JOURNAL OF COMPUTER AND SYSTEM SCIENCES 7, 237–277 (1973)

Strict Deterministic Grammars*

MICHAEL A. HARRISON AND IVAN M. HAVEL†

Department of Computer Science, University of California at Berkeley, Berkeley, California 94720

Received April 5, 1972; revised August 23, 1972

A grammatical definition of a family of deterministic context free languages is presented. It is very easy to decide if a context free grammar is strict deterministic. A characterization theorem involving pushdown automata is proved, and it follows that the strict deterministic languages are coextensive with the family of prefix free deterministic languages. It is possible to obtain an infinite hierarchy of strict deterministic languages as defined by their degree.

1. INTRODUCTION

Among various families of formal languages, the context free languages are the most interesting both theoretically and practically. One of the main theorems in this area is that a set is a context free language if and only if it is accepted by a pushdown automaton. This characterization has been the starting point for a great deal of research particularly in the area of parsing. This has led to the definition and study of deterministic context free languages. There have been relatively few theoretical contributions to this theory since the papers by Knuth [12] and by Ginsburg and Greibach [3]. The main obstacle to a formal mathematical treatment seemed to be the lack of simple and convenient grammatical characterizations. The characterization given by Knuth [12] (by means of $LR(k)$ grammars) is formulated as a property of an infinite set of derivations in a given grammar, and, even if intuitively natural as a formal approach to deterministic parsing methods, it is not a very useful theoretical tool for mathematical investigations. This may be exemplified by the fact that no precise and rigorous mathematical proof of Knuth's characterization theorem has yet been published.

The principal purpose of this paper is to present and utilize a grammatical characterization of a new family of languages called *strict deterministic languages*. This class

* This research was supported by NSF Grant GJ-474.

† Present address: ÚTIA, Vyšehradská 49, Praha 2, Czechoslovakia.

is precisely the class of prefix-free deterministic languages—or, equivalently, the class of languages accepted by a deterministic pushdown automaton with empty store. It turns out that focusing attention on the strict deterministic languages does not force a great loss of generality over studying the entire class of deterministic context free languages. This is because any language in the larger family can be naturally mapped into the smaller class by the addition of an endmarker and all mathematical properties can be preserved under the mapping. Our characterization is based on the existence of a special type of partition defined on the set of letters of a given grammar. The existence of such a partition can be recognized in a very straightforward way by simple tests on the productions.

The present paper is the first of a series of three. In this paper, we give the basic definitions and some preliminary properties and establish two main theorems. The languages which are defined grammatically are characterized by pushdown automata (Theorem 3.5) and, moreover, an infinite hierarchy is established (Theorem 4.2). In [9] the connections between this family and the parsing problem are given. For the first time, certain important properties of $LR(k)$ and bounded right context languages are proved. In [10], the family of real time strict deterministic languages is studied.

The present paper is divided into four sections of which this is the first. In Section 2, strict deterministic grammars and languages are introduced. A procedure is given for deciding if a given context free grammar is strict deterministic. It is then shown that for any strict deterministic grammar, an equivalent strict deterministic Λ-free grammar may be found. In Section 3, pushdown automata are introduced and a variety of families of deterministic context free languages are defined. Proper containments of the families are established. The main theorem of the section characterizes strict deterministic languages by pushdown automata. In Section 4, a nontrivial hierarchy of strict deterministic languages [classified by their degrees] is established. The remainder of Section 1 is devoted to the mathematical definitions and notations which are required.

When X and Y are sets then any set $\rho \subseteq X \times Y$ is a *relation* (between X and Y). Let $\rho \subseteq X \times Y$ and $\sigma \subseteq Y \times Z$. We define

$$\rho\sigma = \{(x, z) \mid x\rho y\sigma z \text{ for some } y\},$$
$$\rho^{-1} = \{(x, y) \mid y\rho x\} \subseteq Y \times X$$

and, if $X = Y$,

$$\rho^0 = \{(x, x) \mid x \in X\} \text{ (the diagonal)},$$
$$\rho^{n+1} = \rho^n\rho,\ n \geqslant 0,$$
$$\rho^* = \bigcup_{n \geqslant 0} \rho^n \text{ (the reflexive and transitive closure of } \rho),$$
$$\rho^+ = \rho^*\rho \text{ (the transitive closure of } \rho).$$

Let $\rho \subseteq X \times X$. ρ is an *equivalence on* X iff $\rho \cdot \rho^* = \rho^{-1}$ (we almost always use the symbol $\equiv$ for equivalence).

Let X be a set. A *partition of* X is a collection $\pi = \{X_1, X_2, \ldots\}$ of nonempty mutually disjoint subsets $X_i \subseteq X$ such that $X = \bigcup_i X_i$. Subsets X_i are called *blocks* of partition π.

We write $X/\equiv$ for the partition induced by an equivalence relation $\equiv$ and $\equiv$ mod π for the equivalence relation induced by a partition π.

Let $\pi_1 = X/\equiv_1$ and $\pi_2 = X/\equiv_2$. We define

$$\pi_1 \leqslant \pi_2 \quad \text{iff} \quad \equiv_1 \subseteq \equiv_2, \tag{1}$$

and two other partitions

$$\pi_1 \cdot \pi_2 = X/(\equiv_1 \cap \equiv_2) \qquad \text{(the } \textit{meet} \text{ of } \pi_1, \pi_2) \tag{2}$$

and

$$\pi_1 + \pi_2 = X/(\equiv_1 \cup \equiv_2)^* \qquad \text{(the } \textit{join} \text{ of } \pi_1, \pi_2). \tag{3}$$

The set of all partitions on X together with the operations meet and join forms a *lattice of partitions of* X.

We define functions as single valued relations which are total together with their domains and ranges using the notation $f: X \to Y$ or $X \to^f Y$. For *partial functions* which correspond to singlevalued relations, we use the notation $f: X \to_p^f Y$ or $X \to_p Y$.

We call an *alphabet* any finite nonempty set of objects, called *letters*. We assume that all alphabets are considered as subsets of a fixed infinite set of letters, say Ω (we will not usually mention this set explicitly).

In our constructions we will often need special auxiliary alphabets whose letters correspond to certain given objects. To make these constructions uniform we formally define for any given finite nonempty set X a special new alphabet $Alph(X)$ by means of a fixed injection $X \to \Omega: x \to \bar{x}$, i.e.,

$$Alph(X) = \{\bar{x} \mid x \in X\}. \tag{4}$$

By convention we always assume that any alphabet of this form is disjoint from other alphabets in the particular context if they were introduced independently. When X is itself an alphabet, the $Alph$ operator produces a new copy of X.

We make a special agreement that the special symbol \$ denotes a new letter which is not a member of any alphabet used in the particular context (which will be always clear). This symbol will usually be used as an "endmarker."

For any alphabet A, by $A^+(A^*)$ we denote the free semigroup (free monoid) generated by A under concatenation. The identity in a free monoid is always denoted by Λ. We use the abbreviation

$$A_\Lambda = A \cup \{\Lambda\}. \tag{5}$$

Elements of A^* are called *strings* (over A); in particular, Λ is called the *empty string*. Subsets of A^* are called *languages* (over A).

We use the term *homomorphism* always for a monoid homomorphism $h: A^* \to B^*$ (A and B are alphabets), i.e., $h: \Lambda \mapsto \Lambda$ and $h: uv \mapsto h(u)\,h(v)$. For a complete specification of such a homomorphism h it is enough to define a function $A \to B$; h is then the *homomorphic extension* of this function.

Let $u, v \in A^*$ be two strings. Then u is a *prefix* (*suffix*) of v iff $v = uw$ ($v = wu$) for some $w \in A^*$; when $w \neq \Lambda$, u is a *proper prefix* (*proper suffix*) of v. We denote by $\lg(w)$ the *length* of w, i.e., the total number of occurrences of letters in w, in particular, $\lg(\Lambda) = 0$. For any $n \geqslant 0$

$$\begin{aligned} &{}^{(n)}w(w^{(n)}) \text{ is the prefix (suffix) of } w \\ &\quad \text{with length } \min(\lg(w), n). \end{aligned} \tag{6}$$

A language $L \subseteq A^*$ is said to be *prefix-free* iff

$$w \in L \quad \text{and} \quad wu \in L \quad \text{implies} \quad u = \Lambda. \tag{7}$$

Among the various operations over strings let us mention the *left quotient*

$$u\backslash v = \begin{cases} v' & \text{if } v = uv' \text{ for some string } v', \\ \text{undefined} & \text{otherwise.} \end{cases} \tag{8}$$

This operation can be extended to languages in a natural way:

$$L_1\backslash L_2 = \{u\backslash v \mid u \in L_1\,,\ v \in L_2\}. \tag{9}$$

The Boolean and Kleene operations over languages are used in the usual way.

If L is a language, then we sometimes write $L\$$ for $L\{\$\}$ even when L is a singleton.

Let us define a *family of languages* as any set of languages (over different alphabets, in general) which contains at least one nonempty language.

2. Strict Deterministic Grammars

In this section the principal objects of our study, namely the strict deterministic grammars and languages, will be introduced. After the necessary definitions, we shall describe a procedure for recognizing strict deterministic grammars among general context-free grammars. Some of the basic properties of strict deterministic grammars will then be stated. The section is concluded by showing that for any strict deterministic grammar we can find an equivalent strict deterministic grammar which is Λ-free.

First we review some basic concepts concerning formal grammars.

DEFINITION 2.1. A *context-free grammar* (hereafter, a *grammar*) G is a 4-tuple

$$G = \langle V, \Sigma, P, S \rangle, \tag{1}$$

where V and Σ are two alphabets, $\Sigma \subseteq V$ (letters in Σ and in $N = V - \Sigma$ are called *terminals* and *nonterminals*, respectively), $S \in N$ and P is a finite relation, $P \subseteq N \times V^*$ (the set of *productions*).

By convention[1] we write $A \to \alpha$ is in P, or sometimes only $A \to \alpha$, instead of $(A, \alpha) \in P$. We also write $A \to \alpha_1 \mid \alpha_2 \mid \cdots \mid \alpha_n$ instead of $A \to \alpha_1$ and $A \to \alpha_2$ and $\cdots$ and $A \to \alpha_n$ (here "|" is a metasymbol not in V). Where the reference to G is important we write $\to_G$ instead of $\to$.

DEFINITION 2.2. Let G be a grammar of the form (1). We define a relation $\Rightarrow \subseteq V^* \times V^*$ as follows. For any $\alpha, \beta \in V^*$ $\alpha \Rightarrow \beta$ iff $\alpha = \alpha_1 A \alpha_2$, $\beta = \alpha_1 \beta_1 \alpha_2$, and $A \to \beta_1$ is in P for some $A \in N$ and α_1, α_2, $\beta_1 \in V^*$. In particular, if $\alpha_1 \in \Sigma^*$ or $\alpha_2 \in \Sigma^*$ we write $\alpha \Rightarrow_L \beta$ or $\alpha \Rightarrow_R \beta$, respectively. Any $\alpha \in V^*$ is called a (*canonical*) *sentential form* iff $S \Rightarrow^* \alpha (S \Rightarrow_R^* \alpha)$.

The *language generated by* G is the language

$$L(G) = \{w \in \Sigma^* \mid S \Rightarrow^* w\}. \tag{2}$$

Two grammars are called *equivalent* iff they generate the same language.

Note that in (2) relation $\Rightarrow^*$ can be replaced by $\Rightarrow_R^*$ or by $\Rightarrow_L^*$. Given a string $w \in L(G)$, the relation $S \Rightarrow^* w$ can be decomposed (in general, not uniquely) to a so-called *derivation* of w:

$$S = \alpha_0 \Rightarrow \alpha_1 \Rightarrow \cdots \Rightarrow \alpha_n = w, \tag{3}$$

where the α_i, $i = 1,\ldots, n$, are sentential forms of G. A number of different techniques have been developed for finding derivations of the form (3) for a given string w. These techniques are usually called *parsing algorithms*.[2] There is an important subfamily of context-free languages, called the *deterministic context-free languages* (the formal definition will be given later) admitting fast and efficient parsing methods. This paper will be concerned with this family of languages and some closely related families. This investigation has led to a new family of grammars which we shall now motivate.

In our new family of grammars, we wish to make certain restrictions on the simul-

[1] We adopt certain preferences in usage of symbols. When talking about grammars we use symbols $A, B, C,\ldots$ for elements of V or N; $a, b, c,\ldots$ for elements of Σ or Σ_A; $\alpha, \beta, \gamma,\ldots$ for elements of V^*; and $u, v, w,\ldots$ for elements of Σ^*.

[2] The reader is referred to a plentiful literature for more details about these techniques and applications to parsing and translation of programming languages, in particular to [6] or [14].

taneous occurrences of substrings in different productions. Intuitively, if $A \to \alpha\beta$ is a production in our grammar then "partial information" about A together with complete information about a prefix α of $\alpha\beta$ yields similar partial information about the next symbol of β when $\beta \neq \Lambda$, or otherwise the complementary information about A when $\beta = \Lambda$. In the formal definition which follows, the intuitive notion of "partial information" is precisely represented by means of a certain partition π of the total vocabulary of the grammar.

DEFINITION 2.3. Let G be a grammar of the form (1), and let π be a partition of the set V of terminal and nonterminal letters of G. Such a partition π is called *strict* iff

1. $\Sigma \in \pi$ and
2. for any $A, A' \in N$ and $\alpha, \beta, \beta' \in V^*$ if $A \to \alpha\beta$, $A' \to \alpha\beta'$, and $A \equiv A' \pmod{\pi}$ then either
   (i) both $\beta, \beta' \neq \Lambda$ and[3] ${}^{(1)}\beta \equiv {}^{(1)}\beta' \pmod{\pi}$ or
   (ii) $\beta = \beta' = \Lambda$ and $A = A'$.

In most cases, the partition π will be clear from the context and we shall write simply $A \equiv B$ instead of $A \equiv B \pmod{\pi}$, and $[A]$ instead of

$$[A]_\pi = \{A' \in V \mid A' \equiv A \pmod{\pi}\}.$$

DEFINITION 2.4. Any grammar G of the form (1) is called *strict deterministic* iff there exists a strict partition π of V. A language L is called a *strict deterministic language* iff $L = L(G)$ for some strict deterministic grammar G.

To justify our terminology, let us make a premature remark that the strict deterministic languages will be shown (in Section 3) to be precisely all those context-free languages which are *deterministic* (in the usual sense) and *prefix-free* (Cf. (7) in Section 1). In a general sense, the *determinism* in its *strict* meaning requires that all actions be determined without any anticipation of the future. Thus, in our automata-theoretic framework, if an acceptance of a string terminates the recognition procedure, the accepted language cannot contain prefixes of its own strings.

We emphasize the fact that the existence of a strict partition is a direct property of productions of the grammar, rather than a property of the (generally infinite) set of all derivations as in the case of $LR(k)$ grammars [12]. There is, however, a "global" property of derivation trees which corresponds in a natural way to the strict determinism of a grammar, [9].

At this point the reader may welcome a few examples of strict deterministic grammars.

[3] Cf. (6) in Section 1 for the definition of ${}^{(1)}\beta$.

EXAMPLE 2.1. Let G_1 be a grammar with the productions[4]

$$S \to aA \mid aB$$
$$A \to aAa \mid bC$$
$$B \to aB \mid bD$$
$$C \to bC \mid a$$
$$D \to bDc \mid c.$$

The blocks of a strict partition are: Σ, $\{S\}$, $\{A, B\}$, $\{C, D\}$. The language is

$$L(G_1) = \{a^n b^k a^n, a^k b^n c^n \mid k, n \geqslant 1\}.$$

EXAMPLE 2.2. Our second example is a grammar for a set of simple arithmetic expressions (enclosed in parentheses). A natural grammar for them might have productions

$$S \to (E)$$
$$E \to E + T \mid T$$
$$T \to T * F \mid F$$
$$F \to S \mid a.$$

Unfortunately, this grammar has no strict partition. However, we can find an equivalent strict deterministic grammar G_2.

$$S \to (E$$
$$E \to T_1 E \mid T_2$$
$$T_1 \to F_1 T_1 \mid F_2$$
$$T_2 \to F_1 T_2 \mid F_3$$
$$F_1 \to (E+ \mid a+$$
$$F_2 \to (E* \mid a*$$
$$F_3 \to (E) \mid a)$$

The blocks of a strict partition for G_2 are Σ, $\{S\}$, $\{E\}$, $\{T_1, T_2\}$, $\{F_1, F_2, F_3\}$.

As we already mentioned, the existence of a strict partition is a direct property of the grammar; this makes the decision problem trivial. For later applications we now present a simple algorithm for testing a grammar for a strict partition. First we introduce a convenient concept.

[4] For simplicity we display in this and all subsequent examples only the set of productions when defining a grammar. The reader will recognize the nonterminals, terminals, and the initial letter without trouble by our conventions.

DEFINITION 2.5. Let $\alpha, \beta \in V^*$ and let A, B be two letters in V such that $A \neq B$, and we have $\alpha = \gamma A \alpha_1$ and $\beta = \gamma B \beta_1$ for some $\gamma, \alpha_1, \beta_1 \in V^*$. Then the pair (A,B) is called the *distinguishing pair of* α *and* β. A distinguishing pair (A, B) is said to be *terminal* iff $A, B \in \Sigma$.

From Definition 2.5 we can make the following observation.

FACT. *Any two strings* $\alpha, \beta \in V^*$ *have a distinguishing pair iff* α *is not a prefix of* β *and* β *is not a prefix of* α. *If they have a distinguishing pair then it is unique.*

We shall now give our algorithm.

ALGORITHM 1. (*The algorithm takes as input any context-free grammar G and determines whether G is strict deterministic or not. In the former case the algorithm produces the minimal*[5] *strict partition of V.*)

Assume that the productions of G consecutively numbered, i.e.,

$$P = \{A_i \to \alpha_i \mid i = 1,..., \mid P \mid\}$$

and all productions are distinct.

Step 1. *Initially define* $\pi = \{\{A\} \mid A \in N\} \cup \{\Sigma\}$. *Set* $i = 0$.

Step 2. *Set* $i = j = i + 1$. (i *and* j *are pointers to two productions in P.*) *If* $i > \mid P \mid$ *go to step* 8.

Step 3. *Set* $j = j + 1$. *If* $j > \mid P \mid$ *go to step* 2.

Step 4. (*Note that* $A_i \to \alpha_i$ *and* $A_j \to \alpha_j$ *are two distinct productions in P.*) *If* $A_i \neq A_j$ *go to step* 3. *If* α_i *and* α_j *have no distinguishing pair* (*i.e., if one is a prefix of the other, in particular, if* $\alpha_i = \alpha_j$) *go to step* 7.

Step 5. *Let* (B, C) *be the unique distinguishing pair of* α_i, α_j. *If* $B = C$ *go to step* 3. *If* $B \in \Sigma$ *or* $C \in \Sigma$ (*both cannot be in* Σ) *go to step* 7.

Step 6. *Replace* $[B]$ *and* $[C]$ *in* π (*note that* $B \neq C$) *by one new block* $[B] \cup [C]$. *Set* $i = 0$ *and go to step* 2.

Step 7. *Halt. G is not a strict deterministic grammar.*

Step 8. *Halt. G is strict deterministic under* π.

The flow of control in the algorithm is schematized in Fig. 1.

Proof of the correctness of the algorithm. The algorithm is relatively simple, and

[5] The partition is minimal with respect to the standard lattice ordering of partitions (cf. (1) in preliminaries). We return to minimal strict partitions later in this section.

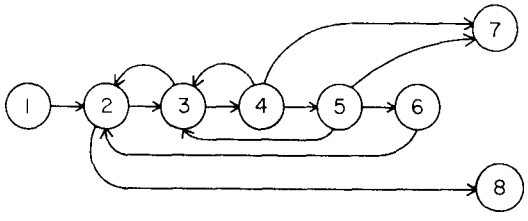


FIGURE 1

we give only an outline of a proof of its correctness. At a later time, we will need the properties which may prevent a grammar from being strict deterministic.

First let us check that the algorithm always halts. From Fig. 1 we can see that all closed loops go through step 3. But step 3 always increases the value of j. Only step 2 can decrease this value but it always increases the value of i. Only step 6 can decrease the value of i but it also decreases the cardinality of π. And $\min \mid \pi \mid = 2$.

Now, let us investigate how a grammar may fail to be strict deterministic. Under the hypothesis and notation of Definition 2.3 we note that if $A = A'$ (in the initial stage of the algorithm) or if $A \equiv A'$ (as a result of preceding stages in any subsequent stage) and if $\beta, \beta' \neq \Lambda$ and ${}^{(1)}\beta, {}^{(1)}\beta' \in N$, the case (i) in condition 2 of the definition can be always satisfied by forcing ${}^{(1)}\beta \equiv {}^{(1)}\beta'$ (step 6 does the job). We do not obtain an essential failure under these circumstances. But the following three kinds of failure are essential. Assume $A \equiv A'$ (forced by preceding stages).

Failure I.

$$A \to \alpha\beta_1$$
$$A' \to \alpha,$$

where $\beta_1 \neq \Lambda$. Taking $\beta = \beta_1$ and $\beta' = \Lambda$ in Definition 2.3 neither (i) nor (ii) can occur. Algorithm 1 goes from step 4 directly to step 7 since $\alpha\beta_1$ and α have no distinguishing pair.

Failure II.

$$A \to \alpha$$
$$A' \to \alpha,$$

where $A \neq A'$. Taking $\beta = \beta' = \Lambda$ we obtain case (ii) with the contradictory requirement $A = A'$. Algorithm 1 halts in the same way as in the case of failure I.

Failure III.

$$A \to \alpha B\beta_1$$
$$A' \to \alpha a\beta_2,$$

where $B \in N$ and $a \in \Sigma$. Taking $\beta = B\beta_1$ and $\beta' = a\beta_2$ in Definition 2.3 we obtain case (i) with the requirement $B \equiv a$ which contradicts condition 1. Algorithm 1 goes from step 5 to step 7.

These are all the possible ways for a grammar to fail to be strict deterministic. If no failure occurs, G is strict deterministic. Also, Algorithm 1 cannot halt in step 7 and since it always terminates, it halts in step 8.

The fact that the partition produced in step 8 is strict and minimal follows from the property of the algorithm that two letters are made equivalent (in step 6) if and only if it is required by Definition 2.3. Q.E.D.

Next several simple but important properties of strict deterministic grammars are given. All of these results are almost direct consequences of Definition 2.3.

DEFINITION 2.6. A grammar G of the form (1) is *reduced* iff either $P = \varnothing$ or for any $A \in V$ we have $S \Rightarrow^* \alpha A \beta \Rightarrow^* w$ for some $\alpha, \beta \in V^*$ and $w \in \Sigma^*$.

THEOREM 2.1. *Any strict deterministic grammar is equivalent to a reduced strict deterministic grammars.*

Proof. The standard construction (cf. [2, 11]) for reducing a grammar consists only of deleting some productions. The result, thus, follows directly from the following claim.

CLAIM. *Let π be a strict partition for a grammar $G = \langle V, \Sigma, P, S\rangle$ and let $P' \subseteq P$. Then π is also strict partition in grammar $G' = \langle V, \Sigma, P', S\rangle$.*

Proof of the Claim. For the sake of contradiction, assume that neither case (i) nor case (ii) in Definition 2.3 is satisfied for a pair of productions $p_1, p_2 \in P'$. But $p_1, p_2 \in P$, and, hence, π cannot be strict even in G. Q.E.D.

The following lemma extends the property of strict partitions from productions to certain derivations in the grammar.

LEMMA 2.2. *Let G be a grammar of the form* (1) *with a strict partition π. Then for any $A, A' \in N$; $\alpha, \beta, \beta' \in V^*$ and $n \geqslant 1$ if $A \Rightarrow_L^n \alpha\beta$, $A' \Rightarrow_L^n \alpha\beta'$ and $A \equiv A'$ then either* (i) *both $\beta, \beta' \neq \Lambda$ and ${}^{(1)}\beta \equiv {}^{(1)}\beta'$ or*

(ii) $\beta = \beta' = \Lambda$ *and* $A = A'$.

Proof. The argument is an induction on n. The basis is immediate since for $n = 1$ the assertion of the lemma is equivalent to condition 2 in Definition 2.3.

Inductive Step. Assume the assertion of the lemma is true for a given $n \geqslant 1$ and consider the case of $n + 1$. We can write

$$A \Rightarrow_L^n wB\beta_1 \Rightarrow_L w\beta_2\beta_1 = \alpha\beta$$

and

$$A' \Rightarrow_L^n w'B'\beta_1' \Rightarrow_L w'\beta_2'\beta_1' = \alpha\beta'$$

for some $B, B' \in N$; $w, w' \in \Sigma^*$ and $\beta_1, \beta_1', \beta_2, \beta_2' \in V^*$ and assume, without loss of generality, that $\lg(\beta_2') \geqslant \lg(\beta_2)$.

Case 1. $0 \leqslant \lg(\alpha) < \lg(w)$. Then both ${}^{(1)}\beta, {}^{(1)}\beta' \in \Sigma$ and, thus, ${}^{(1)}\beta = {}^{(1)}\beta'$.
For the remaining cases we make the following claim.

CLAIM. *In the previous derivations, if w is a prefix of w', or w' is a prefix of w then $w = w'$ and $B = B'$.*

Proof of Claim. Assume, without loss of generality, that w is a prefix of w', i.e., $w' = ww''$ for some $w'' \in \Sigma^*$. We have $A \Rightarrow_L^n wB\beta_1$ and $A' \Rightarrow_L^n ww''B'\beta_1'$. Then by the inductive hypothesis ${}^{(1)}(B\beta_1) = {}^{(1)}(w''B'\beta_1')$, and since for any $a \in \Sigma$, $B \neq a$ (by the property that $\Sigma \in \pi$), we conclude that $w'' = \Lambda$, $w = w'$, and $B = B'$. This completes the proof of the claim.

Case 2. $\lg(w) \leqslant \lg(\alpha) < \lg(w\beta_2)$. In other words, we have $\alpha = w\gamma_1$ and $\beta_2 = \gamma_1\gamma_2$ for some $\gamma_1, \gamma_2 \in V^*$, where $\gamma_2 \neq \Lambda$. Since w is a prefix of w', or conversely (because they are initial subwords of α) we have $w = w'$ and $B = B'$ by the claim. Therefore, we can write also $\beta_2' = \gamma_1\gamma_2'$ for some $\gamma_2' \in V^+$ (we have used the assumption that $\lg(\beta_2') \geqslant \lg(\beta_2)$). Now we have $B \to \gamma_1\gamma_2$, $B' \to \gamma_1\gamma_2'$ and $\gamma_2, \gamma_2' \neq \Lambda$. By the strictness of π (property 2(i) in Definition 2.3) we conclude that

$${}^{(1)}\beta = {}^{(1)}\gamma_2 = {}^{(1)}\gamma_2' = {}^{(1)}\beta'.$$

Case 3. $\lg(w\beta_2) \leqslant \lg(\alpha) \leqslant \lg(w\beta_2\beta_1)$. Again from the claim we have $w = w'$ and $B = B'$. Moreover, β_2 is a prefix of β_2', or conversely. By the strictness of π (property 2(ii) in Definition 2.3) this is possible only if $\beta_2 = \beta_2'$ and $B = B'$. Thus, $wB = wB'$ and the result follows from the inductive hypothesis. Q.E.D.

THEOREM 2.2. *Any strict deterministic language is prefix-free.*

Proof. Let G be a strict deterministic grammar of the form (1) and assume

$$S \Rightarrow_L^n w \qquad \text{and} \qquad S \Rightarrow_L^{n'} wu$$

for some w, $u \in \Sigma^*$ and $n, n' \geqslant 1$. If $n \leqslant n'$ we have

$$S \Rightarrow_L^n w \qquad \text{and} \qquad S \Rightarrow_L^n \alpha \Rightarrow_L^* wu \tag{4}$$

for some $\alpha \in V^*$. On the other hand, if $n' < n$ we have

$$S \Rightarrow_L^{n'} \alpha \Rightarrow_L^* w \qquad \text{and} \qquad S \Rightarrow_L^{n'} wu. \tag{5}$$

In either case for any k, $0 \leqslant k < \lg(\alpha)$, ${}^{(k)}\alpha = {}^{(k)}w$ implies ${}^{(1)}\alpha^{(\lg(\alpha)-k)} \equiv {}^{(1)}w^{(\lg(w)-k)}$ by Lemma 2.2 and then ${}^{(k+1)}\alpha = {}^{(k+1)}w$ since any terminal prefix of α is (by the definition of $\Rightarrow_L$) a prefix of wu in (4) or of w in (5). Having ${}^{(0)}\alpha = \Lambda = {}^{(0)}w$ we conclude that $\alpha = w$ and $u = \Lambda$. Q.E.D.

The following result will often be needed in the sequel.

THEOREM 2.3. *Let G be a reduced strict deterministic grammar of the form* (1). *Then for any $A, B \in N$ and $\alpha \in V^*$,*

$$A \Rightarrow^+ B\alpha \qquad \text{implies} \quad A \not\equiv B.$$

Proof. Let G be as in the theorem and let $A \Rightarrow^+ B\alpha$ for some $A, B \in N$, and $\alpha \in V^*$. Then for some $\alpha' \in V^*$ and $n \geqslant 1$ we have

$$A \Rightarrow_L^n B\alpha'.$$

Assume for the sake of contradiction that $A \equiv B$. Then by Lemma 2.2 for any $A' \in [A]$, $A' \Rightarrow_L^n \beta$ implies ${}^{(1)}\beta \in [A]$, and we obtain for arbitrarily large $k \geqslant 1$ a derivation

$$A \Rightarrow_L^{kn} \beta_k, \tag{6}$$

where $\beta_k \in NV^*$. By the fact that G is reduced also

$$A \Rightarrow_L^m w, \tag{7}$$

where $w \in \Sigma^*$ and $m \geqslant 1$. Applying now Lemma 2.2 to (7) (and using the definition of $\Rightarrow_L$) we obtain that for any $m' \geqslant m$ and $\gamma \in V^*$, $A \Rightarrow_L^{m'} \gamma$ implies $\gamma \in \Sigma V^*$ or $\gamma = \Lambda$ which contradicts (6). Q.E.D.

COROLLARY. *No reduced strict deterministic grammar is left recursive (i.e., for no $A \in N$ and $\alpha \in V^*$, $A \Rightarrow^+ A\alpha$).*

Remark. Lemma 2.2 could be also used to prove that any strict deterministic grammar is unambiguous.

We now turn our attention to the relationship of distinct strict partitions defined for the same grammar. The following example shows that this can indeed be the case.

EXAMPLE 2.4. Let G_3 be a grammar with productions

$$S \to aA$$
$$A \to bB$$
$$B \to a.$$

There are three strict partitions, namely

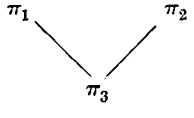
$$\pi_1 = \{\{S, A\}, \{B\}, \Sigma\},$$
$$\pi_2 = \{\{S\}, \{A, B\}, \Sigma\},$$
$$\pi_3 = \{\{S\}, \{A\}, \{B\}, \Sigma\}.$$

THEOREM 2.4. *Let G be a strict deterministic grammar of the form* (1) *and let $\mathscr{P}_G$ be the set of all strict partitions for G. Then $\mathscr{P}_G$ is a semilattice under the meet operation*[6] $(\pi_1, \pi_2) \to \pi_1 \cdot \pi_2$.

Proof. First $\mathscr{P}_G$ is nonempty since G is strict deterministic, by assumption. Since the meet is a lattice operation in the lattice of all partitions of V it is idempotent, commutative, and associative. Therefore, it is enough to prove that for any two strict partitions π_1 and π_2 their meet $\pi_1 \cdot \pi_2$ is again a strict partition.

By definition $\pi_1 \cdot \pi_2 = \{V_1 \cap V_2 \mid V_1 \in \pi_1, V_2 \in \pi_2\}$. We have $\Sigma \in \pi_1 \cdot \pi_2$ since $\Sigma \in \pi_1$ and $\Sigma \in \pi_2$. Let $A, A' \in N$; $\alpha, \beta, \beta' \in V^*$; $A \to \alpha\beta$, $A' \to \alpha\beta'$. If $A \equiv A' \pmod{\pi_1 \cdot \pi_2}$ then $A \equiv A' \pmod{\pi_1}$ and $A \equiv A' \pmod{\pi_2}$. Let $\beta, \beta' \neq \Lambda$. Then ${}^{(1)}\beta \equiv {}^{(1)}\beta' \pmod{\pi_1}$ as well as ${}^{(1)}\beta \equiv {}^{(1)}\beta' \pmod{\pi_2}$ since both π_1 and π_2 are strict. Hence, ${}^{(1)}\beta \equiv {}^{(1)}\beta' \pmod{\pi_1 \cdot \pi_2}$. On the other hand, if $\beta = \Lambda$ then $\beta' = \Lambda$ and $A = A'$ by the strictness of π_1 or π_2. Q.E.D.

Remark. As a consequence of Theorem 2.4 for any strict deterministic grammar there exists a unique *minimal strict partition* π_0. On the other hand, there is no dual concept of a maximal strict partition, in general, and the semilattice $\mathscr{P}_G$ may not be a lattice. For instance, in Example 2.4 semilattice $\mathscr{P}_{G_3} = \{\pi_1, \pi_2, \pi_3\}$ has the structure



DEFINITION 2.7. For any strict partition π in a given grammar define

$$\| \pi \| = \max_{V_i \in \pi - \{\Sigma\}} | V_i |.$$

[6] Cf. (2) in Section 1 for the definition of meet. A semilattice is a partially ordered set in which every two elements have a greatest lower bound.

Note that if $\pi_1 \leqslant \pi_2$ in the standard lattice ordering of partitions (cf. (1) in Section 1) then $\| \pi_1 \| \leqslant \| \pi_2 \|$. In Section 4 we shall use the number $\| \pi_0 \|$, where π_0 is the minimal strict partition for a given grammar G, as an important and nontrivial measure of the complexity of strict deterministic grammars and languages.

When working with formal grammars, it is often convenient to have grammars in a special form where the productions satisfy suitable restrictions. Section 2 will be concluded by establishing that any strict deterministic grammar can be transformed into a form with no Λ productions that is still strict deterministic.

DEFINITION 2.8. Let G be a grammar of the form

$$G = \langle V, \Sigma, P, S \rangle. \tag{8}$$

Any production of the form $A \to \Lambda$ where $A \in N$ is called a *Λ-production*. Grammar G is said to be in *Λ-free form* iff it has no Λ-productions.

We know (cf. Theorem 2.2) that any strict deterministic grammar G generates a prefix-free language. Therefore, either $\Lambda \notin L(G)$ or $L(G) = \{\Lambda\}$. Obviously we cannot have a Λ-free grammar for the latter case, but otherwise we can always eliminate the Λ-productions.

THEOREM 2.5. *Let G be a strict deterministic grammar. Then either $L(G) = \{\Lambda\}$ or there exists an equivalent Λ-free strict deterministic grammar G'.*

Proof. Let G be a strict deterministic grammar of the form (8), and let π be a strict partition of V. First note that if $B \to \Lambda$ is in P then this is the only production for B and, moreover, $[B] = \{B\}$. For, if $B' \equiv B$ and $B' \to \alpha$ for some $B' \in N$, then $\alpha = \Lambda$ and $B' = B$ by property 2 of Definition 2.3.

Assume that if $B \to \Lambda$ then B occurs at most once in the right side of each production of G. There is no loss of generality in this assumption since we can always replace the additional occurrences of B by new nonterminal letters $B_1, B_2, \ldots$ and add new productions $B_1 \to \Lambda$, $B_2 \to \Lambda, \ldots$. The new grammar is clearly equivalent to the original one, and, if k new nonterminals are used, it has a strict partition

$$\pi \cup \{\{B_i\} \mid 1 \leqslant i \leqslant k\}.$$

We shall prove the theorem by induction on $n = |N|$.

Basis. If $n = 1$ then S is the only nonterminal letter of G. Either $S \to \Lambda$ and then $L(G) = \{\Lambda\}$ or G has no Λ-productions.

Inductive step. Assume the result true for grammars with fewer than n nonterminals, $n > 1$ and which have "Λ-variables" occurring at most once in each production. If G has no Λ-productions or if $S \to \Lambda$ is in P (and, thus, $L(G) = \{\Lambda\}$), we are done. Assume $B \in V - \{S\}$ and $B \to \Lambda$ is in P.

Let G' be a new grammar obtained from G by removing $B \to \Lambda$ from P and substituting Λ for B in the remaining rules. Then we can remove B from N, and we obtain a grammar with $n - 1$ nonterminals which will complete the proof. Formally, let $G' = \langle V', \Sigma, P_1 \cup P_2, S \rangle$, where $V' = V - \{B\}$,

$$P_1 = \{A \to \alpha \mid A \to \alpha \text{ in } P,\ A \neq B \text{ and } \alpha \in (V')^*\}, \tag{9}$$

$$P_2 = \{A \to \alpha\beta \mid A \to \alpha B \beta \text{ in } P\}. \tag{10}$$

Clearly $L(G') = L(G)$. We shall show that the partition $\pi' = \pi - \{[B]\}$ is strict in G'. First, $\Sigma \in \pi'$ is immediate. Let A, $A' \in V' - \Sigma$ and $A \equiv A' \pmod{\pi'}$, or equivalently, $A \equiv A' \pmod{\pi}$. Let p, p' be two productions in $P_1 \cup P_2$ of the form $A \to \alpha\beta$ and $A' \to \alpha\beta'$, respectively (α, β, $\beta' \in (V')^*$). We want to show that either

$$\begin{aligned} &\beta, \beta' \neq \Lambda \quad \text{and} \quad {}^{(1)}\beta \equiv {}^{(1)}\beta' \pmod{\pi'}, \quad \text{or} \\ &\beta = \beta' = \Lambda \quad \text{and} \quad A = A'. \end{aligned} \tag{11}$$

This is immediate if both p and $p' \in P_1$ by the strictness of π (condition 2 in Definition 2.3) since then p, $p' \in P$.

Let $p \in P_2$ or $p' \in P_2$ and assume without loss of generality the first case, i.e., $p \in P_2$. Then by (10) there is a production $p_0 \in P$ which contains an occurrence of B. We distinguish two cases.

Case 1. p_0 has the form $A \to_G \alpha_1 B \alpha_2 \beta$ where α_1, $\alpha_2 \in (V')^*$ and $\alpha_1\alpha_2 = \alpha$. Since p' has the form $A' \to_{G'} \alpha_1\alpha_2\beta'$ it cannot be in p_1. For otherwise, by the strictness of π, $B \equiv {}^{(1)}(\alpha_2\beta) \pmod{\pi}$ is in contradiction to the fact that $[B] = \{B\}$ and that B does not occur in $\alpha_2\beta$. Therefore, $p' \in P_2$. By the strictness of π the only possible form of a production in P from which p' is obtained in (10), is $A' \to \alpha_1 B \alpha_2 \beta'$. Now also (11) follows by the strictness of π.

Case 2. p_0 has the form $A \to_G \alpha\beta_1 B \beta_2$, where $\beta_1 \in (V')^+$, $\beta_2 \in (V')^*$, and $\beta_1\beta_2 = \beta$. If $p' \in P_1$ we have $\beta' \neq \Lambda$ and ${}^{(1)}\beta = {}^{(1)}\beta_1 \equiv {}^{(1)}\beta' \pmod{\pi}$ by the strictness of π, and, thus, ${}^{(1)}\beta \equiv {}^{(1)}\beta' \pmod{\pi'}$. Assume now $p' \in P_2$. The corresponding production in P (in (10)) cannot have the form $A' \to \alpha_1 B \alpha_2 \beta'$ (α_1, $\alpha_2 \in V'^*$ and $\alpha_1\alpha_2 = \alpha$) since by the strictness of π we would have ${}^{(1)}(\alpha_2\beta_1) \equiv B$ contradictory to $\beta_1 \neq \Lambda$ and to the assumption that B occurs not more than once in any right side of a production in P. Thus, the production corresponding to p' has the form $A' \to \alpha\beta_1' B \beta_2'$ for some $\beta_1' \in (V')^+$, $\beta_2' \in (V')^*$ such that $\beta_1'\beta_2' = \beta'$. Now, by the strictness of π, ${}^{(1)}\beta = {}^{(1)}\beta_1 \equiv {}^{(1)}\beta_1' = {}^{(1)}\beta' \pmod{\pi}$, and, therefore, ${}^{(1)}\beta \equiv {}^{(1)}\beta' \pmod{\pi'}$. Q.E.D.

There are other normal forms for strict deterministic grammars which also hold. For instance, every strict deterministic language has a strict deterministic grammar which is in Greibach normal form [11].

3. Deterministic Pushdown Automata

The purpose of this section is to establish the relationship between strict deterministic grammars and deterministic pushdown automata. The main result is that the generative power of the strict deterministic grammars is exactly the same as the power of acceptance of deterministic pushdown automata (which accept by empty store). As a consequence, the family of strict deterministic languages and the family of prefix-free deterministic languages coincide.

It will be pointed out that our consideration of the prefix-free case is not as restrictive as it might seem at first sight. There is a simple and natural way of passing from general deterministic languages to their prefix-free form and backwards (cf. Remark following Definition 3.4). On the other hand, we gain a tremendous simplification of our results and proofs by this restriction.

First we shall review the standard terminology and notation concerning deterministic pushdown automata.

Definition 3.1. A *deterministic pushdown automaton* (abbreviated *DPDA*) is an 7-tuple

$$M = \langle Q, \Sigma, \Gamma, \delta, q_0, Z_0, F \rangle, \tag{1}$$

where Q is a finite nonempty set, Σ and Γ are two alphabets, $q_0 \in Q$, $Z_0 \in \Gamma$, $F \subseteq Q$ and δ is a partial function[7]

$$\delta: Q \times \Sigma_\Lambda \times \Gamma \to_p Q \times \Gamma^* \tag{2}$$

with the property that for any $q \in Q$ and $Z \in \Gamma$,

$$\delta(q, \Lambda, Z) \neq \varnothing \quad \text{implies} \quad \delta(q, a, Z) = \varnothing \quad \text{for all} \quad a \in \Sigma. \tag{3}$$

We often use intuitive terminology in which Q is called the *set of states* (of the control), Σ is the *input alphabet*, Γ is the *pushdown alphabet*, q_0 is the *initial state*, Z_0 is the *initial pushdown letter*, and δ the *transition function*. Certain strings over Γ are interpreted as contents of the *pushdown store*; in this interpretation we assume that the bottom of the store is on the left and the top on the right. If some $q, q' \in Q$, $a \in \Sigma_\Lambda$, $Z \in \Gamma$ and $\gamma \in \Gamma^*$ satisfy

$$\delta(q, a, Z) = (q', \gamma), \tag{4}$$

then formula (4) is called a *move* (of DPDA M); in particular, if $a = \Lambda$, (4) is called a *Λ-move*.

Let us mention for completeness, that a (*nondeterministic*) *pushdown automaton*

[7] Cf. Section 1 for our conventions about partial functions. Also recall that $\Sigma_\Lambda = \Sigma \cup \{\Lambda\}$.

(abbreviated PDA) is defined in a similar way as DPDA except that δ is a many-valued finitary function, or, formally, instead of (2) we have

$$\delta: Q \times \Sigma_\Lambda \times \Gamma \to \text{finite subsets of } Q \times \Gamma^*.$$

Then a DPDA can be defined as a PDA satisfying the inequality

$$|\delta(q, \Lambda, Z)| + |\delta(q, a, Z)| \leqslant 1,$$

for all $q \in Q$, $Z \in \Gamma$, and $a \in \Sigma$.

DEFINITION 3.2. Let M be a DPDA of the form (1) and let $\mathcal{Q} = Q \times \Sigma^* \times \Gamma^*$. The *yield relation of* M, $\vdash_M \subseteq \mathcal{Q} \times \mathcal{Q}$ (or $\vdash$ when M is understood) is defined as follows. For any $q, q' \in Q$, $a \in \Sigma_\Lambda$, $w \in \Sigma^*$, $\alpha, \beta \in \Gamma^*$, and $Z \in \Gamma$,

$$(q, aw, \alpha Z) \vdash (q', w, \alpha\beta) \qquad \text{iff} \quad \delta(q, a, Z) = (q', \beta). \tag{5}$$

Set $\mathcal{Q}$ in this definition is the *configuration space* (of M); its elements are *configurations*. The configuration (q_0, w, Z_0) for some $w \in \Sigma^*$ is called the *initial configuration* (*for* w). An instance of the yield relation on the left side of (5) is called a *transition* (from the first configuration to the second) *corresponding to the move* on the right side of (5). Any sequence of configurations $c_0, \ldots, c_n, \ldots \in \mathcal{Q}$ (possibly infinite) such that $c_0 \vdash \cdots \vdash c_n \vdash \cdots$ and, where c_0 is an initial configuration, is called a *computation* (of M).

We now endow a DPDA with an ability to define, or *accept*, certain languages over its input alphabet.

DEFINITION 3.3. Let M be a DPDA of the form (1). For a given $K \subseteq \Gamma^*$ define the language $T(M, K) \subseteq \Sigma^*$ as follows

$$T(M, K) = \{w \in \Sigma^* \mid (q_0, w, Z_0) \vdash^* (q, \Lambda, \alpha) \text{ for some } q \in F \text{ and } \alpha \in K\}. \tag{6}$$

In particular, let

$$T_0(M) = T(M, \Gamma^*),$$
$$T_1(M) = T(M, \Gamma),$$

and

$$T_2(M) = T(M, \Lambda).$$

The customary notation for $T_0(M)$ is $T(M)$ and for $T_2(M)$ in the case when $F = Q$, is Null(M) (cf. [2][8]). Acceptance by $T_1(M)$ is equivalent to the acceptance by re-

[8] It is not hard to show that for any DPDA M there exists a DPDA M' such that $T_2(M) = \text{Null}(M')$.

initializing the pushdown store (cf., e.g., [4]) or by empty store if moves on empty store are allowed (cf., e.g., [5]). It is immediate from (6) that for $i = 1, 2$, $T_i(M) \subseteq T_0(M)$. In the nondeterministic case these three types of acceptance lead to the same family of languages (the context-free languages). This is not, however, the case for DPDA.

DEFINITION 3.4. We define the following three families of languages for $i = 0, 1, 2$.

$$\Delta_i = \{T_i(M) \mid M \text{ is a DPDA}\}.$$

The languages in Δ_2 have not been previously studied. They will be our primary area of investigation as they are identical to the family of languages generated by strict deterministic grammars. Note that Δ_2 does not even contain all regular events but only the prefix free ones.

The family Δ_1 (called "deterministic" by Dikovskii [1] and Greibach [5]) has the advantage over Δ_2 in that it contains all regular languages. It can be shown that Δ_1 is a subfamily of the closure of Δ_2 under the Kleene operations. Finally the languages in Δ_2 are the *deterministic* (*context free*) *languages* first studied in detail by Haines [7] and by Ginsburg and Greibach [3].

Remark. Our restriction to Δ_2 is justified by the fact that there exists a trivial injection

$$\Delta_0 \to \Delta_2 : L \mapsto L\$$$

(so called *endmarking*; strictly speaking the endmarker \$ may depend on L by the condition that $L \subseteq (\Sigma - \{\$\})^*$ for some alphabet Σ). This injection preserves most mathematical properties (as e.g., the cardinality, regularity, equivalence, etc.) or, at least, transforms them to other properties which can be easily recognized (as, e.g., inclusion, properties of substrings, etc.).[9]

First we mention several properties which are simple consequences of the definitions of Δ_0, Δ_1, and Δ_2.

FACT 1. *All languages in Δ_2 are prefix-free.*

Proof. This is an immediate consequence of the definition of $T_2(M)$ since if

$$(q_0, x, Z_0) \vdash^* (q, \Lambda, \Lambda)$$

then

$$(q_0, xy, Z_0) \vdash^* (q, y, \Lambda) \vdash^* (q', \Lambda, \Lambda)$$

is possible only if $y = \Lambda$ (and $q' = q$). Q.E.D.

[9] Also from the point of view of applications to parsing of programming languages there is no loss of generality in assuming endmarkers; as a matter of fact, practical parsing procedures always make this assumption.

Next, we note the containment properties of the Δ_i.

THEOREM 3.1. $\Delta_2 \subset \Delta_1 \subset \Delta_0$ (*all inclusions are proper*).

Proof. Part 1: $\Delta_2 \subseteq \Delta_1$. Let $L \in \Delta_2$ and let M be a DPDA of the form (1) such that $L = T_2(M)$. We shall construct another DPDA M' with the same behavior as M except that M' will have an additional initial pushdown letter Z_0' which is never overwritten. Formally, let

$$M' = \langle Q \cup \{q_0'\}, \Sigma, \Gamma \cup \{Z_0'\}, \delta', q_0', Z_0', F\rangle,$$

where $q_0' \notin Q$, $Z_0' \notin \Gamma$ and

$$\delta'(q, a, Z) = \begin{cases} \delta(q, a, Z) & \text{if } q \in Q, \quad a \in \Sigma_\Lambda \text{ and } Z \in \Gamma; \\ (q_0, Z_0'Z_0) & \text{if } q = q_0', \quad a = \Lambda \text{ and } Z = Z_0'; \\ & \text{undefined otherwise.} \end{cases}$$

Now, for any $x \in \Sigma^*$, first $(q_0', x, Z_0') \vdash_{M'} (q_0, x, Z_0'Z_0)$ and afterwards for any $q \in F$

$$(q_0, x, Z_0'Z_0) \vdash_{M'}^* (q, \Lambda, Z_0') \qquad \text{iff} \quad (q_0, x, Z_0) \vdash_M^* (q, \Lambda, \Lambda).$$

Since L is prefix-free, $q \in F$ implies $q \neq q_0$ and $\delta'(q, \Lambda, Z_0') = \varnothing$. Thus, $x \in T_1(M')$ iff $x \in T_2(M)$. Therefore, $L = T_1(M')$.

Part 2: $\Delta_1 \subseteq \Delta_0$. Let $L \in \Delta_1$ and let M be DPDA of the form (1) such that $L = T_1(M)$. We shall construct another DPDA M' which possesses a special copy of each letter in Γ, to be able to distinguish the bottom of the store. M' simulates M but accepts only with a single letter on the store. Formally, let

$$M' = \langle Q \cup F', \Sigma, \Gamma \cup \Gamma', \delta', q_0, Z_0', F'\rangle,$$

where[10] $\Gamma' = Alph(\Gamma)$, $Z_0' = \bar{Z}_0$, $F' = Alph(F)$, and δ' is defined as follows. For all $Z \in \Gamma$ and $a \in \Sigma_\Lambda$

$$\delta'(q, a, Z) = \delta(q', a, Z) \qquad \text{if} \quad q = q' \in Q \quad \text{or} \quad q = \overline{q'} \in F';$$

$$\delta'(q, a, \bar{Z}) = \begin{cases} (q'', \overline{Z'}\alpha) & \text{if} \quad q = q' \in Q - F \quad \text{or} \quad q = \overline{q'} \in F' \quad \text{and} \\ & \quad \delta(q', a, Z) = (q'', Z'\alpha); \\ (\bar{q}, \bar{Z}) & \text{if} \quad q \in F \quad \text{and} \quad a = \Lambda. \end{cases}$$

Otherwise δ' is not defined.

Now for every $x \in \Sigma^*$, $(q_0, x, \bar{Z}_0) \vdash_{M'}^* (q, \Lambda, \bar{Z}\alpha)$ iff $(q_0, x, Z_0) \vdash_M^* (q, \Lambda, Z\alpha)$.

[10] Cf. (4) in Section 1 for the definition of "*Alph.*"

Here $x \in T_1(M)$ iff $\alpha = \Lambda$ and $q \in F$, hence iff $(q, \Lambda, \bar{Z}\alpha) \vdash_{M'} (\bar{q}, \Lambda, \bar{Z})$, hence iff $x \in T_0(M')$. Thus, $L = T_0(M')$.

Part 3: $\Delta_2 \neq \Delta_1$. We prove this by the following example.

EXAMPLE 3.1. Language $L_1 = a^*$ is in Δ_1 but not in Δ_2. For, L_1 is not prefix-free, and, therefore, $L_1 \notin \Delta_2$. On the other hand, $L_1 = T_1(M)$ where M is a DPDA of the form (1) with $\Sigma = \{a\}$, $Q = F = \{q_0\}$, $\Gamma = \{Z_0\}$ and $\delta(q_0, a, Z_0) = (q_0, Z_0)$ is the only move of M.

Part 4: $\Delta_1 \neq \Delta_0$. Again the proof takes the form of an example.

EXAMPLE 3.2. Language $L_2 = \{a^n b^n \mid n \geqslant 1\} \cup a^*$ is in Δ_0 but not in Δ_1.

First, $L_2 = T_0(M)$ where M is a DPDA of the form (1) with $Q = \{q_0, q_1\}$, $\Sigma = \{a, b\}$, $\Gamma = \{Z_0, Z_1, Z_2\}$, $F = \{q_0\}$ and

$$\delta(q_0, a, Z) = \begin{cases} (q_0, Z_1) & \text{if } Z = Z_0; \\ (q_0, ZZ_2) & \text{if } Z \neq Z_0; \end{cases}$$

and for any $q \in Q$

$$\delta(q, b, Z) = \begin{cases} (q_0, \Lambda) & \text{if } Z = Z_1; \\ (q_1, \Lambda) & \text{if } Z = Z_2; \\ \text{undefined} & \text{if } Z = Z_0. \end{cases}$$

Second, assume that $L_2 = T_1(M)$ for some DPDA M. Since there are only a finite number of accepting configurations (q, Λ, Z), where $q \in F$ and $Z \in \Gamma$, there is a sufficiently large n such that for some $k < n$

$$(q_0, a^n, Z_0) \vdash^* (q, \Lambda, Z)$$

and

$$(q_0, a^k, Z_0) \vdash^* (q, \Lambda, Z),$$

where $q \in F$ and $Z \in \Gamma$.
But since for some $q' \in F$ and $Z' \in \Gamma$,

$$(q_0, a^n b^n, Z_0) \vdash^* (q, b^n, Z) \vdash^* (q', \Lambda, Z'),$$

we also have

$$(q_0, a^k b^n, Z_0) \vdash^* (q, b^n, Z) \vdash^* (q', \Lambda, Z'),$$

and, hence, $a^k b^n \in L_2$ which is a contradiction since $k < n$. Therefore, $L_2 \notin \Delta_1$. Q.E.D.

Our next result is quite simple but informative.

THEOREM 3.2. *$L \in \Delta_2$ iff L is prefix-free and $L \in \Delta_0$. Thus, Δ_2 coincides with the family of prefix-free deterministic languages.*

Proof. The "only if" direction is immediate from Fact 1 and Theorem 3.1.

For the "if" direction, let $L \in \Delta_0$ and let M be a DPDA of the form (1) such that $L = T_0(M)$. We can construct another DPDA M' which simulates M and erases all its store after M accepts. If L is prefix-free then $L = T_2(M')$.

Formally, let

$$M' = \langle Q \cup \{q_f\}, \Sigma, \Gamma, \delta', q_0, Z_0, \{q_f\}\rangle,$$

where $q_f \notin Q$ and

$$\delta'(q, a, Z) = \begin{cases} \delta(q, a, Z) & \text{if } q \notin F, \\ (q_f, \Lambda) & \text{if } q \in F \cup \{q_f\} \text{ and } a = \Lambda, \\ & \text{undefined otherwise.} \end{cases}$$

Assume that L is prefix-free. If

$$(q_0, xy, Z_0) \vdash_M^* (q, y, \alpha) \vdash_M^* (q', \Lambda, \alpha'),$$

then $q' \in F$ implies $q \notin F$ or $y = \Lambda$. Therefore, for any $x \in \Sigma^*$ and $q \in F$,

$$(q_0, x, Z_0) \vdash_M^* (q, \Lambda, \alpha) \quad \text{iff} \quad (q_0, x, Z_0) \vdash_{M'}^* (q, \Lambda, \alpha) \vdash_{M'}^* (q_f, \Lambda, \Lambda),$$

and, thus, $x \in T_0(M)$ iff $x \in T_2(M')$. Hence, $L \in \Delta_2$. Q.E.D.

We now begin the long argument to show that every language in Δ_2 is strict deterministic. Our first lemma puts a DPDA into a more convenient form.

LEMMA 3.1. *Let $L \in \Delta_2$. Then $L = T_2(\hat{M})$ for some* DPDA *$\hat{M}$ with a single final state.*

Proof. Assume $L = T_2(M)$ for some DPDA M of the form (1). Using a similar construction as in the proof that $\Delta_1 \subseteq \Delta_0$ in the last section, we construct another DPDA $\hat{M}$ which can distinguish the bottom of the store. Otherwise it simulates M except that if M erases its store and accepts, $\hat{M}$ does the same by entering the single final state.

Formally, let

$$\hat{M} = \langle Q, \Sigma, \Gamma \cup \mathit{Alph}(\Gamma), \delta', q_0, \bar{Z}_0, \{q_f\}\rangle,$$

where $q_f \in F$ is chosen arbitrarily and for every $q \in Q$, $a \in \Sigma$, and $Z \in \Gamma$

$$\delta'(q, a, Z) = \delta(q, a, Z),$$

$$\delta'(q, a, \bar{Z}) = \begin{cases} (q', \bar{Z}'\gamma) & \text{if } \delta(q, a, Z) = (q', Z'\gamma); \\ (q_f, \Lambda) & \text{if } \delta(q, a, Z) = (q', \Lambda) \text{ and } q' \in F; \\ & \text{undefined otherwise.} \end{cases}$$

Now, for any $w \in \Sigma^*$,

$$\begin{aligned} w \in T_2(\hat{M}) \quad &\text{iff} \quad (q_0, w, \bar{Z}_0) \vdash_{\hat{M}}^* (q, a, \bar{Z}) \vdash_{\hat{M}} (q_f, \Lambda, \Lambda) \text{ for some} \\ &\quad a \in \Sigma_\Lambda, Z \in \Gamma \text{ and } q \in Q \\ &\text{iff} \quad (q_0, w, Z_0) \vdash_{\hat{M}}^* (q, a, Z) \vdash_M (q', \Lambda, \Lambda) \\ &\quad \text{for some } q' \in F \\ &\text{iff} \quad w \in T_2(M). \end{aligned}$$ Q.E.D.

Now we give the construction which carries a DPDA in the required form into a grammar.

DEFINITION 3.5. Let $\hat{M}$ be a DPDA of the form

$$\hat{M} = \langle Q, \Sigma, \Gamma, \delta, q_0, Z_0, \{q_f\} \rangle \tag{7}$$

(i.e., $\hat{M}$ has a single final state). We define the *canonical grammar* $G_{\hat{M}}$ *of* $\hat{M}$ as follows.

$$G_{\hat{M}} = \langle V, \Sigma, P, S \rangle \tag{8}$$

where $V = Alph(Q \times \Gamma \times Q) \cup \Sigma$, $S = \overline{(q_0, Z_0, q_f)}$ and P is defined as follows.[11] For any $a \in \Sigma_\Lambda$, $Z, Z_1, \ldots, Z_k \in \Gamma$; $q, p, q_1, \ldots, q_k \in Q$ and $k \geqslant 1$,

$$\overline{qZq_k} \to a\, \overline{pZ_1q_1}\, \overline{q_1Z_2q_2} \cdots \overline{q_{k-1}Z_kq_k} \text{ is in } P \text{ iff } \delta(q, a, Z) = (p, Z_k \cdots Z_2Z_1); \tag{9}$$

and

$$\overline{qZp} \to a \text{ is in } P \quad \text{iff} \quad \delta(q, a, Z) = (p, \Lambda). \tag{10}$$

(No other productions are in P.) The canonical grammar is the principal construction which leads from pushdown automata to grammars. The same construction is used in standard proofs of equivalence of PDA and context-free grammars (cf., e.g., [2] and [11]).

Before we proceed to a proof that the construction works, we give an example of a canonical grammar constructed for a given DPDA.

EXAMPLE 3.3. DPDA $\hat{M}$ of the form (7) for which

$$T_2(M) = \{a^n b^k a^n, a^n b^k c^k \mid k, n \geqslant 1\}$$

[11] We use the simplified notation $\overline{qZq'}$ instead of $\overline{(q, Z, q')}$ for elements of $Alph(Q \times \Gamma \times Q)$.

is defined as follows: $Q = \{0, 1, 2\}$, $\Sigma = \Gamma = \{a, b, c\}$, $q_0 = 0$, $Z_0 = c$, $q_f = 2$ and δ is defined in the left part of the following table:

$\delta: Q \times \Sigma_\Lambda \times \Gamma \to Q \times \Gamma^*$					P
0	a	c	0	ca	$\overline{0c2} \to a\,\overline{0a1}\;\overline{1c2}$ $\overline{0c2} \to a\,\overline{0a2}\;\overline{2c2}$
0	a	a	0	aa	$\overline{0a1} \to a\,\overline{0a1}\;\overline{1a1}$ $\overline{0a2} \to a\,\overline{0a2}\;\overline{2a2}$
0	b	a	0	b	$\overline{0a1} \to b\,\overline{0b1}$ $\overline{0a2} \to b\,\overline{0b2}$
0	b	b	0	bb	$\overline{0b1} \to b\,\overline{0b1}\;\overline{1b1}$ $\overline{0b2} \to b\,\overline{0b2}\;\overline{2b2}$
0	a	b	1	Λ	$\overline{0b1} \to a$
1	Λ	b	1	Λ	$\overline{1b1} \to \Lambda$
1	a	a	1	Λ	$\overline{1a1} \to a$
1	Λ	c	2	Λ	$\overline{1c2} \to \Lambda$
0	c	b	2	Λ	$\overline{0b2} \to c$
2	c	b	2	Λ	$\overline{2b2} \to c$
2	Λ	a	2	Λ	$\overline{2a2} \to \Lambda$
2	Λ	c	2	Λ	$\overline{2c2} \to \Lambda$

Productions of G_M constructed by (9) and (10) are in the right part of the table (the useless productions were omitted). After elimination of Λ-productions and some simple modifications we arrive at the following grammar

$$G_{\hat{M}} = \langle Alph(Q \times \Gamma \times Q) \cup \Sigma, \Sigma, P, \overline{0c2} \rangle,$$

where P is a follows:

$$\begin{array}{l}
\overline{0c2} \to a\,\overline{0a1} \quad | \quad a\,\overline{0a2} \\
\hline
\overline{0a1} \to a\,\overline{0a1}\,a \quad | \quad b\,\overline{0b1} \\
\overline{0a2} \to a\,\overline{0a2} \quad | \quad b\,\overline{0b2} \\
\hline
\overline{0b1} \to b\,\overline{0b1} \quad | \quad a \\
\overline{0b2} \to b\,\overline{0b2}\,c \quad | \quad c
\end{array}$$

Note that this grammar is "isomorphic" to the grammar from Example 2.1 generating the same language.

It can be easily checked that grammar $G_{\hat{M}}$ in the foregoing example is strict deterministic under the partition indicated by broken horizontal lines. We shall now prove this result in general.

LEMMA 3.2. *For any* DPDA $\hat{M}$ *with a single final state the canonical grammar* $G_{\hat{M}}$ *is strict deterministic.*

Proof. Let $G_{\hat{M}}$ be the canonical grammar of the form (8) for a given DPDA $\hat{M}$ (with a single final state). Define an equivalence $\equiv$ on V such that

$$\begin{aligned}
A \equiv B \quad &\text{iff} \quad A, B \in \Sigma \quad \text{or} \\
A = \overline{qZq'} \quad &\text{and} \quad B = \overline{qZq''}
\end{aligned}$$

for some $q, q', q'' \in Q$, $Z \in \Gamma$. Let $\pi = V/\equiv$. Obviously $\Sigma \in \pi$. Let

$$\begin{aligned}
A = \overline{qZq'} &\to a\gamma = \alpha\beta, \\
A' = \overline{qZq''} &\to a'\gamma' = \alpha\beta',
\end{aligned}$$

where $a, a' \in \Sigma \cup \Lambda$; $\gamma, \gamma' \in (V - \Sigma)^*$. Let us make the following three observations.

Observation 1. Either $a = a' = \Lambda$ or both $a, a' \in \Sigma$. (Otherwise (3) would be violated.)

Observation 2. If $a = a'$ then either $\gamma = \gamma' = \Lambda$ and $q' = q''$, or we have

$$\begin{aligned}
\gamma &= \overline{pZ_1q_1}\;\overline{q_1Z_2q_2} \cdots \overline{q_{k-1}Z_kq'} \\
\gamma' &= \overline{pZ_1q_1'}\;\overline{q_1'Z_2q_2'} \cdots \overline{q_{k-1}Z_kq''}
\end{aligned}$$

for some $p, q_i, q_i' \in Q$, $Z_i \in \Gamma$, and $k \geqslant 1$. In either case $\lg(\gamma) = \lg(\gamma')$. (This observation is a consequence of (9), (10) and of the fact that δ is single valued.)

Observation 3. Either $\beta = \beta' = \Lambda$ or $\beta, \beta' \neq \Lambda$. (Assume for instance $\beta = \Lambda$ but $\beta' \neq \Lambda$. Then $\alpha = \alpha\beta = a\gamma$ and $a'\gamma' = \alpha\beta' = a\gamma\beta'$. Thus, $a' = a$ and by Observation 2 $\lg(\gamma) = \lg(\gamma')$. But now $\lg(\alpha) = \lg(\alpha\beta')$ which contradicts that $\beta' \neq \Lambda$.)

To prove the strictness of π, assume first $\beta \neq \Lambda$. Then by the last observation $\beta' \neq \Lambda$.

Case 1. $\alpha = \Lambda$ and $a \in \Sigma$. Then $a' \in \Sigma$ by observation 1 and both ${}^{(1)}\beta, {}^{(1)}\beta' \in \Sigma$. Hence, ${}^{(1)}\beta = {}^{(1)}\beta'$.

Case 2. $\alpha = a \in \Sigma_\Lambda$. Then $a' = \alpha = a$ and by observation 2 (note that $\gamma \neq \Lambda$) we have ${}^{(1)}\beta = \overline{pZ_1q_1} = \overline{pZ_1q_1'} = {}^{(1)}\beta'$.

Case 3. $\alpha \in \Sigma_\Lambda N^+$. Then again $a = a'$ and by observation 2, $\alpha^{(1)} = \overline{q_{i-1}Z_iq_i} = \overline{q_{i-1}Z_iq_i'}$ for some i, $1 \leqslant i < k$. Thus, $q_i = q_i'$ and ${}^{(1)}\beta = \overline{q_iZ_{i+1}q_{i+1}} = \overline{q_iZ_{i+1}q_{i+1}'} = {}^{(1)}\beta'$.

Now, assume $\beta = \Lambda$, and, thus, $\beta' = \Lambda$. Then $a = a'$ and $\gamma = \gamma'$. By observation 2 $q' = q''$. Hence $A = A'$. Q.E.D.

We must also check that $G_{\hat{M}}$ generates the desired language.

LEMMA 3.3. *Let $\hat{M}$ be a* DPDA *with a single final state and let $G_{\hat{M}}$ be its canonical grammar. Then $L(G_{\hat{M}}) = T_2(\hat{M})$.*

Proof. The result is known from the literature (cf. [2] or [8]) and we give here a rather concise proof for the sake of completeness.

Let $\hat{M}$ and $G_{\hat{M}}$ be as in Definition 3.5, in particular, let $N = Alph(Q \times \Gamma \times Q)$. Define the following three sets

$$H = \{\alpha \in N^* \mid \text{if } \alpha = \alpha_1 \overline{q_1Z_1q_1'}\ \overline{q_2Z_2q_2'}\ \alpha_2 \text{ for some } \alpha_1, \alpha_2 \in N^* \text{ then } q_1' = q_2\},$$

$${}_pN = \{\overline{pZq} \in N \mid Z \in \Gamma, q \in Q\},$$

$$N_q = \{\overline{pZq} \in N \mid p \in Q, Z \in \Gamma\}.$$

Define also the function $f\colon H \to \Gamma^*$ inductively by $\Lambda \mapsto \Lambda$ and $\alpha\ \overline{qZq'} \mapsto Zf(\alpha)$.

CLAIM. *For any $w \in \Sigma^*$, $p, q, q' \in Q$, $Z \in \Gamma$, $\alpha \in H$ and $n \geqslant 1$*

$$\overline{qZq'} \Rightarrow_L^n w\alpha \quad \text{iff} \quad (q, w, Z) \vdash^n (p, \Lambda, f(\alpha)) \text{ and either}$$
$$\alpha \in {}_pNH \cap HN_{q'} \text{ or } (\alpha = \Lambda \text{ and } p = q').$$

Proof of the claim. The argument is an induction on n.

Basis. For $n = 1$ we have $\overline{qZq'} \to_{G_{\hat{M}}} w\alpha$, on the one hand, and $\delta(q, w, Z) = (p, f(\alpha))$, on the other hand. The claim then follows directly from (9) and (10).

Inductive step. Assume the claim true for integers smaller than or equal to n. For the case of length $n+1$ we have first

$$\overline{qZq'} \Rightarrow_L^{n+1} w\alpha \qquad \text{iff} \quad \overline{qZq'} \Rightarrow_L^{n} w_1 Y\alpha_2 \Rightarrow_L w_1 a\alpha_1\alpha_2 , \tag{11}$$

for some $Y \in N$, $w_1 a = w$ and $\alpha_1\alpha_2 = \alpha$. Applying the inductive hypothesis twice we find that (11) is true iff

$$\begin{gathered}(q, w_1 a, Z) \vdash^n (q'', a, f(\alpha_2)\, Z') \vdash (p, \Lambda, f(\alpha_2) f(\alpha_1)) \text{ where} \\ \overline{q''Z'p'} = Y,\ Y \rightarrow_{G_{\hat{M}}} a\alpha_1 ,\ Y\alpha_2 \in HN_{q'} \text{ and either} \\ \alpha_1 \in {}_pNH \cap HN_{p'} \text{ or} \\ (\alpha_1 = \Lambda \text{ and } p = p').\end{gathered} \tag{12}$$

Now, since $\alpha_2 = \Lambda$ implies $p' = q'$, and $\alpha_2 \neq \Lambda$ implies $\alpha_2 \in {}_{p'}NH$, we conclude that (12) is true iff

$$\begin{gathered}(q, w, Z) \vdash^{n+1} (p, \Lambda, f(\alpha)) \text{ and either} \\ \alpha = \alpha_1\alpha_2 \in {}_pNH \cap HN_{q'} \text{ or} \\ (\alpha = \alpha_1 = \alpha_2 = \Lambda \text{ and } p = p' = q').\end{gathered}$$

This proves the claim for $n+1$.

Now to complete the proof of the Lemma it is enough to take $q = q_0$, $Z = Z_0$, $\alpha = \Lambda$, and $p = q' = q_f$ in the previous claim. Q.E.D.

At last, we can state one of the main results of this section.

THEOREM 3.3. *Δ_2 is a subfamily of the family of all strict deterministic languages.*

Proof. This theorem is now a direct consequence of our lemmas and the definition of canonical grammar: Let $L \in \Delta_2$. Then $L = T_2(M)$ for some DPDA M. Then by Lemma 3.1 $L = T_2(\hat{M})$ for some DPDA $\hat{M}$ with a single final state. Now the canonical grammar $G_{\hat{M}}$ is strict deterministic by Lemma 3.2 and $L = L(G_{\hat{M}})$ by Lemma 3.3. Thus, L is strict deterministic language. Q.E.D.

The rigorous proof of the converse to Theorem 3.3 consists of the formalization of a parsing algorithm for strict deterministic grammars on a DPDA. The parsing method is presented in more intuitive form in [9]. Because of space limitations, we we give only the DPDA construction here. The following definition of a canonical DPDA for a given grammar is not optimal from the point of view of memory utilization, but it has the advantage that its state set is in a natural correspondence with the strict partition of the grammar (this correspondence will be exploited later in Section 4).

First we introduce certain notational conventions which will simplify our formalism. Let

$$G = \langle V, \Sigma, P, S \rangle \tag{13}$$

be a grammar with the strict partition

$$\pi = \{\Sigma, V_0, V_1, \ldots, V_m\}, \tag{14}$$

where $m \geqslant 0$ and $V_0 = [S]$. We use special indexed symbols A_{ij} for the nonterminals of G so that for all i $(0 \leqslant i \leqslant m)$ we have

$$V_i = \{A_{i0}, A_{i1}, \ldots, A_{in_i}\}, \tag{15}$$

where $n_i = |V_i| - 1$. (Note that $\max_i n_i = \|\pi\|$.) Moreover, let $A_{00} = S$.

DEFINITION 3.6. Let G be a grammar of the form (13) with strict partition π for which we use the notation from (14) and (15). We define the *canonical DPDA* M_G *for* G as follows.

$$M_G = \langle Q, \Sigma, \Gamma, \delta, q_0, Z_0, \{q_0\} \rangle, \tag{16}$$

where

$$Q = \{q_j \mid 0 \leqslant j < \|\pi\|\},$$

$$\Gamma = \Gamma_1 \cup \Gamma_2,$$

$$\Gamma_1 = \{\overline{V_i, \alpha} \mid A \to \alpha\beta \text{ for some } A \in V_i \text{ and } \alpha, \beta \in V^*\},$$

$$\Gamma_2 = \{\overline{V_i, \alpha, V_j} \mid A \to \alpha B\beta \text{ for some } A \in V_i, B \in V_j \text{ and } \alpha, \beta \in V^*\},$$

$$Z_0 = \overline{[S], \Lambda} = \overline{V_0, \Lambda} \in \Gamma_1,$$

and δ is defined by means of four types of moves as follows. For any $V_i, V_k \in \pi - \{\Sigma\}$, $\alpha \in V^*$, $a \in \Sigma$, and $q_j \in Q$, we have

Type 1. $\delta(q_0, \Lambda, \overline{V_i, \alpha}) = (q_0, \overline{V_i, \alpha, V_k}\ \overline{V_k, \Lambda})$ if $A \to \alpha B\beta$ is in P for some $A \in V_i$, $B \in V_k$ and $\beta \in V^*$;

Type 2. $\delta(q_0, a, \overline{V_i, \alpha}) = (q_0, \overline{V_i, \alpha a})$ if $A \to \alpha a\beta$ is in P for some $A \in V_i$ and $\beta \in V^*$;

Type 3. $\delta(q_0, \Lambda, \overline{V_i, \alpha}) = (q_j, \Lambda)$ if $A_{ij} \to \alpha$ is in P;

Type 4. $\delta(q_j, \Lambda, \overline{V_k, a, V_i}) = (q_0, \overline{V_k, \alpha A_{ij}})$.

Otherwise δ is not defined. Moves of type 1, 2 and 3 are called *detection moves*, the move of type 4 is called the *reduction move*.
First we have to verify that our definition is correct.

LEMMA 3.4. *For any strict deterministic grammar G the object M_G defined in Definition* 3.6 *is a well defined* DPDA.

We have to show that δ is single valued and satisfies (3).
Assume the following two moves:

$$\begin{array}{ccc} (q, a, Z) & & (q, a', Z) \\ \downarrow{\scriptstyle \delta} & & \downarrow{\scriptstyle \delta} \\ (p, \alpha) & \neq & (p', \alpha'). \end{array} \tag{17}$$

We shall show that $a \neq a'$ (hence, the single valuedness), and, moreover, $a, a' \in \Sigma$ (hence, (3)).

Case 1. Both moves in (17) are of the same type in the terminology of Definition 3.6. Then either $a = a' = \Lambda$ or $a, a' \in \Sigma$. Suppose $a = a'$. Then we obtain $(p, \alpha) = (p', \alpha')$ immediately for type 2 and 4 or using the strictness of π for types 1 and 3 (cf. Definition 2.3). Hence, $a \neq a'$ and $a, a' \in \Sigma$.

Case 2. Two moves in (17) are of different types. Then neither of them is type 4 since that is the only type with $Z \in \Gamma_2$. Assume, therefore, $q = q_0$, $Z = \overline{V_i, \alpha}$. Now moves of type 1 and 2 are incompatible since $B \equiv a$ is not possible for $B \in N$ and $a \in \Sigma$, and neither type 1 nor type 2 is compatible with type 3 since $A \equiv A'$, $A \to \alpha\beta$, and $A' \to \alpha$ implies $\beta = \Lambda$ by the strictness of π. Hence, $a \neq a'$. Q.E.D.

The following result is intuitively obvious but the formal proof is rather lengthy.

LEMMA 3.5. *Let M_G be the canonical* DPDA *for a strict deterministic grammar G. Then $T_2(M_G) = L(G)$.*

Proof. Let us partition the yield relation $\vdash$ in accordance with the four possible moves in Definition 3.6:

$$\begin{aligned} &(q, aw, \gamma Z) \vdash_i (q', w, \gamma\alpha) \text{ iff} \\ &\delta(q, a, Z) = (q', \alpha) \text{ is a move of type } i,\ i = 1, 2, 3, 4. \end{aligned} \tag{18}$$

Thus $\vdash = \bigcup_{i=1}^{4} \vdash_i$. Moreover, let us define

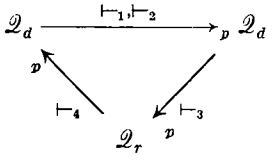
$$\models = (\vdash_1 \cup \vdash_2)^* \vdash_3 . \tag{19}$$

(Intuitively, if we interpret M_G as a parsing algorithm, $\models$ is the *detection phase* and $\vdash_4$ the *reduction phase* of the algorithm.)

Let $\mathcal{Q}$ be the set of configurations of M_G and define two sets $\mathcal{Q}_d$, $\mathcal{Q}_r \subseteq \mathcal{Q}$ as follows.

$$\begin{aligned} \mathcal{Q}_d &= Q \times \Sigma^* \times \Gamma_2^* \Gamma_1 , \\ \mathcal{Q}_r &= Q \times \Sigma^* \times \Gamma_2^* . \end{aligned}$$

Using Definition 3.6 we can interpret relations $\vdash_i$ as partial functions ($\vdash$ is single valued) as follows:



(20)

Now (19) and (20) imply

$$\mathcal{Q}_d \xrightarrow{(\models \vdash_4)^* \models}_p \mathcal{Q}_r .$$

We have the following.

FACT. *If* $c_1 \in \mathcal{Q}_d$ *and* $c_2 \in \mathcal{Q}_r$ *then*

$$c_1 \vdash^* c_2 \qquad \text{implies} \quad c_1(\models \vdash_4)^* \models c_2 . \tag{21}$$

Further we will need the following claim about the contents of the store of M_G.

CLAIM 1. *For any* $(q, w, \gamma), (q', w', \gamma') \in \mathcal{Q}_r$ *(i.e.,* $\gamma, \gamma' \in \Gamma_2^*$*) if* $(q, w, \gamma) \vdash^+ (q', w', \gamma')$ *then* γ *is not a prefix of* γ'.

Proof of the claim. The argument is an induction on the length of γ. (For convenience we use the notation $\gamma \vdash \gamma'$ as an abbreviation for $(q, w, \gamma) \vdash (q', w', \gamma')$ for some $q, q' \in Q$ and $w, w' \in \Sigma^*$.)

Basis. The case $\gamma = \Lambda$ is vacuously true since $\vdash$ is in this case undefined.

Inductive step. Assume the claim proved for all γ_1 shorter than some $\gamma \in \Gamma_2^+$. Now let

$$\gamma = \gamma_1 \overline{V, \alpha, V'} \vdash^+ \gamma',$$

for some $\overline{V, \alpha, V'} \in \Gamma_2$. Then by Definition 3.6 $\gamma_1 \overline{V, \alpha, V'} \vdash_4 \gamma_1 \overline{V, \alpha A} \vdash^* \gamma'$ for some $A \in V$. Now αA in $\overline{V, \alpha A}$ cannot be changed back to α in $\overline{V, \alpha, V'}$ without being erased, i.e., without an application of a type 3 move. Hence, a necessary condition for γ being a prefix of γ' is

$$\gamma_1 \overline{V, \alpha, V'} \vdash^* \vdash_3 \gamma_1 \vdash^* \gamma'.$$

Since $\gamma_1 \neq \gamma'$ we have $\gamma_1 \vdash^+ \gamma'$. By the inductive hypothesis γ_1 is not a prefix of γ', and, hence, γ is not also. This completes the proof of claim 1.

The following two claims are crucial in our proof.

CLAIM 2. *Let $w \in \Sigma^*$ and $i, j \geqslant 0$. If $A_{ij} \Rightarrow^+ w$ then for any $y \in \Sigma^*$ and $\gamma \in \Gamma_2{}^*$*

$$(q_0, wy, \gamma\overline{V_i, \Lambda})(\models \vdash_4)^* \models (q_j, y, \gamma).$$

Let $c_1 = (q_0, wy, \gamma\overline{V_i, \Lambda})$ and $c_2 = (q_j, y, \gamma)$. Assume $A_{ij} \Rightarrow^n w$. The argument is an induction on $n \geqslant 1$.

Basis. $n = 1$. Then $A_{ij} \to w$ and if $w \neq \Lambda$ then for every factorization $w = w_1 a w_2$ ($w_1, w_2 \in \Sigma^*$ and $a \in \Sigma$), we have

$$(q_0, aw_2y, \gamma\overline{V_i, w_1}) \vdash_2 (q_0, w_2y, \gamma\overline{V_i, w_1a}),$$

and in any event we subsequently have

$$(q_0, y, \gamma\overline{V_i, w}) \vdash_3 (q_j, y, \gamma) = c_2.$$

Thus, $c_1 \vdash_2^* \vdash_3 c_2$ or, using (19), $c_1 \models c_2$.

Inductive step. Let $n > 1$ and assume the result proved for all $n' < n$. Let $\alpha \in V^*$ such that

$$A_{ij} \Rightarrow \alpha \Rightarrow^+ w.$$

For any prefix α_1 of α we have one of the following three cases.

Case 1. α_1 is followed by $a \in \Sigma$, i.e., $\alpha = \alpha_1 a \alpha_2$. Then for some suffix w_2 of w, we have $\alpha_2 \Rightarrow^* w_2$ and

$$(q_0, aw_2y, \gamma\overline{V_i, \alpha_1}) \vdash_2 (q_0, w_2y, \gamma\overline{V_i, \alpha_1 a}).$$

Case 2. α_1 is followed by $B \in N$, i.e., $\alpha = \alpha_1 B \alpha_2$. Then for some suffix $w_B w_2$ of w we have $B \Rightarrow^{n'} w_B$ ($n' < n$), $\alpha_2 \Rightarrow^* w_2$ and

$$\begin{aligned}&(q_0, w_B w_2 y, \gamma\overline{V_i, \alpha_1})\\ &\quad\vdash_1 (q_0, w_B w_2 y, \gamma\overline{V_i, \alpha, [B]\ \overline{[B], \Lambda}})\\ &\quad(\models \vdash_4)^* \models (q_k, w_2y, \gamma\overline{V_i, \alpha_1, [B]})\end{aligned}$$

by the inductive hypothesis from $B \Rightarrow^{n'} w_B$, if $B = A_{ik} \in [B]$

$$\vdash_4 (q_0, w_2y, \gamma\overline{V_i, \alpha_1 B}).$$

Case 3. $\alpha_1 = \alpha$. Then

$$(q_0, y, \gamma\overline{V_i, \alpha}) \vdash_3 (q_j, y, \gamma) = c_2.$$

Now starting with configuration c_1, i.e., $\alpha_1 = \Lambda$, we can combine cases 1 and 2 depending on the form of $\alpha \in (\Sigma \cup N)^*$, until we finish with case 3, i.e., $\alpha_1 = \alpha$. Therefore,

$$c_1(\vdash_2 \cup \vdash_1(\models \vdash_4)^* \models \vdash_4)^* \vdash_3 c_2 ;$$

hence,

$$c_1(\vdash_2^* \vdash_1(\models \vdash_4)^* \models \vdash_4)^* \vdash_2^* \vdash_3 c_2 ,$$

using the identity $(\rho \cup \sigma)^* = (\rho^* \sigma)^* \rho^*$ (cf. [13, p. 115, Eq. 1.15]); hence,

$$c_1(\models \vdash_4(\models \vdash_4)^*)^* \models c_2$$

using (19) and the identity $\rho^* \rho = \rho \rho^*$; hence,

$$c_1(\models \vdash_4)^* \vdash c_2 ,$$

using the identity $(\rho\rho^*)^* = \rho^{**} = \rho^*$. This proves claim 2.

Our last claim is essentially the converse of claim 2.

CLAIM 3. *Let* $w, y \in \Sigma^*$, $\gamma \in \Gamma_2^*$ *and* $i, j \geqslant 0$. *If*

$$(q_0 , wy, \gamma\overline{V_i , \Lambda}) \vdash^* (q_j , y, \gamma) \qquad \textit{then} \quad A_{ij} \Rightarrow^+ w.$$

Again let $c_1 = (q_0 , wy, \gamma\overline{V_i , \Lambda})$, $c_2 = (q_j , y, \gamma)$, and let $c_1 \vdash^* c_2$. By (21) $c_1(\models \vdash_4)^n \models c_2$ for some $n \geqslant 0$. Then by Definition 3.6 and (19)

$$c_1(\models \vdash_4)^n (\vdash_1 \cup \vdash_2)^* c_3 \vdash_3 c_2 , \tag{22}$$

where $c_3 = (q_0 , y, \gamma\overline{V_k , \alpha})$. First we show that $V_k = V_i$. Indeed, if $V_k \neq V_i$ we would have (using the notation from the proof of claim 1)

$$\gamma\overline{V_i , \Lambda} \vdash^* \gamma\overline{V_i , \beta} \vdash_3 \gamma \vdash^* \vdash_1 \gamma\overline{V_k , \Lambda} \vdash^* \gamma\overline{V_k , \alpha} \vdash_3 \gamma,$$

for some $\beta \in V^*$ since this is the only way of replacing V_i by V_k in agreement with Definition 3.6. But here $\gamma \vdash^+ \gamma$ is contrary to claim 1. Therefore, $V_i = V_k$ and from $c_3 \vdash_3 c_2$ we have $A_{ij} \to \alpha$.

We proceed by induction on n in (22).

Basis. $n = 0$. Then $c_1(\vdash_1 \cup \vdash_2)^* c_3$. Moreover, no move of type 1 can occur (since that would increase the length of γ). Thus, $c_1 \vdash_2^* c_3 \vdash_3 c_2$. This is possible only if all letters in α are terminal or if $c_1 = c_3$ which occurs if $\alpha = \Lambda$. Hence, $\alpha \in \Sigma^*$, $\alpha = w$, and $A_{ij} \to w$.

Inductive step. Let $n > 1$ and assume the result proved for all $n' < n$. From (22), Definition 3.6, and $A_{ij} \to \alpha$, we conclude that for every prefix α_1 of α there is a unique configuration $c = (q_0, w_2 y, \gamma \overline{V_i, \alpha_1})$ such that $c_1 \vdash^* c \vdash^* c_2$ (the uniqueness follows from claim 1). Here w_2 is a suffix of w. Let $w = w_1 w_2$ and $w_1 \in \Sigma^*$. First we show that $\alpha_1 \Rightarrow^* w_1$ by induction on the length of α_1. This is immediate if $\alpha_1 = \Lambda \Rightarrow^* \Lambda = w_1$. Assume that $\alpha_1 \Rightarrow^* w_1$ was already proved and let $\alpha = \alpha_1 B \alpha_2$ ($B \in V$ and $\alpha_2 \in V^*$). Let us consider two configurations,

$$c = (q_0, w_2, \gamma \overline{V_i, \alpha_1})$$

and

$$c' = (q_0, w_2', \gamma \overline{V_i, \alpha_1 B}).$$

There are two cases.

Case 1. $B = a \in \Sigma$. Then by Definition 3.6 $c \vdash_2 c'$, $w_2 = a w_2'$, and $\alpha_1 a \Rightarrow^* w_1 a$ is a triviality

Case 2. $B \in N$. Then we have

$$\begin{aligned} c &\vdash_1 (q_0, w_2 y, \gamma \overline{V_i, \alpha_1, [B]}\ \overline{[B], \Lambda}), \\ &\vdash^* (q_k, w_2' y, \gamma \overline{V_i, \alpha_1, [B]}), \\ &\vdash_4 c', \end{aligned}$$

assuming that $B = A_{ik} \in [B]$. From (21) we see that the relation $\vdash^*$ can be replaced by $(\models \vdash)^{n'} \models$ for some $n' < n$. Moreover, w_2' is a suffix of w_2, say $w_2 = w_1' w_2'$. Then by the inductive hypothesis $B \Rightarrow^+ w_1'$, and, therefore, $\alpha_1 B \Rightarrow^+ w_1 w_1'$. This ends the inductive proof that $\alpha_1 \to^+ w_1$.

Now, for $\alpha_1 = \alpha$ we have $A_{ij} \Rightarrow \alpha \Rightarrow^* w'$, where w' is a prefix of w. We shall show that $w' = w$. For, assume $w = w'w''$. By claim 2 then

$$c_1 = (q_0, w'w'' y, \gamma \overline{V_i, \Lambda}) \vdash^* (q_j, w'' y, \gamma).$$

But then, to avoid $\gamma \vdash^+ \gamma$ which would contradict claim 1, we must have $(q_j, w'' y, \gamma) = c_2$, and, hence, $w'' = \Lambda$. We conclude $A_{ij} \Rightarrow^+ w$, as asserted. This finishes the proof of claim 3.

Now, to conclude the proof of the lemma we combine claims 2 and 3 and substitute $y = \gamma = \Lambda$, $i = j = 0$ (note that $S = A_{00}$ by convention). We obtain that for any $w \in \Sigma^*$

$$S \Rightarrow^+ w \qquad \text{iff} \quad (q_0, w, \overline{V_0, \Lambda}) \vdash^* (q_0, \Lambda, \Lambda),$$

and, therefore, $w \in L(G)$ iff $w \in T_2(M_G)$. Q.E.D.

We can immediately apply the previous lemma.

THEOREM 3.4. *The family of all strict deterministic languages is a subfamily of* Δ_2.

Proof. The theorem is a direct consequence of the definition of canonical DPDA and of Lemmas 3.4 and 3.5. Q.E.D.

Now, combining Theorems 3.2, 3.3, and 3.4, we obtain one of our main results.

THEOREM 3.5. *The family of strict deterministic languages coincides with the family of prefix-free deterministic languages.*

4. A HIERARCHY OF STRICT DETERMINISTIC LANGUAGES DEFINED BY THEIR DEGREES

In this last section, we show that there is a natural hierarchy of strict deterministic languages as defined by their degree. The degree turns out to be the number of states in the appropriate DPDA and the correspondence holds in the reverse direction as well.

Let M be DPDA of the form

$$M = \langle Q, \Sigma, \Gamma, \delta, q_0, Z_0, F \rangle, \tag{1}$$

as defined in Section 3. We shall consider the number $| Q |$ of states of M. In Section 2 we defined, for a given strict partition π, the number $\| \pi \|$ as the maximal size of the nonterminal blocks in π,

$$\| \pi \| = \max_{V_i \in \pi - \{\Sigma\}} | V_i |. \tag{2}$$

We shall now relate this number to the number of states of the corresponding DPDA.

THEOREM 4.1. *Let L be any language and let $n \geqslant 1$. Then $L = L(G)$ for some strict deterministic grammar with partition π such that $\| \pi \| = n$* iff *$L = T_2(M)$ for some* DPDA *M with n states.*

Proof. The result is immediate in the "only if" direction from Theorem 3.4 and from the fact that the canonical automaton has, by definition, $\| \pi \|$ states. For the "if" direction we first note that the construction of DPDA $\hat{M}$ with $| F | = 1$ in the proof of Lemma 3.1 does not change the number of states. The rest then follows from the proof of Theorem 3.3, in particular, that the partition π used for proving the strict determinism of the canonical grammar (cf. Lemma 3.2) has the property that $\| \pi \| = | Q |$. Q.E.D.

Let us recall from Section 2 (cf. Remark following Theorem 2.4) that every strict deterministic grammar has a unique partition π_0 which is the minimal element in the

semilattice of all strict partitions of G. Also, $\| \pi_0 \| \leqslant \| \pi \|$ for any other strict partition of G.

DEFINITION 4.1. Let G be a strict deterministic grammar. We define the *degree of* G as the number

$$deg(G) = \| \pi_0 \| ,$$

where π_0 is the minimal strict partition for G. For any language $L \in \Delta_2$ define its degree as follows:

$$deg(L) = \min\{deg(G) \mid G \text{ is strict deterministic and } L(G) = L\}.$$

It is interesting to note that the degree of a grammar is invariant under certain transformations. Observing that neither the elimination of useless productions nor the transformation used in proof of Theorem 2.1 changes the maximal cardinality of blocks of strict partitions, we can strengthen our previous results.

FACT. *For any strict deterministic grammar G there is an equivalent reduced and (unless $L(G) = \{\Lambda\}$) Λ-free grammar with the same degree.*

Our next objective will be to show that strict deterministic languages form a nontrivial hierarchy with respect to their degree—or, equivalently, with respect to the minimal number of states of the corresponding DPDA.[12] This result can be intuitively explained by pointing out that the main reason for having more states in a DPDA is when a greater amount of information from the top part of the store has to be preserved while any letters beneath it are read. This consideration leads us to an example which will serve as a basis for the formal proof of the result.

LEMMA 4.1. *Let $\Sigma = \{a, b\}$. Define for any $n \geqslant 1$ the language*

$$L_n = \{a^m b^k a^m b^k \mid 1 \leqslant m,\ 1 \leqslant k \leqslant n\}.$$

Then $L_n = T_2(M)$ for some DPDA *M with n states.*

Proof. Let M_n be a DPDA of the form (1) where

$$Q = \{q_0, \ldots, q_{n-1}\}, \qquad \Gamma = \{Z_0, 0, 1, 2, 3\}, \qquad F = \{q_0\}$$

[12] Let us mention for comparison that any context-free language can be defined as $T_2(M)$ or $T_0(M)$ of some PDA with one or two states, respectively (cf. [2]).

and δ is defined as follows.

$$\begin{aligned}
\delta(q_0, a, Z_0) &= (q_0, 02), \\
\delta(q_0, a, 2) &= (q_0, 12), \\
\delta(q_0, b, 2) &= (q_0, 3), \\
\delta(q_i, b, 3) &= (q_{i+1}, 3) \qquad \text{for} \quad 0 \leqslant i < n-1, \\
\delta(q_i, a, 3) &= (q_i, \Lambda) \qquad \text{for} \quad 0 \leqslant i \leqslant n-1, \\
\delta(q_i, a, 1) &= (q_i, \Lambda) \qquad \text{for} \quad 0 \leqslant i \leqslant n-1, \\
\delta(q_i, b, 0) &= (q_{i-1}, 0) \qquad \text{for} \quad 0 < i \leqslant n-1, \\
\delta(q_0, b, 0) &= (q_0, \Lambda).
\end{aligned}$$

To show that $T_2(M_n) = L_n$ it is enough to observe that a computation leads to acceptance of a string $w \in \Sigma^*$ iff it has the following form [13] for some $m, k \geqslant 1, k \leqslant n$:

$$\begin{aligned}
(q_0, w, Z_0) &\vdash (q_0, a\backslash w, 02) \vdash^{m-1} (q_0, a^m\backslash w, 01^{m-1}2) \\
&\vdash (q_0, a^m b\backslash w, 01^{m-1}3) \vdash^{k-1} (q_{k-1}, a^m b^k\backslash w, 01^{m-1}3) \\
&\vdash (q_{k-1}, a^m b^k a\backslash w, 01^{m-1}) \vdash^{m-1} (q_{k-1}, a^m b^k a^m\backslash w, 0) \\
&\vdash^{k-1} (q_0, a^m b^k a^m b^{k-1}\backslash w, 0) \vdash (q_0, a^m b^k a^m b^k\backslash w, \Lambda),
\end{aligned}$$

where all the quotients are defined and $a^n b^k a^n b^k = w$. Q.E.D.

Next we shall prove that n states are necessary for acceptance of L_n (defined in Lemma 4.1). The formal proof is by no means trivial since we have to prove this for all conceivable DPDA. First we need one general definition and a lemma from [3].

DEFINITION 4.2. Let M be a DPDA of the form (1). We define a relation [14] $\Vdash^* \subseteq \vdash^*$ as follows. For all $q, q' \in Q$, $w, w' \in \Sigma^*$, and $\gamma, \gamma' \in \Gamma^*$
$(q, w, \gamma) \Vdash^* (q', w', \gamma')$ iff

(i) $(q, w, \gamma) \vdash^* (q', w', \gamma')$; and

(ii) $\delta(q', \Lambda, \gamma'^{(1)})$ is undefined (in particular, if $\gamma' = \Lambda$).

A DPDA M is said to be *loop-free* iff for every $w \in \Sigma^*$ there exist $q \in Q$ and $\gamma \in \Gamma^*$ such that $(q_0, w, Z_0) \Vdash^* (q, \Lambda, \gamma)$.

LEMMA 4.2. *Let M be a loop-free* DPDA *of the form* (1) *and let $a \in \Sigma$. Then either*

1. *there exists $n_0 \geqslant 1$ such that for any $q \in Q$, $\gamma \in \Gamma^*$, and $m \geqslant 1$, if*

$$(q_0, a^m, Z_0) \vdash^* (q, \Lambda, \gamma)$$

then $\lg(\gamma) \leqslant n_0$; *or*

[13] Cf. (9) in Section 1 for the definition of quotient.

[14] The symbol $\vdash^{d*}$ is used for $\models^*$ in [3].

2. *there exist* $m_0, f \geqslant 1, q \in Q, \gamma_0 \in \Gamma^*, \eta \in \Gamma^+$, *and* $Z \in \Gamma$ *such that for every* $h \geqslant 0$

$$(q_0, a^{m_0+hf}, Z_0) \vdash^* (q, \Lambda, \gamma_0\eta^h Z) \tag{3}$$

and for any $k \geqslant 0$ *and* $\gamma \in \Gamma^*$

$$(q, a^k, \gamma_0\eta^h Z) \vdash^* (q', \Lambda, \gamma) \qquad \textit{implies} \quad \gamma = \gamma_0\eta^h\gamma', \tag{4}$$

for some $\gamma' \in \Gamma^+$.

This lemma is proved in [3, p. 642]. The present formulation differs from [3] (besides the notation) only by the requirement in 2. that $\eta \in \Gamma^+$ instead of $\eta \in \Gamma^*$. This is completely justifiable since if $\eta = \Lambda$ then 2. is reduced to 1.

LEMMA 4.3. *Let* L_n *be the language from Lemma* 4.1 *for some* $n \geqslant 1$ *and let* M *be* DPDA *such that* $T_2(M) = L_n$. *Then* M *has a least* n *states.*

Proof. Let $M = \langle Q, \Sigma, \Gamma, \delta, q_0, Z_0, F\rangle$ be a DPDA satisfying the assumptions of the lemma and assume that $|Q| < n$. We assume $n \geqslant 2$ since for $n = 1$ the lemma is trivial. Our strategy will be to prove a sequence of claims about M which will eventually lead to a contradiction. We shall investigate an accepting computation of the form

$$(q_0, a^m b^k a^m b^k, Z_0) \vdash^* (q_f, \Lambda, \Lambda), \tag{5}$$

where $1 \leqslant m$, $1 \leqslant k \leqslant n$, and $q_f \in F$. Informally, we distinguish four phases of computation in (5) in a natural way: in each phase one block of the same letter (a^m or b^k) is read. A typical history of the store is illustrated in Fig. 2.

First we show that the content of the store after the first phase is periodic.

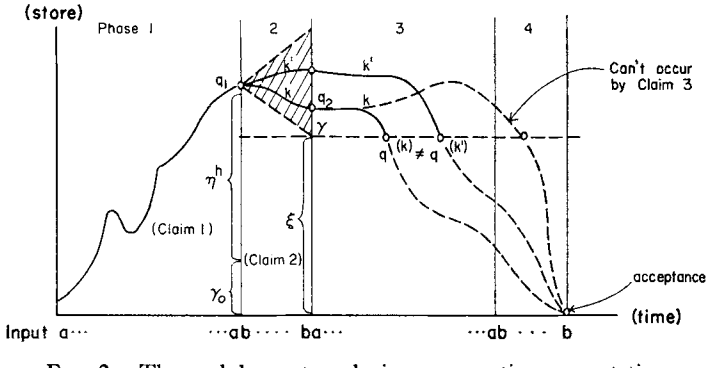


FIG. 2. The pushdown store during an accepting computation.

CLAIM 1. *There exist* m_0, $f \geqslant 1$, $q_1 \in Q$, $\gamma_0 \in \Gamma^*$, $\eta \in \Gamma^+$, *and* $Z_1 \in \Gamma$ *such that for every* $h \geqslant 0$

$$(q_0, a^{m_0+hf}, Z_0) \vdash^* (q_1, \Lambda, \gamma_0\eta^h Z_1). \tag{6}$$

Proof of the claim. To be able to apply Lemma 4.2 we need the loop-free property. Consider, therefore, a DPDA, $M_a = \langle Q, \{a\}, \Gamma, \delta_a, q_0, Z_0, F\rangle$, obtained from M by restriction of input alphabet to $\{a\}$, i.e., for all $q \in Q$ and $Z \in \Gamma$

$$\delta_a(q, a', Z) = \begin{cases} \delta(q, a', Z) & \text{if } a' \in \{a, \Lambda\}; \\ \text{undefined} & \text{if } a' = b. \end{cases}$$

Clearly, for any $m \geqslant 0$, $q \in Q$, and $\gamma \in \Gamma^*$

$$(q_0, a^m, Z_0) \vdash^*_{M_a} (q, \Lambda, \gamma) \quad \text{iff} \quad (q_0, a^m, Z_0) \vdash^*_M (q, \Lambda, \gamma). \tag{7}$$

For any $m \geqslant 1$, since a^m is a prefix of an acceptable string (e.g., $a^m b a^m b$) there exist $q \in Q$ and $\gamma \in \Gamma^*$ such that $(q_0, a^m, Z_0) \vdash^*_M (q, \Lambda, \gamma)$. Hence, by (7) M_a is loop-free. We shall now apply Lemma 4.2 to M_a. For any $m \geqslant 1$, there must be a different configuration (q, Λ, γ) in (7) (otherwise we would have $a^{m'}b^k a^m b^k \in T_2(M)$ for some $m' \neq m$). Now since Q and Γ are finite but m may be arbitrary, there is no bound on the length of γ. Thus, we cannot have case 1 in Lemma 4.2, and we obtain claim 1 as a consequence of (3).

For the rest of the proof we fix symbols m_0, f, γ_0, q_1, and Z_1 with the same meaning as in Claim 1. Our next claim is that during the second phase (during the reading of the b^k) only a bounded number of η's are erased from the store.

CLAIM 2. *Let* $1 \leqslant k \leqslant n$ *and* $h \geqslant 0$. *Then*

$$(q_1, b^k, \gamma_0\eta^h Z_1) \vdash^* (q_2, \Lambda, \gamma_0\eta^{h-s}\gamma)$$

for some $q_2 \in Q$, $\gamma \in \Gamma^*$ *and* $s \leqslant n^2$.

Proof of the claim. Let $q_2 \in Q$ and $\gamma' \in \Gamma^*$ be such that

$$c_1 = (q_1, b^k, \gamma_0\eta^h Z_1) \vdash^* (q_2, \Lambda, \gamma'). \tag{8}$$

Such q_2 and γ' always exist since otherwise M could not accept a string with the prefix $a^{m_0+hf}b^k$ (note (6)). It is enough to show that $\gamma_0\eta^{h-s}$ is a prefix of γ' for some $s \leqslant n^2$.

Assume, for the sake of contradiction, that this is not the case, i.e., that $\gamma_0\eta^{h-n^2}$ is not a prefix of γ'. However, it is a prefix of $\gamma_0\eta^h Z_1$ in c_1, and, therefore, there exists a configuration c_1' such that (8) can be written in the form

$$c_1 \vdash^* c_1' \vdash^* (q_2, \Lambda, \gamma'),$$

and the contents of the store in c_1' is exactly $\gamma_0\eta^{h-n^2}$ (cf. Fig. 3). We shall concentrate on those transitions in $c_1 \vdash^* c_1'$ which are associated with erasing a letter from the original contents of the store (i.e., a letter written before configuration c_1 was entered; cf. thick line on Fig. 3). Let us call these transitions *proper erasing*. Since at most one pushdown letter can be erased at a time, there are, at least, $\lg(\eta^{n^2}Z_1) = n^2 \cdot \lg(\eta) + 1$ proper erasing transitions in $c_1 \vdash^* c_1'$. At most k of the erasing transitions correspond to non-Λ-moves of M since only k input letters are read in (8). Consider the sequence of consecutive transitions corresponding to Λ-moves (we call them *Λ-transitions*) in $c_1 \vdash^* c_1'$ which contain the maximal number $N_{\max}$ of proper erasing transitions. Since $k \leqslant n$, $\lg(\eta) \geqslant 1$ and using the fact that c_1 can be followed only by a non-Λ-

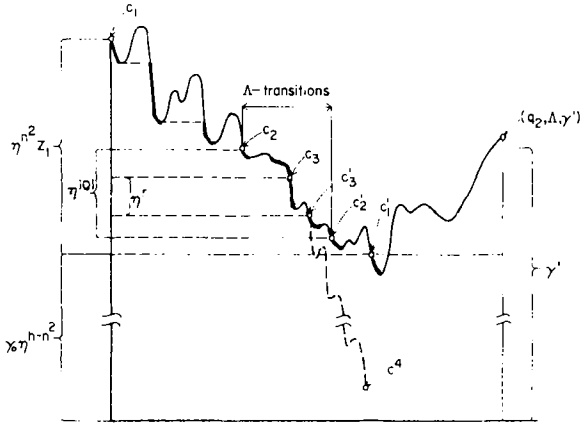


FIG. 3. Phase 2 of the accepting computation.

transition (a consequence of (6)) we obtain the inequality $N_{\max} \geqslant n \cdot \lg(\eta)$. Now any suffix of $\gamma_0\eta^h Z_1$ of length $N_{\max}$ contains, at least, $n - 1$ repetitions of η or, using the assumption that $|Q| < n$, it contains, at least, $|Q|$ repetitions of η. Consequently, there exist two configurations c_2 and c_2' such that $c_1 \vdash^* c_2 \vdash^+ c_2' \vdash^* c_1'$, where $c_2 \vdash^+ c_2'$ consists entirely of Λ-transitions and exactly $|Q|$ repetitions of η are erased from the store during $c_2 \vdash^+ c_2'$ (cf. Fig. 3). Formally, we have for some p, $p' \in Q$, $1 \leqslant j \leqslant k$, and $i \geqslant |Q|$

$$c_1 \vdash^* c_2 = (p, b^j, \gamma_0\eta^{h-i}\eta^{|Q|}) \vdash^t (p', b^j, \gamma_0\eta^{h-i}) = c_2', \tag{9}$$

where $t \geqslant |Q| \cdot \lg(\eta)$. Therefore, at least, two distinct configurations occur in $c_2 \vdash^t c_2'$ such that both have the same state and the same topmost string η in the store. In other words, for some $p'' \in Q$ and $1 \leqslant r \leqslant |Q|$,

$$c_2 \vdash^* c_3 = (p'', b^j, \gamma_0\eta^{h-i}\eta^r) \vdash^+ (p'', b^j, \gamma_0\eta^{h-i}\eta^s) = c_3' \vdash^* c_2', \tag{10}$$

where $s < r$. But this introduces a loop, and (10) must repeat until almost all the periodic part of the store is erased:

$$c_3' \vdash^* (p'', b^j, \gamma_0\eta^{r_0}) = c_4, \tag{11}$$

where $r_0 \leqslant r$. Now let $h' = h + r$. Then

$$\begin{aligned}(q_0, a^{m_0+h'f}b^k, Z_0) &\vdash^* (q_1, b^k, \gamma_0\eta^{h'}Z_1) \text{ by claim 1,}\\ &\vdash^* (p'', b^j, \gamma_0\eta^{h'-i}\eta^r) \text{ by (9) and (10),}\\ &\vdash^* (p'', b^j, \gamma_0\eta^{h-i}\eta^r) = c_3 \vdash^* c_4 \text{ by (10)}\\ &\quad \text{and (11), using } h' = h + r.\end{aligned}$$

Let $m = m_0 + hf$ and $m' = m_0 + h'f \neq m$. We have

$$(q_0, a^m b^k a^m b^k, Z_0) \vdash^* (p'', b^j a^m b^k, \gamma_0\eta^{r_0}) \vdash^* (q_f, \Lambda, \Lambda)$$

on the one hand, and

$$(q_0, a^{m'} b^k a^m b^k, Z_0) \vdash^* (p'', b^j a^m b^k, \gamma_0\eta^{r_0}) \vdash^* (q_f, \Lambda, \Lambda),$$

on the other hand, contradicting that $m' \neq m$ and $a^{m'}b^k a^m b^k \notin L_n = T_2(M)$. Thus, $\gamma_0\eta^{h-n^2}$ is a prefix of γ' which concludes the proof of claim 2.

Thus, for any $h \geqslant n^2$, $m = m_0 + hf$, and for any k, $1 \leqslant k \leqslant n$, we have a common prefix $\xi = \gamma_0\eta^{h-n^2}$ of any possible contents of the pushdown store after the first two phases of the accepting computation

$$(q_0, a^m b^k, Z_0) \vdash^* (q_2, \Lambda, \xi\gamma), \tag{12}$$

where $q_2 \in Q$ and $\gamma \in \Gamma^*$. Here γ is dependent only on k (cf. Fig. 2).

We shall turn our attention to the last two phases of the computation (5).

CLAIM 3. *Let $1 \leqslant k \leqslant n$ and $m \geqslant 1$. There exist $m_k \leqslant m$ and $q^{(k)} \in Q$ such that*

$$(q_2, a^{m_k}, \xi\gamma) \vdash^* (q^{(k)}, \Lambda, \xi), \tag{13}$$

where q_2, ξ, γ are the same as in (12).

Proof of the claim. We have $(q_2, a^m b^k, \xi\gamma) \vdash^* (q_f, \Lambda, \Lambda)$ since $a^m b^k a^m b^k \in T_2(M)$. Therefore, for some prefix w of $a^m b^k$ we have also $(q_2, w, \xi\gamma) \vdash^* (q^{(k)}, \Lambda, \xi)$ for some $q^{(k)} \in Q$. It is enough to show that $\lg(w) \leqslant m$. Suppose the converse, i.e., $w = a^m b^j$ for some $1 \leqslant j \leqslant k$. Then we have

$$(q^{(k)}, b^{k-j}, \xi) \vdash^* (q_f, \Lambda, \Lambda). \tag{14}$$

Here $\xi = \gamma_0 \eta^{h-n^2}$ may be an arbitrarily long string since h is arbitrarily large. An argument similar to the proof of claim 2 leads us to the conclusion that (14) contains a subsequence of Λ-transitions erasing more than $|Q| \cdot (k-j)$ occurrences of η in ξ and, therefore, erasing almost all ξ; but since ξ was not changed during the second and third phases we would have, as in claim 2, that for some $m' \neq m$

$$(q_0, a^{m'}b^k a^m b^k, Z_0) \vdash^* (q^{(k)}, b^{k-j}, \xi') \vdash^* (q_f, \Lambda, \Lambda)$$

(where ξ' differs from ξ only in the number of repetitions of η; we omit the details which are analogous to the arguments in the proof of Claim 2). This contradicts $T_2(M) = L_n$, and, therefore, $w = a^{m_k}$ for some $m_k \leqslant m$. This completes the proof of claim 3.

The proof of Lemma 4.3 can now be completed without difficulty. Let $m \geqslant 1$, $1 \leqslant k \leqslant n$, and $1 \leqslant k' \leqslant n$. Let $q^{(k)}$, $q^{(k')} \in Q$ and m_k, $m_{k'}$ be as in claim 3 (for k and k', respectively). Using (12) and (13) we can write

$$(q_0, a^m b^k a^m b^k, Z_0) \vdash^* (q^{(k)}, a^{m-m_k} b^k, \xi) = c \vdash^* (q_f, \Lambda, \Lambda)$$

and for $m' = m + m_{k'} - m_k$

$$(q_0, a^m b^{k'} a^{m'} b^k, Z_0) \vdash^* (q^{(k')}, a^{m-m_k} b^k, \xi) = c'.$$

By the assumption that $|Q| < \eta$, k and k' can be chosen in such a way that $k \neq k'$ but $q^{(k)} = q^{(k')}$. But then we have $c' = c \vdash^* (q_f, \Lambda, \Lambda)$, and, thus, $a^m b^{k'} a^{m'} b^k \in T_2(M)$ which is our final contradiction and establishes that M cannot have less than n states. Q.E.D.

Now we can state the theorem.

THEOREM 4.2. *For any $n \geqslant 1$ there is a language $L \in \Delta_2$ such that* $\deg(L) = n$.

Proof. Let $n \geqslant 1$ and consider the language L_n from Lemma 4.1. By Lemma 4.1 and Theorem 4.1 $L_n = L(G)$ for a grammar G with strict partition π, $\|\pi\| = n$. Since $\|\pi_0\| \leqslant \|\pi\|$ we have $deg(L_n) \leqslant n$. Assume $deg(L_n) < n$. By Theorem 4.1 there exists a DPDA for L_n with less than n states, contradicting Lemma 4.3. Q.E.D.

Theorem 4.2 establishes the hierarchy of strict deterministic languages under their degree. However, the problem of deciding the degree of a language remains open. In a certain sense this problem is equivalent to the effectiveness of a minimization procedure for a DPDA.

The family of strict deterministic languages of degree 1 turns out to coincide (except for $\{\Lambda\}$) with the family of "simple deterministic languages" defined by Korenjak and Hopcroft. We shall return to this point in [10].

REFERENCES

1. A. JA. DIKOVSKII, On the relationship between the class of all context-free languages and the class of deterministic context-free languages, (Russian) *Algebra i Logika* **8** (1969), 44–64.
2. S. GINSBURG, "The Mathematical Theory of Context-Free Languages," McGraw-Hill, New York, 1966.
3. S. GINSBURG AND S. A. GREIBACH, Deterministic context-free languages, *Information and Control* **9** (1966), 602–648.
4. S. A. GREIBACH, An infinite hierarchy of context-free languages, *J. Assoc. Comput. Mach.* **16** (1969), 91–106.
5. S. A. GREIBACH, Characteristic and ultrarealtime languages, *Information and Control* **18** (1971), 65–98.
6. D. GRIES, "Compiler Construction for Digital Computers," John Wiley and Sons, New York, 1971.
7. L. H. HAINES, "Generation and Recognition of Formal Languages," Ph.D. thesis, M.I.T., 1965.
8. M. A. HARRISON, On the relation between grammars and automata, *in* "Advances of Information Sciences 4," (J. T. Tou, Ed.), pp. 39–92, Plenum Press, New York, 1972.
9. M. A. HARRISON AND I. M. HAVEL, On the parsing of deterministic languages, submitted for publication.
10. M. A. HARRISON AND I. M. HAVEL, Real time strict deterministic languages, to appear, *SIAM J. Computing,* (1973).
11. J. E. HOPCROFT AND J. D. ULLMAN, "Formal Languages and Their Relation to Automata," Addison-Wesley, Reading, MA, 1969.
12. D. E. KNUTH, On the translation of languages from left to right, *Information and Control* **8** (1965), 607–639.
13. A. SALOMAA, "Theory of Automata," Pergamon Press, New York, 1969.
14. J. D. ULLMAN, "Applications of Language Theory to Compiler Design," Technical Report No. 95, Princeton University; to appear in "Theoretical Computer Science" (A. V. Aho, Ed.), Prentice Hall, Englewood Cliffs, NJ, 1972.