数理解析研究所講究録
第 494 巻 1983 年 46-57

REMARKS ON REAL-TIME DETERMINISTIC CONTEXT-FREE LANGUAGES

Yoshihide Igarashi
Department of Computer Science
Gunma University, Kiryu, 376 Japan

1. Introduction

The context-free languages are most important language family for the study of compiler design techniques and language specifications. In particular, characterizations of deterministic context-free languages by automata are important for parsing algorithms [3][7]. Several subclasses of deterministic context-free languages have been studied in a way that we ask whether placing restrictions on the deterministic pushdown automata affects the family of languages accepted [4][5][6][10]. The real-time deterministic context-free languages are one of such subclasses.

In this paper we establish a pumping lemma for the real-time deterministic context-free languages. The lemma is an interesting character of the subclass and useful to show that a given deterministic context-free language is not real-time.

In the main we employ the definitions and notation given in standard texts such as [3] or [8]. If w is a word (i.e., a string of symbols), $|w|$ denotes its length. ε denotes the word of zero length. If x is a pair of words, $|x|$ denotes the length of its second component (i.e., if $x = (q, \alpha)$, $|x| = |\alpha|$). If S is a set, $\#(S)$ denotes the number of elements in S. A deterministic pushdown automaton (abbreviated DPDA) is a deterministic acceptor with a one-way input tape, a pushdown tape, and a finite state control. It can be specified by a 7-tuple $(Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$, where

(1) Q is a finite set of states,

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

(3) Γ is a finite set of pushdown symbols (the pushdown alphabet),

(4) q_0 is in Q (the initial state),

(5) Z_0 is in Γ (the start symbol),

(6) $F \subseteq Q$ (the set of final states), and

(7) δ is a mapping from $Q \times (\Sigma \cup \{\varepsilon\}) \times \Gamma$ to the finite subsets of $Q \times \Gamma^*$ which has the following restrictions: For each q in Q and Z in Γ (a) either $\delta(q, a, Z)$ contains exactly one element for all a in Σ and $\delta(q, \varepsilon, Z) = \emptyset$, or $\delta(q, \varepsilon, Z)$ contains exactly one element and $\delta(q, a, Z) = \emptyset$ for each a in Σ, and (b) if $\delta(q, \pi, Z_0) \neq \emptyset$ for π in $\Sigma \cup \{\varepsilon\}$, then $\delta(q, \pi, Z_0) = \{(p, Z_0\gamma)\}$ for some p in Q and γ in Γ^*.

Certain strings over Γ are interpreted as the contents of the pushdown store. We assume that the bottom of the store is on the left and top on the right. A configuration is a pair from $Q \times \Gamma^*$. The initial configuration (q_0, Z_0) is denoted by c_s. A DPDA makes a move $(q, \alpha A) \overset{\pi}{\vdash} (p, \alpha\gamma)$ if and only if there is some transition $\delta(q, \pi, A) = (p, \gamma)$. In particular, if $\pi = \varepsilon$, it is called an ε-move. If π is in Σ, then this symbol is considered to have been read. A computation is a sequence of such moves through successive configurations. Suppose w is a string over Σ. If we obtain configuration c' from configuration c by the successive read of w, the computation is denoted by $c \overset{w}{\vdash} c'$. A word w is accepted by DPDA $M = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$ if for some configuration c with the first component of c belonging to F, $(q_0, Z_0) \overset{w}{\vdash} c$. The language accepted by M is denoted by $L(M)$. That is, $L(M) = \{w \text{ in } \Sigma^* \mid c_s = (q_0, Z_0) \overset{w}{\vdash} c$, the first component of c belongs to F}. The language accepted by a DPDA is called a deterministic context-free language (abbreviated DCFL).

Let $c \overset{w}{\vdash} c'$ be a computation. c_1 is a stacking configuration in the computation if and only if it is not followed by any configuration of height $\leqq |c_1|$ in the computation. Note that, whether or not c_1 is a stacking configuration depends on what computation is considered. That is, if we say that c_1 is a stacking configuration in the computation $c \overset{w}{\vdash} c'$, it means that c_1 is a stacking configuration for the whole of $c \overset{w}{\vdash} c'$.

DPDA $M = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$ is said to be quesi-real-time if and only if there exists an integer $t \geqq 0$ such that for any q, q' in Q and γ, γ' in Γ^* $(q, \gamma) \overset{\varepsilon}{\vdash} \cdots \overset{\varepsilon}{\vdash} (q', \gamma')$ implies that the number of steps of this computation is not greater than t. In particular, M is said to be real-time if and only if $t = 0$ (i.e., if and only if $\delta(q, \varepsilon, Z) = \emptyset$ for all q in Q and Z in Γ). A language L is called (quesi-) real-time if and only if $L = L(M)$ for some (quesi-) real-time DPDA M. Our (quesi-) real-time DCFL's correspond to Δ_0-(quesi-) real-time languages defined in [4] and [6]. It is known that the class of quesi-real-time DCFL's coincides with the class of real-time DCFL's [4][6].

2. Pumping Lemmas for Real-Time DCFL's

The pumping lemma and Ogden's lemma are useful and fundamental properties of CFL's [1][3][9][11]. Wise has established a necessary and sufficient version of the classic pumping lemma for CFL's [13], and Jaffe has established a necessary and sufficient pumping lemma for regular languages [9]. Stanat has recently shown another characterization of regular languages using a modified pumping lemma [12]. It is also interesting to ask whether we can derive a useful pumping lemma for each of well-known subclasses of

DCFL's, or to ask whether we can establish a necessary and sufficient pumping lemma for such a subclass.

In this section we first show a simple pumping lemma for real-time DCFL's. Then we show a version of the pumping lemma which will be useful to show that a language is not a real-time DCFL.

Definition 1. Let L be a language (i.e., a subset of Σ^*). x in Σ^* is equivalent under L to y in Σ^* (denoted by $x \equiv_L y$) if and only if for any w in Σ^* both xw and yw are in L or both xw and yw are not in L.

The relation $\equiv_L$ is an equivalence relation on Σ^*. $x \not\equiv_L y$ means that x and y are not equivalent under L.

Lemma 1 (Simple pumping lemma for real-time DCFL's). Let L be a real-time DCFL. Then there are a pair of constants $k_1 > 0$ and k_2, depending only on L, that satisfy the following property (*):

(*) If $x_1, x_2, \ldots, x_n$ are n strings on Σ such that

(*-1) for any $1 \leqq i < j \leqq n$ $x_i \not\equiv_L x_j$, and

(*-2) for each i ($1 \leqq i \leqq n$) there is y_i in Σ^* satisfying

(*-2-1) $x_i y_i$ is in L, and

(*-2-2) $|y_i| \leqq (\log_2 n)/k_1 + k_2$,

then for at least one r ($1 \leqq r \leqq n$) we may write $x_r = x_{r_1} x_{r_2} x_{r_3}$ such that

(*-3) $|x_{r_2}| \geqq 1$, and

(*-4) for all $t \geqq 0$ $x_{r_1} (x_{r_2})^t x_{r_1} y_r$ is in L.

Proof. Let L be recognized by a real-time DPDA $M = (Q, \Sigma, \Gamma, \delta, q_0, Z_0$ F). Without loss of generality we may assume that $\#(\Gamma)$ is not less than 2. For w in Σ^* let $CONF_M(w)$ be the configuration of M when input string w has been read (i.e., $c_s = (q_0, Z_0) \overset{w}{\vdash} CONF_M(w)$). Let $k_1 = \log_2 \#(\Gamma)$ and $k_2 = (\log_2(\#(\Gamma) - 1) - \log_2 \#(Q))/\log_2 \#(\Gamma) - \#(Q)\#(\Gamma) - 1$. Let $x_1, \ldots, x_n$ be n strings over Σ that satisfy (*-1) and (*-2) above, and let $h = \max\{|CONF_M(x_i)| \mid 1 \leqq i \leqq n\}$. From (*-1) all of $CONF_M(x_1)$, $CONF_M(x_2)$, . . . ,$CONF_M(x_n)$ are distinct. Therefore, $\#(Q)(1 + \#(\Gamma) + . . . + (\#(\Gamma))^{h-1}) \geqq n$. Note that the leftmost symbol of the pushdown store is always Z_0. Solving this inequality we have

$$\begin{aligned} h &> (\log_2 n + \log_2(\#(\Gamma) - 1) - \log_2\#(Q))/\log_2\#(\Gamma) \\ &= (\log_2 n)/k_1 + k_2 + \#(Q)\#(\Gamma) + 1. \end{aligned}$$

Let r be an index such that $h = |CONF_M(x_r)|$. From this inequality and (*-2-2) $|CONF_M(x_r)| > \#(Q)\#(\Gamma) + 1 + |y_r|$. Therefore, for the whole computation of the input string $x_r y_r$ there are at least $\#(Q)\#(\Gamma) + 1$ stacking configurations among the configurations from c_s to $CONF_M(x_r)$. Hence,there are at least two configurations in this part such that their pairs of the states and top pushdown tape symbols are identical. Let these configurations be $CONF_M(x_{r_1})$ and $CONF_M(x_{r_1}x_{r_2})$. Since $x_r y_r$ is in L, for all $t \geqq 0$ $x_{r_1}(x_{r_2})^t x_{r_3} y_r$ is in L, where $x_r = x_{r_1}x_{r_2}x_{r_3}$ and $|x_{r_2}| \geqq 1$. Q. E. D.

The notation $CONF_M$ introduced in the above proof will be used in the following. The above lemma is not strong enough to use it as a tool for proving that a given DCFL is not real-time. For example, $L = \{a^i b^j c^k a^i \mid i \geqq 0, j \geqq k \geqq 0\}$ is not a real-time DCFL. However, we cannot lead any contradiction by using Lemma 1 from the assumption that L is a real-time DCFL. We, therefore, are requested to prepare a powerful version of Lemma 1 for this purpose. This situation is analogous to the fact that Ogden's lemma is more powerful than the classic pumping lemma for CFL's. The next lemma is such a version for real-time DCFL's.

Lemma 2 (Strong pumping lemma for real-time DCFL's). Let L be a real-time DCFL. Then there are constants k_1, $k_2 > 0$ and k_3, depending only on L, that satisfy the following property (*):

(*) Let n be an integer such that $n > k_1$, and let m be an integer. If there are n strings $x_1, \ldots, x_n$ on Σ such that for each pair of i and j ($1 \leqq i \leqq n$, $1 \leqq j \leqq m$) there is a string y_{ij} satisfying

(*-1) for each i ($1 \leqq i \leqq n$) and for any pair of j_1 and j_2 ($1 \leqq j_1 < j_2 \leqq m$) $x_i y_{ij_1} \not\equiv_L x_i y_{ij_2}$,

(*-2) for any pair of i_1 and i_2 ($1 \leqq i_1 < i_2 \leqq n$) and for any pair of j_1 and j_2 ($1 \leqq j_1 \leqq m$, $1 \leqq j_2 \leqq m$) the concatenation of x_{i_1} and any initial substring of $y_{i_1j_1}$ and the concatenation of x_{i_2} and any initial substring of $y_{i_2j_2}$ are not equivalent under L (i.e., if $\overline{y_{i_1j_1}}$ is an initial substring of $y_{i_1j_1}$, and if $\overline{y_{i_2j_2}}$ is an initial substring of $y_{i_2j_2}$, then $x_{i_1}\overline{y_{i_1j_1}} \not\equiv_L x_{i_2}\overline{y_{i_2j_2}}$), and

(*-3) for each pair of i ($1 \leqq i \leqq n$) and j ($1 \leqq j \leqq m$) there exists a string w_{ij} such that $x_i y_{ij} w_{ij}$ is in L and $|w_{ij}| \leqq (\log_2 m)/k_2 + k_3$,

then there exists at least one pair of p and q ($1 \leqq p \leqq n$, $1 \leqq q \leqq m$) such that

(*-4) we may write $x_p = x_{p_1} x_{p_2} x_{p_3}$, where $|x_{p_2}| \geqq 1$, and

(*-5) for all $t \geqq 0$ $x_{p_1}(x_{p_2})^t x_{p_3} y_{pq} w_{pq}$ is in L.

<u>Proof.</u> Let L be accepted by a real-time DPDA $M = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$. Without loss of generality we may assume that $\#(\Gamma)$ is not less than 2. The proof will proceed as the proof of the previous lemma. Let $k_1 = \#(Q)(1 + \#(\Gamma) + \ldots + (\#(\Gamma))^{\#(Q)\#(\Gamma)})$ and $k_2 = \log_2 \#(\Gamma)$, and let $k_3 = (\log_2 (\#(\Gamma) - 1) - \log_2 \#(Q))/\log_2 \#(\Gamma) - \#(Q)\#(\Gamma) - 1$. If $m \leqq k_1$, then $(\log_2 m)/k_2 + k_3 < 0$. In this case, for any pair of i ($1 \leqq i \leqq n$) and j ($1 \leqq j \leqq m$) there does not exist w_{ij} satisfying (*-3). Therefore, in this case the assertion of the lemma holds. We suppose that $m > k_1$ and that there

exist x_i $(1 \leq i \leq n)$, y_{ij} $(1 \leq i \leq n,\ 1 \leq j \leq m)$ and w_{ij} $(1 \leq i \leq n,\ 1 \leq j \leq m)$ satisfying (*-1), (*-2) and (*-3), where $n > k_1$.

Consider the following classes of strings in Σ^*.

$A(1) = \{x_1y_{11}, x_1y_{12}, \ldots, x_1y_{1m}\}$

$A(2) = \{x_2y_{21}, x_2y_{22}, \ldots, x_2y_{2m}\}$

$\vdots$

$A(n) = \{x_ny_{n1}, x_ny_{n2}, \ldots, x_ny_{nm}\}$.

From (*-1) for each i $(i \leq i \leq n)$ all of $\mathrm{CONF}_M(x_iy_{i1}), \ldots, \mathrm{CONF}_M(x_iy_{im})$ should be distinct. Therefore, for each i $(1 \leq i \leq n)$ there exists at least one element in $A(i)$, say $x_iy_{ij_i}$, such that $|\mathrm{CONF}_M(x_iy_{ij_i})| \geq g$, where g is the least integer satisfying $\#(Q)(1 + \#(\Gamma) + \ldots + (\#(\Gamma))^{g-1}) \geq m$. Let these strings be $x_1y_{1j_1}, \ldots, x_ny_{nj_n}$. For each i $(1 \leq i \leq n)$ let $\tilde{y}_{ij_i}$ be an initial substring of y_{ij_i} such that $|\mathrm{CONF}_M(x_i\tilde{y}_{ij_i})| = \min\{|\mathrm{CONF}_M(x_i\overline{y}_{ij_i})| \mid \overline{y}_{ij_i}$ is an initial substring of $y_{ij_i}\}$. From (*-2) all of $\mathrm{CONF}_M(x_1\tilde{y}_{1j_1}), \ldots, \mathrm{CONF}_M(x_n\tilde{y}_{nj_n})$ should be distinct. From this fact and $n > k_1$ there exists at least one element, say $x_p\tilde{y}_{pj_p}$, among $x_1\tilde{y}_{1j_1}, \ldots, x_n\tilde{y}_{nj_n}$ such that $|\mathrm{CONF}_M(x_p\tilde{y}_{pj_p})| \geq \#(Q)\#(\Gamma) + 2$. That is, for any initial substring $\overline{y}_{pj_p}$ of y_{pj_p} $|\mathrm{CONF}_M(x_p\overline{y}_{pj_p})| \geq \#(Q)\#(\Gamma) + 2$. Hence, for the computation from c_s to $\mathrm{CONF}_M(x_py_{pj_p})$ there are at least $\#(Q)\#(\Gamma) + 1$ stacking configurations in the first $|x_p|$ steps. Since $|\mathrm{CONF}_M(x_py_{pj_p})| \geq g$ and $|w_{pj_p}| \leq (\log_2 m)/k_2 + k_3$, the height of the pushdown tape during the last $|w_{pj_p}|$ steps of $c_s = (q_0, Z_0) \vdash \ldots \vdash \mathrm{CONF}_M(x_py_{pj_p}w_{pj_p})$ is at least $\#(Q)\#(\Gamma) + 2$. Hence, for the computation $c_s \vdash \ldots \vdash \mathrm{CONF}_M(x_py_{pj_p}w_{pj_p})$ the first $\#(Q)\#(\Gamma) + 1$ stacking configurations locate in the first $|x_p|$ steps of the computation. Thus there are at least two stacking configurations in the first $|x_p|$ steps of the computation $c_s \vdash \ldots \vdash \mathrm{CONF}_M(x_py_{pj_p}w_{pj_p})$ such that their pairs of states and top pushdown tape symbols are identical. Let these configurations be

$CONF_M(x_{p1})$ and $CONF_M(x_{p1}x_{p2})$, where $|x_{p2}| \geqq 1$. Removing or repeating the part of the computation corresponding to x_{p2} does not affect the last state of the whole computation. Since $x_p y_{pj_p} w_{pj_p}$ is in L, for all $t \geqq 0$ $x_{p1}(x_{p2})^t x_{p3} y_{pq} w_{pq}$ is in L, where $q = j_p$ and $x_p = x_{p1}x_{p2}x_{p3}$. Q. E. D.

For a certain string in a real-time DCFL Lemma 2 specifies a range of the pumping position of the string, whereas Lemma 1 does not. This specification of the pumping position is indispensable to use the lemma as a tool to show that a given language is not a real-time DCFL.

3. Applications

Strong pumping lemma (Lemma 2) guarantees a scheme for proving that a given language is not a real-time DCFL. We show this proving scheme by examples.

Example 1. $L_1 = \{a^i b^j a^i,\ a^j b^i c^i \mid i, j \geqq 1\}$

Harrison and Havel proved that L_1 is not a Δ_2-real-time language (Theorem 2.4 of [4]). The class of Δ_2-real-time languages is properly included in the class of Δ_0-real-time languages [4] (i.e., real-time DCFL's of this paper). By using Lemma 2 we can easily show that L_1 is not a real-time DCFL.

Assume for the sake of contradiction that L_1 is a real-time DCFL. Let k_1, k_2 and k_3 be constants described in Lemma 2. Let $n > k_1$ and let m be an integer such that $n \leqq (\log_2 m)/k_2 + k_3$. We choose $x_i = a^i$, $y_{ij} = b^j$ and $w_{ij} = a^i$ for each i ($1 \leqq i \leqq n$) and each j ($1 \leqq j \leqq m$). Then (*-1), (*-2) and (*-3) are satisfied. Then from (*-4) and (*-5) for some pair of i and j we may write $a^i = a^{i_1}a^{i_2}a^{i_3}$, where $i_2 \geqq 1$ and for all $t \geqq 0$ $a^{i_1}(a^{i_2})^t a^{i_3} b^j a^i$ is in L_1. This is a contradiction. We, therefore, conclude that L_1 is not a real-time DCFL.

Lemma 2 is powerful enough for our purpose. In fact, we do not know at present any DCFL that is not real-time but that cannot be proved by Lemma 2 not to be real-time. However, it may be valuable to prepare a version of Lemma 2 that seems to be easier for the reader to use it. In the rest of this section we describe such a version although it is essentially the same as Lemma 2.

Definition 1. Let f(n) be a function from nonnegative integers to nonnegative integers. A language L is f(n)-characteristic if and only if the following property (*) is satisfied:

(*) For arbitrary positive integers n and m there exist n strings $x_1, \ldots, x_n$ and $n \times m$ strings y_{ij} ($1 \leqq i \leqq n$, $1 \leqq j \leqq m$) such that

(*-1) for each i ($1 \leqq i \leqq n$) and for any pair of j_1 and j_2 ($1 \leqq j_1 < j_2 \leqq m$) $x_i y_{j_1} \not\equiv_L x_i y_{ij_2}$,

(*-2) for any pair of i_1 and i_2 ($1 \leqq i_1 < i_2 \leqq n$), any j_1 and j_2 ($1 \leqq j_1 \leqq m$, $1 \leqq j_2 \leqq m$), the concatenation of x_{i_1} and any initial substring of $y_{i_1 j_1}$ and the concatenation of x_{i_2} and any initial substring of $y_{i_2 j_2}$ are not equivalent under L, and

(*-3) for any pair of i and j there exists a string w_{ij} such that

(*-3-1) $|w_{ij}| \leqq f(n)$,

(*-3-2) $x_i y_{ij} w_{ij}$ is in L, and

(*-3-3) for any non-null substring x_i'' of x_i, there exists a non-negative integer t such that $x_i'(x_i'')^t \bar{x}_i y_{ij} w_{ij}$ is not in L, where $x_i = x_i' x_i'' \bar{x}_i$.

Lemma 3. If there is a function f(n) such that L is f(n)-characteristic, then L is not a real-time DCFL.

Proof. Let L be f(n)-characteristic. Assume for the sake of contradiction that L is accepted by a real-time DPDA $M = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$.

Let n and m be integers such that $n > k_1$ and $f(n) \leqq (\log_2 m)/k_2 + k_3$, where k_1, k_2 and k_3 are constants given in the proof of Lemma 2. Let x_i $(1 \leqq i \leqq n)$, y_{ij} $(1 \leqq i \leqq n,\ 1 \leqq j \leqq m)$ and w_{ij} $(1 \leqq i \leqq n,\ 1 \leqq j \leqq m)$ be strings satisfying conditions (*-1), (*-2) and (*-3) of Definition 1. These strings satisfy conditions (*-1), (*-2) and (*-3) of Lemma 2. Therefore, (*-4) and (*-5) of Lemma 2 should hold since L is assumed to be a real-time DCFL. However, (*-4) and (*-5) of Lemma 2 are contrary to (*-3-3) of Definition 1. We, therefore, conclude that our assumption is wrong. That is, L is not a real-time DCFL. Q. E. D.

<u>Example 2.</u> $L_2 = \{a^i b^j a^i,\ a^i b^j c b^j a^i \mid i, j \geqq 1\}$. This language has been given by Gisburg and Greibach(2) as an example of a DCFL that is not real-time. By using Lemma 3 we prove that L_2 is not a real-time DCFL. Let $f(n) = n$. For $n > 1$ and $m \geqq 1$ we choose $x_i = a^i$ $(1 \leqq i \leqq n)$, $y_{ij} = b^j$ and $w_{ij} = a^i$ $(1 \leqq i \leqq n,\ 1 \leqq j \leqq m)$. Then (*-1), (*-2) and (*-3) in Definition 1 hold. That is, L_2 is n-characteristic. From Lemma 3 L_3 is not a real-time DCFL.

<u>Example 3.</u> $L_3 = \{a^i b^j c^r a^i \mid i \geqq 1,\ j \geqq r \geqq 1\}$. Let $f(n) = n + 1$. For $n \geqq 1$ and $m \geqq 1$ we choose $x_i = a^i$ $(1 \leqq i \leqq n)$, $y_{ij} = b^j$ $(1 \leqq i \leqq n,\ 1 \leqq j \leqq m)$ and $w_{ij} = ca^i$ $(1 \leqq i \leqq n,\ 1 \leqq j \leqq m)$. Then (*-1), (*-2) and (*-3) in Definition 1 hold. Therefore, L_3 is (n + 1)-characteristic, and from Lemma 3 it is not a real-time DCFL.

<u>Example 4.</u> $L_4 = \{a^i b^j c^p d^q \mid i, j, p, q \geqq 1,\ i \neq q \text{ and } j \neq p\}$. Let $f(n) = n! + n + 1$. For $n \geqq 1$ and $m \geqq 1$ we choose $x_i = a^i$ $(1 \leqq i \leqq n)$, $y_{ij} = b^{j+1}$ $(1 \leqq i \leqq n,\ 1 \leqq j \leqq m)$ and $w_{ij} = cd^{i!+i}$ $(1 \leqq i \leqq n,\ 1 \leqq j \leqq m)$. Then it is obvious that (*-1), (*-2), (*-3-1) and (*-3-2) in Definition 1 hold. For any non-null substring a^r of a^i $r = |a^r|$ is a divisor of $i!$. Thus we can write $a^{i-r}(a^r)^{(i!/r)+1} = a^{i!+i}$. Therefore, for any r $(1 \leqq r \leqq i)$ and

$t = i!/r$, $a^{i-r}(a^r)^{t+1}b^{j+1}cd^{i!+i} = a^{i!+i}b^{j+1}cd^{i!+i}$ is not in L_4. Thus (*-3-3) in Definition 1 hold, too. Therefore, L_4 is $(n!+n+1)$-characteristic, and from Lemma 3 it is not a real-time DCFL.

Note that $L_5 = \{a^i b^j c^r a^i \mid 1 \leqq j \leqq r,\ i \geqq 1\}$ is a real-time DCFL. Therefore, for any function $f(n)$ L_5 is not $f(n)$-characteristic. For example, suppose that for $n \geqq 1$ and $m \geqq 1$ we choose $x_i = a^i$ $(1 \leqq i \leqq n)$, and $y_{ij} = b^j$ $(1 \leqq i \leqq n,\ 1 \leqq j \leqq m)$. In this case, when m is sufficiently large compared with $f(n)$, say $m = 2\ f(n)$, we cannot choose any w_{ij} $(1 \leqq i \leqq n,\ 1 \leqq j \leqq m)$ that satisfies (*-3-1) and (*-3-2) in Definition 1 simultaneously. Therefore, these choices of x_i $(1 \leqq i \leqq n)$ and y_{ij} $(1 \leqq j \leqq m)$ are not successful to show that L_5 would be $f(n)$-characteristic.

We do not know at present whether Lemma 2 is a sufficient condition for real-time DCFL's. We invite the reader to consider the following problems worthy of further investigation:

(1) Is Lemma 2 a necessary and sufficient condition for real-time DCFL's ?

(2) Find an elegant characterization of real-time DCFL's that is a necessary and sufficient condition for real-time DCFL's.

(3) Find an elegant characterization of each subclass of DCFL's.

References

[1] C. Bader and A. Moura, A generalization of Ogden's lemma, J. ACM 29 (1982) 404-407.

[2] S. Ginsburg and S. Greibach, Deterministic context-free languages, Information and Control 9 (1966) 620-648.

[3] M. A. Harrison, Introduction to Formal Language Theory (Addison-Wesley, Reading, MA, 1978).

[4] M. A. Harrison and I. M. Havel, Real-time strict deterministic languages, SIAM J. Comput. 1 (1972) 333-349.

[5] M. A. Harrison and I. M. Havel, Strict deterministic grammars, J. Comput System Sci. 7 (1973) 237-277.

[6] M. A. Harrison and I. M. Havel, On a family of deterministic grammars, in: M. Nivat, ED., Automata, Languages and Programming (North-Holland, Amsterdam, 1973) 413-442.

[7] M. A. Harrison and I. M. Havel, On the parsing of deterministic languages, J. ACM 21 (1974) 525-548.

[8] J. E. Hopcroft and J. D. Ullma, Formal Languages and Their Relation to Automata (Addison-Wesley, Reading, MA. 1969).

[9] J. Jaffe, A necessary and sufficient pumping lemma for regular language SIGACT NEWS 10(2) (1978) 48-49.

[10] A. J. Korenjak and J. E. Hopcroft, Simple deterministic languages, Proc 7th IEEE Symposium on Switching and Automata Theory (1966) 36-46.

[11] W. Ogden, A helpful result for proving inherent ambiguity, Mathematical Systems Theory 2 (1968) 191-194.

[12] D. F. Stanat, A pumping lemma for regular languages, SIGACT NEWS 14(1) (1982) 36-37.

[13] O. S. Wise, A strong pumping lemma for context-free languages, Theoretical Computer Sci. 3 (1976) 359-369.