Simple deterministic machines

By N. T. KHANH

§ 1. Introduction

Classes of formal languages are frequently characterized by different types of accepting automata. It is interesting to note that deterministic languages, i.e., languages accepted by deterministic automata are very difficult to characterize by other properties. However, as it will be shown in this paper, if we restrict ourselves to the so called simple deterministic machines then the corresponding language classes can be characterized by the prefix-free property. A language L is called prefix-free if and only if for every pair of words (x, y): $x \in L$ and $xy \in L$ jointly imply $y = \lambda$, where λ is the empty word. The hierarchy of simple deterministic machines will thus correspond to the intersections of the classes of deterministic languages in the Chomsky hierarchy with the family of all prefix-free languages. Simple machines introduced by E. P. FRIEDMAN in [3] will be compared with our simple deterministic pushdown machines and we show that the class of languages accepted by the former ones constitute a proper subset of those accepted by the latter machines. This means that although the languages accepted by Friedman's simple machines are prefix-free, they do not include every prefix-free deterministic language. The classes of languages characterized by our simple deterministic machines have different closure properties under the usual operations. We also define some specific operations with respect to the prefix-free property. The usefulness of our simple deterministic machines can be seen also from the properties of the corresponding language classes.

§ 2. Prefix-free languages

Definition 2.1. A language L is said to be prefix-free if for every pair of words (x, y): $x \in L$ and $xy \in L$ imply $y = \lambda$. The family of all prefix-free languages is denoted by $\mathscr{L}_p$. We can prove that $\mathscr{L}_p$ is closed under intersection and concatenation, but it is not closed under complementation and union.

Definition 2.2. Let L_1, L_2 be two languages over the alphabet Σ.
a) For x, y in Σ^*, write $x \prec y$ if $y = xz$ for some z in $\Sigma^* - \{\lambda\}$.
b) The p-quotient of L_1 by L_2 is defined by

$$L_1 p L_2 = \{y \in L_1 / \text{ if } x \prec y \text{ then } x \notin L_2\}.$$

c) The p-union of two languages L_1 and L_2 is defined by

$$L_1 \cup_p L_2 = (L_1 p L_2) \cup (L_2 p L_1).$$

It is easy to see that

$$L_1 \cup_p L_2 = L_1 \cup L_2 - \{y_1 \in L_1 / \text{ there is an } x < y_1 \text{ with } x \in L_2\} - \{y_2 \in L_2 / \text{ there is an } x < y_2 \text{ with } x \in L_1\}.$$

Theorem 2.1. The family $\mathscr{L}_p$ is closed under p-quotient and p-union.

Proof. Let L_1, L_2 be prefix-free languages. It should be clear that $L_1 p L_2$ is prefix-free. We now prove that $L_1 \cup_p L_2$ is prefix-free. Assume on the contrary that there is an $x \in L_1 \cup_p L_2$ with $xy \in L_1 \cup_p L_2$ for some $y \neq \lambda$. Let $x \in L_1 p L_2$. (The case where $x \in L_2 p L_1$ is similar.) Since L_1 is prefix-free and $y \neq \lambda$, it suffices to consider the case where $xy \in L_2 p L_1$. But, by Definition 2.2, we can easily see that if $xy \in L_2 p L_1$ for $y \neq \lambda$ then $x \notin L_1$, i.e., $x \notin L_1 p L_2$ and the contradiction arises. □

Definition 2.3. Let Σ and Δ be two disjoint alphabets, and w any fixed string in Δ^*. We define the homomorphism $h_w: \Sigma^* \to (\Sigma \cup \Delta)^*$, such that

a) $h_w(\lambda) = \lambda$,

b) $h_w(a) = aw$ for all $a \in \Sigma$,

c) $h_w(PQ) = h_w(P) h_w(Q)$ for all $P, Q \in \Sigma^*$.

For a language L over Σ, we define

$$h_w(L) = \{h_w(P) / P \in L\}.$$

Theorem 2.2. A language $L \subseteq \Sigma^*$ is prefix-free if and only if $h_w(L) \subseteq (\Sigma \cup \Delta)^*$ is prefix-free.

Proof. The case where $w = \lambda$ is trivial, so we assume that $w = a_1 \ldots a_n$ for some $a_1, \ldots, a_n \in \Sigma$ with $n \geqq 1$. Similarly, we can assume that $L \neq \{\lambda\}$.

Part 1. $L \in \mathscr{L}_p \to h_w(L) \in \mathscr{L}_p$.

Let $x \in h_w(L)$ and $xy \in h_w(L)$, then there are $P \in L$ and $PQ \in L$ such that $x = h_w(P)$, $xy = h_w(PQ) = h_w(P) h_w(Q)$. Since L is prefix-free, $Q = \lambda$. Consequently, $y = h_w(Q) = h_w(\lambda) = \lambda$, Thus $h_w(L) \in \mathscr{L}_p$.

Part 2. $h_w(L) \in \mathscr{L}_p \to L \in \mathscr{L}_p$.

Assume on the contrary that there are $x \in L$ and $xy \in L$ for some $y \neq \lambda$. It is clear that $P = h_w(x) \in h_w(L)$, $PQ = h_w(xy) \in h_w(L)$ and $Q = h_w(y) \neq \lambda$. Thus, $h_w(L)$ is not prefix-free and the contradiction arises. □

§ 3. Simple finite deterministic machines

In this section first we investigate a special kind of finite deterministic automata called simple finite deterministic machines (abbreviated SFD-machines), and prove that the family of all languages accepted by SFD-machines is the intersection of the two families $\mathscr{L}_3$ and $\mathscr{L}_p$. Further, we note that this family is not closed under complementation and union, but it is closed under concatenation, intersection, p-quotient,

p-union and homomorphism h_w. The proofs of these facts will not be presented in this paper.

Let us consider the standard definition of a finite deterministic automaton (abbreviated FD-automaton, see [1]). That is: Let $M=(K, \Sigma, \delta, q_0, H)$ be an FD-automaton, where K is the set of states, Σ is the set of inputs, q_0 is an element of K (the initial state), H is a subset of K (the set of final states), and δ is a mapping from $K\times\Sigma$ to K.

Notation. Given an FD-automaton M let $\underset{M}{\vdash}$ be the relation on $K\times\Sigma^*$ defined as follows. For $a\in\Sigma$, $w\in\Sigma^*$, $q, p\in K$

$$qaw \underset{M}{\vdash} pw \text{ iff } \delta(q, a)=p.$$

We let $\overset{*}{\underset{M}{\vdash}}$ denote the transitive closure of $\underset{M}{\vdash}$. Finally, we define the language accepted by M to be

$$L(M) = \{w\in\Sigma^*/q_0 w \overset{*}{\underset{M}{\vdash}} p \quad \text{for some} \quad p\in H\}.$$

Definition 3.1. a) A simple finite deterministic machine (abbreviated SFD-machine is a 5-tuple $M=(K, \Sigma, \delta, q_0, H)$, where K, Σ, q_0, H are the same as in the definition of an FD-automaton and δ is a mapping from $(K-H)\times\Sigma$ to K. Similarly, we define the language accepted by an SFD-machine M to be

$$L(M) = \{w\in\Sigma^*/q_0 w \overset{*}{\underset{M}{\vdash}} p \quad \text{for some} \quad p\in H\}.$$

b) A language L is said to be a simple finite deterministic language (abbreviated sfd-language), if $L=L(M)$ for some SFD-machine. The family of all sfd-languages is denoted by $\mathscr{L}_{\text{sd3}}$.

Remark. For simplicity of Definition 3.1 we have restricted the definition of the mapping δ to $K-H$, so δ is not complete. By the following theorem we shall see that the SFD-machine is a special kind of the FD-automaton.

Theorem 3.1. Let L be any language over the alphabet Σ. L is an sfd-language if and only if L is prefix-free and $L\in\mathscr{L}_3$.

Proof. Part 1. $L\in\mathscr{L}_{\text{sd3}}\rightarrow L\in\mathscr{L}_p\cap\mathscr{L}_3$.

Let $L=L(M)$ for an SFD-machine $M=(K, \Sigma, \delta, q_0, H)$. Since the domain of δ is $K-H$, we can easily see that $L\in\mathscr{L}_p$. We now prove that $L\in\mathscr{L}_3$.

Construct an FD-automaton M' from M as follows: Let $M'=(K\cup\{\bar{q}\}, \Sigma, \delta', q_0, H)$, where $\bar{q}\notin K$ and δ' is defined so that

1) for every $q\in K-H$, $a\in\Sigma$: $\delta'(q; a)=\delta(q, a)$,

2) for all $p\in H\cup\{\bar{q}\}$, $a\in\Sigma$: $\delta'(p, a)=\bar{q}$.

It is clear that $L(M')=L(M)$. Thus, $L\in\mathscr{L}_p\cap\mathscr{L}_3$.

Part 2. $L\in\mathscr{L}_3\cap\mathscr{L}_p\rightarrow L\in\mathscr{L}_{\text{sd3}}$.

By Theorem 3.3 in [1], we may assume that $L=L(M)$, where $M=(K, \Sigma, \delta, q_0, H)$ is an FD-automaton. We construct an SFD-machine M' from M as follows.

Let $M'=(K, \Sigma, \delta', q_0, H)$, where δ' is defined so that
1) for every $q\in K-H$, $a\in\Sigma$: $\delta'(q, a)=\delta(q, a)$,
2) for all $p\in H$, $a\in\Sigma$: $\delta'(p, a)$ is undefined.
We now prove that $L(M')=L(M)$.
$\langle\subseteqq\rangle$. By the definition of δ', we can easily see that if $w\in L(M')$ then $w\in L(M)$.
$\langle\supseteqq\rangle$. We shall prove that if $w\notin L(M')$ then $w\notin L(M)$. We now have two cases to consider:

Case 1. Let $w=w_1aw_2$ for $a\in\Sigma$, $w_1, w_2\in\Sigma^*$, and $q_0w_1aw_2\overset{*}{\underset{M'}{\vdash}}paw_2$ for some $p\in H$.

It is easy to see that $q_0w_1\overset{*}{\underset{M}{\vdash}}p$ for $p\in H$. Thus, $w_1\in L(M)$. Since $L(M)$ is prefix-free and $y=aw_2\neq\lambda$, $w=w_1y\notin L(M)$.

Case 2. Let $q_0w\overset{*}{\underset{M'}{\vdash}}q$ for some $q\notin H$.

It is clear that $q_0w\overset{*}{\underset{M}{\vdash}}q$ for $q\notin H$. Thus, $w\notin L(M)$. □

§ 4. Simple deterministic pushdown machines

In this section we investigate a special kind of deterministic pushdown automata known as simple deterministic pushdown machines (abbreviated SDP-machines) and prove that the family of all languages accepted by SDP-machines is the intersection of $\mathscr{L}_p$ and the family of all deterministic context-free languages. Furthermore, we can prove that this family is not closed under intersection, complementation and union, but it is closed under concatenation, homomorphism h_w, and L_1pL_2, $L_1\cup_pL_2$ are accepted by SDP-machines if L_1 is accepted by an SDP-machine and L_2 is an sfd-language. In this paper, however, we do not present all these proofs.

Let us consider the standard definition of a deterministic pushdown automaton (abbreviated DP-automaton, see [2]). That is: Let $M=(K, \Sigma, \Gamma, \delta, q_0, z_0, H)$ be a DP-automaton, where K is the set of states, Σ is the input alphabet, Γ is the pushdown alphabet, $q_0\in K$ is the initial state, $z_0\in\Gamma$ is the initial pushdown symbol, $H\subseteqq K$ is the set of final states, and δ is a mapping from $K\times(\Sigma\cup\{\lambda\})\times\Gamma$ to $K\times\Gamma^*$ satisfying the following conditions: for each $q\in K, z\in\Gamma$ either (i) $\delta(q, \lambda, z)$ is undefined and $\delta(q, a, z)$ contains exactly one element for all $a\in\Sigma$, or (ii) $\delta(q, \lambda, z)$ contains exactly one element and $\delta(q, a, z)$ is undefined for all $a\in\Sigma$.

Notation. Given a DP-automaton M let $\underset{M}{\vdash}$ be the relation on $K\times\Sigma^*\times\Gamma^*$ defined as follows

$$\text{for } q, p\in K, a\in\Sigma\cup\{\lambda\}, w\in\Sigma^*, z\in\Gamma, \alpha, \beta\in\Gamma^*,$$
$$(q, aw, \alpha z)\underset{M}{\vdash}(p, w, \alpha\beta) \quad \text{iff} \quad \delta(q, a, z)=(p, \beta).$$

Let $\overset{*}{\underset{M}{\vdash}}$ denote the transitive closure of $\underset{M}{\vdash}$. Finally, we define the language accepted

by M to be

$$L(M)=\{w\in\Sigma^*/(q_0, w, z_0)\overset{*}{\underset{M}{\vdash}}(p, \lambda, \alpha) \text{ for some } p\in H, \alpha\in\Gamma^*\}.$$

A language L is said to be deterministic context-free if $L=L(M)$ for some DP-automaton M. The family of all deterministic context-free languages is denoted by $\mathscr{L}_{d2}$.

Definition 4.1. a) A simple deterministic pushdown machine (abbreviated SDP-machine) is a 7-tuple $M=(K, \Sigma, \Gamma, \delta, q_0, z_0, H)$, where K, Σ, Γ, q_0, z_0 and H are the same as in the definition of a DP-automaton, and δ is a mapping from $(K-H)\times(\Sigma\cup\{\lambda\})\times\Gamma$ to $K\times\Gamma^*$ satisfying the following conditions: for each $q\in K-H, z\in\Gamma$ either (i) $\delta(q, \lambda, z)$ is undefined and $\delta(q, a, z)$ contains exactly one element for all $a\in\Sigma$ or (ii) $\delta(q, \lambda, z)$ contains exactly one element and $\delta(q, a, z)$ is undefined for all $a\in\Sigma$.

b) An input string is accepted by the SDP-machine M when the entire tape has been processed and the actual state is a final state. That is

$$L(M)=\{w\in\Sigma^*/(q_0, w, z_0)\overset{*}{\underset{M}{\vdash}}(p, \lambda, \alpha) \text{ for some } p\in H\}.$$

A language L is said to be simple deterministic context-free (abbreviated sdc-language) if $L=L(M)$ for some SDP-machine M. Finally, the family of all sdc-languages is defined by $\mathscr{L}_{sd2}$.

Theorem 4.1. Let L be any language over the alphabet Σ. L is an sdc-language if and only if L is deterministic context-free and prefix-free.

Proof. Part 1. $L\in\mathscr{L}_{sd2}\rightarrow L\in\mathscr{L}_{d2}\cap\mathscr{L}_p$.

Let $L=L(M)$ for an SDP-machine $M=(K, \Sigma, \Gamma, \delta, q_0, z_0, H)$. By the definition of δ, we can easily see that L is prefix-free. We now prove that $L\in\mathscr{L}_{d2}$. Construct a deterministic pushdown automaton M' from M as follows.

Let $M=(K\cup\{\bar{q}\}, \Sigma, \Gamma, \delta', q_0, z_0, H)$, where $\bar{q}\notin K$ and δ' is defined as

1) for every $q\in K-H, a\in\Sigma\cup\{\lambda\}, z\in\Gamma$, $\delta'(q, a, z)=\delta(q, a, z)$,

2) for all $p\in H\cup\{\bar{q}\}, z\in\Gamma$, $\delta'(p, \lambda, z)=(\bar{q}, \lambda)$.

It is clear that $L(M')=L(M)$. Thus, $L\in\mathscr{L}_{d2}\cap\mathscr{L}_p$.

Part 2. $L\in\mathscr{L}_{d2}\cap\mathscr{L}_p\rightarrow L\in\mathscr{L}_{sd2}$.

Let $L=L(M)$ for a DP-automaton $M=(K, \Sigma, \Gamma, \delta, q_0, z_0, H)$. Construct an SDP-machine M' from M as follows.

Let $M'=(K, \Sigma, \Gamma, \delta', q_0, z_0, H)$, where δ' is defined so that

1) for every $q\in K-H, z\in\Gamma, a\in\Sigma\cup\{\lambda\}$, $\delta'(q, a, z)=\delta(q, a, z)$,

2) for all $p\in H, z\in\Gamma, a\in\Sigma\cup\{\lambda\}$: $\delta'(p, a, z)$ is undefined.

We now prove that $L(M')=L(M)$.

($\subseteq$). By the construction of δ', we can easily see that if $w\in L(M')$ then $w\in L(M)$.

($\supseteq$) We shall prove that if $w\notin L(M')$ then $w\notin L(M)$. We now have three cases to consider:

Case 1. Let $w=w_1aw_2$ for $a\in\Sigma$, $w_1, w_2\in\Sigma^*$, and $(q_0, w_1aw_2, z_0)\overset{*}{\underset{M'}{\vdash}}(p, aw_2, \alpha)$ for some $p\in H$.

It is clear that $(q_0, w_1, z_0)\overset{*}{\underset{M}{\vdash}}(p, \lambda, \alpha)$ for $p\in H$. Consequently, $w_1\in L(M)$. Since $L(M)$ is prefix-free and $y=aw_2\neq\lambda$, $w\notin L(M)$.

Case 2. Let $w=w_1w_2$, where $w_1, w_2\in\Sigma^*$, and $(q_0, w_1w_2, z_0)\overset{*}{\underset{M'}{\vdash}}(q, w_2, \lambda)$ for some $q\in K-H$.

It is clear that $(q_0, w_1w_2, z_0)\underset{M}{\vdash}(q, w_2, \lambda)$ for $q\in K-H$. Thus, $w\notin L(M)$.

Case 3. Let $(q_0, w, z_0)\overset{*}{\underset{M'}{\vdash}}(q, \lambda, \alpha z)$ for some $q\in K-H$, $z\in\Gamma$ such that $\delta'(q, \lambda, z)$ is undefined.

It is easy to see that $w\notin L(M)$. □

In the following part we want to deal with a subfamily of simple deterministic context-free languages known as simple context-free languages (E. P. Friedman 1977).

Definition 4.2. (Definition 2.1 in [3]). a) A simple machine is a 4-tuple $M=(\Sigma, \Gamma, \delta, z_0)$, where Σ is a finite input alphabet, Γ is a finite pushdown alphabet, $z_0\in\Gamma$ is the initial pushdown symbol, δ is the partial transition function from $(\Sigma\cup\{\lambda\})\times\Gamma$ to Γ^* satisfying the following conditions: for each $z\in\Gamma$ either (i) $\delta(\lambda, z)$ is undefined and $\delta(a, z)$ contains exactly one element for all $a\in\Sigma$; or (ii) $\delta(\lambda, z)$ contains exactly one element and $\delta(a, z)$ is undefined for all $a\in\Sigma$.

Let $\underset{M}{\vdash}$ be the relation on $\Sigma^*\times\Gamma^*$ defined as follows: for each $a\in\Sigma\cup\{\lambda\}$, $w\in\Sigma^*$, $z\in\Gamma$, $\alpha, \beta\in\Gamma^*$, $(aw, \alpha z)\underset{M}{\vdash}(w, \alpha\beta)$ if $\delta(a, z)=\beta$.

Let $\overset{*}{\underset{M}{\vdash}}$ denote the transitive closure of $\underset{M}{\vdash}$. Finally, we define the language accepted by the simple machine M to be $L(M)=\{w\in\Sigma^*/(w, z_0)\overset{*}{\underset{M}{\vdash}}(\lambda, \lambda)\}$.

b) A language L is said to be simple context-free (abbreviated sc-language) if $L=L(M)$ for some simple machine M. It is easy to see that if L is an sc-language then L must be prefix-free. The family of all sc-languages is denoted by $\mathscr{L}_{sc}$.

Theorem 4.2. a) For every SFD-machine M, there is a simple machine M' such that $L(M')=L(M)$.

b) There is a simple machine M_1 such that $L(M_1)\notin\mathscr{L}_{sd3}$.

Proof. a) Let $M=(K, \Sigma, \delta, q_0, H)$ be an SFD-machine. We now construct the simple machine M' from M as follows.

Let $M'=(\Sigma, \Gamma', \delta', z_{q_0})$, where $\Gamma'=\{z_q/q\in K\}$ and δ' is defined so that

1) for each $q\in K-H$, $a\in\Sigma$, if $\delta(q, a)=p$ then $\delta'(a, z_q)=z_p$,

2) for all $p\in H$: $\delta'(\lambda, z_p)=\lambda$.

It is clear that $L(M')=L(M)$.

b) To prove the second statement, we reconsider the non-regular language (which is not an sfd-language), first seen in [2] $L=\{a^nb^n/n\geq 1\}$.

We now provide a simple machine M_1 which accepts this language $M_1=(\{a, b\}, \{Z_0, A, E\}, \delta_1, Z_0)$, where δ_1 is defined so that

1) $\delta_1(a, Z_0)=A$,
2) $\delta_1(a, A)=AA$,
3) $\delta_1(b, A)=\lambda$,
4) $\delta_1(b, Z_0)=E$,
5) $\delta_1(\lambda, E)=$E. □

Theorem 4.3. a) For every simple machine M, there is an SDP-machine M' such that $L(M')=L(M)$.

b) There is an SDP-machine M_1 such that $L(M_1)$ is not sc-language.

Proof. a) Let $M=(\Sigma, \Gamma, \delta, Z_0)$ be a simple machine. Without loss of generality, we may assume again that for arbitrary $a\in\Sigma\cup\{\lambda\}$ and $Z\in\Gamma$ if $\delta(a, Z)=\alpha$ then $\alpha\in(\Gamma-\{Z_0\})^*$. In the opposite case we can introduce a new initial pushdown symbol $\bar{Z}_0$ and take the new machine $\bar{M}=(\Sigma, \Gamma\cup\{\bar{Z}_0\}, \bar{\delta}, \bar{Z}_0)$, where $\bar{\delta}(a, \bar{Z}_0)=\delta(a, Z_0)$ and $\bar{\delta}(a, Z)=\delta(a, Z)$ for each $Z\in\Gamma$, $a\in\Sigma\cup\{\lambda\}$.

Construct an SDP-machine M' from M as follows. Let $M'=(K, \Sigma, \Gamma', \delta', q_0, Z_0, \{q_h\})$, where $K=\{q_0, q_h\}$, $\bar{\Gamma}=\Gamma\cup\{\bar{Z}_0\}$ for $\bar{Z}_0\notin\Gamma$, and δ' is defined so that for each $a\in\Sigma\cup\{\lambda\}$, $Z\in\Gamma-\{Z_0\}$

1) if $\delta(a, Z_0)=\alpha$ then $\delta'(q_0, a, Z_0)=(q_0, \bar{Z}_0\alpha)$,
2) if $\delta(a, Z)=\alpha$ then $\delta'(q_0, a, Z)=(q_0, \alpha)$,
3) $\delta'(q_0, \lambda, \bar{Z}_0)=(q_h, \lambda)$.

First by induction on the length of $w\in\Sigma^*$ we can easily prove that $(w, Z_0)\overset{*}{\underset{M}{\vdash}}(\lambda, \alpha)$ iff $(q_0, w, Z_0)\overset{*}{\underset{M'}{\vdash}}(q_0, \lambda, \bar{Z}_0\alpha)$. Now, let $w\in\Sigma^*$, then

$$w\in L(M) \leftrightarrow (w, Z_0)\overset{*}{\underset{M'}{\vdash}}(\lambda, \lambda)$$

$$\leftrightarrow (q_0, w, Z_0)\overset{*}{\underset{M'}{\vdash}}(q_0, \lambda, \bar{Z}_0)\overset{*}{\underset{M'}{\vdash}}(q_h, \lambda, \lambda)$$

$$\leftrightarrow w\in L(M').$$

b) To prove the second statement, we reconsider the non sc-language first seen in [3] $L=\{a^iba^ib/i\geqq 1\}\cup\{a^ica^ic/i\geqq 1\}$. We can easily check that the following SDP-machine M_1 accepts this language.

Let $M_1=(K, \{a, b, c\}, \Gamma, \delta_1, q_0, Z_0, \{q_h\})$, where $K=\{q_0, q_1, q_2, \bar{q}, q_h\}$, $\Gamma=\{Z_0, A\}$ and δ_1 is defined so that

1) 1.a) $\delta_1(q_0, a, Z_0) = (q_0, Z_0AA)$,
   1.b) $\delta_1(q_0, a, A) = (q_0, AA)$,
   1.c) $\delta_1(q_0, b, A) = (q_1, \lambda)$,
   1.d) $\delta_1(q_0, c, A) = (q_2, \lambda)$,
   1.e) $\delta_1(q_0, b, Z_0) = \delta_1(q_0, c, Z_0) = (\bar{q}, \lambda)$;

2) 2.a) $\delta_1(q_1, a, A) = (q_1, \lambda)$,
2.b) $\delta_1(q_1, b, A) = (\bar{q}, \lambda)$,
2.c) $\delta_1(q_1, c, A) = (\bar{q}, \lambda,)$,
2.d) $\delta_1(q_1, b, Z_0) = (q_h, \lambda)$;

3) 3.a) $\delta_1(q_2, a, A) = (q_2, \lambda)$,
3.b) $\delta_1(q_2, c, A) = (\bar{q}, \lambda)$,
3.c) $\delta_1(q_2, b, A) = (\bar{q}, \lambda)$,
3.d) $\delta_1(q_2, c, Z_0) = (q_h, \lambda)$;

4) $\delta_1(\bar{q}, \lambda, Z_0) = \delta_1(\bar{q}, \lambda, A) = (\bar{q}, \lambda)$. □

Theorem 4.4. There exists a prefix-free context-free language which is not an sdc-language.

Proof. Let Σ be a finite nonempty alphabet. By Corollary 1 to Theorem 3.5 in [2], we can easily see that $L=\{ww^R/w\in\Sigma^*\}$ is a context-free language which is not deterministic context-free, where w^R is the mirror image of w. Let c be a symbol not is Σ, and set $L_1=L\cdot\{c\}=\{ww^Rc/w\in\Sigma^*\}$. It is easy to see that $L_1\in\mathscr{L}_2\cap\mathscr{L}_p$. We now prove that $L_1\notin\mathscr{L}_{sd2}$. Assume on the contrary that $L_1\in\mathscr{L}_{sd2}$. By Corollary to Theorem 3.4 in [2], if $L_1=L\cdot\{c\}$ is deterministic context-free then L is deterministic context-free, and the contradiction arises. Consequently, $L_1=\{ww^Rc/x\in\Sigma^*\}$ is a prefix-free context-free language which is not an sdc-language. □

§ 5. Simple deterministic linear bounded machines

In this section we investigate a special kind of deterministic linear bounded automata called simple deterministic linear bounded machines (abbreviated SDLB-machines), and prove that the family of all languages accepted by SDLB-machines is the intersection of the family $\mathscr{L}_p$ and the family of all deterministic context-sensitive languages. Furthermore, we mention without proof that this family is closed under concatenation, intersection, p-quotient, p-union and homomorphism h_w; but it is not closed under complementation and union.

Let us consider the standard definition of a deterministic linear bounded automaton (abbreviated DLB-automaton, see [7]). That is: Let $M=(\Gamma, K, \delta, q_0, H)$ be a DLB-automaton, where Γ is the tape alphabet, K is the set of states, $q_0\in K$ is the initial state, $H\subseteq K$ is a set of final states and $\delta: K\times\Gamma\to K\times(\Gamma\cup\{R, L\})$ is the mapping satisfying the condition: for arbitrary $q\in K$ and $x\in\Gamma$, $\delta(q, x)$ contains exactly one element.

Notation. An instantaneous configuration is a word of the form w_1qw_2, where $q\in K, w_1, w_2\in\Gamma^*$ and $w_1w_2\neq\lambda$. Given a DLB-automaton M let $\underset{M}{\vdash}$ be the relation on configurations of M defined as follows. For $q, p\in K, x, y\in\Gamma, w_1, w_2\in\Gamma^*$

$$w_1qxw_2 \underset{M}{\vdash} w_1pyw_2 \quad \text{iff} \quad \delta(q, x) = (p, y),$$

$$w_1qxw_2 \underset{M}{\vdash} w_1xpw_2 \quad \text{iff} \quad \delta(q, x) = (p, R),$$

$$w_1yqxw_2 \underset{M}{\vdash} w_1pyxw_2 \quad \text{iff} \quad \delta(q, x) = (p, L).$$

Let $\overset{*}{\underset{M}{\vdash}}$ denote the transitive closure of $\underset{M}{\vdash}$. Finally, we define the language accepted by a DLB-automaton M to be $L(M)=\{w\in\Sigma^*/q_0w\overset{*}{\underset{M}{\vdash}}\alpha p \text{ for some } p\in H\}$, where $\Sigma\subseteqq\Gamma$. A language L is said to be deterministic context-sensitive (abbreviated dcs-language) if $L=L(M)$ for some DLB-automaton M. The family of all dcs-languages is denoted by $\mathscr{L}_{dl}$.

Lemma 5.1. Let L be a dcs-language over the alphabet Σ. Then there is a DLB-automaton $M'=(\Gamma', K', \delta', q_0', H')$ such that

i) $L=L(M')$,

ii) $\delta'\colon K'\times\Gamma'\to K'\times((\Gamma'-\Sigma)\cup\{R, L\})$ is the mapping satisfying the following condition: for arbitrary $q\in K'$ and $a\in\Sigma$, there is a $z\in\Gamma'-\Sigma$ such that $\delta'(q, a)=(p, z)$, i.e., there are no forms $\delta(q, a)=(p, R)$ or $\delta(q, a)=(p, L)$.

Proof. Let $L=L(M)$ for a DLB-automaton $M=(\Gamma, K, \delta, q_0, H)$. We now construct a DLB-automaton M' from M as follows. Let $M'=(\Gamma', K', \delta', q_0', H')$, where $\Gamma'=\Gamma\cup\{a'/a\in\Sigma\}$, $K'=K$, $q_0'=q_0$, $H'=H$ and δ' is defined so that

1) for arbitrary $a\in\Sigma$ and $q\in K$, $\delta'(q, a)=(q, a')$,

2) for arbitrary $x\in\Gamma-\Sigma$ and $q\in K$

2.a) if $\delta(q, x)=(p, i)$ for an $i\in\{R, L\}$ then $\delta'(q, x)=(p, i)$,

2.b) if $\delta(q, x)=(p, y)$ then $\delta'(q, x)=(p, \bar{y})$, where

$$\bar{y}=\begin{cases} y & \text{if } y\in\Gamma-\Sigma,\\ y' & \text{if } y\in\Sigma,\end{cases}$$

3) for arbitrary $q\in K$ and $a\in\Sigma$.

3.a) if $\delta(q, a)=(p, i)$ for an $i\in\{R, L\}$ then $\delta'(q, a')=(p, i)$,

3.b) if $\delta(q, a)=(p, y)$ then $\delta'(q, a')=(p, \bar{y})$, where

$$\bar{y}=\begin{cases} y & \text{if } y\in\Gamma-\Sigma,\\ y' & \text{if } y\in\Sigma.\end{cases}$$

It is clear that $L(M')=L(M)$ and the condition ii) is satisfied. □

Definition 5.1. a) A simple deterministic linear bounded machine (abbreviated SDLB-machine) is a 6-tuple $M=(\Gamma, K, \Sigma, \delta, q_0, H)$, where Γ is the tape alphabet, K is the set of states, Σ is the input alphabet for $\Sigma\cap\Gamma=\emptyset$, $q_0\in K$ is the initial state, $H\subseteqq K$ is a set of final states, and $\delta\colon K\times(\Gamma\cup\Sigma)\to K\times(\Gamma\cup\{R, L\})$ is the mapping satisfying the following conditions

i) for arbitrary $q\in K-H$ and $x\in\Gamma\cup\Sigma$: $\delta(q, x)$ contains exactly one element,

ii) for every $p\in H$: $\delta(p, a)$ contains exactly one element if $a\in\Gamma$ and it is undefined if $a\in\Sigma$.

b) An instantaneous configuration and the relation $\overset{*}{\underset{M}{\vdash}}$ are defined as in the case of a DLB-automaton. We define the language accepted by a SDLB-machine M to be

$$L(M)=\{w\in\Sigma^*/q_0w\overset{*}{\underset{M}{\vdash}}\alpha p \text{ for some } p\in H\}.$$

A language L is said to be simple deterministic context-sensitive (abbreviated sdcs-language) if $L=L(M)$ for some SDLB-machine M. The family of all sdcs-languages is denoted by $\mathscr{L}_{\text{sdl}}$.

Theorem 5.2. Let L be any language over the alphabet Σ. L is an sdcs-language if and only if L deterministic context-sensitive and prefix-free.

Proof. Part 1. $L\in\mathscr{L}_{\text{sdl}}\rightarrow L\in\mathscr{L}_{d1}\cap\mathscr{L}_p$.

Let $L=L(M)$ for an SDLB-machine $M=(\Gamma, K, \Sigma, \delta, q_0, H)$. By condition ii) of δ, we can easily see that if M is an SDLB-machine then $L(M)$ must be prefix-free. We now prove that $L\in\mathscr{L}_{d1}$. Construct a DLB-automaton M' from M as follows.

Let $M'=(\Gamma', K', \delta', q_0, H)$, where $\Gamma'=\Gamma\cup\Sigma$, $K'=K\cup\{\bar{q}\}$ for $\bar{q}\notin K$, and δ' is defined so that

1) for every $q\in K-H$: $\delta'(q, x)=\delta(q, x)$ for all $x\in\Gamma\cup\Sigma$,

2) for every $p\in H$

$$\delta'(p, a)=\begin{cases}\delta(p, a) & \text{if } a\in\Gamma,\\ (\bar{q}, R) & \text{if } a\in\Sigma,\end{cases}$$

3) $\delta'(\bar{q}, x)=(\bar{q}, R)$ for all $x\in\Gamma\cup\Sigma$.

It is clear that $L(M')=L(M)$. Thus $L\in\mathscr{L}_{d1}\cap\mathscr{L}_p$.

Part 2. $L\in\mathscr{L}_{d1}\cap\mathscr{L}_p\rightarrow L\in\mathscr{L}_{\text{sdl}}$.

Without loss of generality, we may assume that $L=L(M)$ for a DLB-automaton $M=(\Gamma, K, \delta, q_0, H)$ satisfying condition ii) of Lemma 5.1. We now construct an SDLB-machine M' from M as follows. Let $M'=(\Gamma', K, \Sigma, \delta', q_0, H)$, where $\Gamma'=\Gamma-\Sigma$, δ' is defined so that

1) for every $q\in K-H$: $\delta'(q, x)=\delta(q, x)$ for all $x\in\Gamma'\cup\Sigma$,

2) for every $p\in H$

$$\delta'(p, a)=\begin{cases}\delta(p, a) & \text{if } a\in\Gamma',\\ \text{undefined} & \text{if } a\in\Sigma.\end{cases}$$

We now prove that $L(M')=L(M)$.

($\subseteq$). Let $w\in L(M')$, i.e., $q_0w\overset{*}{\underset{M'}{\vdash}}\alpha p$ for some $p\in H$. By the definition of δ', we obtain: $q_0w\overset{*}{\underset{M}{\vdash}}\alpha p$ for $p\in H$. Thus $w\in L(M)$.

($\supseteq$). We shall prove that if $w\notin L(M')$ then $w\notin L(M)$. There are two cases to consider:

Case a) Let $w=w_1aw_2$ for $a\in\Sigma$, $w_1, w_2\in\Sigma^*$ and $q_0w_1aw_2\overset{*}{\underset{M'}{\vdash}}\alpha paw_2$ for some $p\in H$.

It is clear that $q_0w_1\overset{*}{\underset{M}{\vdash}}\alpha p$ for $p\in H$. Since L is prefix-free and $y=aw_2\neq\lambda$, we obtain: $w=w_1y\notin L(M)$.

Case b) Let $q_0w\overset{*}{\underset{M'}{\vdash}}\alpha q$ for some $q\in K-H$.

It is clear that $q_0w\overset{*}{\underset{M}{\vdash}}\alpha q$ for $q\notin H$. Thus $w\notin L(M)$. □

Theorem 5.3. a) For every SDP-machine M, there is an SDLB-machine M' such that $L(M')=L(M)$.

b) There is an SDLB-machine M_1 such that $L(M_1)\notin\mathscr{L}_{sd2}$.

Proof. a) Part a) holds, due to Theorem 5.2 and the following statement (Theorem 3 in [5]) "A context-free language is accepted by a deterministic linear bounded automaton".

b) To prove part b), we reconsider the non context-free language

$$L=\{a^nb^nc^n/n\geqq 1\}$$

(which is not an sdc-language either), first seen in [1], [4], [7].

We now construct an SDLB-machine M_1 which accepts this language. Let $M_1=(\Gamma_1, K_1, \{a,b,c\}, \delta_1, q_0, \{q_h\})$, where

$$\Gamma_1=\{A,B,C,X,Y\},\quad K_1=\{q_0,q_1,q_2,\ldots,q_{11},q_{12},q,q_h\},$$

δ_1 is defined so that:

1) $\delta_1(q_0,a)=(q_0,A)$, $\delta_1(q_0,A)=(q_1,R)$, $\delta_1(q_1,a)=(q_1,X)$, $\delta_1(q_1,X)=(q_1,R)$, $\delta_1(q_1,b)=(q_1,B)$, $\delta_1(q_1,B)=(q_2,R)$, $\delta_1(q_2,b)=(q_2,Y)$, $\delta_1(q_2,Y)=(q_2,R)$.
2) $\delta_1(q_2,c)=(q_2,C)$, $\delta_1(q_2,C)=(q_3,L)$, $\delta_1(q_3,Z)=(q_3,L)$, for $Z\in\{Y,B,X\}$.
3) $\delta_1(q_3,A)=(q_4,R)$, $\delta_1(q_4,X)=(q_4,A)$, $\delta_1(q_4,A)=(q_5,R)$, $\delta_1(q_4,B)=(q_{12},R)$.
4) $\delta_1(q_5,X)=(q_6,R)$, $\delta_1(q_5,B)=(q_7,R)$.
5) $\delta_1(q_6,Y)=(q_3,X)$, $\delta_1(q_6,Z)=(q_6,R)$ for $Z\in\{B,X\}$.
6) $\delta_1(q_7,X)=(q_7,R)$, $\delta_1(q_7,Y)=(q_8,X)$.
7) $\delta_1(q_8,X)=(q_8,L)$, $\delta_1(q_8,B)=(q_9,R)$, $\delta_1(q_9,X)=(q_9,B)$, $\delta_1(q_9,B)=(q_{10},R)$.
8) $\delta_1(q_{10},X)=(q_{11},R)$, $\delta_1(q_{10},C)=(q_{10},R)$.
9) $\delta_1(q_{11},c)=(q_8,C)$, $\delta_1(q_8,C)=(q_8,L)$, $\delta_1(q_{11},Z)=(q_{11},R)$ for $Z\in\{X,C\}$.
10) $\delta_1(q_{10},c)=(q_{12},C)$, $\delta_1(q_{12},C)=(q_h,R)$.
11) $\delta_1(\bar{q},Z)=(\bar{q},A)$ for $Z\in(\Gamma_1-\{A\})\cup\{a,b,c\}$, $\delta_1(q_h,Z)=(\bar{q},A)$ for $Z\in\Gamma_1$, $\delta_1(\bar{q},A)=(\bar{q},R)$.
12) In all other cases for arbitrary $q\in K_1-\{\bar{q},q_h\}$ and $x\in\Gamma_1\cup\{a,b,c\}$, $\delta_1(q,x)=(\bar{q},A)$.

It is easy to see that if $w\in\{a,b,c\}^*-\{a^nb^nc^n/n\geqq 1\}-\{\lambda\}$, then

$$q_0w\underset{M'}{\vdash}\alpha q',\quad\text{where}\quad\alpha\in\Gamma_1^*,\quad\text{and}$$

$$q'=\begin{cases}q_1 & \text{if}\quad w=a^n,\\ q_2 & \text{if}\quad w=a^nb^m,\\ \bar{q} & \text{otherwise.}\end{cases}$$

Consequently, $w\notin L(M_1)$.

We now check that $w=a^nb^nc^n\in L(M_1)$ for all $n\geqq 1$. Indeed, for $n=1$

$$q_0abc \overset{*}{\underset{M_1}{\vdash}} ABq_2C \overset{*}{\underset{M_1}{\vdash}} q_3ABC \underset{M_1}{\vdash} Aq_4BC \underset{M_1}{\vdash} ABq_{12}C \underset{M_1}{\vdash} ABCq_h.$$

Similarly, for all $n\geqq 1$

$$q_0a^{n+1}b^{n+1}c^{n+1} \overset{*}{\underset{M_1}{\vdash}} AX^nBY^nq_2Cc^n \overset{*}{\underset{M_1}{\vdash}} A^{n+1}BX^{n-1}q_8XCc^n$$
$$\overset{*}{\underset{M_1}{\vdash}} A^{n+1}B^{n+1}C^nq_{10}c \underset{M_1}{\vdash} A^{n+1}B^{n+1}C^nq_{12}C \underset{M_1}{\vdash} A^{n+1}B^{n+1}C^{n+1}q_h.$$

Consequently, $a^nb^nc^n\in L(M_1)$ for all $n\geqq 1$.

§ 6. Simple deterministic Turing-machines

In this section we investigate a special kind of deterministic Turing-machines known as simple deterministic Turing-machines, (abbreviated SDT-machines), and prove that the family of all languages accepted by SDT-machines is the intersection of the two classes $\mathscr{L}_0$ and $\mathscr{L}_p$. Furthermore, we can prove that this family is closed under concatenation, intersection, p-quotient, p-union, and homomorphism h_w; but it is not closed under complementation and union. In this paper we do not prove these statements.

Definition 6.1. a) A deterministic Turing-machine (abbreviated DT-machine) is a 6-tuple $M=(K, \Gamma, \Sigma, \delta, q_0, H)$, where K is the set of states, Γ is the set of tape symbols, one of these, usually denoted by B, is the blank, $\Sigma\subseteqq\Gamma-\{B\}$ is the set of input symbols, $q_0\in K$ is the initial state, $H\subseteqq K$ is the set of final states, and δ: $K\times\Gamma\to K\times(\Gamma-\{B\})\times\{R, L\}$ is the mapping satisfying the following condition: for arbitrary $q\in K$ and $z\in\Gamma$, $\delta(q, z)$ contains exactly one element.

b) We denote a configuration of the DT-machine M by w_1qw_2 or qBw for $w, w_1, w_2\in(\Gamma-\{B\})^*$ and $w_1w_2\neq\lambda$. Let $\underset{M}{\vdash}$ be the relation on configurations of M given as follows. For arbitrary $q\in K, x, y\in\Gamma-\{B\}$ and $w_1, w_2\in(\Gamma-\{B\})^*$

1) $w_1qxw_2 \underset{M}{\vdash} w_1zpw_2$ if $\delta(q, x)=(p, z, R)$,
2) $w_1yqxw_2 \underset{M}{\vdash} w_1pyzw_2$ if $\delta(q, x)=(p, z, L)$,
3) $qxw_2 \underset{M}{\vdash} pBzw_2$ if $\delta(q, x)=(p, z, L)$,
4) $qBw_2 \underset{M}{\vdash} zpw_2$ if $\delta(q, B)=(p, z, R)$,
5) $qBw_2 \underset{M}{\vdash} pBzw_2$ if $\delta(q, B)=(p, z, L)$.

Let $\overset{*}{\underset{M}{\vdash}}$ denote the transitive closure of $\underset{M}{\vdash}$. Finally, define the language accepted by the DT-machine M to be $L(M)=\{w\in\Sigma^*/q_0w \overset{*}{\underset{M}{\vdash}} \alpha p$ for some $p\in H, \alpha\in(\Gamma-\{B\})^*\}$.

Remark. In this definition the tape of the Turing-machine is infinitely extensible to the left, but is totally bounded to the right by the end of the tape. By a carry

forward algorithm to the left (M. DAVIS, 1958, [6]), we can prove that this is the equivalent of Turing-machine definition in [1] such that its tape is totally bounded to the left by the end of the tape.

Lemma 6.1. Let L be a type-0 language over the alphabet Σ. Then there is a DT-machine $M'=(K', \Gamma', \Sigma, \delta', q_0', H')$ such that

i) $L=L(M')$,

ii) the mapping δ' satisfies the following condition: for arbitrary $q\in K'$ and $x\in\Gamma'$ if $\delta'(q, x)=(p, z, i)$ for an $i\in\{R, L\}$ then $z\notin\Sigma'$.

Proof. By the Theorem 6.3 in [1], we may assume that $L=L(M)$ for a DT-machine $M=(K, \Gamma, \Sigma, \delta, q_0, H)$.

We now construct a DT-machine M' from M as follows. Let $M'=(K', \Gamma', \Sigma, \delta', q_0', H')$, where $K'=K, \Gamma'=\Gamma\cup\{a'/a\in\Sigma\}, q_0'=q_0, H'=H$, and δ' is defined so that

1) for arbitrary $q\in K$ and $x\in\Gamma$: if $\delta(q, x)=(p, z, i)$ for $i\in\{R, L\}$ then $\delta'(q, x)=(p, \bar{z}, i)$, where

$$\bar{z}=\begin{cases} z & \text{if} \quad z\in\Gamma-\Sigma-\{B\}, \\ z' & \text{if} \quad z\in\Sigma, \end{cases}$$

2) for arbitrary $q\in K$ and $a\in\Sigma$: $\delta'(q, a')=\delta'(q, a)$.

It is clear that $L(M')=L(M)$ and the condition ii) is satisfied. □

Definition 6.2. a) A simple deterministic Turing-machine (abbreviated SDT-machine), is a 6-tuple $M=(K, \Gamma, \Sigma, \delta, q_0, H)$, where K is the set of states, Γ is the set of tape symbols; one of these, usually denoted by B, is the blank, Σ is the set of input symbols for which $\Sigma\cap\Gamma=\emptyset$, and $\delta: K\times(\Gamma\cup\Sigma)\to K\times(\Gamma-\{B\})\times\{R, L\}$ is the mapping satisfying the following conditions

i) for arbitrary $q\in K-H$ and $x\in\Gamma\cup\Sigma$: $\delta(q, x)$ contains exactly one element,

ii) for each $p\in H$: $\delta(p, a)$ is undefined if $a\in\Sigma$, and it contains exactly one element for all $a\in\Gamma$.

b) The relation $\overset{*}{\underset{M}{\vdash}}$ is defined as in the case of a DT-machine. Finally we define the language accepted by an SDT-machine M to be $L(M)=\{w\in\Sigma^*/q_0 w\overset{*}{\underset{M}{\vdash}}\alpha p$ for some $p\in H, \alpha\in(\Gamma-\{B\})^*\}$. A language L is said to be simple deterministic type-0 (abbreviated sd0-language) if $L=L(M)$ for some SDT-machine M. The family of all sd0-languages is denoted by $\mathscr{L}_{\text{sd0}}$.

Theorem 6.2. Let L be any language over the alphabet Σ. L is an sd0-language if and only if L is prefix-free and $L\in\mathscr{L}_0$.

Proof. Part 1. $L\in\mathscr{L}_{\text{sd0}}\to L\in\mathscr{L}_0\cap\mathscr{L}_p$.

Let $L=L(M)$ for an SDT-machine $M=(K, \Gamma, \Sigma, \delta, q_0, H)$. By the definition of δ, we can easily see that $L\in\mathscr{L}_p$. On the other hand, it is easy to see that an SDT-machine is a Turing-machine. Consequently, $L(M)\in\mathscr{L}_0\cap\mathscr{L}_p$.

Part 2. $L\in\mathscr{L}_0\cap\mathscr{L}_p\to L\in\mathscr{L}_{\text{sd0}}$.

By Lemma 6.1, we may assume that $L=L(M)$ for a DT-machine $M=(K, \Gamma, \Sigma, \delta, q_0, H)$ satisfying the condition ii) of Lemma 6.1, i.e., for arbitrary $q\in K$ and $x\in\Gamma$ if $\delta(q, x)=(p, z, i)$ then $z\notin\Sigma$, where $i\in\{R, L\}$. We now construct the SDT-machine M' from M as follows. Let $M'=(K, \Gamma', \Sigma, \delta', q_0, H)$, where $\Gamma'=\Gamma-\Sigma$, δ' is defined as

1) for arbitrary $q\in K-H$ and $x\in\Gamma'\cup\Sigma$: $\delta'(q, x)=\delta(q, x)$,

2) for each $p\in H$

$$\delta'(p, a)=\begin{cases}\delta(p, a) & \text{if } a\in\Gamma',\\ \text{undefined} & \text{if } a\in\Sigma.\end{cases}$$

We prove that $L(M')=L(M)$.

($\subseteq$). Let $w\in L(M')$, i.e., $q_0w\overset{*}{\underset{M'}{\vdash}}\alpha p$ for some $p\in H$. It is clear that: $q_0w\overset{*}{\underset{M}{\vdash}}\alpha p$ for $p\in H$. Thus, $w\in L(M)$.

($\supseteq$). We shall prove that if $w\notin L(M')$ then $w\notin L(M)$. We have two cases to consider:

Case 1. Let $w=w_1aw_2$, for $a\in\Sigma, w_1, w_2\in\Sigma^*$, and $q_0w_1aw_2\overset{*}{\underset{M'}{\vdash}}\alpha paw_2$ for some $p\in H$.

It is easy to see that: $q_0w_1\overset{*}{\underset{M}{\vdash}}\alpha p$ for $p\in H$, i.e., $w_1\in L(M)$. Since $L(M)$ is prefix-free and $y=aw_2\neq\lambda$, $w=w_1y\notin L(M)$.

Case 2. Let $q_0w\overset{*}{\underset{M'}{\vdash}}\alpha p$ for some $q\in K-H$.

It is clear that: $q_0w\overset{*}{\underset{M}{\vdash}}\alpha q$ for $q\notin H$. Thus, $w\notin L(M)$. □

Theorem 6.3. a) For every SDLB-machine M, there is an SDT-machine M' such that $L(M')=L(M)$.

b) There is an SDT-machine M_1 such that $L(M_1)$ is not an sdcs-language.

Proof. a) Part a) is implied, by Theorems 5.2., 6.2 and the following statement "A context-sensitive language is of type-0".

b) By Theorem III/9.4 in [7], there is a type-0 language $L\in\{a, b\}^*$ which is not context-sensitive. Without loss of generality, we may assume that $\lambda\notin L$.

First, we can easily check that $L_1=L\cdot\{c\}=\{wc/w\in L\}\in\mathscr{L}_0\cap\mathscr{L}_p$. Consequently, $L_1\in\mathscr{L}_{\text{sd0}}$. We now prove that $L_1\notin\mathscr{L}_1$ (i.e., L_1 is not an sdcs-language either). Assume on the contrary that $L_1\in\mathscr{L}_1$. We consider the following homomorphism h: $\{a, b, c\}^*\to\{a, b\}^*$ such that

$$h(\lambda)=\lambda,\quad h(a)=a,\quad h(b)=b,\quad h(c)=\lambda.$$

It is clear that if $x\in L_1=L\cdot\{c\}$, then $\lg(h(x))=\lg(x)-1$ (where $\lg(x)$ denotes the length of x). On the other hand, it can be easily seen that if $x\in L_1$ then $\lg(x)\geqq 2$. Consequently, for all $x\in L_1$: $2\lg(h(x))=2(\lg(x)-1)\geqq\lg(x)$, i.e., h is termed a 2-linear erasing with respect to L_1 (this definition can be found in [7]). By Theorem III/10.4 in [7], if $L_1\in\mathscr{L}_1$ then $L=h(L_1)\in\mathscr{L}_1$, and the contradiction arises. Thus, L is an sd0-language which is not an sdcs-language. □

In the final part we wish to deal with the two memory simple machine that is the equivalent of an SDT-machine.

Definition 6.3. a) A two memory simple machine (abbreviated TS-machine) is a 6-tuple $M=(\Sigma, \Gamma, \Gamma', \delta, z_0, z_0')$, where Σ is the set of input symbols, Γ and Γ' are two sets of pushdown symbols, $z_0\in\Gamma$, $z_0'\in\Gamma'$ are two initial symbols of two pushdown stores, and the mapping $\delta: \Gamma\times(\Sigma\cup\{\lambda\})\times\Gamma'\to\Gamma^*\times\Gamma'^*$ satisfies the following conditions: for arbitrary $z\in\Gamma$ and, $z'\in\Gamma'$ either (i) $\delta(z, \lambda, z')$ is undefined and $\delta(z, a, z')$ contains exactly one element for all $a\in\Sigma$; or (ii) $\delta(z, \lambda, z')$ contains exactly one element and $\delta(z, a, z')$ is undefined for all $a\in\Sigma$.

b) A configuration of M is a triplet (α, w, α'), where $w\in\Sigma^*$, $\alpha\in\Gamma^*$, $\alpha'\in\Gamma'^*$. We define the operator $\underset{M}{\vdash}$ on configurations of M as follows. For arbitrary $a\in\Sigma\cup\{\lambda\}$, $w\in\Sigma^*$, $z\in\Gamma$, $z'\in\Gamma'$, $\alpha, \beta\in\Gamma^*$ and $\alpha', \beta'\in\Gamma'^*$: $(\alpha z, aw, \alpha' z')\underset{M}{\vdash}(\alpha\beta, w, \alpha'\beta')$ if $\delta(z, a, z')=(\beta, \beta')$. Let $\overset{*}{\underset{M}{\vdash}}$ denote the transitive closure of $\underset{M}{\vdash}$. Finally, we shall be concerned with the acceptance of an input tape by empty pushdown stores. Accordingly, we define the language accepted by a TS-machine M to be

$$L(M) = \{w\in\Sigma^*/(z_0, w, z_0')\overset{*}{\underset{M}{\vdash}}(\lambda, \lambda, \lambda)\}.$$

Theorem 6.4. Let L be any language over the alphabet Σ. L is an sd0-language if and only if L is accepted by some TS-machine M.

Proof. Part 1. Let $L=L(M)$ for an SDT-machine $M=(K, \Gamma, \Sigma, \delta, q_0, H)$. Without loss of generality, we may assume again that: for arbitrary $q\in K$, and $a\in\Sigma\cup\{\lambda\}$ if $\delta(q, a)=(p, z, i)$ then $p\neq q_0$, where $i\in\{R, L\}$. We now construct the TS-machine M_1 from M as follows. Let $M_1=(\Sigma, \Gamma_1, \Gamma_1', \delta_1, q_0, \$)$, where $\Gamma_1=K\cup\Gamma\cup\{\$\}$, $\Gamma_1'=(K-\{q_0\})\cup\Gamma\cup\{\$\}$ for $\$\notin K\cup\Gamma$, and δ_1 is defined so that:

1) For arbitrary $q\in K-H-\{q_0\}$ and $a\in\Sigma$:
   a) if $\delta(q_0, a)=(p, x, R)$ then $\delta_1(q_0, a, \$)=(Bxp, \$)$,
   b) if $\delta(q_0, a)=(p, x, L)$ then $\delta_1(q_0, a, \$)=(B, \$xp)$,
   c) if $\delta(q, a)=(p, x, R)$ then $\delta_1(q, a, \$)=(xp, \$)$,
   d) if $\delta(q, a)=(p, x, L)$ then $\delta_1(q, a, \$)=(\lambda, \$xp)$.
2) For arbitrary $p\in H$ and $y\in\Gamma-\{B\}$:
   a) $\delta_1(p, \lambda, \$)=(\lambda, \$)$,
   b) $\delta_1(y, \lambda, \$)=(\lambda, \$)$,
   c) $\delta_1(B, \lambda, \$)=(\lambda, \lambda)$.
3) For arbitrary $q\in K-\{q_0\}$ and $z\in\Gamma$:
   a) if $\delta(q, z)=(p, x, R)$ then $\delta_1(q, \lambda, z)=(xp, \lambda)$,
   b) if $\delta(q, z)=(p, x, L)$ then $\delta_1(q, \lambda, z)=(\lambda, xp)$,
   c) $\delta_1(q_0, \lambda, y)=(\$, \$)$ for all $y\in\Gamma$.
4) For arbitrary $y\in\Gamma-\{B\}$ and $q\in K-\{q_0\}$:
   a) $\delta_1(y, \lambda, q)=(q, y)$,
   b) $\delta_1(B, \lambda, q)=(Bq, B)$.
5) For arbitrary $y_1\in\Gamma$ and $y_2\in\Gamma$: $\delta_1(y_1, \lambda, y_2)=(\$, \$)$.
6) For arbitrary $q_1\in K$ and $q_2\in K-\{q_0\}$: $\delta_1(q_1, \lambda, q_2)=(\$, \$)$.
7) For each $z\in\Gamma_1'=(K-\{q_0\})\cup\Gamma\cup\{\$\}$: $\delta_1(\$, \lambda, z)=(\$, \$)$.

It is easy to see that

$$w\in L(M) \leftrightarrow q_0 w \overset{*}{\underset{M}{\vdash}} \alpha p \quad \text{for some} \quad p\in H$$

$$\leftrightarrow \{(q_0, w, \$) \overset{*}{\underset{M_1}{\vdash}} (B\alpha p, \lambda, \$) \quad \text{for} \quad p\in H$$

$$\underset{M_1}{\vdash} (B\alpha, \lambda, \$) \overset{*}{\underset{M_1}{\vdash}} (B, \lambda, \$) \underset{M_1}{\vdash} (\lambda, \lambda, \lambda)\}$$

$$\leftrightarrow w\in L(M_1).$$

Thus, $L=L(M_1)$ for the TS-machine M_1.

Part 2. Let $L=L(M)$ for a TS-machine M.

By the acceptance of an input tape by empty pushdown stores, it can be easily seen that L is prefix-free. On the other hand, by the Church's thesis, $L\in\mathscr{L}_0$. Consequently, $L\in\mathscr{L}_{\text{sd0}}$. □

Finally, we prove that every sd0-language equals to a homomorphic image of the intersection of two simple deterministic context-free languages.

Theorem 6.5. Every sd0-language L can be expressed in the form $L=h(L_1\cap L_2)$, where h is a homomoprhism and L_1, L_2 are simple context-free languages.

Proof. By Theorem 6.4, we may assume that $L=L(M)$, where $M=(\Sigma, \Gamma, \Gamma', \delta, z_0, z_0')$ is a TS-machine. First, we set:

$$\Sigma_1 = \{x_{[z,z']}/z\in\Gamma, z'\in\Gamma'\}, \bar{\Gamma} = \{[z, z']/z\in\Gamma, z'\in\Gamma'\}.$$

We now construct two simple machines M_1 and M_2 from M in the following way. Let $M_1=(\Sigma', \Gamma_1, \delta_1, z_0)$, $M_2=(\Sigma', \Gamma_2, \delta_2, z_0)$, where $\Sigma'=\Sigma\cup\Sigma_1$, $\Gamma_1=\Gamma\cup\bar{\Gamma}\cup\{\$\}$, $\Gamma_2=\Gamma'\cup\bar{\Gamma}\cup\{\$\}$, and δ_1, δ_2 are defined as follows:

1) For arbitrary $y, z\in\Gamma$ and $y', z'\in\Gamma'$:

a) $\delta_1(x_{[y,z']}, z) = \begin{cases} [z, z'] & \text{if} \quad y = z \\ \$ & \text{if} \quad y \neq z, \end{cases}$

b) $\delta_2(x_{[z\ y']}, z') = \begin{cases} [z, z'] & \text{if} \quad y' = z', \\ \$ & \text{if} \quad y' \neq z', \end{cases}$

c) $\delta_1(a, z) = \$$, $\delta_2(a, z') = \$$ for all $a\in\Sigma$.

2) For arbitrary $z\in\Gamma$ and $z'\in\Gamma'$:

a) The case where $\delta(z, \lambda, z')$ is defined.

If $\delta(z, \lambda, z')=(\alpha, \alpha')$ then $\delta_1(\lambda, [z, z'])=\alpha$, $\delta_2(\lambda, [z, z'])=\alpha'$.

b) The case where $\delta(z, \lambda, z')$ is undefined.

For every $a\in\Sigma$, if $\delta(z, a, z')=(\alpha, \alpha')$ then $\delta_1(a, [z, z'])=\alpha$, $\delta_2(a, [z, z'])=\alpha'$, and $\delta_1(b, [z, z'])=\$$, $\delta_2(b, [z, z'])=\$$ for all $b\in\Sigma_1$.

3) $\delta_1(\lambda, \$)=\$$, $\delta_2(\lambda, \$)=\$$.

Let h be the homomorphism of Σ' into Σ defined by

$$h(a)=\begin{cases} a & \text{if } a\in\Sigma, \\ \lambda & \text{if } a\in\Sigma_1. \end{cases}$$

We now prove that $L(M)=h(L_1\cap L_2)$ for $L_1=L(M_1)$ and $L_2=L(M_2)$. First, we can easily check that for arbitrary $a\in\Sigma$, $z\in\Gamma$, $z'\in\Gamma'$, α, $\alpha_1\in\Gamma^*$, β, $\beta_1\in\Gamma'^*$,

$$(\alpha z, a, \beta z') \overset{*}{\underset{M}{\vdash}} (\alpha_1, \lambda, \beta_1) \quad \text{iff} \quad \begin{cases} \text{there is } u\in\Sigma_1^* \text{ such that} \\ (ua, \alpha z) \overset{*}{\underset{M_1}{\vdash}} (\lambda, \alpha_1) \quad \text{and} \\ (ua, \beta z') \overset{*}{\underset{M_2}{\vdash}} (\lambda, \beta_1). \end{cases} \tag{6.5.1}$$

Then, we prove by induction on the length of $w=a_1\dots a_n\in\Sigma^*$ that

$$(z_0, a_1\dots a_n, z_0') \overset{*}{\underset{M}{\vdash}} (\alpha, \lambda, \beta) \quad \text{iff} \quad \begin{cases} \text{there are } u_1, \dots, u_n\in\Sigma_1^* \text{ such that} \\ (u_1a_1\dots u_na_n, z_0) \overset{*}{\underset{M_1}{\vdash}} (\lambda, \alpha) \quad \text{and} \\ (u_1a_1\dots u_na_n, z_0') \overset{*}{\underset{M_2}{\vdash}} (\lambda, \beta). \end{cases} \tag{6.5.2}$$

Indeed, the case where $w=a\in\Sigma$ is trivial.

Assume that statement (6.5.2) is valid for all $w\in\Sigma^*$ with $\lg(w)<n$. We now consider the word $w=a_1\dots a_{n-1}a_n$, and let $w_1=a_1\dots a_{n-1}$. Since $\lg(w_1)<n$, statement (6.5.2) is true and we have

$$(z_0, w_1, z_0') \underset{M}{\vdash} (\alpha_1 z, \lambda, \beta_1 z') \quad \text{iff} \quad \begin{cases} \text{there are } u_1, \dots, u_{n-1}\in\Sigma_1 \text{ such that} \\ (u_1a_1\dots u_{n-1}a_{n-1}, z_0) \overset{*}{\underset{M_1}{\vdash}} (\lambda, \alpha_1 z) \\ (u_1a_1\dots u_{n-1}a_{n-1}, z_0') \overset{*}{\underset{M_2}{\vdash}} (\lambda, \beta_1 z'). \end{cases}$$

On the other hand, by statement (6.5.1), we can easily see that

$$(\alpha_1 z, a_n, \beta_1 z') \overset{*}{\underset{M}{\vdash}} (\alpha, \lambda, \beta) \quad \text{iff} \quad \begin{cases} \text{there is } u_n\in\Sigma_1^* \text{ such that} \\ (u_na_n, \alpha_1 z) \overset{*}{\underset{M_1}{\vdash}} (\lambda, \alpha), \quad \text{and} \\ u_na_n, \beta_1 z') \overset{*}{\underset{M_2}{\vdash}} (\lambda, \beta). \end{cases}$$

Thus, statement (6.5.2) holds. Finally, for $w=a_1 \dots a_n \in \Sigma^*$

$$w = a_1 \dots a_n \in L(M) \quad \text{iff} \quad (z_0, a_1 \dots, a_n, z_0') \overset{*}{\underset{M}{\vdash}} (\lambda, \lambda, \lambda)$$

$$\text{iff} \quad \begin{cases} \text{there are } u_1, \dots, u_n \in \Sigma_1 \text{ such that} \\ (u_1 a_1 \dots u_n a_n, z_0) \overset{*}{\underset{M_1}{\vdash}} (\lambda, \lambda) \\ (u_1 a_1 \dots u_n a_n, z_0') \overset{*}{\underset{M_2}{\vdash}} (\lambda, \lambda) \end{cases}$$

$$\text{iff} \quad \begin{cases} \text{there are } u_1, \dots, u_n \in \Sigma_1^* \text{ such that} \\ u_1 a_1 \dots u_n a_n \in L_1 \cap L_2 \text{ and} \\ h(u_1 a_1 \dots u_n a_n) = a_1 \dots a_n \in h(L_1 \cap L_2). \end{cases} \quad \square$$

Corollary 6.6. Every sd0-language can be expressed in the form $L=h(L_1 \cap L_2)$, where h is a homomorphism, and L_1, L_2 are two sdc-languages.

Acknowledgement. The author would like to thank J. Demetrovics, I. Peák and Gy. Révész for th eir careful readings of several versions of the manuscript and their many helpful suggestions for its revision. This paper is very much improved by their contributions.

COMPUTER AND AUTOMATION INSTITUTE
HUNGARIAN ACEDEMY OF SCIENCE
KENDE U. 13-17.
BUDAPEST, HUNGARY
H-1502

References

[1] HOPCROFT, J., J. ULLMAN, Formal languages and their relation of automata, Addison-Wesley, 1969.
[2] GINSBURG, S., A. GREIBACH, Deterministic context-free languages, *Inform. and Control*, v. 9, 1966, pp. 620—648.
[3] FRIEDMAN, E. P., Simple context-free languages and free monadic recursion schemes, *Math. Systems Theory*, v. 11, 1971, pp. 9—28.
[4] RÉVÉSZ, GY., Bevezetés a formális nyelvek elméletébe, Akadémiai Kiadó, Budapest, 1979.
[5] KURODA, S. Y., Classes of languages and linear-bounded automata, *Inform. and Control*, v. 7, 1964, pp. 207—223.
[6] DAVIS, M., Computability and unsolvability, McGraw-Hill, New York, 1958.
[7] SALOMAA, A., Formal languages, Academic Press, 1973.

(Received Oct. 31, 1980)