hereafter, we will not write the collections of letters of the alphabet Σ for which we have $\delta = 3$ since their occurrence influences not the execution of operations of the procedure Π but only the representation completeness.

As a result of decomposition, we obtain the collection

$$[5, X_1 a \vee X_2, Y_2 \vee Y_1 a \vee Y_5 a, 0]. \tag{39}$$

We apply the substitution operation to collection (39) and substitute $\langle X_1 \rangle = ba.$ Then we obtain the direct system consisting of one collection $[6, a, a, 2]$ labeled by $\delta = 2$ and the inverse system

$$[6, X_1, Y_1 \vee Y_5, 0]. \tag{40}$$

Applying the composition operations to collections (40) and (35), i.e., multiplying the components of system (40) by the symbol a from the right, we obtain the collection

$$[7, X_1 a, Y_1 a \vee Y_5 a, 0]. \tag{41}$$

Applying the splitting operation to the pair of collections (40), (41), we obtain the collection

$$[8, X_2, Y_2, 0]. \tag{42}$$

Now, collections (40) and (42) replace collection (39) that is labeled as an already used collection ($\delta = 1$). Thus, this branch of the tree is also subdivided into two new branches outgoing from collections (40) and (42).

We apply the comparison algorithm separately to each branch. Let us first consider collection (40). After the replacement operation, we obtain the collection

$$[7, aX_1 a \vee bX_3, a(Y_1 a \vee Y_5 a) \vee b(Y_3 \vee a), 0] \tag{43}$$

and apply the operations of complexation by the first letter to it. We obtain two collections and reduce them by, respectively, a and b from the left (decomposition),

$$[8, X_1 a, Y_1 a \vee Y_5 a, 3], \tag{44}$$

$$[8, X_3, Y_3 \vee a, 0]. \tag{45}$$

We have $\delta = 3$ in collection (44) since it has already occurred at previous steps (collection (41)). In collection (45), we make a replacement and apply the splitting operation to its result. We obtain the collections

$$[9, a, a, 2],$$

$$[9, bX_3, b(Y_3 \vee a), 0]. \tag{46}$$

We decompose collection (46) by selecting the symbol b from the left. As a result, we obtain a collection for which we have $\delta = 3$ since it has already occurred in the form (45). This reasoning completes the consideration of the branch of the tree outgoing from system (40). The labels of all the collections of this branch equal $\delta > 0$.

Let us now consider the tree branch determined by collection (42). We apply the replacement operation to collection (42), obtain the collection

$$[9, aX_2 \vee bX_4, aY_2 \vee b(Y_4 \vee c), 0], \tag{47}$$

and apply the complexation operation to this collection. As a result, we obtain the following collections:

$$[10, aX_2, aY_2, 0], \tag{48}$$

$$[10, bX_4, b(Y_4 \vee c), 0]. \tag{49}$$

After reducing collection (48) by the symbol a from the left, it is transformed into the collection $[11, X_2, Y_2, 3]$ coincident with collection (42) and, after reducing collection (49) by the symbol b, it is transformed into the collection

$$[10, X_4, Y_4 \vee c, 0]. \tag{50}$$

Applying the replacement operation to collection (50), we obtain

$$[11, bX_4 c \vee c, bY_4 c \vee bcc \vee c, 0]. \tag{51}$$

If we repeatedly apply splitting operations to collection (51) to separate the word c and then the operation of decomposition from the left and from the right, then we obtain the collection $[12, X_4, Y_4 \vee c, 3]$ that coincides with collection (50). This completes the application of the comparison algorithm to this branch of the tree specified by $\mathfrak{R}_V$.

Now, the unique collection labeled by $\delta = 0$ is collection (38). Reducing it by b from the left, we obtain the collection

$$[5, X_3 \vee X_4, Y_3 \vee a \vee Y_4 \vee c, 0]. \tag{52}$$

After replacement of variables in it, we obtain

$$[6, bX_3 \vee a \vee bX_4c \vee c, bY_3 \vee ba \vee bY_4c \vee bcc \vee a \vee c, 0]. \tag{53}$$

Applying the splitting operation, we eliminate the words *a* and *c* from collection (53) and reduce it by *b* from the left. As a result, we obtain the collection

$$[7, X_3 \vee X_4c, Y_3 \vee a \vee (Y_4 \vee c)c, 0]. \tag{54}$$

Using collection (45) obtained earlier, collection (54) can be split into collection (45) with the label $\delta = 3$ and the collection

$$[8, X_4c, (Y_4 \vee c)c, 0]. \tag{55}$$

Reducing collection (55) by *c* from the right, we obtain a collection with the label $\delta = 3$ as coincident with collection (50) considered earlier.

Since all the collections of the tree specified by $\mathfrak{R}_V$ are labeled by $\delta > 0$, the comparison algorithm comes to an end at this point. Since no collection has been labeled by $\delta = 4$, we can apply Theorem 1 according to which the DPDAs being compared are equivalent and represent the same language.

7. NEW INVESTIGATIONS OF THE EQUIVALENCE PROBLEM

The works of G. Senizergues [12, 13] published in 1997 and 2001 are also devoted to the investigation of the decidability of the equivalence problem for DPDAs. We note distinctive features of [13] since the decidability of the equivalence problem for DPDAs is proved in it along with other results.

1. In [13], the entire presentation is strictly formalized, and each step and passage are more algebraized than those in [8, 11]. This approach allows one to construct a new theoretical approach to the investigation of relations specified for DPDAs rather then to consider individual algorithms of comparison of automata. Therefore, many additional mathematical and applied results related to equivalence problems are obtained in [13].

2. The general approach of [13] consists of analyzing series or trees over a Boolean semiring, which allows one to consider a structure as the whole without considering traversals of such a tree along its branches. The proof of the main result is formulated as the proof of the recursiveness of an equivalence relation for deterministic rational Boolean series (Theorem 87 from [13]) while in [11] only a concrete resolving algorithm is constructed for DPDAs and it is shown that the resolving procedure is primitive recursive. The result on primitive recursiveness was not explicitly formulated in [11] but directly follows from the description of the algorithm.

3. In [13], sets of operations are investigated that are various modifications and variants of a system of operations and generate various deductive systems. In this regard, [13] is devoted to the construction and investigation of some class of algebraic theories in which the equivalence of DPDAs is a partial but important problem.

4. In [13], the decidability of the equivalence problem is shown as a corollary for some program schemes, namely, for monadic recursive schemes (which was shown in [11]) and for polyadic recursive schemes (a new result!). (At the same time, in [13], the fact is not anywhere even mentioned that the result on the decidability of the equivalence problem for DPDAs was originally published five years before in [12].)

8. EQUIVALENCE PROBLEMS FOR PARTICULAR CLASSES OF GRAMMARS

The equivalence problem for DPDAs is closely related to equivalence problems that arise in considering representations of CF languages in the form of various grammatical schemes. To decrease the language processing time, special schemes of grammatical representation are introduced that make it possible to perform syntactic analysis more efficiently than in the general case of a CF language. Various variants of such classes are presented in [14].

The decidability of the equivalence problem is shown below in the form of corollaries of Theorem 1 for several classes of grammars considered in [14, Ch. 5]; the definitions of the grammars themselves are also given in this work. Since each of the classes generates exactly one class of deterministic CF languages, the equivalence problem for any pair of languages from any class is decidable.

COROLLARY 1. The equivalence problem for languages specified by simple precedence grammars or invertible weak-precedence grammars is decidable.

COROLLARY 2. The equivalence problem for languages specified by simple mixed-strategy-precedence grammars is decidable.

COROLLARY 3. The equivalence problem for languages specified by bounded right context (BRC) grammars and (1,1)-BRC grammars is decidable.

COROLLARY 4. The equivalence problem for languages specified by invertible (2,1)-precedence grammars is decidable.

This paper presents the main ideas and methods for proving the decidability of the equivalence problem for DPDAs, an example of using the developed algorithm, and the solution of equivalence problems for several classes of grammars generating languages equivalent to deterministic CF languages as a corollary of the result obtained.

REFERENCES

1. S. Ginsburg, The Mathematical Theory of Context-Free Languages [Russian translation], Mir, Moscow (1970).
2. D. E. Knuth, "On the translation of languages from left to right," Inform. and Control, **8**, 607–639 (1965).
3. M. Harrison and I. Havel, "Strict deterministic grammars," J. Comp. and System Sci., **7**, No. 3, 237–277 (1973).
4. A. J. Korenjac and J. E. Hopcroft, "Simple deterministic languages," in: Proc. 7th Ann. IEEE Switching and Automata Theory Conf., Berkly (1966), pp. 36–46.
5. L. G. Valiant, "The equivalence problem for deterministic finite-turn pushdown automata," Inform. and Control, **25**, 123–133 (1974).
6. M. Oyamaguchi, "The equivalence problem for real-time D.P.D.A's," J. Assoc. Comput. Mach., **34**, 731–760 (1987).
7. V. Yu. Romanovskii, "The equivalence problem for real-time deterministic pushdown automata," Kibernetika, No. 2, 13–23 (1986).
8. V. Yu. Meitus, "The equivalence problem for real-time strict deterministic pushdown automata," Kibernetika, No. 5, 14–25 (1989).
9. D. Rosenkrantz and R. Stearns, "Properties of deterministic topdown grammars," Inform. and Control., **17**, 226–256 (1970).
10. L. Valiant and M. Paterson, "Deterministic one-counter automata," J. Comput. System Sci., **10**, 340–350 (1975).
11. V. Yu. Meitus, "Decidability of the equivalence problem for deterministic pushdown automata," Cybernetics and Systems Analysis, No. 5, 20–45 (1992).
12. G. Senizergues, "The equivalence problem for deterministic pushdown automata is decidable," in: Proc. ICALP 97, Lect. Notes in Comput. Sci., **1256**, 671–681 (1997).
13. G. Senizergues, "L(A) = L(B)? decidability results from complete formal systems," Theoret. Comput. Sci., **251**, 1–166 (2001).
14. A. Aho and J. Ulman, Theory of Parsing, Translation, and Compiling [Russian translation], Vol. 1, Mir, Moscow (1978).
15. W. Kuich, "Semirings and formal power series: Their relevance to formal languages and automata," in: G. Rozenberg and A. Salomaa (Eds.), Handbook of Formal Languages, Springer, Berlin (1996), pp. 609–677.