数理解析研究所講究録
1148 巻 2000 年 29-34



# Semi-Right-Terminating 一意解析可能文法による決定性文脈自由言語族の特徴付け

# Characterizing the Class of Deterministic Context-Free Languages by Semi-Right-Terminating Uniquely Parsable Grammars

広島大学工学部 森田憲一 (Kenichi Morita) *
広島大学工学部 李 佳 (Jia Lee) †

## Abstract

A uniquely parsable grammar (UPG) introduced by Morita et. al (1997) is a kind of phrase structure grammar, in which parsing can be performed without backtracking. It is known that UPGs and their three subclasses form a "deterministic Chomsky hierarchy". Especially, the class of RC-UPGs (UPGs with right-terminating and context-free-like rules) exactly characterizes the class of deterministic context-free languages (DCFLs). In this paper, we newly introduce a semi-right-terminating grammar (SR-G) and a semi-right-terminating UPG (SR-UPG). We show that the classes of SR-Gs and SR-UPGs exactly characterize the classes of context-free languages and deterministic context-free languages, respectively. Although an SR-UPG is a variant of an RC-UPG, it is simpler than the latter as a framework for characterizing the class of DCFLs.

## 1. Introduction

A uniquely parsable grammar (UPG) [4] is a kind of phrase structure grammar whose rewriting rules satisfy the following condition: If a suffix of the righthand side of a rule matches with a prefix of that of some other rule, then these overlapping portions remain unchanged by the reverse application of the rules. By this condition, UPGs can be parsed without backtracking. Furthermore, it is known that the class of UPGs and its three subclasses form a "deterministic Chomsky hierarchy". That is, the classes of (unrestricted) UPGs, M-UPGs (monotonic UPGs), RC-UPGs (UPGs with right-terminating and context-free-like rules), and REG-UPGs (regular UPGs) exactly characterize the classes of deterministic Turing machines, deterministic linear-bounded automata, deterministic pushdown automata, and deterministic finite automata, respectively.

In this paper, we introduce a semi-right-terminating grammar (SR-G) and a semi-right-terminating UPG (SR-UPG). They are grammars having semi-right-terminating rules (SR-rules), each of which is of the form $\alpha \to \beta t$, $\alpha t \to \beta t$, $\alpha\$ \to \beta\$$, $\$A \to \$t$, or $\$A\$ \to \$\$$, where $\alpha$ and $\beta$ are non-empty nonterminal strings, $A$ is a nonterminal, $t$ is a terminal, and \$ is an end-marker. We prove that the classes of SR-Gs and SR-UPGs exactly characterize the classes of context-free languages (CFLs) and deterministic context-free languages (DCFLs), respectively. Although an SR-UPG has some similarity with an RC-UPG, it is simpler than the latter in its definition. Hence, this gives another characterization of the class of DCFLs.

## 2. Definitions

### 2.1. Uniquely Parsable Grammars

We first give definitions of a grammar with an end-marker, a uniquely parsable grammar, and some other related notions that are needed in the following sections (see e.g. [2, 5, 6] for basic notions on formal languages, and [4] for a uniquely parsable grammar).

**Definition 2.1** *A grammar (with an end-marker) is a system*

$$G = (N, T, P, S, \$),$$

*where $N$ and $T$ are sets of nonterminals and terminals respectively ($N \cap T = \emptyset$), $S$ is a start*

*morita@ke.sys.hiroshima-u.ac.jp
†lijia@ke.sys.hiroshima-u.ac.jp

*symbol* ($S \in N$)*, and* \$ *is an end-marker* ($\$ \notin (N \cup T)$ )*. P is a set of rewriting rules of the following form:*

$$\alpha \to \beta,\ \$\alpha \to \$\beta,\ \alpha\$ \to \beta\$,$$
$$\$\alpha\$ \to \$\beta\$,\ \text{or } \$A\$ \to \$\$,$$

*where* $\alpha, \beta \in (N \cup T)^+$, $\alpha \neq \beta$, $A \in N$, *and* $\alpha$ *contains at least one nonterminal.*

**Definition 2.2** *Let* $G = (N, T, P, S, \$)$ *be a grammar, and* $\eta$ *be a string in* $(N \cup T \cup \{\$\})^+$. *A rule* $\alpha \to \beta$ ($\alpha, \beta \in (N \cup T \cup \{\$\})^+$) *in P is said to be* applicable *to* $\eta$ *if* $\eta = \gamma\alpha\delta$ *for some* $\gamma, \delta \in (N \cup T \cup \{\$\})^*$. *Applying* $\alpha \to \beta$ *to* $\eta$ *we obtain* $\zeta = \gamma\beta\delta$*, and say* $\zeta$ *is* directly derived *from* $\eta$ *in G. This is written as* $\eta \Rightarrow \zeta$. *Let* $\stackrel{*}{\Rightarrow}$ *denote the reflexive and transitive closure of* $\Rightarrow$. *An n-step derivation is denoted by* $\eta \stackrel{n}{\Rightarrow} \zeta$. *The* language $L(G)$ *generated by G is defined by* $L(G) = \{w \in T^* \mid \$S\$ \stackrel{*}{\Rightarrow} \$w\$\}$.

**Definition 2.3** *Let* $G = (N, T, P, S, \$)$ *be a grammar, and* $\eta$ *be a string in* $(N \cup T \cup \{\$\})^+$. *A rule* $\alpha \to \beta$ *in P is said to be* reversely applicable *to* $\eta$ *if* $\eta = \gamma\beta\delta$ *for some* $\gamma, \delta \in (N \cup T \cup \{\$\})^*$. *Reversely applying* $\alpha \to \beta$ *to* $\eta$ *we obtain* $\zeta = \gamma\alpha\delta$*, and say* $\eta$ *is* directly reduced *to* $\zeta$. *We write it as* $\eta \Leftarrow \zeta$. *Apparently,* $\eta \Leftarrow \zeta$ *iff* $\zeta \Rightarrow \eta$. *The relations* $\stackrel{*}{\Leftarrow}$ *and* $\stackrel{n}{\Leftarrow}$ *are also defined similarly to* $\stackrel{*}{\Rightarrow}$ *and* $\stackrel{n}{\Rightarrow}$.

**Definition 2.4** *Let* $G = (N, T, P, S, \$)$ *be a grammar. If P satisfies the following condition (the "UPG-condition"), then G is called a* uniquely parsable grammar *(UPG).*

1. *The righthand side of each rule is neither S,* $\$S$, $S\$$, *nor* $\$S\$$.
2. *For any two rules* $r_1 = \alpha_1 \to \beta_1$ *and* $r_2 = \alpha_2 \to \beta_2$ *in P* ($r_1$ *and* $r_2$ *may not be distinct rules) the next statements hold.*
   (a) *If* $\beta_1 = \beta_1'\delta$ *and* $\beta_2 = \delta\beta_2'$ *for some* $\delta, \beta_1', \beta_2' \in (N \cup T \cup \{\$\})^+$, *then* $\alpha_1 = \alpha_1'\delta$ *and* $\alpha_2 = \delta\alpha_2'$ *for some* $\alpha_1', \alpha_2' \in (N \cup T \cup \{\$\})^*$.
   (b) *If* $\beta_1 = \gamma\beta_2\gamma'$ *for some* $\gamma, \gamma' \in (N \cup T \cup \{\$\})^*$, *then* $r_1 = r_2$.

The UPG-condition 2(a) requires that if some proper suffix of the righthand side of $r_1$ matches with some proper prefix of that of $r_2$, then the lefthand sides of $r_1$ and $r_2$ also contain them as a suffix and a prefix, respectively. The condition 2(b) says there is no pair of distinct rules $r_1$ and $r_2$ such that the righthand side of $r_2$ is a substring of that of $r_1$.

The following Theorem shows that any given string $w \in T^*$ can be parsed without backtracking provided that $w \in L(G)$.

**Theorem 2.1** [4] *Let* $G = (N, T, P, S, \$)$ *be a UPG, and let* $\eta$ *be a string in* $(N \cup T \cup \{\$\})^+$. *If* $\eta \stackrel{n}{\Leftarrow} \$S\$$, *then* $\eta \Leftarrow \xi \stackrel{n-1}{\Leftarrow} \$S\$$ *for any string* $\xi$ *such that* $\eta \Leftarrow \xi$ $(n = 1, 2, \cdots)$.

The next Corollary states that, in a UPG, parsing can always be performed in a unique way by a leftmost reduction (see [4] for the definition of a leftmost reduction $\underset{\text{lmr}}{\Leftarrow}$).

**Corollary 2.1** [4] *Let* $G = (N, T, P, S, \$)$ *be a UPG, and let* $\eta$ *be a string in* $(N \cup T \cup \{\$\})^+$. *If* $\eta \stackrel{n}{\Leftarrow} \$S\$$, *then* $\eta \underset{\text{lmr}}{\stackrel{n}{\Leftarrow}} \$S\$$ $(n = 1, 2, \cdots)$.

**Definition 2.5** *A rewriting rule of the following form is called a* right-terminating rule *(R-rule), where* $\alpha \in N^+$, $\beta \in N^*$, $x \in T^+$.

$$\alpha \to \beta x,\ \$\alpha \to \$\beta x,$$
$$\alpha\$ \to \beta x\$,\ \text{or } \$\alpha\$ \to \$\beta x\$.$$

*A rewriting rule of the following form is called a* context-free-like rule *(C-rule), where* $A \in N$, $\alpha \in N^+$.

$$A \to \alpha,\ \$A \to \$\alpha,\ A\$ \to \alpha\$,$$
$$\$A\$ \to \$\alpha\$,\ \text{or } \$A\$ \to \$\$.$$

*Let* $G = (N, T, P, S, \$)$ *be a UPG. G is called an* RC-UPG *iff every rule in P is either an R-rule or a C-rule.*

It is known that the class of RC-UPGs exactly characterizes the class of languages accepted by deterministic pushdown automata (DPDAs) (i.e., the class of DCFLs) as stated in the following Theorem. We use the notation $\mathcal{L}[\mathcal{A}]$ to describe the class of languages generated (accepted, respectively) by the class $\mathcal{A}$ of grammars (automata).

**Theorem 2.2** [4] $\mathcal{L}$[RC-UPG] $= \mathcal{L}$[DPDA].

### 2.2. Semi-Right-Terminating Grammars

We newly introduce a semi-right-terminating grammar, and a semi-right-terminating UPG here.

**Definition 2.6** *A rewriting rule of the following form is called a* semi-right-terminating rule *(SR-rule), where $\alpha, \beta \in N^+, A \in N$, and $t \in T$.*

$$\alpha \to \beta t,\ \ \alpha t \to \beta t,\ \ \alpha\$ \to \beta\$,$$
$$\$A \to \$t,\ \ or\ \$A\$ \to \$\$.$$

*Let $G = (N, T, P, S, \$)$ be a grammar. $G$ is said to be a* semi-right-terminating grammar (SR-G) *iff every rule in $P$ is an SR-rule.*

**Definition 2.7** *Let $G = (N, T, P, S, \$)$ be an SR-G. $G$ is said to be a* semi-right-terminating uniquely parsable grammar (SR-UPG) *iff $G$ satisfies the UPG-condition in Definition 2.4.*

The next Lemma states that SR-UPGs can also be defined by adding a simple constraint to SR-Gs.

**Lemma 2.1** *Let $G = (N, T, P, S, \$)$ be an SR-G. $G$ is an SR-UPG iff $G$ satisfies the following conditions.*

*(i) There is no rule in $P$ whose righthand side is $S\$$.*

*(ii) There is no pair of distinct rules $\alpha_1 \to \beta_1$ and $\alpha_2 \to \beta_2$ in $P$ such that $\beta_1 = \gamma\beta_2$ for some $\gamma \in N^*$.*

*Proof.* The "only if" part is obvious. The "if" part is also easy to prove as follows: First, it is clear that, if $G$ is an SR-G, then the righthand side of a rule in $P$ can be neither $S$, $\$S$, nor $\$S\$$ from the definition of an SR-rule. We can also see that, if $G$ is an SR-G, $P$ automatically satisfies the UPG-condition 2(a). This is because the righthand side of each rule is of the form $\beta t, \beta\$, \$t$, or $\$\$$ ($\beta \in N^+, t \in T$). Furthermore, if $G$ is an SR-G and $\beta_2$ is a substring of $\beta_1$ for a pair of distinct rules $\alpha_1 \to \beta_1$ and $\alpha_2 \to \beta_2$ in $P$, then $\beta_2$ must be a suffix of $\beta_1$ by the same reason as above. Hence, the Lemma follows. □

### 2.3. Pushdown Automata

We give here definitions of nondeterministic and deterministic pushdown automata which will be needed later (see e.g., [2, 5, 6] for the detail).

**Definition 2.8** *A pushdown automaton (PDA) is a system defined by*

$$M = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F),$$

*where $Q$ is a set of states, $\Sigma$ is a set of input symbols, $\Gamma$ is a set of stack symbols, $q_0 \in Q$ is an initial state, $Z_0 \in \Gamma$ is an initial stack symbol, and $F (\subseteq Q)$ is a set of final states. $\delta : Q \times (\Sigma \cup \{\varepsilon\}) \times \Gamma \to 2^{(Q \times \Gamma^*)}$ is a transition function.*

*An instantaneous description (ID) of a PDA $M$ is a triple $(q, w, \alpha) \in Q \times \Sigma^* \times \Gamma^*$. It denotes a computational configuration of $M$, where $M$ is in the state $q$, the unread input string is $w$, and the stack string is $\alpha$ (the leftmost symbol of $\alpha$ is at the top of the stack). The relation $\vdash$ between IDs that represents the transition of computational configuration is defined as follows: For any $q, p \in Q$, $a \in \Sigma \cup \{\varepsilon\}$, $w \in \Sigma^*$, $A \in \Gamma$, and $\alpha, \beta \in \Gamma^*$, $(q, aw, A\alpha) \vdash (p, w, \beta\alpha)$ iff $(p, \beta) \in \delta(q, a, A)$. Let $\overset{*}{\vdash}$ be the reflexive and transitive closure of $\vdash$. A string $w \in \Sigma^*$ is said to be accepted by $M$, iff $(q_0, w, Z_0) \overset{*}{\vdash} (q_f, \varepsilon, \alpha)$ for some $q_f \in F$ and $\alpha \in \Gamma^*$.*

**Definition 2.9** *Let $M = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$ be a PDA. $M$ is called a* deterministic pushdown automaton *(DPDA) iff (1) and (2) hold.*

*(1) $|\delta(q, a, A)| \le 1$ for every $(q, a, A) \in Q \times (\Sigma \cup \{\varepsilon\}) \times \Gamma$.*

*(2) For each $q \in Q$ and $A \in \Gamma$, if $(q, \varepsilon, A) \neq \emptyset$, then $(q, a, A) = \emptyset$ for all $a \in \Sigma$.*

## 3. Characterizing the Class of Deterministic Context-Free Languages by SR-UPGs

**Lemma 3.1**

*(I) For any PDA $M$, there is an SR-G $G_M$ such that $L(M) = L(G_M)$.*

*(II) For any DPDA $M$, there is an SR-UPG $G_M$ such that $L(M) = L(G_M)$.*

*Proof.* (I) Let $M = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$ be an arbitrary PDA. We can assume, without loss of generality, that the initial stack symbol $Z_0$ is never popped. We construct an SR-G $G_M = (N, T, P, S, \$)$ such that $L(M) = L(G_M)$ in the following manner. The sets $N$ and $T$ of nonterminals and terminals of $G_M$ are as follows (we assume $S \notin \Gamma$):

$$N = \{S\} \cup \Gamma \cup (\Gamma \times Q \times \Sigma) \cup (\Gamma \times Q)$$
$$T = \Sigma$$

The set $P$ of rules is as follows:

(1) For each $A \in \Gamma$, and $q_f \in F$, include the following rules in $P$.

$$S\$ \rightarrow AS\$$$
$$S\$ \rightarrow (A, q_f)\$$$

(2) For each $p, q \in Q$, $a, b \in \Sigma$ and $A, B \in \Gamma$, if $(p, \varepsilon) \in \delta(q, a, A)$ then include the following rules in $P$.

$$(B, p, b) \rightarrow B(A, q, a)b$$
$$(B, p)\$ \rightarrow B(A, q, a)\$$$

(3) For each $p, q \in Q$, $a, b \in \Sigma$, $A, B \in \Gamma$, and $\gamma \in \Gamma^*$, if $(p, B\gamma) \in \delta(q, a, A)$ then include the following rules in $P$ ($\gamma^R$ denotes the reversal string of $\gamma$).

$$\gamma^R(B, p, b) \rightarrow (A, q, a)b$$
$$\gamma^R(B, p)\$ \rightarrow (A, q, a)\$$$

(4) For each $p, q \in Q$, $a \in \Sigma$, $b \in \Sigma \cup \{\$\}$, and $A, B \in \Gamma$, if $(p, \varepsilon) \in \delta(q, \varepsilon, A)$ then include the following rule in $P$.

$$(B, p, a)b \rightarrow B(A, q, a)b$$

Furthermore, if $q \notin F$ then include the following rule.

$$(B, p)\$ \rightarrow B(A, q)\$$$

(5) For each $p, q \in Q$, $a \in \Sigma$, $b \in \Sigma \cup \{\$\}$, $A, B \in \Gamma$, and $\gamma \in \Gamma^*$, if $(p, B\gamma) \in \delta(q, \varepsilon, A)$ then include the following rule in $P$.

$$\gamma^R(B, p, a)b \rightarrow (A, q, a)b$$

Furthermore, if $q \notin F$ then include the following rule.

$$\gamma^R(B, p)\$ \rightarrow (A, q)\$$$

(6) For each $a \in \Sigma$, include the following rule in $P$.

$$\$(Z_0, q_0, a) \rightarrow \$a$$

(7) Include the following rule in $P$.

$$\$(Z_0, q_0)\$ \rightarrow \$\$$$

It is easy to see that each rule is an SR-rule. Hence $G_M$ is an SR-G.

The computing process of $M$ is simulated by a reduction process in $G_M$. An ID of $M$

$$(q,\ a_1 \cdots a_j,\ A_1 \cdots A_k)$$

is represented by the following string in $G_M$, where $q \in Q$, $a_1 \cdots a_j \in \Sigma^*$, and $A_1 \cdots A_k \in \Gamma^+$ ($k \geq 1$ because the initial stack symbol is never popped). In the following, we also call such strings IDs of $M$.

In the case $j \geq 1$: $\$A_kA_{k-1} \cdots A_2(A_1, q, a_1)a_2 \cdots a_j\$$

In the case $j = 0$: $\$A_kA_{k-1} \cdots A_2(A_1, q)\$$

We first show that if a terminal string $w = a_1 \cdots a_j \in T^*$ is generated by $G_M$ then $M$ accepts $w$. Since $w \in L(G_M)$, there is a reduction $\$w\$ \stackrel{*}{\Leftarrow} \$S\$$. Consider this reduction process. First, $\$w\$$ must be reduced by a rule in (6) (or the rule (7) if $w = \varepsilon$), because the other rules cannot be used for a terminal string. This yields $\$(Z_0, q_0, a_1)a_2 \cdots a_j\$$ (or $\$(Z_0, q_0)\$$ if $w = \varepsilon$), an initial ID of $M$. After that, the rules in (2)–(5) are used to reduce it, and new IDs appear successively. Note that in the case of an ID with $j \geq 1$, the rules in (2),(3) or the first rules in (4),(5) are used, while in the case of $j = 0$, the second rules in (4),(5) are used. It is easy to see that $M$'s movement is simulated correctly by the rules in (2)–(5). Finally, a string of the form $\$\alpha(A, q_f)\$$ ($\alpha \in \Gamma^*$, $A \in \Gamma$, $q_f \in F$) must appear. Otherwise, rules in (1) cannot be used to reduce the string into $\$S\$$. Since the string $\$\alpha(A, q_f)\$$ represents a final ID of $M$, the string $w$ is accepted by $M$.

Conversely, suppose a string $w = a_1 \cdots a_j \in \Sigma^*$ is accepted by $M$. That is, the following relation holds for some $q_f \in F$ and $A_1 \cdots A_k \in \Gamma^+$.

$$(q_0, w, Z_0) \overset{*}{\vdash} (q_f, \varepsilon, A_1 \cdots A_k)$$

We show that there is a reduction from $\$w\$$ to $\$S\$$. First, reducing $\$w\$$ by a rule in (6) (or (7) if $w = \varepsilon$), a string representing the initial ID is obtained. Then, each step of $M$ is simulated by the rules in (2)–(5). Since $M$ accepts $w$, $\$w\$$ must be reduced to a final ID

$$\$A_k A_{k-1} \cdots A_2 (A_1, q_f)\$.$$

After that, by the rules in (1) it is reduced to $\$S\$$. Thus, $\$w\$ \overset{*}{\Leftarrow} \$S\$$. Therefore $\$w\$$ is generated by $G_M$. By above, $L(G_M) = L(M)$.

(II) Let $M = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$ be an arbitrary DPDA. An SR-UPG $G_M$ that simulates $M$ is constructed in exactly the same way as shown in (I), and the proof of $L(G_M) = L(M)$ is also the same. The only thing we must show is $G_M$ is indeed an SR-UPG, i.e., it satisfies the conditions (i) and (ii) in Lemma 2.1.

It is clear that $G_M$ satisfies the condition (i). We can also verify that the righthand side of each rule cannot be a suffix of that of any other rule because $M$ is a DPDA (this is done by checking all the pairs of rule schemes in (1)–(7)). Hence, $G_M$ satisfies (ii). □

In order to show $\mathcal{L}$[PDA] $\supseteq$ $\mathcal{L}$[SR-G] and $\mathcal{L}$[DPDA] $\supseteq$ $\mathcal{L}$[SR-UPG], we first prove the following Lemma.

**Lemma 3.2** *Let $G = (N, T, P, S, \$)$ be an SR-G, and $w_0 \in T^*$ be a terminal string. For any $n\ (= 1, 2, \cdots)$ and any $\xi_n \in (N \cup T)^*$, if $\$w_0\$ \overset{n}{\Leftarrow} \$\xi_n\$$, then there exist $\eta_n \in N^+$ and $w_n \in T^*$ such that $\xi_n = \eta_n w_n$.*

*Proof.* It is proved by a simple induction on $n$. The case $n = 1$: It is clear, because only one of the rules of the form $\$A \to \$t$ or $\$A\$ \to \$\$$ can be used to reduce $\$w_0\$$.

The case $n > 1$: Consider a reduction $\$w_0\$ \overset{n-1}{\Leftarrow} \$\xi_{n-1}\$ \Leftarrow \$\xi_n\$$. From the inductive hypothesis, there exist $\eta_{n-1} \in N^+$ and $w_{n-1} \in T^*$ such that $\xi_{n-1} = \eta_{n-1} w_{n-1}$. Since only the rules of the form $\alpha \to \beta t$, $\alpha t \to \beta t$, or $\alpha\$ \to \beta\$$ can be used to reduce $\$\eta_{n-1} w_{n-1}\$$, there exist $\eta_n \in N^+$ and $w_n \in T^*$ such that $\xi_n = \eta_n w_n$. □

**Lemma 3.3**

*(I) For any SR-G $G$, there is a PDA $M_G$ such that $L(G) = L(M_G)$.*

*(II) For any SR-UPG $G$, there is a DPDA $M_G$ such that $L(G) = L(M_G)$.*

*Proof.* (I) Let $G = (N, T, P, S, \$)$ be an arbitrary SR-G. To prove the lemma, it suffices to show that there is a PDA $M_G = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$ that accepts the language $L(G)\{\$\} = \{w\$ \mid w \in L(G)\}$. Because, the class of context-free languages is closed under the quotient operation with a regular set (see e.g., [2]).

The sets of input and stack symbols of $M_G$ are as follows: $\Sigma = T \cup \{\$\}$, $\Gamma = N \cup T \cup \{\$, Z_0\}$. Let $\hat{R}$ be the maximum length of righthand sides of the rules in $P$. $M_G$ has an "internal" stack in its finite-state control that can store up to $\hat{R}$ symbols, besides its "real" stack. The whole stack of $M_G$ is formed by attaching the internal stack on the top of the real stack, and we call it simply a "stack" from now on. We assume that the internal stack always keeps as many symbols as it can by getting them from the real stack (except $Z_0$). Given an input $w\$ \in T^*\{\$\}$, $M_G$ simulates a reduction process of $\$w\$$ in the following manner.

1. Push the symbol \$ into the stack.
2. Read one input symbol and push it into the stack.
3. If the contents of the stack is $\$S\$Z_0$, then halt in a final state, else go to 4.
4. If some rule of $G$ is reversely applicable to the string stored in the internal stack, then nondeterministically choose one of such rules and perform the reduction in the internal stack and go to 5. If no such rule exists, then halt in a non-final state.
5. If the top symbol of the stack is a terminal or \$, then go to 3, else go to 2.

First we show that if $w \in L(G)$, then $w\$ \in L(M_G)$. Assume $w \in L(G)$, i.e., $\$w\$ \overset{m}{\Leftarrow} \$S\$$ holds for some $m$. Then for each $n$ $(0 \le n \le m)$, there are $\zeta_n \in \{\$\}N^*$, $a_n \in T \cup \{\$\}$, and $x_n \in T^*\{\varepsilon, \$\}$ such that $\$w\$ \overset{n}{\Leftarrow} \zeta_n a_n x_n$ from Lemma 3.2. We now give $w\$$ to $M_G$ as an input. Let $D_k = (q, x, \gamma)$ be the ID of $M_G$ at the $k$-th execution of Step 3 (note that, here, $\gamma$

represents the contents of the whole stack).

We claim that there are choices of movements of $M_G$ that satisfy the following condition: For each $n$ $(0 \le n \le m)$,

$$D_{n+1} = (q^{(n+1)},\ x_n,\ a_n \zeta_n^R Z_0)$$

holds for some $q^{(n+1)} \in Q$. This is proved by an induction on $n$.

The case $n = 0$: By Steps 1 and 2, $\zeta_0$ and $a_0$ are pushed into the stack, because in this case $\zeta_0 = \$$ and $a_0 x_0 = w\$$. Thus $D_1 = (q^{(1)},\ x_0,\ a_0 \zeta_0^R Z_0)$ holds for some $q^{(1)} \in Q$.

The case $n > 0$: By the induction hypothesis, we assume there are choices of movements of $M_G$ such that $D_{k+1} = (q^{(k+1)},\ x_k,\ a_k \zeta_k^R Z_0)$ for $k = 1, \cdots, n-1$. Let $\alpha \rightarrow \beta$ be a rule used in

$$\zeta_{n-1} a_{n-1} x_{n-1} \Leftarrow \zeta_n a_n x_n.$$

Since $\alpha \rightarrow \beta$ is an SR-rule, $\beta$ must be a suffix of $\zeta_{n-1} a_{n-1}$. Hence, this reduction can be performed at Step 4 in the internal stack. After that, if the top symbol is not a terminal, then an input symbol is read and pushed into the stack by the steps 5 and 2. Thus, $D_{n+1} = (q^{(n+1)},\ x_n,\ a_n \zeta_n^R Z_0)$ holds for some $q^{(n+1)}$.

By the claim just proved, we see $D_{m+1} = (q^{(m+1)},\ \varepsilon,\ \$S\$Z_0)$ holds, because $\zeta_m = \$S$, $a_m = \$$, and $x_m = \varepsilon$. Hence the input $w\$$ is accepted by $M_G$ at Step 3. Therefore $L(G)\{\$\} \subseteq L(M_G)$ holds.

On the other hand, it is clear that if $w \notin L(G)$ then $M_G$ does not accept $w\$$, because $M_G$ performs only legal reductions in $G$. Consequently, $L(M_G) = L(G)\{\$\}$ is concluded.

(II) Next, we consider the case that $G = (N, T, P, S, \$)$ is an SR-UPG. We can construct a DPDA $M_G$ that accepts the language $L(G)\{\$\}$ exactly the same way as in (I) (note that the class of deterministic context-free languages is also closed under the quotient operation with a regular set [1]). Because, by Lemma 2.1, there is at most one rule which is reversely applicable to a string of the form $\zeta_n a_n x_n$, and thus the step 4 is performed deterministically as well as the other steps. Hence, $M_G$ is a DPDA. The proof of $L(M_G) = L(G)\{\$\}$ is the same as (I) except that $M_G$ is deterministic. $\square$

By Lemmas 3.1 and 3.3, we can obtain the following Theorem.

**Theorem 3.1**

$$\begin{aligned} \mathcal{L}[\text{SR-G}] &= \mathcal{L}[\text{PDA}] \\ \mathcal{L}[\text{SR-UPG}] &= \mathcal{L}[\text{DPDA}] \end{aligned}$$

## 4. Concluding Remarks

We introduced an SR-G and an SR-UPG, and proved that they characterize the classes of CFLs and DCFLs. It is well known that the class of DCFLs is equal to the class of languages generated by the class of LR($k$) grammars [3]. But it is rather complex to test whether a given context-free grammar is an LR($k$) grammar. SR-UPGs give much simpler grammatical characterization of DCFLs than LR($k$) grammars. Furthermore, an SR-UPG is obtained from an SR-G by adding the following simple constraint (besides some minor constraint): There is no pair of rules such that the righthand side of a rule is a suffix of that of the other. Hence, the difference between deterministic and nondeterministic pushdown automata is also characterized simply in a grammatical formalism.

## References


[1] Ginsburg, S, and Greibach, S., Deterministic context free languages, *Inf. Control*, **9**, 620–648 (1966).

[2] Hopcroft, J. E., and Ullman J. D., *Introduction to automata theory, languages, and computation*, Addison-Wesley, Reading, Massachusetts (1979).

[3] Knuth, D. E., On the translation of languages from left to right, *Inf. Control*, **8**, 607–639 (1965).

[4] Morita, K., Nishihara, N., Yamamoto, Y., and Zhang, Z., A hierarchy of uniquely parsable grammar classes and deterministic acceptors, *Acta Informatica*, **34**, 389–410 (1997).

[5] Rozenberg, G., and Salomaa, A. (eds.), *Handbook of formal languages*, Vols.1–3, Springer-Verlag, Berlin (1997).

[6] Sippu, S., and Soisalon-Soininen, E., *Parsing theory*, Vols. I and II, Springer-Verlag, Berlin (1988,1990).