Acta Math. Vietnam., 1976, Tom 1, № 2, pp 47 - 66.

PUSHDOWN AUTOMATA WITH MANY PUSHDOWN STORE TAPES

ĐỖ LONG VÂN and PHAN ĐÌNH DIỆU
Institute of Mathematics, Hanoi

§1. INTRODUCTION

As is well-known, the notion of pushdown store was introduced since 1954 in [11] by Burks, Warren and Wright, and then it was used in machine translation. Some years later, this notion was formalized in pushdown acceptors by Chomsky [1] in finding the machine model of context-free languages. From that time, the pushdown automata (with one pushdown store tape) were investigated extensively by many authors, either as a subject of automata theory or a means of mathematical theory of languages (see [1] - [10]).

The conception of pushdown stores is used widely in many theoretical and practical domains of computer science. And in many cases, it is useful to organize such memory that consists of not only a single pushdown store tape but many ones together. Therefore, it is of interest to study a mathematical model of pushdown automata with some pushdown store tapes. In this paper, such a model is considered, and its capability and properties are investigated. The definitions and necessary notions are introduced in §2. In §3 we shall examine some closure properties of classes of languages under consideration. In §4 a necessary condition for languages recognizable by deterministic pushdown automata with many pushdown store tapes in real time will be formulated. Relations between classes of languages under consideration will be established in §5. Finally, §6 is devoted to proving the unsolvability of some algorithmic problems.

§2. DEFINITIONS AND NECESSARY NOTIONS

2.1. We recall some notions. Let Σ be a finite alphabet. We denote the set of all words in the alphabet Σ by Σ^*, the empty word by $\wedge$ and $\Sigma^+ = \Sigma^* \setminus \{\wedge\}$. Every subset $L \subseteq \Sigma^*$ is called a language over the alphabet

Σ. For any word $u \in \Sigma^*$, $l(u)$ denotes the length of u, $\hat{u}$ is the word obtained from u by writing the letters of u in reverse order, e.g., if $u = \sigma_1 \sigma_2 \dots \sigma_n$ then $\hat{u} = \sigma_n \sigma_{n-1} \dots \sigma_1$, $\hat{\Lambda} = \Lambda$. Let A be a set the elements of which are some letters or some words over some alphabet Σ. In the sequel we shall use the notation A^n to denote either Cartesian product of A with itself n times, then the members of A^n will be the n-tuples of elements of A - or concatenation of language A with itself n times - then the members of A^n will be words over the alphabet Σ. Which meaning this notation has in every case will be clear by context. The cardinality of a set A is denoted by $|A|$.

2.2. A (nondeterministic) *pushdown automata with n pushdown store tapes* (abbreviated nP) is a 7-tuple

$$A = \langle S, \Sigma, \Gamma, \delta, s_0, \overline{Z}_0, F \rangle$$

where

S is a nonempty finite set of *states*,

Σ is a nonempty finite set of *input symbols* (the input alphabet),

Γ is a nonempty finite set of *pushdown symbols*,

$s_0 \in S$ is the *initial state*,

$\overline{Z}_0 = (Z_{01}, \dots, Z_{0n}) \in \Gamma^n$, where Z_{0i} $(i = 1, \dots, n)$ is called the *initial pushdown symbol* of the i-th pushdown store tape,

$F \subseteq S$ is the set of *final states*,

δ is a map from $S \times \Sigma \times \Gamma^n$ into the family of the finite subsets of $S \times (\Gamma^*)^n$ and called the *transition function*. Moreover, in order that the pushdown store tapes never become empty during working process, the transition function δ must satisfy this condition: if $(s', \overline{\mathrm{Y}}) \in \delta(s, \sigma, \overline{Y})$, then $(\forall i)\ (1 \leqslant i \leqslant n)\ (Y_i = Z_{0i} \rightarrow \mathrm{Y}_i \in \{Z_{0i}\}\ \Gamma^*)$, where $\overline{\mathrm{Y}} = (\mathrm{Y}_1, \dots, \mathrm{Y}_n)$, $\overline{Y} = (Y_1, \dots, Y_n)$.

Every member of $S \times \Sigma^* \times (\Gamma^*)^n$ is called a *configuration* of automaton A.

In the set of configurations of A, we introduce the binary relation $\vdash_{A}$ called *configuration transition* which is defined as follows: for any $s, s' \in S$; $\sigma \in \Sigma$, $w \in \Sigma^*$; $\bar{\alpha}, \bar{Y} \in (\Gamma^*)^n$; $\bar{Y} \in \Gamma^n$ if $(s', \bar{\Upsilon}) \in \delta(s, \sigma, \bar{Y})$, then $(s, \sigma w, \bar{\alpha}\, \bar{Y}) \vdash_{A} (s', w, \bar{\alpha}\, \bar{\Upsilon})$, where $\bar{\alpha}\bar{\beta} = (\alpha_1, \ldots, \alpha_n)(\beta_1, \ldots, \beta_n) = (\alpha_1\beta_1, \ldots, \alpha_n\beta_n)$.

The reflexive and transitive closure of relation $\vdash_{A}$ is denoted by $\overset{*}{\vdash}_{A}$.

We say that the word $w \in \Sigma^*$ is *accepted* (*in realtime*) *by* A if there exist $\bar{\alpha} \in (\Gamma^*)^n$ and $s \in F$ such that

$$(s_0, w, \bar{Z}_0) \overset{*}{\vdash}_{A} (s, \Lambda, \bar{\alpha})$$

The set of all words accepted by A is called *the language recognizable by* A and denoted by $L(A)$.

A pushdown automaton with n pushdown store tapes A is called a *deterministic* one (abbreviated nDP) if for any $(s, \sigma, \bar{Y}) \in S \times \Sigma \times \Gamma^n$, the set $\delta(s, \sigma, \bar{Y})$ consists of one and only one element.

In the sequel we shall need the following notations:

For any natural number $n \geqslant 1$, $\mathcal{A}_{nP}[\Sigma]$ denotes the class of all nP's with the alphabet Σ.

$$\mathcal{L}_{nP}[\Sigma] = \left\{ L \subseteq \Sigma^* \;\middle|\; \exists A \in \mathcal{A}_{nP}[\Sigma] : \; L(A) = L \right\}$$

$$\mathcal{A}_{nP} = \bigcup_{\Sigma} \mathcal{A}_{nP}[\Sigma] \; ; \; \mathcal{A}_P = \bigcup_{n=1}^{\infty} \mathcal{A}_{nP}$$

$$\mathcal{L}_{nP} = \bigcup_{\Sigma} \mathcal{L}_{nP}[\Sigma] \; ; \; \mathcal{L}_P = \bigcup_{n=1}^{\infty} \mathcal{L}_{nP} .$$

The notations $\mathcal{A}_{nDP}[\Sigma]$, $\mathcal{L}_{nDP}[\Sigma]$, $\mathcal{A}_{nDP}$, $\mathcal{A}_{DP}$, $\mathcal{L}_{nDP}$, $\mathcal{L}_{DP}$ are introduced similarly.

The following inclusions are immediate from the definitions:

$$\mathcal{L}_{nDP} \subseteq \mathcal{L}_{(n+1)DP} \tag{2.1}$$

$$\mathcal{L}_{nP} \subseteq \mathcal{L}_{(n+1)P} \tag{2.2}$$

$$\mathcal{L}_{nDP} \subseteq \mathcal{L}_{nP} \tag{2.3}$$

§ 3. SOME CLOSURE PROPERTIES

3.1. THEOREM 1. (i) *If* $L \in \mathcal{L}_{nDP}[\Sigma]$ *then* $\overline{L} = \Sigma^* \setminus L \in \mathcal{L}_{nDP}[\Sigma]$. (ii) *If* $L_1 \in \mathcal{L}_{nDP}[\Sigma]\left(\mathcal{L}_{nP}[\Sigma]\right)$ *and* $L_2 \in \mathcal{L}_{mDP}[\Sigma]\left(\mathcal{L}_{mP}[\Sigma]\right)$ *then* $L_1 \cap L_2$, $L_1 \cup L_2 \in \mathcal{L}_{(n+m)DP}[\Sigma]\left(\mathcal{L}_{(n+m)P}[\Sigma]\right)$.

Proof. (i) It is evident that if $A = \langle S, \Sigma, \Gamma, \delta, s_o, \overline{Z}_o, F \rangle$ is in $\mathcal{A}_{nDP}[\Sigma]$ such that $L(A) = L$ then $\overline{A} = \langle S, \Sigma, \Gamma, \delta, s_o, \overline{Z}_o, S \setminus F \rangle$ is also in $\mathcal{A}_{nDP}[\Sigma]$ and $L(\overline{A}) = \overline{L}$.

(ii) Let us have $L_1 = L(A_1)$, $L_2 = L(A_2)$, where

$A_1 = \langle S_1, \Sigma, \Gamma_1, \delta_1, s_o, \overline{X}_o, F_1 \rangle$ and $A_2 = \langle S_2, \Sigma, \Gamma_2, \delta_2, t_o, \overline{Y}_o, F_2 \rangle$ are in $\mathcal{A}_{nDP}[\Sigma]$ and $\mathcal{A}_{mDP}[\Sigma]$ respectively. We construct an automaton $A \doteq \langle S, \Sigma, \Gamma, \delta, u_o, \overline{Z}_o, F \rangle$, where $S = S_1 \times S_2 \cup \{R\}$, R is a new symbol called the refusing state of A, $\Gamma = \Gamma_1 \cup \Gamma_2$, $u_o = (s_o, t_o)$, $\overline{Z}_o = (\overline{X}_o, \overline{Y}_o)$(1), $F = F_1 \times F_2$; the transition function δ is defined as follows:

1) $\delta((s,t), \sigma, (\overline{X}, \overline{Y})) = ((s', t'), (\overline{\alpha}, \overline{\beta}))$ if $\overline{X} \in \Gamma_1^n$, $\overline{Y} \in \Gamma_2^m$ and $\delta_1(s, \sigma, \overline{X}) = (s', \overline{\alpha})$, $\delta_2(t, \sigma, \overline{Y}) = (t', \overline{\beta})$.

2) $\delta(s, \sigma, \overline{Z}) = (R, \overline{Z})$ in all other cases.

It is evident that $A \in \mathcal{A}_{(n+m)DP}[\Sigma]$ and $L(A) = L_1 \cap L_2$, that is $L_1 \cap L_2 \in \mathcal{L}_{(n+m)DP}[\Sigma]$.

The (n + m) DP recognizing the language $L_1 \cup L_2$ is constructed similarly, only now $F = F_1 \times S_2 \cup S_1 \times F$.

In the case of nondeterministic languages the proof is made in the same way.

(1) Here $(\overline{X}_o, \overline{Y}_o)$ stands for $(X_{o1}, \ldots, X_{on}, Y_{o1}, \ldots, Y_{om})$

From Theorem 1 we have

COROLLARY. (i) *The class $\mathcal{L}_{DP}[\Sigma]$ and, therefore, the class $\mathcal{L}_{DP}$ are closed under union, intersection and complementation.*

(ii) *The class $\mathcal{L}_{P}[\Sigma]$ and, therefore, the class $\mathcal{L}_{P}$ are closed under union and intersection.*

3.2. Haines [12] and Greibach [13] have proved that

$$\mathcal{L}_{1P} = \mathcal{L}_{cF}\,, \tag{3.1}$$

where $\mathcal{L}_{cF}$ denotes the class of all context-free languages. Hence, by (2.3)

$$\mathcal{L}_{1DP} \subseteq \mathcal{L}_{cF}\,. \tag{3.2}$$

We now consider the following languages over the alphabet $\{a\,,\,b\}$

$$\begin{aligned} L_1 &= \left\{ a^m\, b^n\, a^n \,\middle|\, n,\, m \geqslant 1 \right\} \\ L_2 &= \left\{ a^n\, b^n\, a^m \,\middle|\, n,\, m \geqslant 1 \right\} \end{aligned} \tag{3.3}$$

It is not difficult to construct the 1DPs recognizing the languages L_1 and L_2, that is, L_1 and L_2 belong to the class $\mathcal{L}_{1DP}$.

Obviously also

$$L_1 \cap L_2 = \left\{ a^n\, b^n\, a^n \,\middle|\, n \geqslant 1 \right\}. \tag{3.4}$$

On the other hand, as has been known (see, e.g.,[10]) the language $L = \left\{ a^n\, b^n\, a^n \,\middle|\, n \geqslant 1 \right\}$ is not context-free. Thus *the class $\mathcal{L}_{1DP}$ is not closed under intersection.* From this and (i) of Theorem 1 it follows that *the class $\mathcal{L}_{1DP}$ is not closed under union also.* However, as in the case of context-free languages, it is not difficult to prove the following fact :

THEOREM 2. *If $L_1 \in \mathcal{L}_{nDP}\left(\mathcal{L}_{nP}\right)$ and $L_2 \in \mathcal{L}_R$, where $\mathcal{L}_R$ is the class of all regular languages, then $L_1 \cap L_2\,,\ L_1 \cup L_2\,,\ L_1 \setminus L_2 \in \mathcal{L}_{nDP}\left(\mathcal{L}_{nP}\right)$*

3.3. It has been shown in [12] that for every recursively enumerable language $E \subseteq \Sigma^*$ there exist two deterministic context-free languages $L_1, L_2 \subseteq (\Sigma')^+$ and a homomorphism $h\,:\,(\Sigma')^* \rightarrow \Sigma^*$ such that

$$E = h\ (L_1 \cap L_2). \tag{3.5}$$

By (3.1), L_1, $L_2 \in \mathcal{L}_{1P}$, and therefore by (ii) of Theorem 1, $L_1 \cap L_2 \in \mathcal{L}_{2P}$. Thus, *every recursively enumerable language is the homomorphism image of some language in the class* $\mathcal{L}_{2P}$.

Since any language of the class $\mathcal{L}_P$ is recursive, from the existence of recursively enumerable but not recursive languages it follows that, for any $n \geqslant 2$, the class $\mathcal{L}_{nP}$ is not closed with respect to homomorphisms. However, in the following we shall show a special class of homomorphisms, under which these classes are closed.

Let Σ and Σ' be two arbitrary alphabets. Each one-to-one map τ from Σ into the set $(\Sigma')^k$ of all words of length k in Σ' $(k \geqslant 1)$ is called a *length k-encoding* or a *k-encoding of Σ into Σ'*. Note that such k-encoding always exists provided $|\Sigma'| \geqslant 2$. It is also clear that each k-encoding of Σ into Σ' produces a monoid isomorphism from Σ^* into $(\Sigma')^*$, which is defined by

$$\begin{aligned} \tau(\Lambda) &= \Lambda \\ \tau(w\sigma) &= \tau(w)\ \tau(\sigma), \end{aligned} \tag{3.6}$$

where $w \in \Sigma^*$, $\sigma \in \Sigma$. We have

THEOREM 3. *If* $L \in \mathcal{L}_{nDP}[\Sigma] \left(\mathcal{L}_{nP}[\Sigma] \right)$ *and τ is an arbitrary k-encoding of Σ into Σ', then* $\tau L \in \mathcal{L}_{nDP}[\Sigma'] \left(\mathcal{L}_{nP}[\Sigma'] \right)$

Proof. Let $L = L(A)$, where

$$A = \langle S, \Sigma, \Gamma, \delta, s_0, \overline{Z}_0, F \rangle$$

is in $\mathcal{A}_{nDP}[\Sigma]$.

We construct the automaton

$$A' = \langle S', \Sigma', \Gamma, \delta', s_0', \overline{Z}_0, F \rangle$$

where $S' = S \times \{ w \in (\Sigma')^* \mid l(w) \leqslant k-1 \} \cup \{ R \}$, R is a new symbol, $s' = (s_0, \Lambda)$, $F' = F \times \{ \Lambda \}$, the transition function $\delta' : S' \times \Sigma' \times \Gamma^n \to S' \times (\Gamma^*)^n$ is defined by

1) $\delta'((s, w), \sigma', \overline{Y}) = ((s, w\sigma'), \overline{Y})$ if $l(w) \leqslant k-2$

2) $\delta'((s, w), \sigma', \overline{Y}) = ((s', \Lambda), \overline{Y})$ if $l(w) = k-1$

and there exists $\sigma \in w$ such that $\tau(\sigma) = w\sigma'$ and $(s', \overline{Y}) = \delta(s, \sigma, \overline{Y})$.

3) $\delta'(s', \sigma', \overline{Y}) = (R, \overline{Y})$ in all other cases

It is evident that $A' \in \mathcal{A}_{nDP}[\Sigma']$ and $L(A') = \tau L$. So $\tau L \in \mathcal{L}_{nDP}[\Sigma]$. In the case of nondeterministic languages, the proof is similar.

REMARK. *Note that the proofs of Theorems 1, 2, 3 are constructive ones. This will be used in § 6 when the algorithmic problems are considered.*

§ 4. A NECESSARY CONDITION FOR LANGUAGES IN THE CLASS $\mathcal{L}_{DP}$

4.1. Let L be a language over the alphabet Σ. For every integer $k \geqslant 1$ we define an equivalence relation $E_k \pmod L$ in Σ^* by

$$u\, E_k\, v \pmod L \Leftrightarrow \forall\, w \Big(l(w) \leqslant k \rightarrow (u\,w \in L \leftrightarrow v\,w \in L) \Big) \tag{4.1}$$

Then, for any $k \geqslant 1$, $G_L(k)$ is defined to be the number of equivalence classes in Σ^* determined by the relation $E_k \pmod L$

$$G_L(k) = \text{ord}\ E_k \pmod L \tag{4.2}$$

THEOREM 4. *For every language L in the class $\mathcal{L}_{DP}$ there exists a positive constant c such that*

$$G_L(k) \leqslant c^k$$

Proof. Let $L = L(A)$, where

$$A = \langle S, \Sigma, \Gamma, \delta, s_0, \overline{Z}_0, F \rangle$$

is some nDP, $|S| = m$, $|\Gamma| = l$. Without loss of generality we can assume that $l \geqslant 1$, otherwise, A will become a finite automaton, L a regular language, and then, there exists a positive constant c such that $G_L(k) \leqslant c$ for any $k \geqslant 1$.

For every integer $k \geqslant 0$ we define an equivalence relation $\overset{k}{\sim}$ in $(\Gamma^*)^n$ by

$$\bar{\alpha} \overset{k}{\sim} \bar{\beta} \Leftrightarrow (\forall i = 1,\dots, n) \Big((\alpha_i = \beta_i) \vee (\alpha_i = \alpha'_i\, \Upsilon_i \;\&\; \beta_i = \beta'_i\, \Upsilon_i \;\&\; l(\Upsilon_i) = k) \Big) \tag{4.3}$$

It is obvious that $\text{ord}\,(\overset{k}{\sim}) = (1 + l + \dots + l^k)^n = \left(\dfrac{l^{k+1} - 1}{l-1}\right)^n$

Directly from the definition of relation $\overset{k}{\sim}$ we have, for any $k > 0$

$$\forall \bar{\alpha}, \bar{\beta} \in (\Gamma^*)^n\, \forall\, \overline{Y} \in \Gamma^n [(\bar{\alpha}\, \overline{Y} \overset{k}{\sim} \bar{\beta}\, \overline{Y}) \rightarrow \forall\, \overline{Y} \in (\Gamma^*)^n (\bar{\alpha}\, \overline{Y} \overset{k-1}{\sim} \bar{\beta}\, \overline{Y})] \tag{4.4}$$

Then, for every $k \geqslant 0$, we denote by $\equiv_k$ the equivalence relation in $S \times (\Gamma^*)^n$ which is given by

$$(s, \bar{\alpha}) \equiv_k (t, \bar{\beta}) \Leftrightarrow (s = t) \ \& \ (\bar{\alpha} \overset{k}{\sim} \bar{\beta}) \tag{4.5}$$

Obviously,

$$\text{ord}\,(\equiv_k) = m \left(\frac{l^{k+1} - 1}{l - 1}\right)^n < m.\, l^{n\,(k+1)}$$

Hence, by choosing $c = (m\, l)^{2n}$, we have

$$\text{ord}\ (\equiv_k) \leqslant c^k \tag{4.6}$$

for any $k \geqslant 1$.

Now with every word $w \in \Sigma^*$ we associate a map w from the set $S \times \{Z_{01}\}\Gamma^* \times \cdots \times \{Z_{0n}\}\Gamma^*$ into itself determined by

$$w\ (s, \bar{\alpha}) = (t, \bar{\beta}) \Leftrightarrow (s, w, \bar{\alpha}) \overset{*}{\underset{A}{\vdash\!\!-\!\!-}} (t, \Lambda, \bar{\beta}) \tag{4.7}$$

Clearly, the map w defined by (4.7) is one-valued because A is a deterministic automaton.

It is obvious that

$$\Lambda\ (s, \bar{\alpha}) = (s, \bar{\alpha})$$

For any $(s, \bar{\alpha})$ and $(t, \bar{\beta})$ in $S \times \{Z_{01}\}\ \Gamma^* \times \cdots \times\ \{Z_{0n}\}\ \Gamma^*$ we have

$$(s, \bar{\alpha}) \equiv_k (t, \bar{\beta}) \Rightarrow \forall\, w \in \Sigma^* \Big(\, l\,(w) \leqslant k \rightarrow w\ (s, \bar{\alpha}) \equiv_0 w\ (t, \bar{\beta}) \Big) \tag{4.8}$$

Indeed, for $k = 0$, (4.8) is obviously true. Suppose that $k > 0$ and (4.8) has been proved for $k - 1$. By the definition, $(s, \bar{\alpha}) \equiv_k (t, \bar{\beta})$ implies $s = t$ and $\bar{\alpha} \overset{k}{\sim} \bar{\beta}$. Then, since $k > 0$, $\bar{\alpha}$ and $\bar{\beta}$ can always be presented in the form

$$\bar{\alpha} = \bar{\alpha}'\ \bar{Y}\ \ ;\ \ \bar{\beta} = \bar{\beta}'\ \bar{Y}$$

with $\bar{Y} \in \Gamma^n$.

Let w be any word in Σ^* such that $l\ (w) \leqslant k$. If $w = k$, then

$$\Lambda\ (s, \bar{\alpha}) = (s, \bar{\alpha}) \equiv_k (t, \bar{\beta}) = \Lambda\ (t, \bar{\beta})$$

which implies

$$\Lambda\ (s, \bar{\alpha}) \equiv_0 \Lambda\ (t, \bar{\beta})$$

Suppose $w \neq \Lambda$, $w = \sigma\, w'$. Then

$$\begin{aligned} w\ (s, \bar{\alpha}) &= \sigma\ w'\ (s, \bar{\alpha}'\ \bar{Y}) = w'\ (s', \bar{\alpha}'\ \bar{Y}) \\ w\ (t, \bar{\beta}) &= \sigma\ w'\ (s, \bar{\beta}'\ \bar{Y}) = w'\ (s', \bar{\beta}'\ \bar{Y})\,, \end{aligned} \tag{4.9}$$

where $(s', \overline{Y}) = \delta(s, \sigma, \overline{Y})$.

By (4.4), $\overline{\alpha} \overset{k}{\sim} \overline{\beta}$ implies $\overline{\alpha'}\,\overline{Y} \overset{k-1}{\sim} \overline{\beta'}\,\overline{Y}$. Hence

$$(s', \overline{\alpha'}\,\overline{Y}) \equiv_{k-1} (s', \overline{\beta'}\,\overline{Y}) \, .$$

But then, by the inductive assumption, we have

$$w' (s', \overline{\alpha'}\,\overline{Y}) \equiv_0 (s', \overline{\beta'}\,\overline{Y}).$$

Together with (4.9), this gives us

$$w (s, \overline{\alpha}) \equiv_0 w (t, \overline{\beta}) \, .$$

Thereby (4.8) has been proved.

As a special case of (4.8) we have

$$u (s_0, \overline{Z}_0) \equiv_k v (s_0, \overline{Z}_0) \Rightarrow \forall w \in \Sigma^* \Big(l (w) \leqslant k \rightarrow u\, w (s_0, \overline{Z}_0) \equiv_0 vw (s_0, \overline{Z}_0) \Big) \quad (4.10)$$

Now, if in Σ^* we define the equivalence relation F_k by

$$u \; F_k \; v \Leftrightarrow u (s_0, \overline{Z}_0) \equiv_k v (s_0, \overline{Z}_0) \qquad (4.11)$$

and note that, for any $u, v \in \Sigma^*$, $u (s_0, \overline{Z}_0) \equiv_0 v (s_0, \overline{Z}_0) \Rightarrow (u \in L \leftrightarrow v \in L)$,
Then by virtue of (4.10) and (4.1), we have:

$$uF_k v \Rightarrow uE_k v \pmod L \qquad (4.12)$$

Finally, by (4.2), (4.12), (4.11), and (4.6), we have

$$G_L(k) = \text{ord } E_k \,(\text{mod } L) \leqslant \text{ord } F_k \leqslant \text{ord} \,(\equiv_k) \leqslant c^k ,$$

and the theorem is proved.

4.2. As has been shown in 4.1, the complexity characteristic $G_L(k)$ of every language L in $\mathcal{L}_{DP}$ is limited by some exponential function c^k. In the following we shall show that, for any given exponential function c^k, there can be found a language L in $\mathcal{L}_{DP}$, even in $\mathcal{L}_{1DP}$, such that its complexity characteristic $G_L(k)$ exceeds the function c^k.

For every integer $n \geqslant 1$, we consider the following language over the alphabet $\{0, 1, c\}$

$$L_n = \Big\{ \alpha_{11}\alpha_{12} \ldots \alpha_{1n}\alpha_{21}\alpha_{22} \ldots \alpha_{2n} \ldots \alpha_{m1}\alpha_{m2} \ldots \alpha_{mn} \subset w \,\Big|\, m \geqslant 1 ; \alpha_{ij} \in \{0,1\}$$
$$(1 \leqslant i \leqslant m), 1 \leqslant j \leqslant n ; \exists j (1 \leqslant j \leqslant n) \, \alpha_{1j}\alpha_{2j} \ldots \alpha_{mj} = \hat{w} \Big\} \qquad (4.13)$$

LEMMA 1. *For any $n \geqslant 1$, the language L_n defined by (4.13) belongs to class $\mathcal{L}_{1DP}[\{0, 1, c\}]$*

Proof. The $1DP$ A recognizing L_n is constructed as follows:

$$A = \langle S, \Sigma, \Gamma, \delta, s_0, Z_0, F \rangle ,$$

where

$$S = \{s_0, s_1, s_2\} \cup \{0,1\}^n \cup \{R\}$$

$$\Sigma = \{0, 1, c\}$$

$$\Gamma = \{Z_0\} \cup \bigcup_{i=1}^{n} \{0,1\}^i \cup \{\tilde{w} \mid w \in \{0,1\}^n\}$$

$$F = \{s_2\}$$

$\delta : S \times \Sigma \times \Gamma \to S \times \Gamma^*$ is defined by

1) $\delta(s_0, \sigma, Z_0) = (s_0, Z_0 Y)$ if $\sigma \in \{0,1\}$, $Y = \sigma$

2) $\delta(s_0, \sigma, w) = (s_0, Y)$ if $\sigma \in \{0,1\}$, $w \in \bigcup_{i=1}^{n-1} \{0,1\}^i$, $Y = w\sigma$

3) $\delta(s_0, \sigma, w) = (s_1, Y_1 Y_2)$ if $\sigma \in \{0,1\}$, $w \in \{0,1\}^n$, $Y_1 = \tilde{w}$, $Y_2 = \sigma$

4) $\delta(s_0, c, w) = (1^n, Y)$ if $w \in \{0,1\}^n$, $Y = \tilde{w}$

5) $\delta(s_1, \sigma, w) = (s_1, Y)$ if $\sigma \in \{0,1\}$, $w \in \bigcup_{i=1}^{n-1} \{0,1\}^i$, $Y = w\sigma$

6) $\delta(s_1, \sigma, w) = (s_1, Y_1 Y_2)$ if $\sigma \in \{0,1\}$, $w \in \{0,1\}^n$, $Y_1 = w$, $Y_2 = \sigma$

7) $\delta(s_1, c, w) = (1^n, Y)$ if $w \in \{0,1\}^n$, $Y = w$

8) $\delta(u, \sigma, w) = (v, \Lambda)$ if $\sigma \in \{0,1\}$, $u, v, w \in \{0,1\}^n$ and

$$\forall j\ (1 \leqslant j \leqslant n) \qquad \text{comp}_j\, v = \begin{cases} \text{comp}_j\, u & \text{if } \sigma = \text{comp}_j\, w^{(1)} \\ 0 & \text{if } \sigma \neq \text{comp}_j\, w \end{cases}$$

9) $\delta(u, \sigma, w) = (s_2, \Lambda)$ if $\sigma \in \{0,1\}$, $(u, w) \in \{0,1\}^n$ and
$\exists j\ (1 \leqslant j \leqslant n)\ (\sigma = \text{comp}_j\, w)\ \&\ (\text{comp}_j\, u = 1)$

10) $\delta(s, \sigma, Y) = (R, Y)$ in all other cases.

It is not difficult to verify that $L(A) = L_n$, hence $L_n \in \mathcal{L}_{1DP}[\{0, 1, c\}]$.

(1) Comp$_j$ u denotes the j-th letter of word u (from left to right), e. g.; if $u = 01011$ then comp$_2$ $u = 1$, comp$_3$ $u = 0$.

The lemma is proved.

LEMMA 2. *For any given positive constant c, there exists a natural number n such that*

$$G_{L_n}(k) \geqslant c^k$$

for any k large enough.

Proof. Let

$$\mathcal{W}_n = \left\{ U \subseteq \{0,1\}^k \,\middle|\, |U| = n \right\}$$

Obviously,

$$|\mathcal{W}_n| = C^n_{2^k} \tag{4.14}$$

With every set $U = \left\{ \alpha_{11}\alpha_{21}\ldots\alpha_{k1}, \alpha_{12}\alpha_{22}\ldots\alpha_{k2}, \ldots, \alpha_{1n}\alpha_{2n}\ldots\alpha_{kn} \right\}$ in $\mathcal{W}_n$ we associate a word

$$\xi_U = \alpha_{11}\alpha_{12}\ldots\alpha_{1n}\alpha_{21}\alpha_{22}\ldots\alpha_{2n}\ldots\alpha_{k1}\alpha_{k2}\ldots\alpha_{kn}\, c\,.$$

Clearly

$$\xi_U \hat{w} \in L \Leftrightarrow w \in U$$

Hence,

$$(\forall\, U, V \in \mathcal{W}_n)\,(U \neq V \rightarrow \xi_U \not E_k\, \xi_V \ (\mathrm{mod}\ L_n))$$

therefore,

$$G_{L_n}(k) = \text{ ord } E_k \ (\mathrm{mod}\ L_n) \geqslant |\mathcal{W}_n| = C^n_{2^k} \tag{4.15}$$

By choosing $n \geqslant \log c + 1$ and $k > \log n!$, we have

$$nk - \log n! \geqslant (\log c + 1)\,k - \log n! > k \log c\,.$$

For any k large enough, this implies

$$n \log (2^k - n) - \log n! \geqslant k \log c\,.$$

or

$$\frac{(2^k - n)^n}{n!} \geqslant c^k\,. \tag{4.16}$$

Since

$$C^n_{2^k} = \frac{2^k(2^k-1)\ldots(2^k-n+1)}{n!} \geqslant \frac{(2^k-n)^n}{n!}\,,$$

by (4.15) and (4.16) we obtain

$$G_{L_n}(k) \geqslant c^k$$

for any k large enough. This proves the lemma.

LEMMA 3. *If L is any language over the alphabet Σ and τ is any l-encoding of Σ into Σ', then*

$$G_{\tau L}(l\,k) \geqslant G_L(k) \tag{4.17}$$

for any $k \geqslant 1$.

Proof. Assume $u, v \in \Sigma^*$ and $u \not\equiv_k v \pmod{L}$. Then there must exist some word $w \in \Sigma^*$ with $l(w) \leqslant k$ such that one and only one of the words uw and vw belongs to the language L , e. g. , $uw \in L$ and $vw \notin L$ (for the case $uw \notin L$ and $vw \in L$, the proof is similar). Then we have :

$$\begin{aligned} \tau(u\,w) &= \tau(u)\,\tau(w) \in \tau\,L \\ \tau(v\,w) &= \tau(v)\,\tau(w) \notin \tau\,L \end{aligned} \tag{4.18}$$

Since $l(\tau(w)) = l.\,l(w) \leqslant l.\,k$, (4.18) implies

$$\tau(u) \not\equiv_{lk} \tau(v) \pmod{\tau\,L}$$

Thus, the number of equivalence classes in $(\Sigma')^*$ determined by the relation $E_{lk} \pmod{\tau\,L}$ is not less than the number of equivalence classes in Σ^* determined by $E_k \pmod{L}$. In other words we have

$$G_{\tau L}(lk) \geqslant G_L(k)$$

for any $k \geqslant 1$, and thus the lemma is proved.

THEOREM 5. *Let Σ be any alphabet consisting of at least two letters. Then, for any given positive constant c, however large it may be, there exists a language L in the class $\mathcal{L}_{1DP}[\Sigma]$ such that*

$$G_L(k) \geqslant c^k$$

for any k large enough.

Proof. Choose τ to be an arbitrary l-encoding of $\{0, 1, c\}$ into Σ. By Lemma 2, there exists n such that

$$G_{L_n}(k) \geqslant c^{(l+1)k} \tag{4.19}$$

for any natural number k large enough.

By Lemma 1, $L_n \in \mathcal{L}_{1DP}[\{0, 1, c\}]$ we take $L = \tau L_n$. Then, by Theorem 3 $L \in \mathcal{L}_{1DP}[\Sigma]$. By using the Lemma 3, from (4.19) we obtain

$$G_L(lk) \geqslant c^{(l+1)k} \tag{4.20}$$

for any natural number k large enough.

It is not difficult to note that, for every natural number $h \geqslant (l+1)l$, there exists a natural number k such that

$$lk \leqslant h \leqslant (l+1)k \tag{4.21}$$

Since the functions $G_L(k)$ and c^k are increasing, by virtue of (4.20) and (4.21) we have

$$G_L(h) \geqslant G_L(lk) \geqslant c^{(l+1)k} \geqslant c^h$$

for any natural number h large enough. Theorem is proved.

§5. RELATIONS TO WELL-KNOWN CLASSES OF LANGUAGES

5.1. It is clear that any finite automaton can be stimulated by some $1DP$, Thus the class $\mathcal{L}_R$ of all regular languages is a subclass of $\mathcal{L}_{1DP}$.

As is well-known, a language L is regular if and only if there exists a constant c such that $G_L(k) \leqslant c$ for any k. So, languages satisfying the conditions of Theorem 5 with $c > 1$ cannot be regular. In other words, *over any alphabet Σ consisting of at least two letters, there are the language recognizable by deterministic pushdown automata with one pushdown store tape in real time but not being regular.*

5.2. Let $\Sigma = \{0, 1\}$. Consider language Σ^*A, where $A = \{w \in \Sigma^* \mid w = \widehat{w}\ \&\ l(w) \geqslant 3\}$. It is not difficult to verify that Σ^*A is a context-free language. On the other hand, in [14] Cole has shown

$$G_{\Sigma^*A}(k) > 2^{2^{(k-3)/2}}$$

Hence, by Theorem 4, Σ^*A cannot belong to $\mathcal{L}_{DP}$. Thus, *over any alphabet Σ con-*

sisting of at least two letters, there are context-free languages not recognizable by any deterministic pushdown automaton in real time however large his number of pushdown store tapes may be.

5. 3. It has been shown in 3. 2 that language $L = \{ a^n b^n a^n \mid n \geqslant 1 \}$ is intersection of two languages in $\mathcal{L}_{1DP}$. By (ii) of Theorem 1, it must belong to $\mathcal{L}_{2DP}$. Thus, *over any alphabet* Σ *consisting of at least two letters, there are languages recognizable by deterministic pushdown automata with two pushdown store tapes in real time but not being context-free.*

It is not difficult to show that for any given nDP one can construct a linear bounded automaton stimulating the word of this nDP. Therefore, *any language in* $\mathcal{L}_{DP}$ *is a context-sensitive one.*

5. 4. As a immédiate corollary of 5. 2. and inclusions (2. 1), (2. 2), (2. 3), (3. 1) we have : for every $n \geqslant 1$, *over any alphabet* Σ *consisting of at least two letters, there are languages recognizable by nPs but not recognizable by nDPs. In other words,* $\mathcal{L}_{nDP}$ *is a proper subclass of* $\mathcal{L}_{nP}$.

Then, by 5.3. and (2.1), (2.2), (2.3), (3.1), we have : *over any alphabet* Σ *consisting of at least two letters, there are languages recognizable by (2DPs) but not recognizable by 1Ps (1DPs). In other words,* $\mathcal{L}_{1P}\left(\mathcal{L}_{1DP}\right)$ *is a proper subclass of* $\mathcal{L}_{2P}\left(\mathcal{L}_{2DP}\right)$.

REMARK. In [15] Aanderaa has proved a fundamental result about the hierarchy of the classes $\mathcal{L}_{nDP}$. Namely, he has proved that, for any $n \geqslant 1$, there are languages recognizable by pushdown automata with pushdown memory tapes in real time but not recognizable by Turing machines with n memory tapes in real time. Thereby one obtains the hierarchy :

$$\mathcal{L}_{1DP} \subsetneqq \mathcal{L}_{2DP} \subsetneqq \cdots \subsetneqq \mathcal{L}_{nDP} \subsetneqq \cdots \tag{5.1}$$

In order to obtain this result Aanderaa has used a rather complicated technique of overlapping. In section 4.1. we have given a simple necessary condition for languages in the whole class $\mathcal{L}_{DP}$, but it is very difficult to find the simple necessary conditions for languages in every class $\mathcal{L}_{nDP}$, by which one can distinguish the classes $\mathcal{L}_{nDP}$ and $\mathcal{L}_{(n+1)DP}$. It seems that in order to prove the hierarchy (5.1) one can find the examples of languages, which are simpler than those of Aanderaa, e. g., the language

$$L_n = \left\{ 1^{i_1} 0 1^{i_2} 0 \dots 0 1^{i_{n+1}} 0 1^{i} \,\middle|\, i_j \geqslant 1,\ \exists i_j = i \right\}$$

is obviously in $\mathcal{L}_{(n+1)DP}$, and it seems that L_n does not belong to $\mathcal{L}_{nDP}$, but no proof of this fact is known!

§6. SOME ALGORITHMICAL PROBLEMS

For any pair (x, y), where $x = (x_1, \dots, x_n)$, $y = (y_1, \dots, y_n)$ are two n-tuples $(n \geqslant 1)$ of non-empty words in the alphabet $\{0,1\}$. We define the following languages over the alphabet $\{0,1,c\}$

$$L(x,y) = \{01^{i_k} \dots 01^{i_2}\, 01^{i_1}\, cw \mid k \geqslant 1 \;;\; 1 \leqslant i_j \leqslant n \;\; (1 \leqslant j \leqslant k) \;;$$

$$w = x_{i_1} x_{i_2} \dots x_{i_k} = y_{i_1} y_{i_2} \dots y_{i_k} \}$$

$$M(x,y) = \{01^{i_k} \dots 01^{i_2}\, 01^{i_1}\, cwc\, 1^{i_1}\, 01^{i_2}\, 0 \dots 1^{i_k}\, 0 \mid k \geqslant 1 \,;\, 1 \leqslant i_j \leqslant n$$

$$(1 \leqslant j \leqslant k) \,;\; w = x_{i_1} x_{i_2} \dots x_{i_k} = y_{i_1} y_{i_2} \dots y_{i_k} \}$$

LEMMA 4. *There is no algorithm allowing us, for any pair (x, y), to decide*

(i) *whether $L(x,y) = \varnothing$ or not,*

(ii) *whether $L(x,y)$ is finite or not.*

Proof. It is clear that $L(x,y) = \varnothing$ when and only when the Post's correspondence problem for (x,y) has no solutions. Since the Post's correspondence problem is undecidable, the empty problem for languages $L(x,y)$ is also undecidable. Note that $L(x,y)$ is either empty or infinite that is, $L(x,y)$ is finite when and only when $L(x, y) = \varnothing$. Therefore, the undecidability of finiteness problem for languages $L(x, y)$ follows from the undecidability of empty problem for them. Q.E.D.

LEMMA 4'. *Let Σ be any alphabet consisting of at least two letters. Let τ be any k-encoding of $\{0, 1, c\}$ into Σ. Then, there is no algorithm allowing us, for any pair (x, y) to decide*

(i) *whether $\tau L(x, y) = \varnothing$ or not,*

(ii) *whether $\tau L(x, y)$ is finite or not,*

(iii) *whether $\Sigma^* \setminus \tau L(x, y) = \Sigma^*$ or not.*

Proof. Note that τ is a one-to-one map from $\{0, 1, c\}^*$ into Σ^*. Then (i) and (ii) follow from (i) and (ii) of Lemma 4 respectively. The unsolvability of problem (iii) follows from the unsolvability of problem (i). Q.E.D.

LEMMA 5. *For any given pair (x, y), we can construct*

(i) *an automaton $A \in \mathcal{A}_{2DP}\,[\,\{0, 1, c\}\,]$ such that*

$$L(A) = L(x, y)$$

(ii) *an automaton* $A \in \mathcal{A}_{3DP}$ $[\{0, 1, c\}]$ *such that*

$$L(A) = M(x, y)$$

Proof. (i) Instead of constructing in detail the automaton $A \in \mathcal{A}_{2DP}$ $[\{0, 1, c\}]$ recognizing $L(x, y)$, we describe the working process of A when it reads an input word.

If the input word has not the form $01^{i_k} \dots 01^{i_2} 01^{i_1} c\, w$, then A goes into the refusing state at some moment and then it is always in this state, that is, in this case the input word is refused by A.

If the input word has the form as shown above, the automaton A works as follow :

(a) While reading the part $01^{i_k} \dots 01^{i_2} 01^{i_1}$ of the input word which is a code of sequence of numbers $i_k, \dots, i_2, i_1$, the automaton A records the words $\hat{x}_{i_k} \dots \hat{x}_{i_2} \hat{x}_{i_1}$ and $\hat{y}_{i_k} \dots \hat{y}_{i_2} \hat{y}_{i_1}$ in his first and second pushdown stores respectively.

(b) After the symbol c of the input word is read, A compares the part w of the input word simultaneously with two words $x_{i_1} x_{i_2} \dots x_{i_k} = \widehat{\hat{x}_{i_k} \dots \hat{x}_{i_2} \hat{x}_{i_1}}$ and $y_{i_1} y_{i_2} \dots y_{i_k} = \widehat{\hat{y}_{i_k} \dots \hat{y}_{i_2} \hat{y}_{i_1}}$ which have been stored in two pushdown stores of A by the step (a).

If $w \neq x_{i_1} x_{i_2} \dots x_{i_k}$ or $w \neq y_{i_1} y_{i_2} \dots y_{i_k}$, then A goes into the refusing state at some moment of the comparison process and then it is always in this state, i. e., the input word is refused.

If $w = x_{i_1} x_{i_2} \dots x_{i_k} = y_{i_1} y_{i_2} \dots y_{i_k}$, then after reading w, A goes into some final state, i. e., the input word is accepted by A.

Thus we have proved that $L(A) = L(x, y)$.

(ii) Similarly.

LEMMA 5'. *Let Σ be any alphabet consisting of at least two letters. Let τ be any k-encoding of $\{0, 1, c\}$ into Σ. Then, for any given pair (x, y), we can construct*

(i) *an automaton* $A \in \mathcal{A}_{2DP}$ $[\Sigma]$ *such that*

$$L(A) = \tau\ L(x, y)$$

(ii) *an automaton* $A \in \mathcal{A}_{2DP}\ [\Sigma]$ *such that*

$$L(A) = \Sigma^* - \tau L(x, y)$$

(iii) *an automaton* $A \in \mathcal{A}_{3DP}\ [\Sigma]$ *such that*

$$L(A) = \tau M(x, y)$$

Proof. (i) follows from (i) of Lemma 5 and Theorem 3 (see remark in the end of § 3). (ii) follows from (i) and (i) of Theorem 1. (iii) follows from (ii) of Lemma 5 and Theorem 3.

THEOREM 6. *Let* Σ *be any alphabet consisting of at least two letters. Then, there is no algorithm allowing us, for any* $A \in \mathcal{A}_{2DP}\ [\Sigma]$, *to decide*

(i) *whether* $L(A) = \varnothing$ *or not,*

(ii) *whether* $L(A)$ *is finite or not,*

(iii) *whether* $L(A) = \Sigma^*$ *or not.*

Proof. If there were an algorithm deciding (i), then, by (i) of Lemma 5' there would be an algorithm allowing us, for any pair (x, y), to decide, whether $\tau L(x, y) = \varnothing$ or not. But such an algorithm, by (i) of Lemma 4', is impossible.

(ii) and (iii) can be proved similarly by the help of (i) of Lemma 5', (ii) of Lemma 4' and (ii) of Lemma 5', (iii) of Lemma 4' respectively.

As an immediate corollary of Theorem 6 we have

THEOREM 7. *Let* Σ *be any alphabet consisting of at least two letters. Then, there is no algorithm allowing us, for any* $A_1, A_2 \in \mathcal{A}_{2DP}\ [\Sigma]$, *to decide*

(i) *whether* $L(A_1) = L(A_2)$ *or not,*

(ii) *whether* $L(A_1) \subseteq L(A_2)$ *or not,*

(iii) *whether* $L(A_1) \subsetneq L(A_2)$ *or not.*

LEMMA 6. *For any given pair* (x, y), *the language* $M(x, y)$ *does not contain any infinite context-free language.*

Proof. We first note that every word $w \in M(x, y)$ has the form

$$w = e_1 c e_2 c \hat{e}_1$$

where $e_1 = 01^{i_k} \ldots 01^{i_2} 01^{i_1}$, $k \geqslant 1$, $1 \leqslant i_j \leqslant n$ $(1 \leqslant j \leqslant k)$, and $e_2 = x_{i_1} x_{i_2} \ldots x_{i_k} = y_{i_1} y_{i_2} \ldots y_{i_k}$. Thus the word w is completely defined by his part e_1.

Assume that $M(x, y)$ contains some infinite context-free language L.

Since L is a context-free language, by the Bar-Hillel's criterion (see, e. g., [10]) there are two positive integers p and q such that if $w \in L$ and $l(w) \geqslant p$ then w can be represented in the form $w = u_1\, u\, u_2\, v\, u_3$, where $l(u\,v) \geqslant 1, l(u\,u_2\,v) \leqslant q$ and $u_1\, u^m\, u_2\, v^m\, u_3 \in L$ for any $m \geqslant 1$.

Then since L is an infinite language over a finite alphabet, the set $\{\, l(w) \mid w \in L \,\}$ must be infinite. Hence, as $l(x_i)$, $l(y_i) \geqslant 1$ for any i $(1 \leqslant i \leqslant n)$, there must be found a word $w = e_1\, c\, e_2\, c\, \hat{e}_1 \in L$ such that $l(w) \geqslant p$ and $l(e_2) \geqslant q$. But then

$$w = e_1\, c\, e_2\, c\, \hat{e}_1 = u_1\, u\, u_2\, v\, u_3 \tag{6.1}$$

where $l(u\,v) \geqslant 1$, $l(u\,u_2\,v) \leqslant q$ and $u_1\, u^m\, u_2\, v^m\, u_3 \in L$ for any $m \geqslant 1$.

Clearly,

$$l(w) = 2\,l(e_1) + l(e_2) + 2 = l(u_1) + l(u\,u_2\,v) + l(u_3)$$

hence

$$l(u_1) + l(u_3) = 2\,l(e_1) + 2 + l(e_2) - l(u\,u_2\,v) \geqslant 2\,l(e_1) + 2$$

Thus either $l(u_1) \geqslant l(e_1) + 1$ or $l(u_3) \geqslant l(e_1) + 1$. Suppose $l(u_1) \geqslant l(e_1) + 1$ (for the case $l(u_3) \geqslant l(e_1) + 1$ the argument is similar).

Since $w' = u_1\, u^2\, u_2\, v^2\, u_3 \in L$, there must be e_3 and e_4 such that

$$w' = u_1\, u^2\, u_2\, v^2\, u_3 = e_3\, c\, e_4\, c\, \hat{e}_3 \tag{6.2}$$

By $l(u_1) \geqslant l(e_1) + 1$ and (6.1), it is clear that $e_1 c$ is an initial segment of u_1. Then, note that neither e_1 nor e_3 contain c, by (6.2), we have $e_1 = e_3$. But then $w = w'$, which is impossible because $l(w') = l(w) + l(u\,v) > l(w)$. This contradiction proves the lemma.

LEMMA 6'. *Let Σ be any alphabet consisting of at least two letters. Let τ be any k-encoding of $\{0,1,c\}$ into Σ. Then, for any given pair (x, y), the language*

Proof. As has been known, the class of all context-free languages is closed with respect to inverse homomorphism. Hence, if $\tau\, M(x,y)$ contained some infinite context-free language L, then $M(x,y) = \tau^{-1}(\tau\, M(x,y))$ would contain the infinite context-free language $\tau^{-1} L$, that contradicts Lemma 6.

THEOREM 8. *Let Σ be any alphabet consisting of at least two letters. Then there is no algorithm allowing us, for any $A \in \mathcal{A}_{3DP}[\Sigma]$, to decide*

(i) *whether $L(A) \in \mathcal{L}_{cF}$ or not,*

(ii) *whether $L(A) \in \mathcal{L}_{1DP}$ or not,*

(iii) *whether $L(A) \in \mathcal{L}_{R}$ or not.*

Proof. (i) If there were an algorithm deciding the problem (i), then, by (iii) of Lemma 5', there would be an algorithm deciding if $\tau M(x,y)$ is context-free. Since $\tau M(x,y)$ is either empty or infinite, by Lemma 6', $\tau M(x,y)$ is context-free when and only when $\tau\, M(x,y) = \varnothing$. Thus there would be an algorithm deciding the empty problem for languages $M(x,y)$, that contradicts (iii) of Lemma 4'.

Received March 12, 1974

REFERENCES

[1] CHOMSKY, N., *Context-free grammars and pushdown storage,* M.I.T. Res. Lab. Electron. Quart. Prog. Rept., *1962,* 65.

[2] EVEY, R.J., *The Theory and application of pushdown-store machines,* Mathematical languistics and automatic translation, Harward Univ., Computation Lab. Rept., *1963,* NSP-10.

[3] SCHUTZENBERGER, N.P., *Context-free languages and pushdown automata,* Information and control, *1963,* 6, 246-264.

[4] GRAY, J.N., HARRISON, M.A., IBARRA, O.H., *Two-way pushdown automata,* Information and control, *1967,* 11, 30-70.

[5] GREIBACH, S. A., *A note on pushdown store automata and regular systems,* Proc. Amer. Math. Soc., *1967,* 18, 263-268.

[6] GINSBURG, S., SPANIER, E. H., *Finite-turn pushdown automata,* SIAM J. Control, *1966,* 4, 429-453.

[7] ROVAN, B., *Bounded pushdown automata,* Kybernetica, *1969,* 4, 261-265.

[8] ULLMAN, J.D., *Pushdown automata with bounded backtrack,* System development Corp. Rept., *1965,* TM 738/022/06.

[9] LETICHETSKII, A.A., *Representation of context-free languages in automata with pushdown memory,* Kibernetika, *1965,* 2, 80-84 (in russian).

[10] GINSBURG, S., *The mathematical theory of context-free languages*, Mc Graw-Hill book company, *1966.*

[11] BURKS, A.W., WARREN, D. W., WRIGHT, J. B., *An analysis of logical machine using parenthesis-free notation*, Mathematical tables and other aids to computation, *1954*, 8, 53-57.

[12] HAINES, L.H., *Generation and recognition of formal languages*, Ph. D. dissertation, MIT, Cambridge, Mass., June *1965.*

[13] GREIBACH, S.A., *A new normal-form theorem for context-free phrase structure grammars*, J. ACM, *1965*, 12, 1, 42-52.

[14] COLE, S.N., *Real-time computation by n-dimensional iterative arrays of finite state machines*, IEEE trans., *1969*, C-18, 349-365.

[15] AANDERAA, S. O., *On k-tape versus (k-1)-tape real-time computation*, SIAM-AMS Proceedings, *1974*, 7,75-96.