THE EQUIVALENCE PROBLEM FOR REAL-TIME STRICT DETERMINISTIC PUSHDOWN AUTOMATA

V. Yu. Meitus

UDC 510.53.519.713

A comparison method is described for a pair of real-time strict deterministic pushdown automata, capable of deciding their equivalence.

The complexity of the general equivalence problem of deterministic pushdown automata (DPA) has encouraged attempts to solve the equivalence problem for various subclasses of the DPA class, defined by supplementary conditions and constraints on the member automata. In particular, decidability of the equivalence problem has been proved for two subclasses of real-time DPA (RTDPA) — simple [1] and strict [2, 3], and also for the entire RTDPA class [4].

In this paper, we prove decidability of the equivalence problem for strict RTDPA using a new algebraic method, which constructs and analyzes a sequence of categories from a given pair of RTDPA being compared. This method also may be applied to the general DPA equivalence problem.

1. AUTOMATA AND SYSTEMS OF EQUATIONS

A real-time deterministic pushdown automaton is an ordered tuple of symbols $\mathfrak{A} = (S, \Sigma, \Gamma, \delta, q_0, Z_0, F)$, where S is the finite state set, Σ and Γ respectively are the finite sets of input symbols and pushdown symbols (the input and the pushdown alphabets) of the automaton, $q_0 \in S$ is the starting state, $Z_0 \in \Gamma$ is the starting pushdown symbol, $F \in S$ is the final state. The function $\delta: \Sigma \times S \times \Gamma \to S \times \Gamma^*$ associates to each triple of symbols $(a, \alpha, A) \in \Sigma \times S \times \Gamma$ at most one element from the set $S \times \Gamma^*$, and the empty word ε is not an element of Σ. Each element θ of the graph of the function δ is usually represented in the form θ: $(a, \alpha, A) \to (\beta, \gamma)$; it is called a rule of the automaton $\mathfrak{A}$.

If $\gamma = A\gamma'$, this means that in state α the automaton reads the input symbol a and the pushdown symbol A and goes from state α to state β adding a string γ' after the symbol A in the pushdown; if $\gamma = \varepsilon$, then the automaton goes to state β after erasing the symbol A from the pushdown.

Using the rules of the automaton $\mathfrak{A}$, we can inductively define the computation relation $C_{\mathfrak{A}}$ on the set $\Sigma^* \times S \times \Gamma^*$: 1) $(ax, \alpha, \xi A) \Rightarrow (x, \beta, \xi\gamma) \in C_{\mathfrak{A}}$ if a rule θ: $(a, \alpha, A) \to (\beta, \gamma)$ exists in $\mathfrak{A}$; 2) if $(ax, \delta, \xi) \Rightarrow (ay, \alpha, \zeta A) \in C_{\mathfrak{A}}$ and a rule θ: $(a, \alpha, A) \to (\beta, \gamma)$ exists, then $(x, \delta, \xi) \Rightarrow (y, \beta, \zeta\gamma) \in C_{\mathfrak{A}}$; 3) $C_{\mathfrak{A}}$ contains no other elements.

The language accepted by the RTDPA $\mathfrak{A}$ is the set of those and only those words in the alphabet Σ for which a computation exists starting in the state q_0 with the pushdown symbol Z_0 and ending in state F, i.e., $\mathcal{L}(\mathfrak{A}) = \{w \mid (w, q_0, Z_0) \Rightarrow (F, \gamma)\}$.

A RTDPA $\mathfrak{A}$ is called strict if for any word $w \in \mathcal{L}(\mathfrak{A})$ and a corresponding computation $(w, q_0, Z_0) \Rightarrow (F, \gamma)$ the string $\gamma \in \Gamma^*$ is empty, i.e., on passing to the final state the automaton erases the starting pushdown symbol Z_0. In this paper, we consider only strict RTDPA. We stress again two properties of this class of automata. First, every automaton includes rules of the form $(a, \alpha, A) \to (\beta, \gamma)$, where $a \neq \varepsilon$, and at most one rule exists for each triple (a, α, A). Second, accepting a word, the automaton $\mathfrak{A}$ starts working in the state q_0 with starting pushdown symbol Z_0 and ends working in state F with an empty pushdown tape.

The equivalence problem is stated as follows: for any pair of automata $\mathfrak{A}$ and $\mathfrak{B}$, decide if the languages $\mathcal{L}(\mathfrak{A})$ and $\mathcal{L}(\mathfrak{B})$ are identical or not. The equivalence problem is decidable if an algorithm exists answering this question. If $\mathcal{L}(\mathfrak{A}) = \mathcal{L}(\mathfrak{B})$, then the automata $\mathfrak{A}$ and $\mathfrak{B}$ are called equivalent. Otherwise, $\mathfrak{A}$ and $\mathfrak{B}$ are inequivalent.

Translated from Kibernetika, No. 5, pp. 14-25, September-October, 1989. Original article submitted June 22, 1988.

0011-4235/89/2505-0581$12.50

With each DPA we associate a system of equations, which may be regarded as an algebraic representation of the automaton. We start by introducing some notation. Let $\mathfrak{A}$ be a strict RTDPA. For each pushdown symbol $P \in \Gamma$ we denote by W^P the set of rules of automaton $\mathfrak{A}$ that erase the symbol P, i.e., rules of the form $(a, \alpha, P) \to (\beta, \varepsilon)$. For each pair $(\alpha, P) \in S \times \Gamma$, we denote by $V^{\alpha P}$ the set of all rules of automaton $\mathfrak{A}$ which, reading an input symbol from the alphabet Σ and a pushdown symbol P in state α, adjoin the empty string γ to P, i.e., rules of the form $(a, \alpha, P) \to (\rho, P)$. Finally by $U^{\alpha P}$ we denote the set of all rules of automaton $\mathfrak{A}$ which, reading an input symbol from the alphabet Σ in state α with symbol P at the top of the pushdown, adjoin to P the nonempty string γ, i.e., rules of the form $(a, \alpha, P) \to (\beta, P\gamma)$.

Note that we do not consider RTDPA with rules that simultaneously erase the top symbol in the pushdown and write a nonempty string into the pushdown which starts with a symbol other than the erased symbol, i.e., the automaton does not contain rules of the form $(a, \alpha, P) \to (\beta, Q\gamma)$ (P is erased and $Q \neq P$).

For each rule θ: $(a, \alpha, P) \to (\beta, PQ_m \ldots Q_1) \in U^{\alpha P}$, construct the set $\Phi_\theta = \{[(\alpha_1, b_1), (\alpha_2, b_2), \ldots, (\alpha_m, b_m)]\}$, where each pair $(\alpha_p, b_p) \in S \times \Sigma$ $(p = 1,\ldots,m)$ is associated with the rule $(b_p, \alpha_p, Q_p) \to (\rho_p, \varepsilon) \in W^{Q_p}$ erasing the symbol Q_p. There is a one-to-one correspondence between the sets $W^{Q_1} \times \ldots \times W^{Q_m}$ and Φ_θ. Indeed, if $(\theta_1, \ldots, \theta_m) \in W^{Q_1} \times W^{Q_2} \times \ldots \times (W^{Q_m}$, $\theta_p : (b_p, \alpha_p, Q_p) \to (\rho_p, \varepsilon)$, then the collection of pairs $[(\alpha_1, b_1), \ldots, (\alpha_m, b_m)] \in \Phi_\theta$ can be associated to the tuple $(\theta_1,\ldots,\theta_m)$: the p-th pair (α_p, b_p) is obtained by interchanging the first two components in the left-hand side of rule θ_p. Since the automaton $\mathfrak{A}$ is deterministic, this correspondence is one-to-one, and therefore Φ_θ is representable as the Cartesian product $\Phi_\theta = \Phi^{Q_1} \times \ldots \times \Phi^{Q_m}$, where $\Phi^{Q_p} = \{(\alpha_p, b_p) \,|\, (b_p, \alpha_p, Q_p) \to (\rho_p, \varepsilon) \in W^{Q_p}\}$ for $p = 1,\ldots,m$.

For any rule θ: $(b, \alpha, Q) \to (\rho, \varepsilon)$, the pair (α, b) can be denoted by the symbol [Q] if the specific values of α, b for this case are irrelevant, introducing the notation $[Q]_i$ and $[Q]_j$ for different pairs. If the specific value of the state ρ is also immaterial and only its association with a rule erasing the symbol Q is relevant, then we use the notation $\mu[Q]$. Thus, a representative of the set of rules W^Q for a given pushdown symbol Q in general may be written as the rule $([Q], Q) \to (\mu[Q], \varepsilon)$. For specific values of b, α, Q, the state $\rho = \mu[Q]$ may be denoted by the symbol $\mu(\alpha, b, Q)$.

Let us now construct a system of equations given a strict RTDPA $\mathfrak{A}$. For each pair $(\alpha, Q) \in S \times \Gamma$ introduce the set of variables $X_{\beta b}^{\alpha Q}$, where the pair $(\beta, b) \in \Phi^Q$. With the variable $X_{\beta b}^{\alpha Q}$ associate a set of words in the alphabet Σ, namely the language $\hat{X}_{\beta b}^{\alpha Q} = \{xb \,|\, x \in \Sigma^*$ and there exists a computation $(x, \alpha, Q) \Rightarrow (\beta, Q)$ which does not erase the symbol Q from the pushdown even once$\}$.

In other words, if the automaton $\mathfrak{A}$ starts in state α with symbol Q at the top of the pushdown and reads the word x from the input tape, then this symbol Q is never erased in the course of the computation, and at the end of the computation the automaton goes to state β, such that the viewed cell on the pushdown tape contains the same symbol Q.

The only exceptions are variables of the form $X_F^{\alpha Z_0}$ for which the set of words $\hat{X}_F^{\alpha Z_0}$ contains words that take the automaton $\mathfrak{A}$ from state α to final state F with erasure of the starting pushdown symbol Z_0.

For each variable $X_{\beta b}^{\alpha Q}$, the system of equations $\mathfrak{A}(\bar{X})$ defined by automaton $\mathfrak{A}$ contains the following equation:

$$X_{\beta b}^{\alpha Q} = b\delta_{\alpha\beta} \vee \bigvee_{\eta \in V^{\alpha Q},\, \eta:(a,\alpha',Q)\to(\rho,Q)} a X_{\beta b}^{\rho Q} \vee$$

$$\vee \bigvee_{\substack{\theta \in U^{\alpha Q} \\ \theta:(a,\alpha,Q)\to(\rho,QT_m\ldots T_1)}} \bigvee_{[(\alpha_1,b_1),\ldots,(\alpha_m,b_m)]\in\Phi_\theta} a X_{\alpha_1 b_1}^{\rho T_1} \times X_{\alpha_2 b_2}^{\mu(\alpha_1,b_1,T_1)T_2} \ldots X_{\beta b}^{\mu(\alpha_m,b_m,T_m)Q}, \tag{1.1}$$

where $\delta_{\alpha\beta} = \varnothing$ if $\alpha \neq \beta$ and $\delta_{\alpha\alpha} = \varepsilon$. Moreover, the equations for the variables $X_F^{\alpha Z_0}$ contain an additional component $\bigvee_{(a,\alpha,Z_0)\to(F,\varepsilon)} a$.

The multiplication of variables in this equation defines the concatenation of the languages represented by these variables, and the disjunction operation, denoted by the symbol $\vee$, defines the union of the corresponding sets. Details of the general theory linking pushdown automata, context-free languages, and systems of equations are presented in [5]. Note that the third disjunctive expression in (1.1) may be rewritten in abbreviated form as $\bigvee_{\theta\in U^{\alpha Q}} a\mathfrak{A}_{\beta b}^{\rho Q T_m \ldots T_1}$, where $\mathfrak{A}_{\beta b}^{\rho Q T_m \ldots T_1}$ replaces the expression $\bigvee_{\Phi_\theta} X_{\alpha_1 b_1}^{\rho, T_1} \ldots X_{\beta b}^{\mu(\alpha_m, b_m, T_m)Q}$. Expressions also can be introduced for any subsequence of the sequence $QT_m \ldots T_1$. Thus,

$$\mathfrak{A}_{\beta b}^{\rho Q T_m \ldots T_1} = \bigvee_{\Phi_\theta = \Phi^{T_1} \times \ldots \times \Phi^{T_m}} X_{\alpha_1 b_1}^{\rho T_1} \ldots X_{\beta b}^{\mu[T_m]Q} =$$

$$= \bigvee_{\Phi^{T_1}} X_{[T_1]}^{\rho T_1} \mathfrak{A}_{\beta b}^{\mu[T_1]QT_m \ldots T_2} = \bigvee_{\Phi^{T_1} \times \Phi^{T_2}} X_{[T_1]}^{\rho T_1} \cdot X_{[T_2]}^{\mu[T_1]T_2} \mathfrak{A}_{\beta b}^{\mu[T_2]Q \ldots T_2},$$

and in general

$$\mathfrak{A}_{\beta b}^{\rho Q T_m \dots T_1} = \bigvee_{\Phi^{T_1} \times \dots \times \Phi^{T_k}} X_{[T_1]}^{\rho T_1} \cdot X_{[T_2]}^{\mu [T_1] T_2} \dots \mathfrak{A}_{\beta b}^{\mu [T_k] Q \dots T_{k+1}} = \bigvee_{[T_k] \in \Phi^{T_k}} \mathfrak{A}_{[T_k]}^{\rho T_k \dots T_1} \mathfrak{A}_{\beta b}^{\mu [T_k] Q \dots T_{k+1}}.$$

These arguments prove Lemma 9 of Sec. 2.

The principle of construction of Eq. (1.1) can be informally described as follows. If the word $xb \in \hat{X}_{\beta b}^{\alpha Q}$ ($\alpha \neq \beta$), then the computation defined by the word xb should start with a rule contained in one of the sets $V^{\alpha Q}$ or $U^{\alpha Q}$, since only these rules do not erase the symbol Q from the pushdown. For example, suppose that the computation defined by xb starts with the rule θ: $(a, \alpha, Q) \to (\rho, QT_m \dots T_1)$. Since it ends with the pair (β, Q), subsequent transitions will erase all the symbols $T_1, \dots, T_m$ from the pushdown, i.e., $xb = ax_0b$, where $x_0 = x_1 b_1 x_2 b_2 \dots x_m b_m x_{m+1}$, the word x_1 defines the computation $(x_1, \rho, T_1) \Rightarrow (\alpha_1, T_1)$, and b_1 erases the symbol T_1 by the rule θ_1: $(b_1, \alpha_1, T_1) \to (\rho_1, \varepsilon)$ from the set W^{T_1}. Then the word x_2 defines the computation $(x_2, \rho_1, T_2) \Rightarrow (\alpha_2, T_2)$ and the symbol b_2 erases T_2. This is continued until the entire string $T_1 \dots T_m$ has been erased, after which the last word x_{m+1} defines the computation $(x_{m+1}, \rho_m, Q) \Rightarrow (\beta, Q)$.

The sought system is constructed as follows. First write an equation of the form (2.1) for the variable $X_F^{q_0 Z_0}$ and then for all the variables $X_{\beta b}^{\alpha Q}$ entering the right-hand side of the equation for $X_F^{q_0 Z_0}$. Continue this process until the equations have been constructed for all the variables entering the right-hand side of at least one equation in the system. Since the total number of possible variables is finite, the construction process ends after finitely many steps. Denote the resulting system by $\mathfrak{A}(\bar{X})$, where $\bar{X} = (\dots, X_{\beta b}^{\alpha Q}, \dots)$ is the set of variables.

The minimum solution of the system $\mathfrak{A}(\bar{X})$ is the set of languages $\bar{\mathcal{L}} = \{\hat{X}_{\beta b}^{\alpha Q} | X_{\beta b}^{\alpha Q} \in \bar{X}\}$, which, when substituted for the variables $X_{\beta b}^{\alpha Q}$ in $\mathfrak{A}(\bar{X})$, transform the right- and the left-hand sides of each equation into a pair of equal sets, and for any other solution $\{\tilde{X}_{\beta b}^{\alpha Q} \mid X_{\beta b}^{\alpha Q} \in \bar{X}\}$ we have $\hat{X}_{\beta b}^{\alpha Q} \subseteq \tilde{X}_{\beta b}^{\alpha Q}$ for all variables $X_{\beta b}^{\alpha Q}$ of the system $\mathfrak{A}(\bar{X})$.

In what follows, if the fixed indices of the variable $X_{\beta b}^{\alpha Q} \in \bar{X}$ are immaterial for our specific discussion, we will denote the variables from $\bar{X}$ simply by X or by X_i if there are several variables.

The notion of the language $\hat{X}$ represented by the variable $\bar{X}$ can be extended to disjunctive polynomials. If $F = \bigvee_i X_{i1} \dots X_{ik_i}$ is a polynomial, then the language $\hat{F}$ is defined as $\hat{F} = \bigcup_i \widehat{X_{i1} \dots X_{ik_i}} = \bigcup_i \hat{X}_{i1} \dots \hat{X}_{ik_i}$.

THEOREM 1. Let $\hat{X}_F^{q_0 Z_0}$ be the first component of the minimum solution $\hat{X}$ of the system $\mathfrak{A}(\bar{X})$. Then $X_F^{q_0 Z_0} = \mathcal{L}_{\mathfrak{A}}$, where $\mathcal{L}_{\mathfrak{A}}$ is a language accepted by the strict RTDPA $\mathfrak{A}$.

The minimum solution theorems are proved by a well-known standard technique, and we accordingly omit the proof.

For each variable X, define the norm $\|X\|$ as $\|X\| = \min_{x \in \hat{X}} |x|$, where $|x|$ is the length of the word x in the language $\hat{X}$. The concept of norm can be extended to arbitrary disjunctive polynomials,

$$\| \bigvee X_{i1} \dots X_{ik_i} \| = \min_i \| X_{i1} \dots X_{ik_i} \| = \min_i \sum_{m=1}^{k_i} \| X_{im} \|.$$

We have previously considered the expressions $\mathfrak{A}_{\beta b}^{\rho Q T_m \dots T_1} = \mathfrak{A}_{\beta b}^{\rho \xi}$, where ξ replaces the string $QT_m, \dots, T_1$, the so-called characteristic of the expression. In what follows, if the specific value of βb is immaterial, we can replace it with the symbol "$\cdot$", writing $\mathfrak{A}_{\cdot}^{\rho \xi}$. If the superscripts are also immaterial, the expression will be simply denoted by $\mathfrak{A}$.

The following lemma relates to variables entering the same expression $\mathfrak{A}_{\beta b}^{\rho Q T_m \dots T_1} = \bigvee_{p=1}^{m'} X_{\alpha_p b_p}^{\rho T_1} \mathfrak{A}_{\beta b}^{\varphi_p Q T_m \dots T_2}$.

LEMMA 1. If the word $x \in \hat{X}_{\alpha_p b_p}^{\rho T_1}$, then there exists no word y such that $xy \in \hat{X}_{\alpha_q b_q}^{\rho T_1}$ ($1 \leqslant q \leqslant m'$).

Indeed, by the condition of the lemma $x \in \hat{X}_{\alpha_p b_p}^{\rho T_1}$, the word x erases the symbol T_1 from the pushdown, whereas existence of the word y and the inclusion $xy \in \hat{X}_{\alpha_q b_q}^{\rho T_1}$ imply that the symbol T_1 is first erased only by the last letter of the word y, a contradiction. Q.E.D.

2. CATEGORY SYSTEMS AND PROCEDURE II

We will analyze the equivalence problem of strict RTDPA $\mathfrak{A}$ and $\mathfrak{B}$ using a general method which involves construction of a category system and its direct limit. The notion of category system $\mathcal{K}$ was introduced in [6] by analogy with A-systems of [7, Sec. 111.1] and inductive systems of [8]. The elements of a category system are categories, and the mappings (morphisms) of the elements are functors. A category system constructed from the automata $\mathfrak{A}$, $\mathfrak{B}$ consists of a sequence of categories $\{\mathfrak{C}_i \mid i \geqslant 0\}$, such that for each pair $\mathfrak{C}_i$, $\mathfrak{C}_k$ ($i \leqslant k$) there exists a functor $G_{ik} : \mathfrak{C}_i \to \mathfrak{C}_k$, and if $i \leq k \leq l$, then $G_{ik} \cdot G_{kl} = G_{il}$. The

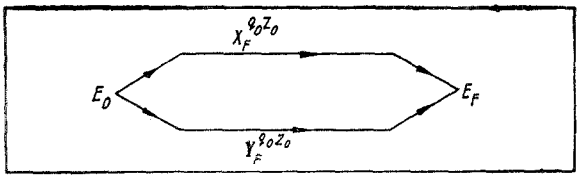

Fig. 1

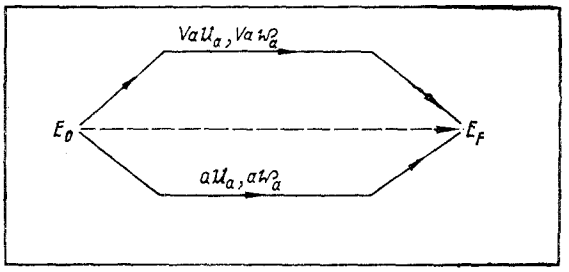

Fig. 2

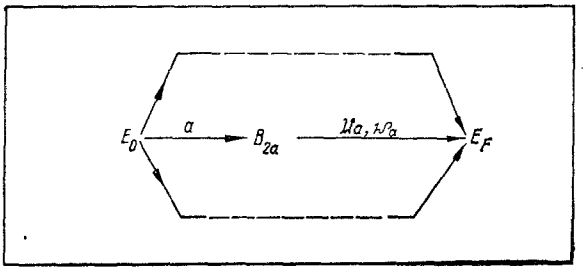

Fig. 3

morphisms of the categories $\mathfrak{E}_i$ are representable by disjunctive polynomials on the alphabet $\overline{X} \cup \overline{Y} \cup \Sigma \cup \overline{Z}$, where $\overline{X}, \overline{Y}$ are the alphabets of the variables entering the systems $\mathfrak{A}(\overline{X})$ and $\mathfrak{B}(\overline{Y})$, and the alphabet $\overline{Z}$ will be defined later (this is an alphabet of auxiliary variables).

The categories $\mathfrak{E}_i \in \mathcal{K}$ are constructed successively by a set of operations incorporated in some procedure Π. These operations determine the transition from $\mathfrak{E}_i$ to $\mathfrak{E}_{i+1}$, by inserting new morphisms and new objects in the category $\mathfrak{E}_j$. The functor G_{ii+1} is an inclusion functor and the category $\mathfrak{E}_i$ is a subcategory of the category $\mathfrak{E}_{i+1}$ for all $i \geq 0$.

In the process of construction of the system $\mathcal{K}$, a set of morphism pairs R_i is defined for each category $\mathfrak{E}_i$.

The starting category $\mathfrak{E}_0$ is defined by a diagram (Fig. 1) consisting of two objects E_0, E_F and a pair of morphisms $X_F^{q_0 Z_0}$, $Y_F^{q_0 Z_0}$. Identity morphisms are omitted in this and other diagrams.

The set R_0 consists of a single pair of morphisms $X_F^{q_0 Z_0} \doteq Y_F^{q_0 Z_0}$, that form a conditional equality. This concept is introduced because in what follows conditional equality may be either a true, exact equality or an inequality. All that follows usually relates to conditional equalities, and the qualifier "conditional" will be omitted. A pair of morphisms linked by an equality contributes to the category one arrow labeled by this pair.

A certain language can be associated with each morphism represented by a disjunctive polynomial, and the morphisms are accordingly compared by the corresponding languages. In particular, two morphisms $\mathfrak{U}$, $\mathfrak{V}$ are said to be equal (exact equality) if the corresponding languages $\hat{\mathfrak{U}}$, $\hat{\mathfrak{V}}$ are identical, i.e., $\mathfrak{U} = \mathfrak{V}$ if and only if $\hat{\mathfrak{U}} = \hat{\mathfrak{V}}$.

Procedure Π includes the following operations: replacement, aggregation, decomposition, splitting, substitution. We define the operations of procedure Π by constructing the categories $\mathfrak{E}_1, \mathfrak{E}_2, \mathfrak{E}_3$ of the system $\mathcal{K}$. The replacement operation uses the morphism $\bigvee X_i \mathfrak{U}_i$ in which every disjunctive term starts with the variable X_i to construct a new morphism $\bigvee_i f_i(\overline{X}) \mathfrak{U}_i$,

where $X_i = f_i(\overline{X})$ is an equation for the variable X_i in the system $\mathfrak{A}(\overline{X})$. On passing from $\mathfrak{C}_0$ to $\mathfrak{C}_1$, if $X_F^{q_0 Z_0} = \bigvee_{a \in \Sigma} a\mathfrak{U}_a$, $Y_F^{q_0 Z_0} = \bigvee_{a \in \Sigma} a\mathfrak{B}_a$ are equations for the variables $X_F^{q_0 Z_0}$ and $Y_F^{q_0 Z_0}$, then application of the replacement operation to the equality

$$X_F^{q_0 Z_0} \doteq Y_F^{q_0 Z_0}, \tag{2.1}$$

produces new morphisms $\bigvee a\mathfrak{U}_a$, $\bigvee a\mathfrak{B}_a$ of the category $\mathfrak{C}_1$ and a new conditional equality $\bigvee_a a\mathfrak{U}_a \doteq \bigvee_a a\mathfrak{B}_a$, which is included in R_1. Equality (2.1) is then marked as used.

The aggregation operation passes from the morphisms $\bigvee a\mathfrak{U}_a$, $\bigvee a\mathfrak{B}_a$ to the set of morphisms $\{\mathfrak{U}_a, \mathfrak{B}_a \mid a \in \Sigma\}$ and includes new equalities $\{\mathfrak{U}_a \doteq \mathfrak{B}_a \mid a \in \Sigma\}$ in R_1 (marking the equality $\bigvee a\mathfrak{U}_a \doteq \bigvee a\mathfrak{B}_a$ as used). Thus, $R_1 = R_0 \cup \{\bigvee a\mathfrak{U}_a \doteq \bigvee a\mathfrak{B}_a\} \cup \{a\mathfrak{U}_a \doteq a\mathfrak{B}_a \mid a \in \Sigma\}$, and the category $\mathfrak{C}_1$ is defined by the diagram in Fig. 2.

The decomposition operation passes from the pair of equalities $\mathfrak{U} \doteq \mathfrak{B}$ and $\mathfrak{U}\mathfrak{U}' \doteq \mathfrak{B}\mathfrak{B}'$ to the pair $\mathfrak{U} \doteq \mathfrak{B}$, $\mathfrak{U}' \doteq \mathfrak{B}'$ or from the pair of equalities $\mathfrak{U}\mathfrak{U}' \doteq \mathfrak{B}\mathfrak{B}'$ and $\mathfrak{U}' \doteq \mathfrak{B}'$ to the pair $\mathfrak{U} \doteq \mathfrak{B}$, $\mathfrak{U}' \doteq \mathfrak{B}'$. Applying the decomposition operation to each equality $a\mathfrak{U}_a \doteq a\mathfrak{B}_a$, which is a particular case of the equality $\mathfrak{U}\mathfrak{U}' \doteq \mathfrak{B}\mathfrak{B}'$ for $\mathfrak{U} = \mathfrak{B} = a$, we obtain the system of equalities $\{\mathfrak{U}_a \doteq \mathfrak{B}_a \mid a \in \Sigma\}$. Then $R_2 = R_1 \cup \{\mathfrak{U}_a \doteq \mathfrak{B}_a \mid a \in \Sigma\}$. The diagram of category $\mathfrak{C}_2$ is given in Fig. 3, which shows only the new arrows not included in the diagram of category $\mathfrak{C}_1$.

The substitution operation is applied to equalities of the form

$$\bigvee_{p=1}^{k} X_p \mathfrak{U}_p \doteq \mathfrak{B}_{\cdot}^{\psi\xi}, \tag{2.2}$$

where $\mathfrak{B}_{\cdot}^{\psi\xi}$ is the expression defined by the state ψ and the characteristic ξ, and the expression in the left-hand side is nonlinear. A certain language $\hat{X}_p$ is associated with each variable X_p, and we can select the shortest word from this language. This word will be denoted by $\langle X_p \rangle$ and called a representing morphism. Substitute $\langle X_p \rangle$ in equality (2.2). Since $\langle X_p \rangle \in \hat{X}_p$, we obtain the expression in the left-hand side. If $\langle X_p \rangle$ is substituted in automaton $\mathfrak{B}$ in state ψ with the pushdown containing the string ξ, the automaton will go to some state ψ_p, part of the string ξ will be erased, and a new string η_p will be adjoined to the remaining initial piece ξ_p. In other words, for each $\langle X_p \rangle$ we obtain a new equality

$$\mathfrak{U}_p \doteq \mathfrak{B}_{\cdot}^{\psi_p \xi_p \eta_p}, \tag{2.3}$$

and on the whole we obtain a system of equalities, which are called the direct system of equality (2.2). Note that the starting subword of the word $\langle X_p \rangle$ cannot erase the entire sequence ξ, because this would produce an empty word with zero norm in the right-hand side and an expression with norm not less than $\min_p \| \mathfrak{U}_p \| > 0$ in the left-hand side. The expression obtained by substituting the word $\langle X_p \rangle$ in $\mathfrak{B}$ will be denoted by the symbol $\int \langle X_p \rangle \mid \mathfrak{B}$. Thus, from (2.3), we have $\int \langle X_p \rangle \mid \mathfrak{B}_{\cdot}^{\psi\xi} = \mathfrak{B}_{\cdot}^{\psi_p \xi_p \eta_p}$.

Let $q = \min_p | \xi_p |$. From each ξ_p separate a string of symbols of length $q - \zeta A$. Then each equality (2.3) can be written as

$$\mathfrak{U}_p \doteq \mathfrak{B}_{\cdot}^{\psi_p \zeta A \xi_p \eta_p}, \tag{2.4}$$

where $\zeta A \xi_p = \xi_p$. But

$$\mathfrak{B}_{\cdot}^{\psi_p \zeta A \xi_p \eta_p} = \bigvee_{[A]} \mathfrak{B}_{A]}^{\psi_p A \xi_p \eta_p} \mathfrak{B}_{\cdot}^{\mu[A]\xi}. \tag{2.5}$$

Left-multiply each equality in (2.4) by X_p and consider the disjunctive union over all p. Then we have

$$\bigvee_{p=1}^{k} X_p \mathfrak{U}_p \doteq \bigvee_{p=1}^{k} X_p \mathfrak{B}_{\cdot}^{\psi_p \zeta A \xi_p \eta_p} = \bigvee_{p=1}^{k} \bigvee_{[A]} X_p \mathfrak{B}_{[A]}^{\psi_p A \xi_p \eta_p} \mathfrak{B}_{\cdot}^{\mu[A]\xi}. \tag{2.6}$$

But by (2.2) the left-hand side of (2.6) is equal to $\mathfrak{B}_{\cdot}^{\psi\xi}$, and $\mathfrak{B}_{\cdot}^{\psi\xi} = \bigvee_{[A]} \mathfrak{B}_{[A]}^{\psi A \bar{\xi}} \mathfrak{B}_{\cdot}^{\mu[A]\xi}$, because ζA is the initial substring of the string ξ. Hence we obtain

$$\bigvee_{[A]} \bigvee_{p} X_p \mathfrak{B}_{[A]}^{\psi_p A \xi_p \eta_p} \mathfrak{B}_{\cdot}^{\mu[A]\xi} \doteq \bigvee_{[A]} \mathfrak{B}_{[A]}^{\psi A \bar{\xi}} \mathfrak{B}_{\cdot}^{\mu[A]\xi} \tag{2.7}$$

or, equating the expressions for equal $\mathfrak{B}^{\mu[A]\xi}$,

$$\bigvee_p X_p \mathfrak{B}_{[A]}^{\psi_p A \zeta_p \eta_p} \doteq \mathfrak{B}_{[A]}^{\psi A \xi}. \tag{2.8}$$

Equality (2.8) is written out for each pair [A], and all these equalities jointly constitute the inverse system of equality (2.2).

Note that all the equalities (2.8) were written out assuming distinct $\mathfrak{B}^{\mu[A]\xi}$. In reality, for different pairs $[A]_i \neq [A]_j$ we may have $\mu[A]_i = \mu[A]_j$. Then clearly, $\mathfrak{B}^{\mu[A]_i\xi} = \mathfrak{B}^{\mu[A]_j\xi}$. In this case, instead of the pair of equalities of the form (2.8) for $[A]_i$ and $[A]_j$, the inverse system contains only one equality

$$\bigvee_p X_p (\mathfrak{B}_{[A]_i}^{\psi_p A \zeta_p \eta_p} \vee \mathfrak{B}_{[A]_j}^{\psi_p A \zeta_p \eta_p}) \doteq \mathfrak{B}_{[A]_i}^{\psi A \xi} \vee \mathfrak{B}_{[A]_j}^{\psi A \xi}. \tag{2.8.1}$$

On passing from category $\mathfrak{E}_i$ to $\mathfrak{E}_{i+1}$, the systems (2.4) and (2.8) are adjoined to the set R_i.

If, for instance, (2.2) coincides with the equality $\mathfrak{U}_a \doteq \mathfrak{B}_a$, then the diagram of category $\mathfrak{E}_3$ has the form shown in Fig. 4: the solid arrows correspond to the index p, the wavy arrows to the direct system, and the broken arrows to the inverse system. The objects $B_{3[A]}$ and B_{3p} are new relative to Fig. 3 for category $\mathfrak{E}_2$.

The comparison algorithm is used for transition to category $\mathfrak{E}_4$ and subsequent construction of the category system $\mathfrak{K}$. The first steps of the algorithm were demonstrated for the construction of the categories $\mathfrak{E}_1$, $\mathfrak{E}_2$, $\mathfrak{E}_3$. Now let $\mathfrak{E}_r$ be the category to which the comparison algorithm is to be applied in the given step and R_r the corresponding set of pairs (conditional equalities). We define the following steps of the comparison algorithm.

1. Operations of procedure Π included in the comparison algorithm are applied to all unmarked equalities included in R_r.

2. First the splitting operation is applied to all equalities for which this is feasible. Splitting takes us from the pair of equalities $\mathfrak{U} \vee \mathfrak{U}' \doteq \mathfrak{B} \vee \mathfrak{B}'$, $\mathfrak{U} \doteq \mathfrak{B}$ to the pair $\mathfrak{U}' \doteq \mathfrak{B}'$, $\mathfrak{U} \doteq \mathfrak{B}$. If the equality being split corresponds to a wavy (broken) arrow, then after splitting the new equality also corresponds to a wavy (broken) arrow.

3. If an equality corresponds to a wavy arrow of category $\mathfrak{E}_r$, then the substitution operation is applied to it. Any equality to which the operation has been applied is marked as used. Substitution produces a direct and an inverse system. For each equality of the inverse system, a broken arrow is introduced in category $\mathfrak{E}_{r+1}$. For each equality $\mathfrak{U}_a \doteq \mathfrak{B}_a$ of the direct system, a wavy arrow is introduced if $\mathfrak{U}_a$ is nonlinear and a broken arrow if $\mathfrak{U}_a = X$.

4. The replacement operation and then the decomposition operation are applied to all equalities from R_r that correspond to broken arrows. The resulting equalities are associated to wavy arrows if the right-hand side of the equality is nonlinear and to broken arrows otherwise.

5. All the new equalities are included in R_{r+1}, and the added arrows and objects form the diagram of category $\mathfrak{E}_{r+1}$ (they are adjoined to the diagram of category $\mathfrak{E}_r$).

6. The comparison algorithm includes additional rules for marking conditional equalities. Let $\mathfrak{U} \doteq \mathfrak{B}$ be a given equality.

Rule ⟨r1⟩. The morphisms $\mathfrak{U}$ and $\mathfrak{B}$ are written in the alphabet Σ and are equal as sets of words.

Rule ⟨r2⟩. The equality $\mathfrak{U} \doteq \mathfrak{B}$ added to the set R_{r+1} is already contained in the set R_k ($k \leq r$).

In both cases, the equality $\mathfrak{U} \doteq \mathfrak{B}$ is marked as used, and the corresponding arrow in $\mathfrak{E}_{r+1}$ is replaced with a solid arrow.

7. The comparison algorithm stops if in some step either all the rules in R_r are already marked or one of the following negative conditions is satisfied for the equality $\mathfrak{U} \doteq \mathfrak{B}$.

Rule ⟨m1⟩. The morphisms $\mathfrak{U}$, $\mathfrak{B}$ are written in the alphabet Σ and are unequal as sets of words.

Rule ⟨m2⟩. $\|\mathfrak{U}\| \neq \|\mathfrak{B}\|$.

If rules ⟨m1⟩ and ⟨m2⟩ are satisfied, we can find a shortest word in one of the languages $\hat{X}_F^{q_0 Z_0}$, $\hat{Y}_F^{q_0 Z_0}$ which is not contained in the other language.

Regarding the rule ⟨r2⟩ we should note that, comparing the automata $\mathfrak{A}$ and $\mathfrak{B}$, we seek the shortest word which can prove inequivalence of the automata. Therefore its search by the first occurrence of the equality $\mathfrak{U} \doteq \mathfrak{B}$ in R_k will produce a shorter word than that determined by the second occurrence of the same equality in R_{r+1}, $k < r + 1$.

An operation of procedure Π is called correct if its application to the equality $\mathfrak{U} \doteq \mathfrak{B}$ produces new equalities $\mathfrak{U}_p \doteq \mathfrak{B}_p$ that satisfy the following condition: if the languages $\hat{\mathfrak{U}}$, $\hat{\mathfrak{B}}$ are identical, then $\hat{\mathfrak{U}}_p = \hat{\mathfrak{B}}_p$ for all p, and if $\hat{\mathfrak{U}} \neq \hat{\mathfrak{B}}$, then there is an index p_0 such that $\hat{\mathfrak{U}}_{p_0} \neq \hat{\mathfrak{B}}_{p_0}$. The transition from category $\mathfrak{E}_p$ to category $\mathfrak{E}_{p+1}$ will be called correct if all the operations applied to the morphisms from R_p when constructing the set R_{p+1} are correct.

THEOREM 2. Let $\mathfrak{A}$ and $\mathfrak{B}$ be two strict real-time deterministic pushdown automata, and $\mathcal{K} = \{\mathfrak{E}_p, G_{pk} \mid 0 \leqslant p \leqslant k < \infty\}$ the category system constructed from category $\mathfrak{E}_0$ by the comparison algorithm. If the transition from $\mathfrak{E}_p$ to $\mathfrak{E}_{p+1}$ is correct for all categories $\mathfrak{E}_p$ of the system, then the equivalence problem is decidable for the automata $\mathfrak{A}$ and $\mathfrak{B}$.

All the operations of procedure Π, with the possible exception of the substitution operation, are always correct (we will prove this in Sec. 3). For the substitution operation, consider the following example. Let

$$X_1\mathfrak{U}_1 \vee X_2\mathfrak{U}_2 \doteq Y_1\mathfrak{B}_1 \vee Y_2\mathfrak{B}_2, \tag{2.9}$$

where

$$X_1 = X_1' \vee X_2',\ \mathfrak{U}_1 = (\mathfrak{R}' \vee \mathfrak{R}'')\,\mathfrak{B}_1 \vee \mathfrak{R}_1\mathfrak{B}_2,\ \mathfrak{U}_2 = \mathfrak{B}_2,\ Y_1 = X_1'(\mathfrak{R}'Y' \vee \mathfrak{R}'') \vee X_1''(\mathfrak{R}' \vee \mathfrak{R}''),\ Y_2 = X_1'\mathfrak{R}'Y'' \vee (X_1' \vee X_1'')\,\mathfrak{R}_1 \vee X_2,\ Y' = \int \langle X_1' \rangle \mid Y_1,\ X'' = \int \langle X_1'' \rangle \mid Y_2.$$

Equality (2.9) will be exact if $\mathfrak{B}_1 = Y'\mathfrak{B}_1 \vee Y''\mathfrak{B}_2$. But in this case, the inverse system comprising the equalities $X_1(\mathfrak{R}' \vee \mathfrak{R}'') \doteq X_1'(\mathfrak{R}'Y' \vee \mathfrak{R}'') \vee X_1''(\mathfrak{R}' \vee \mathfrak{R}'')$, $X_1\mathfrak{R}_1 \vee X_2 \doteq X_1'\mathfrak{R}'Y'' \vee X_1\mathfrak{R}_2 \vee X_2$ is obviously not a system of exact equalities, although (2.9) is exact.

The comparison algorithm can be modified in such a way that even with incorrect substitution operation, it will solve the equivalence problem of strict RTDPA. We thus have the following theorem.

THEOREM 3. The equivalence problem of strict real-time deterministic automata is decidable.

Theorems 2 and 3 are proved in Sec. 4.

3. PROOF OF AUXILIARY RESULTS

We will prove that the operations of procedure Π are correct.

LEMMA 2 (replacement). The transition from the equality

$$\bigvee_{p=1}^{m} X_p\mathfrak{U}_p \doteq \bigvee_{q=1}^{n} Y_q\mathfrak{B}_q \tag{*}$$

to the equality

$$\bigvee_{p=1}^{m} f_p(\bar{X})\,\mathfrak{U}_p \doteq \bigvee_{q=1}^{n} g_q(\bar{Y})\,\mathfrak{B}_q \tag{**}$$

is correct.

In the condition of the lemma, $\bar{X}_p = f_p(\bar{X})$ is the equation for X_p from the system $\mathfrak{A}(\bar{X})$, and $Y_q = g_q(\bar{Y})$ is the equation for Y_q from the system $\mathfrak{B}(\bar{Y})$. Since $X_1 = f_1(\bar{X})$ is an exact equality, the transition from (*) to the equality

$$f_1(\bar{X})\,\mathfrak{U}_1 \vee \bigvee_{p=2}^{m} X_p\mathfrak{U}_p \doteq \bigvee_{q=1}^{n} g_q(\bar{Y})\,\mathfrak{B}_q \tag{***}$$

is correct, because (*) differs from (***) only in the notation of the variable X_1 for which the languages $\hat{X}_1 = f_1(\widehat{\bar{X}})$. Applying this transformation to all variables X_p ($2 \leq p \leq m$), Y_q ($1 \leq q \leq n$) in (*), we successively obtain equality (**). Since the transition is correct for each individual variable, the transition from (*) to (**) is thus also correct on the whole. Q.E.D.

LEMMA 3 (aggregation). Replacement of the equality $\bigvee_{i=1}^{m} a_i\mathfrak{U}_i \doteq \bigvee_{i=1}^{m} a_i\mathfrak{B}_i$, where $a_i \neq a_j$ for $i \neq j$, by a system of m equalities $\{a_i\mathfrak{U}_i \doteq a_i\mathfrak{B}_i \mid 1 \leqslant i \leqslant m\}$ is a correct operation. Indeed, if the starting equality is exact, then each equality in the derived system obtained by aggregation is also exact. If there exists a word $x \in \bigvee a_i\mathfrak{U}_i$, $x \notin \bigvee a_i\mathfrak{B}_i$, then $x = a_{i_0}x'$ and $x' \in \mathfrak{U}_{i_0}$. But then $x' \notin \mathfrak{B}_{i_0}$, since otherwise the word x would belong to $a\mathfrak{B}_{i_0}$, i.e., be contained in $\bigvee a_i\mathfrak{B}_i$. Q.E.D.

LEMMA 4. Let X be the following system included in R_i: $\chi = (X\mathfrak{U} \doteq Y\mathfrak{B},\ X \doteq Y)$. Then the transition from X to the system $X^{\wedge} = (\mathfrak{U} \doteq \mathfrak{B},\ X \doteq Y)$ included in R_{i+1} is a correct operation.

Proof. If $X \neq Y$, then the lemma is obvious. Thus assume that $X = Y$ and show that X is an epimorphism in the category $\mathfrak{E}_{i+1}$, i.e., $X\mathfrak{U} \doteq Y\mathfrak{B}$ implies $\mathfrak{U} \doteq \mathfrak{B}$. Let $X\mathfrak{U} = Y\mathfrak{B}$ and assume that X is not an epimorphism, i.e., $\mathfrak{U} \neq \mathfrak{B}$. Then there exists a word y which is contained, say, in $\hat{\mathfrak{U}}$ and $y \notin \mathfrak{B}$. Let x be the shortest word from the language $\hat{X}$. Then $xy \in \widehat{X\mathfrak{U}}$ and by the equality $X\mathfrak{U} = Y\mathfrak{B}$, we have $xy \in \widehat{Y\mathfrak{B}}$. Since x is a shortest word from $\hat{Y}$, there should exist a word y_1

such that $y = y_1y_2$ and $xy_1 \in \hat{Y}$. But this is impossible by definition of the variable $Y \in \overline{X} \cup \overline{Y}$ and Lemma 1. The contradiction establishes that X is an epimorphism.

Now let $X\mathfrak{U} \neq Y\mathfrak{B}$. Then there exists a shortest word xy contained, say, in $\widehat{X\mathfrak{U}}$ and $xy \notin \widehat{Y\mathfrak{U}}$, $x \in \hat{X}$, $y \in \hat{\mathfrak{U}}$. But by the equality X = Y, we have $x \in \hat{Y}$ and therefore $y \notin \mathfrak{B}$, whence $\mathfrak{U} \neq \mathfrak{B}$. Q.E.D.

LEMMA 5 (splitting). Let the equalities of the system $X = \{\mathfrak{U} \vee \mathfrak{U}' \doteq \mathfrak{B} \vee \mathfrak{B}',\ \mathfrak{U}' \doteq \mathfrak{B}'\}$ be included in the set R_i and $\mathfrak{U} \cap \mathfrak{U}' = \varnothing, \mathfrak{B} \cap \mathfrak{B}' = \varnothing$. Then inclusion of the system $X^{\wedge} = \{\mathfrak{U} \doteq \mathfrak{B}, \mathfrak{U}' \doteq \mathfrak{B}'\}$ in R_{i+1} is a correct operation.

In order to prove this assertion, it suffices to consider the case when $\mathfrak{U}' = \mathfrak{B}'$. But then replacement of $\mathfrak{U} \vee \mathfrak{U} \doteq \mathfrak{B} \vee \mathfrak{B}'$ with the equality $\mathfrak{U} \doteq \mathfrak{B}$ is obviously correct. Q.E.D.

The following lemma is the dual of Lemma 4.

LEMMA 6. Let $X = \{\mathfrak{U}X \doteq \mathfrak{B}Y,\ X \doteq Y\}$ be a system of equalities included in R_i. Then the transition from X to the system $X^{\wedge} = \{\mathfrak{U} \doteq \mathfrak{B}, X \doteq Y\}$ included in R_{i+1} is a correct operation.

Similarly to the proof of Lemma 4, we may assume that X = Y. We have to show that X is a monomorphism in category $\mathfrak{E}_{i+1}$. Assume the contrary, i.e., $\mathfrak{U}X = \mathfrak{B}Y$ and $\mathfrak{U} \neq \mathfrak{B}$ after right-dividing by X. By assumption, there exists a shortest word, say, $z \in \hat{\mathfrak{U}}$ and $z \notin \hat{\mathfrak{B}}$. Let $x \in \hat{X}$ be a shortest word in the language $\hat{X} = \hat{Y}$ (by assumption). Then $zx \in \widehat{\mathfrak{U}X}$ and $zx \in \widehat{\mathfrak{B}Y}$. Thus, there exists a word $z' \in \hat{\mathfrak{B}}$, $z' \in \hat{\mathfrak{U}}$, $|z'| < |z|$, $z = z'z''$. But then both words z' and z starting with z' are contained in the language $\hat{\mathfrak{U}}$, which is impossible. Indeed, if $\mathfrak{U} = \mathfrak{U}_{[A]}^{\alpha A\xi}$, then the word $z \in \hat{\mathfrak{U}}_{[A]}^{\alpha A\xi}$ and it defines a computation that takes the automaton from state α to state $\mu[A]$ erasing the string ξ from the pushdown. But the word z' has previously erased ξ, because $z' \in \hat{\mathfrak{U}}_{[A]}^{\alpha A\xi}$. The contradiction proves the lemma for this case.

Now let $\mathfrak{U}X \neq \mathfrak{B}Y$ and after right-dividing $\mathfrak{U} = \mathfrak{B}$. Right-multiply this equality by X = Y. The result is a contradiction with the starting assumption. Q.E.D.

The following results present correctness conditions and proof of correctness of the substitution operation.

Rewrite equality (2.2) with the first pushdown symbol Q in explicit form: $\bigvee_{[Q]} X_{[Q]}^{\alpha Q}\mathfrak{U}_{\cdot}^{\mu[Q]\eta} \doteq \mathfrak{B}_{\cdot}^{\psi\xi}$, i.e., replacing X_p with $X_{[Q]}^{\alpha Q}$.

LEMMA 7. $\hat{X}_{[Q]_i}^{\alpha Q} \cap \hat{X}_{[Q]_j}^{\alpha Q} = \varnothing$ for $i \neq j$.

Indeed, for deterministic pushdown automata the same word cannot define two different computations $(x, \alpha, Q) \Rightarrow [Q]_i$ and $(x, \alpha, Q) \Rightarrow [Q]_j$. Q.E.D.

LEMMA 8. If (2.2) is an exact equality, then each equality of the direct system (2.3) is also exact.

Indeed, if the direct system contains at least one inequality, e.g., $\mathfrak{U}_p \neq \mathfrak{B}_{\cdot}^{\psi_p\xi_p\eta_p}$, i.e., a word $x \in \hat{\mathfrak{U}}_p$, $x \notin \hat{\mathfrak{B}}_{\cdot}^{\psi_p\xi_p\eta_p}$ exists, then it would imply that the word $\langle X_p\rangle x \notin \hat{\mathfrak{B}}_{\cdot}^{\psi\xi}$, thus contradicting the assumption that equality (2.2) is exact. Similarly, if there exists a word $y \in \mathfrak{B}_{\cdot}^{\psi_p\xi_p\eta_p}$ and $y \notin \hat{\mathfrak{U}}_p$, then the word $\langle X_p\rangle y \in \hat{\mathfrak{B}}_{\cdot}^{\psi\xi}$, but $\langle X_p\rangle y \notin \bigvee_{p=1}^{k} X_p\mathfrak{B}_p$. Q.E.D.

LEMMA 9. The transition from the expression $\mathfrak{B}_{\cdot}^{\psi_p\xi A\zeta_p\eta_p}$ to the expression $\bigvee_{[A]} \mathfrak{B}_{[A]}^{\psi_p A\zeta_p\eta_p}\mathfrak{B}_{\cdot}^{\mu[A]\zeta}$ defines an exact equality (2.5).

The lemma was proved in Sec. 2

LEMMA 10. Replacement of equality (2.2) by equality (2.7) and the direct system (2.3) is correct.

Proof. Assume that equality (2.2) is exact. Then Lemma 8 implies that each equality in the direct system is exact. Right-multiplying each equality of the direct system (2.3) by X_p and examining their disjunction, we obtain

$$\bigvee_p X_p\mathfrak{U}_p \doteq \bigvee_p X_p\mathfrak{B}_{\cdot}^{\psi_p\xi_p\eta_p}. \tag{3.1}$$

But

$$\bigvee X_p\mathfrak{U}_p \doteq \mathfrak{B}_{\cdot}^{\psi\xi} = \bigvee_{[A]} \mathfrak{B}_{[A]}^{\psi A\xi}\mathfrak{B}_{\cdot}^{\mu[A]\zeta} = \bigvee_p X_p\mathfrak{B}_{\cdot}^{\psi_p\xi_p\eta_p} = \bigvee_p \bigvee_{[A]} X_p\mathfrak{B}_{[A]}^{\psi_p A\zeta_p\eta_p}\mathfrak{B}_{\cdot}^{\mu[A]\zeta}.$$

Thus, equality (2.7) is also exact, which proves the lemma in this case.

Now assume that (2.2) is not exact. Then for the pair of languages defined by the left- and right-hand sides of this equality there exists a word which is contained in one language and is not contained in the other language. For definiteness, let $x \in \bigvee_p \hat{X}_p\hat{\mathfrak{U}}_p$, $x \notin \hat{\mathfrak{B}}_{\cdot}^{\psi\xi}$, and x is the shortest of these words. If x starts with a representing morphism, i.e., $x = \langle x_{p_0}\rangle x'$, then $x' \in \hat{\mathfrak{U}}_{p_0}$, but $x' \notin \mathfrak{B}_{\cdot}^{\psi_{p_0}\xi_{p_0}\eta_{p_0}}$, because otherwise x would be contained in $\hat{\mathfrak{B}}_{\cdot}^{\psi\xi}$. Then the equality $\mathfrak{U}_{p_0} \doteq \mathfrak{B}_{\cdot}^{\psi_{p_0}\xi_{p_0}\eta_{p_0}}$ of the direct system is inexact, and the lemma is true.

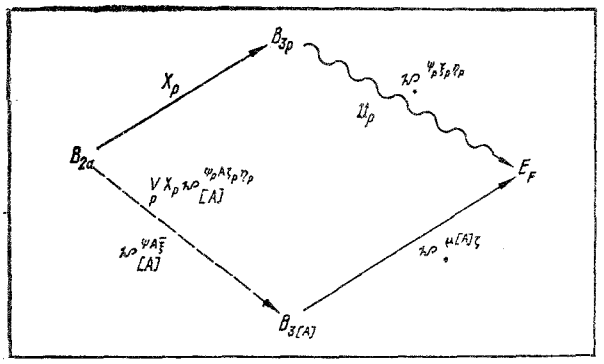

Fig. 4

If x does not start with a representing morphism, then either the direct system may contain an inexact equality or all the equalities will be exact. The latter is possible because the words on which (2.2) is an inequality will be omitted on passing to the direct system: while constructing the direct system, we replace the variable X_p with a word — the representing morphism $\langle X_p \rangle$. If the system (2.3) is inexact, then the transition defined by the lemma is correct. If (2.3) is exact, then we will show that equality (2.7) is inexact in this case. Indeed, since (2.3) is exact, we conclude that (3.1) is exact, i.e., $x \in \bigvee_p \hat{X}_p \hat{\mathfrak{V}}^{\psi_p \xi_p \eta_p}$.

But then x belongs to the left-hand side of equality (2.7) and by assumption does not belong to the right-hand side. Thus, (2.7) is an exact equality, and the lemma is true.

Now assume that there exists a word $y \in \hat{\mathfrak{V}}^{\varphi\iota}$ and $y \notin \bigvee_p \hat{X}_p \hat{u}_p$. Arguing as previously, we can show that either the direct system is inexact or the fact that (3.1) is exact implies that the word $y \notin \bigvee_p \hat{X}_p \hat{\mathfrak{V}}^{\psi_p \xi_p \eta_p}$ and (2.7) is inexact. Q.E.D.

Let $\mathfrak{V}_0, \mathfrak{V}_1, \ldots, \mathfrak{V}_k$ be a given system of expressions. We say that the expression $\mathfrak{V}_0$ depends on the system $\{\mathfrak{V}_1, \ldots, \mathfrak{V}_k\}$ if there exist expressions $\mathfrak{W}_i$ $(0 \leqslant i \leqslant k)$ such that $\mathfrak{V}_0 = \bigvee_{i=0}^{k} \mathfrak{W}_i \mathfrak{V}_i$ and if $\mathfrak{V}_0 \neq \varnothing$, then $\| \mathfrak{V}_0 \| \geqslant 1$. Otherwise, we say that $\mathfrak{V}_0$ is independent of $\mathfrak{V}_1, \ldots, \mathfrak{V}_k$.

The characteristic system $\varkappa$ of equality (2.7) is the set of expressions $\{\mathfrak{V}_{\cdot}^{\mu[A]\xi}\}$, entering this equality. The system $\varkappa$ is independent if any expression $\mathfrak{V}^{\mu[A]\xi}$ is independent of the remaining expressions in $\varkappa$. For example, if $\mathfrak{V}_{\cdot}^{\mu[A]_i\xi} = \mathfrak{V}_{\cdot}^{\mu[A]\xi}{}_j$ for $i \neq j$, then obviously the system $\varkappa$ is independent, because taking $\mathfrak{W}_k = \varnothing$ for $k \neq j$ and $\mathfrak{W}_j = \varepsilon$, we obtain the equality $\mathfrak{V}_{\cdot}^{\mu[A]_i\xi} = \bigvee_k \mathfrak{W}_k \mathfrak{V}_{\cdot}^{\mu[A]_k\xi}$, which signifies dependence of $\mathfrak{V}^{\mu[A]_i\xi}$.

LEMMA 11. If the characteristic system κ of equality (2.7) is independent, then the substitution operation replacing equality (2.2) with the direct and inverse systems of equalities (equalities (2.3) and (2.8) respectively) is correct.

Proof. By Lemma 10, the transition from (2.2) to equality (2.7) and the direct system is correct. Therefore, in order to prove Lemma 11 is suffices to show that replacement of equality (2.7) with the inverse system is correct.

First assume that (2.7) is exact, but the inverse system (2.8) contains for the pair $[A]_0$ the inequality

$$T \neq M, \tag{3.2}$$

where $T = \bigvee_{p=1}^{k} X_p \mathfrak{V}_{[A]_0}^{\psi_p A_p \xi_p \eta_p}$, and $M = \mathfrak{V}_{[A]_0}^{\psi A \bar{\xi}}$. This means that there exists a word $v \in \hat{T}$, $v \in \hat{X}_{p_0} \hat{\mathfrak{V}}_{[A]_0}^{\psi_{p_0} A \xi_{p_0} \eta_{p_0}}$ and $v \notin M$, or conversely $v \in \hat{M}$ and $v \notin \hat{T}$. In the first case, since (2.7) is exact, we have that the subset of words $v\hat{\mathfrak{V}}^{\mu[A]_0\xi} \subset \bigvee_{[A]} \hat{\mathfrak{V}}_{[A]}^{\psi A \bar{\xi}} \mathfrak{V}_{\cdot}^{\mu[A]\xi}$. Then three cases are possible: 1) for some pair $[A]_1 \neq [A]_0$ we have $v \in \hat{\mathfrak{V}}_{[A]_1}^{\psi A \bar{\xi}}$, because $v \notin M$; 2) there exists a starting subword v_1 of the word $v = v_1 v_2$ such that $v_1 \in \hat{\mathfrak{V}}_{[A]_1}^{\psi A \bar{\xi}}$, and $v_2 \mathfrak{V}_{\cdot}^{\mu[A]_0\xi} \subset \mathfrak{V}_{\cdot}^{\mu[A]_1\xi}$, and in this case the word v_1 defines a computation on which the automaton goes from state ψ with pushdown string ξ to state $\mu[A]_1$ and the string $A\bar{\xi}$ is erased from the pushdown; or 3) the word v erases from the pushdown only part of the string $A\bar{\xi}$ and at least does not erase the symbol A, i.e., it defines the computation $(v, \psi, A\bar{\xi}) \Rightarrow (\varepsilon, \psi_v, A\bar{\xi})$.

In case 1, substituting v in equality (2.7), we obtain $\mathfrak{V}_{\cdot}^{\mu[A]_0\xi} = \mathfrak{V}_{\cdot}^{\mu[A]_1\xi}$, i.e., κ is a dependent system. In case 2, substituting v_1 in T, we obtain an expression $\bigvee_{p=1}^{k} \mathfrak{W}_{[A]}^{(p)}$, where the symbol $\mathfrak{W}_{[A]}^{(p)} = \int v_1 | X_p \mathfrak{V}_{[A]}^{\psi_p A \xi_p \eta_p}$. The characteristic of each

$\mathfrak{B}_{[A]}^{(p)}$ is positive, because only the entire word v can erase the string $A\xi_p\eta_p$ in full. But then from (2.7) we obtain the equality $\bigvee_{p=1}^{k} \mathfrak{B}_{[A]}^{(p)} \mathfrak{B}_{\cdot}^{\mu[A]\xi} \doteq \mathfrak{B}_{\cdot}^{\mu[A]_1\xi}$, which signifies dependence of $\varkappa$. Finally, in case 3, substituting v in (2.7), we obtain the equality $\mathfrak{B}_{\cdot}^{\mu[A]_0\xi} \doteq \bigvee_{[A]} \mathfrak{B}_{[A]}^{\psi_v A\bar{\xi}} \mathfrak{B}_{\cdot}^{\mu[A]\xi}$, which also determines that system $\varkappa$ is dependent.

But this contradicts the assumption of the lemma for all three cases, and is thus impossible. The proof for the case when $v \in M$ and $v \notin T$ is similar. We have thus proved the lemma for the case when (2.7) is an exact equality.

It remains to consider the case when (2.7) is an inequality, and all the equalities of the inverse system are exact. Right-multiply each equality in the inverse system respectively by $\mathfrak{B}_{\cdot}^{\mu[A]\xi}$ and consider their disjunction. The resulting equality is exact by construction and coincides with (2.7). The contradiction completes the proof of the lemma. Q.E.D.

We should stress again that dependence of the characteristic system obviously will not cause an inequality to change into exact equalities by substitution.

We will need some new definitions for the following proofs. We denote by σ the length of the representing morphism corresponding to the variables from the alphabet $\overline{X} \cup \overline{Y}$, i.e.,

$$\sigma = \max_{Z \in \overline{X} \cup \overline{Y}} |\langle Z \rangle|,$$

and by λ^0 the maximum length of the strings from Γ^* entering the rules of the automata $\mathfrak{A}$ and $\mathfrak{B}$, i.e., $\lambda^0 = \max\{\lambda_{\mathfrak{A}}, \lambda_{\mathfrak{B}}\}$, where $\lambda_{\mathfrak{A}} = \max_{\alpha, R}\{m + 1 \mid (a, \alpha, R) \to (\beta, RT_m \dots T_1) \in \mathfrak{U}_{\cdot}^{\alpha R}\}$ and $\lambda_{\mathfrak{B}}$ is defined similarly.

LEMMA 12. Successive application of replacement and aggregation operations may increase the norm of the morphism by not more than $\sigma\lambda^0 - 1$.

Indeed, for an arbitrary strict RTDPA $\mathfrak{A}$, the norm of any variable $X \in \overline{X}$ satisfies the inequality $1 \leq \|X\| \leq \sigma$. As a result of replacement of the variable X and aggregation, we can substitute the expression $\mathfrak{U}_{\cdot}^{\beta RT_m \dots T_1}$ for X, and $\|\mathfrak{U}_{\cdot}^{\beta RT_m \dots T}\| \leqslant \sigma\lambda^{0i}$. Therefore, the norm of the morphism may increase at most by $\sigma\lambda^0 - 1$. Q.E.D.

LEMMA 13. Substitution of a representating morphism will reduce the length of the characteristic of any expression $\mathfrak{B}^{\Phi\xi}$ by not more than σ.

Indeed, the maximum reduction of the length of the characteristic is achieved when each symbol of the word $x = \langle X \rangle$ erases only one symbol from the sequence ξ. Since $|x| \leq \sigma$, the length of ξ may be reduced by not more than σ. Q.E.D.

THEOREM 4. Assume that as a result of substitution the equality $\bigvee_{p=1}^{k} X_p \mathfrak{U}_p \doteq \mathfrak{B}_{\cdot}^{\Psi\xi}$ is replaced with the direct system of equalities $\overline{\mathfrak{U}_p \doteq \mathfrak{B}_{\cdot}^{\psi_p \zeta A \zeta_p \eta_p}}$ $(1 \leqslant p \leqslant k)$ and the inverse system of equalities $\bigvee_{p=1}^{k} X_p \mathfrak{B}_{[A]}^{\psi_p A \zeta_p \eta_p} \doteq \mathfrak{B}_{[A]}^{\psi A \xi}$. Then $|\zeta_p \eta_p| < (\lambda^0 + 1)\sigma$.

Indeed $|\zeta_p \eta_p|$ attains its maximum if substitution of the word $\langle X_{p_1} \rangle$ erases the largest possible piece of the characteristic ξ of length not exceeding σ, and substitution of another word $\langle X_{p_2} \rangle$ adjoins to ξ the longest sequence of length not exceeding $\sigma\lambda^0 - 1$ (Lemmas 12 and 13). Therefore, $|\zeta_p \eta_p| \leqslant \max_{1 \leqslant p \leqslant k} |\zeta_p \eta_p| < \sigma\lambda^0 + \sigma = (\lambda^0 + 1)\sigma$. Q.E.D.

4. PROOF OF THE MAIN THEOREMS

Proof of Theorem 2. Assume that the process of construction of the category system $\mathcal{K}$ stops after a finite number of steps. In this case, the category $\mathfrak{C} = \lim_{\to} \mathcal{K}$ will contain a finite number of objects and morphisms, which can be analyzed in order to decide whether the automata $\mathfrak{A}$ and $\mathfrak{B}$ are equivalent or not. If at least one negative condition was applied in the process of construction of the system $\mathcal{K}$, causing the comparison algorithm to stop, then we can apply the corresponding inequality in order to construct a word which is accepted by one automaton and is not accepted by the other automaton: by correctness of all operations, application of a negative condition implies that $\hat{X}_F^{q_0 Z_0} \neq \hat{Y}_F^{q_0 Z_0}$.

If during the construction of the system $\mathcal{K}$ the comparison algorithm stopped because no unmarked rules remained, then no word belongs to one of the languages $\hat{X}_F^{q_0 Z_0}$, $\hat{Y}_F^{q_0 Z_0}$ and does not belong to the other. Indeed, rules are marked in two cases: either an operation of procedure Π is applied or the rules $\langle r1 \rangle$, $\langle r2 \rangle$ are applied. In the first case, since the operations are correct, a word on which inequivalence of automata can be proved will not disappear. In the second case, when $\langle r1 \rangle$ is applied, equality of morphisms is effectively testable, and the application of $\langle r2 \rangle$ has been previously considered in the analysis of the comparison algorithm, when we showed that marking of equalities by rule $\langle r2 \rangle$ also does not eliminate inequivalence-proving words. Since all the equalities are nevertheless marked, no inequivalence-proving words exist and the automata are equivalent.

We will now show that for any strict RTDPA $\mathfrak{A}$ and $\mathfrak{B}$ the comparison algorithm stops after a finite number of steps. Let

$$\mathfrak{W}^{(m)} \doteq \mathfrak{W}^{(n)} \tag{4.1}$$

be an arbitrary unmarked equality in R_k. The numbers m and n are the lengths of the characteristics of the expressions $\mathfrak{W}^{(m)}$ and $\mathfrak{W}^{(n)}$ respectively. By the comparison algorithm, the pair of morphisms corresponds either to a wavy or a broken arrow. In the first case, the substitution operation is applied to (4.1) and in the second the replacement and aggregation operations, followed by the substitution operation. During substitution, if the length of the characteristic of the expression in the right-hand side is sufficiently large, then operations with pushdown memory defined by a representing morphism affect only a finite piece of this characteristic, and therefore the length of the characteristic associated with the left-hand expression in each equality of the inverse system is also bounded. The characteristic of the left-hand expression in each equality of the direct system diminishes by 1 with each application of the substitution operation, and may be reduced to 1 in finitely many steps, producing a linear expression in the left-hand side.

Since the characteristic of the left-hand expression of each equality associated with strict RTDPA is of bounded length, the characteristic of the right-hand expression is also of bounded length, and therefore the number of possible equalities determined by characteristics of fixed length is jointly bounded during the application of the comparison algorithm. Since one of the equalities is marked as used in each step, all the equalities will have been marked after a finite number of steps and the algorithm will stop.

This reasoning can be formalized as follows. Let Λ be the number $\Lambda = (\lambda^0 + 1)(\sigma + 1) + 2$. Apply the comparison algorithm to equalities (4.1) while $m < \Lambda - \lambda^0$. Application of the operations of procedure Π naturally increase the lengths of the characteristics m, n, and in some step of construction of the system $\mathscr{K}$ either the characteristic m falls in the interval

$$\Lambda - \lambda^0 \leqslant m \leqslant \Lambda - 1, \tag{4.2}$$

or the operations of procedure Π are no longer applicable and the comparison algorithm stops. In the second case, finiteness of the category $\varinjlim \mathscr{K}$ is obvious and the theorem is true. Therefore let us consider the case when inequality (4.2) holds. For the direct system, substitution operations successively reduce the number m, pushing it outside the left bound in (4.2). Regarding the inverse system, the length of the characteristic of each left-hand expression in any equality does not exceed $\Lambda - \lambda^0$ by Theorem 4, and after replacement and aggregation it does not exceed $\Lambda - 1$ (Lemma 12), i.e., it also satisfies inequality (4.2). Thus, the number Λ limits the length of the characteristics of the left-hand expressions entering any equality included in the set R_i for any i. Then from the definition of the norm, we obtain

$$\| \mathfrak{W}^{(m)} \| < \Lambda\sigma, \tag{4.3}$$

because each symbol of the characteristic of the expression $\mathfrak{W}^{(m)}$ is associated with a variable whose norm does not exceed σ. The length n of the characteristic of the expression $\mathfrak{W}^{(n)}$ may not exceed $\Lambda\sigma$ on the assumption that each symbol of the characteristic of the expression $\mathfrak{W}^{(m)}$ is associated with a variable of norm 1. Thus,

$$n < \Lambda\sigma. \tag{4.4}$$

Inequalities (4.2) and (4.4) limit the total number of equalities which are obtained by applying the comparison algorithm to categories of the system $\mathscr{K}$. Therefore the process of construction of the system $\mathscr{K}$ terminates after finitely many steps, and the equivalence problem of strict RTDPA is decidable under the conditions of Theorem 2. Q.E.D.

Proof of Theorem 3. Let $\mathfrak{A}$ and $\mathfrak{B}$ be strict RTDPA. Apply the comparison algorithms to these automata for as long as possible. If the negative stopping conditions were not activated in the process, then the automata are equivalent, because none of the operations transforms an inequality into a system of equalities and the reasoning applied in the proof of Theorem 2 holds in this case also.

If the comparison algorithm stopped due to application of one of the negative rules, then two alternatives are possible. First, the algorithm has constructed a word x that belongs to only one of the languages $\hat{X}_F^{q_0 Z_0}$, $\hat{Y}_F^{q_0 Z_0}$. In this case, the automata $\mathfrak{A}$ and $\mathfrak{B}$ are inequivalent and the problem has been solved in a finite number of steps. In the second case, the appearance of the word x is attributable to incorrectness of one of the previously applied substitution operations, and this incorrectness was determined by violation of the independence assumption of the characteristic system κ. Then the word obtained in this way may be used to establish dependence as in the proof of Lemma 11, i.e., for the characteristic system $\varkappa = \{\mathfrak{B}_{\cdot}^{\mu[A]_j\zeta} \mid 0 \leqslant j \leqslant J\}$ of equality (2.7), we can obtain from (2.7) the equality

$$\mathfrak{B}_{\cdot}^{\mu[A]_0\zeta} \doteq \bigvee_{j=0}^{J} \mathfrak{B}_{[A]_j}^{\Phi A\xi_0} \mathfrak{B}_{\cdot}^{\mu[A]_j\zeta}. \tag{4.5}$$

Here three cases are possible. The first when $\mathfrak{B}_{[A]_j}^{\Phi A\xi_0} = \varnothing$ for all $j \neq k$, $\mathfrak{B}_{[A]_k}^{\Phi A\xi_0} = \varepsilon$. The second when $\mathfrak{B}_{[A]_0}^{\Phi A\xi_0} = \varnothing$, and (4.5) has the form

$$\mathfrak{B}_{\cdot}^{\mu[A]_0\zeta} \doteq \bigvee_{j=1}^{J} \mathfrak{B}_{[A]_j}^{\Phi A\xi_0} \mathfrak{B}_{\cdot}^{\mu[A]_j\zeta} \tag{4.5.1}$$

and finally the third when $\| \mathfrak{B}_{[A]_0}^{\Phi A\xi_0} \| \geqslant 1$.

In the first and second case, the element $\mathfrak{B}_{\cdot}^{\mu[A]_0\zeta}$ is eliminated from the system κ, which is replaced with the system $\kappa' = \{\mathfrak{B}_{\cdot}^{\mu[A]_j\zeta} \mid 1 \leqslant j \leqslant J\}$. For the first case, we respectively eliminate from the inverse system two equalities $\bigvee_p X_p \mathfrak{B}_{[A]_l}^{\Psi_p A\zeta_p\eta_p} \doteq \mathfrak{B}_{[A]_l}^{\Psi A\xi}$, where i = 0, k, and introduce a new equality $\bigvee_p X_p (\mathfrak{B}_{[A]_0}^{\Psi_p A\zeta_p\eta_p} \vee \mathfrak{B}_{[A]_k}^{\Psi_p A\zeta_p\eta_p}) \doteq \mathfrak{B}_{[A]_0}^{\Psi A\bar{\xi}} \vee \mathfrak{B}_{[A]_k}^{\Psi A\bar{\xi}}$, which is an analog of equality (2.8.1).

In the second case, the inverse system is also changed. Substituting for $\mathfrak{B}^{\mu[A]_0\zeta}$ in (2.7) its expression from (4.5.1), we obtain

$$\begin{aligned} &\bigvee_p X_p \mathfrak{B}_{[A]_0}^{\Psi_p A\zeta_p\eta_p} \bigvee_{j=1}^{J} \mathfrak{B}_{[A]_j}^{\Phi A\xi_0} \mathfrak{B}_{\cdot}^{\mu[A]_j\zeta} \vee \bigvee_{j=1}^{J} \bigvee_p X_p \mathfrak{B}_{[A]_j}^{\Psi_p A\zeta_p\eta_p} \times \\ &\times \mathfrak{B}_{\cdot}^{\mu[A]_j\zeta} = \bigvee_{j=1}^{J} \bigvee_p X_p (\mathfrak{B}_{[A]_j}^{\Psi_p A\zeta_p\eta_p} \vee \mathfrak{B}_{[A]_0}^{\Psi_p A\zeta_p\eta_p} \mathfrak{B}_{[A]_j}^{\Phi A\xi_0}) \times \\ &\times \mathfrak{B}_{\cdot}^{\mu[A]_j\zeta} \doteq \bigvee_{j=1}^{J} (\mathfrak{B}_{[A]_j}^{\Psi A\bar{\xi}} \vee \mathfrak{B}_{[A]_0}^{\Psi A\bar{\xi}} \mathfrak{B}_{[A]_j}^{\Phi A\xi_0}) \mathfrak{B}_{\cdot}^{\mu[A]_j\zeta}, \end{aligned} \tag{2.7.1}$$

whence the new inverse system is given by

$$\begin{gathered} \bigvee_p X_p (\mathfrak{B}_{[A]_j}^{\Psi_p A\zeta_p\eta_p} \vee \mathfrak{B}_{[A]_0}^{\Psi_p A\zeta_p\eta_p} \mathfrak{B}_{[A]_j}^{\Phi A\xi_0}) \doteq \mathfrak{B}_{[A]_j}^{\Psi A\bar{\xi}} \vee \mathfrak{B}_{[A]_0}^{\Psi A\bar{\xi}} \mathfrak{B}_{[A]_j}^{\Phi A\xi_0}, \\ (1 \leqslant j \leqslant J), \end{gathered} \tag{2.8.2}$$

which contains one equality fewer than the preceding inverse system constructed for the characteristic system κ.

In the third case, equality (4.5) has the form

$$\mathfrak{B}_{\cdot}^{\mu[A]_0\zeta} \doteq \mathfrak{B}_{[A]_0}^{\Phi A\xi_0} \mathfrak{B}_{\cdot}^{\mu[A]_0\zeta} \vee \bigvee_{j=1}^{J} \mathfrak{B}_{[A]_j}^{\Phi A\xi_0} \mathfrak{B}_{\cdot}^{\mu[A]_j\zeta}$$

or

$$\mathfrak{B}_{\cdot}^{\mu[A]_0\zeta} \doteq \{\mathfrak{B}_{[A]_0}^{\Phi A\xi_0}\}^* \bigvee_{j=1}^{J} \mathfrak{B}_{[A]_j}^{\Phi A\xi_0} \mathfrak{B}_{\cdot}^{\mu[A]_j\zeta}, \tag{4.6}$$

where the symbol { }* stands for iteration of the event between braces.

Introduce a new variable $Z_{[A]_0}^{\Phi A\xi_0}$, defined by the exact equality

$$Z_{[A]_0}^{\Phi A\xi_0} = \mathfrak{B}_{[A]_0}^{\Phi A\xi_0} Z_{[A]_0}^{\Phi A\xi_0} \vee \varepsilon. \tag{4.7}$$

Then (4.6) may be rewritten in the form

$$\mathfrak{B}_{\cdot}^{\mu[A]_0\zeta} \doteq Z_{[A]_0}^{\Phi A\xi_0} \bigvee_{j=1}^{J} \mathfrak{B}_{[A]_j}^{\Phi A\xi_0} \mathfrak{B}_{\cdot}^{\mu[A]_j\zeta}. \tag{4.8}$$

Substituting (4.8) in (2.7), we obtain

$$\bigvee_p \bigvee_{j=1}^{J} X_p (\mathfrak{B}_{[A]_j}^{\Psi_p A\zeta_p\eta_p} \vee \mathfrak{B}_{[A]_0}^{\Psi_p A\zeta_p\eta_p} Z_{[A]_0}^{\Phi A\xi_0} \mathfrak{B}_{[A]_j}^{\Phi A\xi_0}) \mathfrak{B}_{\cdot}^{\mu[A]_j\zeta} \doteq \bigvee_{j=1}^{J} (\mathfrak{B}_{[A]_j}^{\Psi A\bar{\xi}} \vee \mathfrak{B}_{[A]_0}^{\Psi A\bar{\xi}} Z_{[A]_0}^{\Phi A\xi_0} \mathfrak{B}_{[A]_j}^{\Phi A\xi_0}) \mathfrak{B}_{\cdot}^{\mu[A]_j\zeta}. \tag{4.9}$$

In (4.9), the variable $Z_{[A]_0}^{\Phi A\xi_0}$ is necessarily followed by one of the expressions $\mathfrak{B}_{[A]_j}^{\Phi A\xi_0}$ $(1 \leqslant j \leqslant J)$. Therefore, introduce J variables $Z_{0j}^{\Phi A\xi_0} = Z_{[A]_0}^{\Phi A\xi_0} \cdot \mathfrak{B}_{[A]_j}^{\Phi A\xi_0}$ with the norm $\| Z_{0j}^{\Phi A\xi_0} \| = \| \mathfrak{B}_{[A]_j}^{\Phi A\xi_0} \|$, defined by the equations

$$Z_{0j}^{\Phi A\xi_0} = \mathfrak{B}_{[A]_0}^{\Phi A\xi_0} Z_{0j}^{\Phi A\xi_0} \vee \mathfrak{B}_{[A]_j}^{\Phi A\xi_0} \tag{4.10}$$

and rewrite (4.9) in the form

$$\bigvee_p \bigvee_{j=1}^{J} X_p (\mathfrak{B}_{[A]_j}^{\psi_p A \zeta_p \eta_p} \vee \mathfrak{B}_{[A]_0}^{\psi_p A \zeta_p \eta_p} Z_{0j}^{\varphi A \xi_0}) \mathfrak{B}_{\cdot}^{\mu[A]_j \zeta} \doteq \bigvee_{j=1}^{J} (\mathfrak{B}_{[A]_j}^{\psi A \bar{\xi}} \vee \mathfrak{B}_{[A]_0}^{\psi A \bar{\xi}} Z_{0j}^{\varphi A \xi_0}) \mathfrak{B}_{\cdot}^{\mu[A]_j \zeta}. \tag{4.11}$$

Equality (4.11) with the characteristic system $\varkappa' = \{\mathfrak{B}_{\cdot}^{\mu[A]_j \zeta} \mid j \geqslant 1\}$ replaces equality (2.7) and is continued to be used instead of (2.7) in the comparison algorithm. From (4.11) we obtain the inverse system

$$\bigvee_p X_p (\mathfrak{B}_{[A]_j}^{\psi_p A \zeta_p \eta_p} \vee \mathfrak{B}_{[A]_0}^{\psi_p A \zeta_p \eta_p} Z_{0j}^{\varphi A \xi_0}) \doteq \mathfrak{B}_{[A]_j}^{\psi A \bar{\xi}} \vee \mathfrak{B}_{[A]_0}^{\psi A \bar{\xi}} Z_{0j}^{\varphi A \xi_0},$$

($1 \leq j \leq J$), which replaces the system (2.8).

If the system $\varkappa'$ subsequently proves to be dependent and the comparison algorithm produces a word that establishes this dependence, then the entire process of elimination of the dependent component from the system $\varkappa'$ is repeated. However, the number of such dependences is finite because any characteristic system is finite. The variables $Z_{k\cdot}^{\varphi\xi}$ will be called auxiliary variables.

The existence of incorrect operations requires modification of the comparison algorithm in order to allow for the possible stopping of the algorithm due to incorrectness of some operations and to determine the subsequent actions of the algorithm in this case. This modified algorithm includes a procedure that checks whether the words defining the inequality $X_F^{q_0 Z_0} \neq Y_F^{q_0 Z_0}$ are accepted by the automata $\mathfrak{A}$ and $\mathfrak{B}$, and if the words are accepted by both automata (i.e., an incorrect substitution operation has been used) then it applies these words to find the dependence in the characteristic system, because substitution is correct for an independent system.

If the incorrect substitution operation was applied on passing from category $\mathfrak{C}_k$ to $\mathfrak{C}_{k+1}$, then after eliminating the dependent component and changing the inverse system, the comparison algorithm is reapplied to category $\mathfrak{C}_k$, and all the other categories $\mathfrak{C}_l$ ($l > k$) are omitted. Let us analyze how the introduction of auxiliary variables affects the application of the modified comparison algorithm.

Auxiliary variables clearly do not affect the substitution operations of variables from the alphabet $\bar{X} \cup \bar{Y}$ and subsequent aggregation and decomposition operations. Consider the substitution operation. The presence of auxiliary variables is immaterial for the substitution of a representative morphism in the left-hand side of any equality, because the variables Z are always preceded by basic variables, and substitution is applied to the variables entering an expression of the form $X_1 \vee X_2 Z$ only after replacement of the variables X_1, X_2. If a case arises when we need to evaluate $\int x \mid Z\mathfrak{B}$, then first Z is replaced using (4.10) and then the corresponding substitution is computed.

Let us apply the modified algorithm to solve the equivalence problem for a pair of automata. Construct the category system $\mathcal{K}_1$ by successively applying the comparison algorithm. This system is finite. If only rules ⟨r1⟩ and ⟨r2⟩ were applied during its construction, then the automata are equivalent. Otherwise, a word x_1 can be found corresponding to the negative stopping condition.

Check the inclusion $x_1 \in \hat{X}_F^{q_0 Z_0}$, $x_1 \in \hat{Y}_F^{q_0 Z_0}$. If one of the inclusions is false, then the automata are inequivalent. If both inclusions hold, then use x_1 to find the corresponding dependence and from this point reapply the comparison algorithm to construct the category system $\mathcal{K}_2$, i.e., until the comparison algorithm stops again.

Since the total number of possible dependences is finite, either the modified comparison algorithm will produce in some step the category system $\mathcal{K}_N$ that establishes equivalence of the automata or a word x_N will be found which is accepted by one automaton only, so that the two automata are inequivalent.

Since this result can be obtained in finitely many steps, the equivalence problem is decidable for strict RTDPA. Q.E.D.

In conclusion, I would like to acknowledge the valuable comments of L. P. Lisovik.

LITERATURE CITED

1. A. Korenjak and D. Hopcroft, "Simple deterministic languages," in: Languages and Automata [Russian translation], Mir, Moscow (1975), pp. 71-96.
2. V. Yu. Romanovskii, "Equivalence problem for strict real-time deterministic pushdown automata," Kibernetika, No. 5, 49-59 (1980).
3. M. Oyamaguchi, Y. Inagaki, and N. Honda, "The equivalence problem for real-time strict deterministic languages," Inform. Contr., **45**, No. 1, 90-115 (1980).
4. V. Yu. Romanovskii, "The equivalence problem in real-time deterministic pushdown automata," Kibernetika, No. 2, 12-23 (1986).

5. A. A. Letichevskii, "Representation of context-free languages in automata with pushdown memory," Kibernetika, No. 2, 80-84 (1965).
6. V. Yu. Meitus, "Transforming categories and finite transformers, I," Kibernetika, No. 2, 26-34 (1978).
7. P. M. Cohn, Universal Algebra, Harper and Row, New York (1965).
8. I. Bucur and A. Deleanu, Introduction to the Theory of Categories and Functors, Wiley, New York (1968).

FAULT DETECTION IN CASCADED AUTOMATA

M. A. Spivak and D. E. Chernyi

UDC 519.713.4

A new method is proposed for fault detection in cascaded automata, which does not require finding the transition-output tables of the cascade and essentially allows for the specific structure of the cascade connections.

Consider two finite automata, $A = (S, X, Y, \delta_1, \lambda_1)$ and $B = (T, Y \times Z, U, \delta_2, \lambda_2)$. A cascade connection of the automata A and B is a new automaton $A \textcircled{1} B = (S \times T, X \times Z, U, \delta, \lambda)$ defined by the diagram of Fig. 1 or by the formulas

$$\begin{aligned}\delta((s, t), (x, z)) &= (\delta_1(s, x), \delta_2(t, \lambda_1(s, x), z)),\\ \lambda((s, t), (x, z)) &= \lambda_2(t, \lambda_1(s, x), z).\end{aligned} \tag{1}$$

Let X^* be the set of all words in the alphabet X, including the empty word e. The following formula is derived from (1) by induction on the length of the word $(p, r) \in (X \times Z)^*$:

$$\lambda((s, t), (p, r)) = \lambda_2(t, \lambda_1(s, p), r).$$

If the input alphabet Z consists of a single element, i.e., the input channel Z of the automaton B is actually fictitious, the cascade connection of the automata A and B is essentially the superposition of these automata (see [1]).

Assume that possible faults of the automata A and B (including the case when the automata A and B are nonfaulty) are described by the classes of automata $K = \{A_i \mid i \in I\}$ and $L = \{B_j \mid j \in J\}$, where $A_i = (S_i, X, Y, \delta_{1i}, \lambda_{1i})$ and $B_j = (T_j, Y \times Z, U, \delta_{2j}, \lambda_{2j})$. From the general definition of an experiment identifying an automaton in a specified class of automata (see [2]), the word $(p, r) \in (X \times Z)^*$ will be called an identifying word for the class of automata $K \textcircled{1} L = \{A_i \textcircled{1} B_j \mid (i, j) \in I \times J\}$ if

$$\begin{aligned}&(\forall (i_1, j_1), (i_2, j_2) \in I \times J)(\forall (s, t) \in S_{i_1} \times T_{j_1})\\ &(\forall (s', t') \in S_{i_2} \times T_{j_2})((i_1, j_1) \neq (i_2, j_2) \Rightarrow\\ &\Rightarrow \lambda_{2j_1}(t, \lambda_{1i_1}(s, p), r) \neq \lambda_{2j_2}(t', \lambda_{1i_2}(s', p), r)).\end{aligned} \tag{2}$$

Consider a simple preset experiment with an unknown automaton $A_i \textcircled{1} B_j$ from the class $K \textcircled{1} L$ (see [2]), which involves applying to the automaton input a word (p, r) that satisfies condition (2) and observing the corresponding output word $\lambda_{2j}(t, \lambda_{1i}(s, p), r)$. It is easy to see that the result of this experiment will identify the specific fault in each component of the automaton.

Identifying words for the class of automata $K \textcircled{1} L$ can be determined by Gill's standard method [2]. However, Gill's method has two significant shortcomings: 1) it requires constructing transition–output tables of all the automata $A_i \textcircled{1} B_j$; 2) it totally ignores additional information concerning the structure of the automaton $A_i \textcircled{1} B_j$ as a cascade connection of two automata. An alternative algorithm is known, which will find an identifying word for the class of automata $K \textcircled{1} L$ and at the same time is free from these shortcomings. This algorithm enumerates a certain finite (although very large) set of words

Translated from Kibernetika, No. 5, pp. 26-31, September-October, 1989. Original article submitted January 9, 1988.

0011-4235/89/2505-0594$12.50