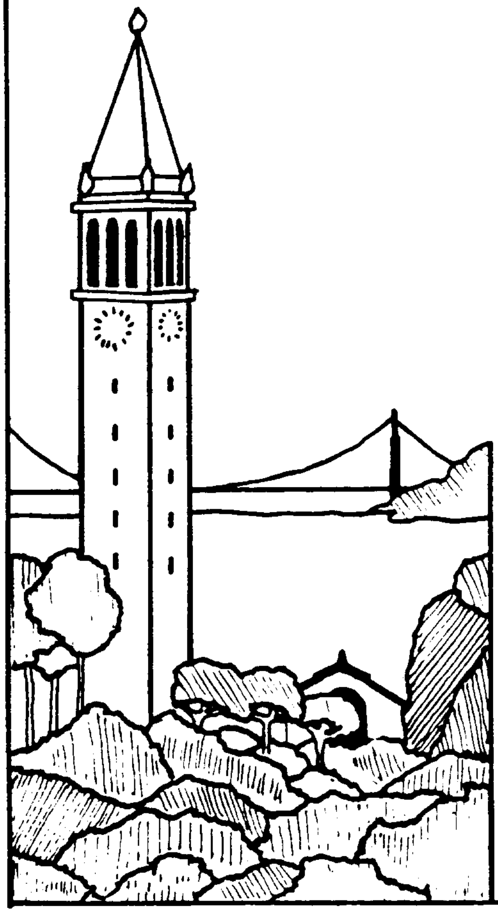
Iteration Theorems for Deterministic Families of Languages

Michael A. Harrison



Report No. UCB/CSD 86/271

December 1985

Computer Science Division (EECS)
University of California
Berkeley, California 94720

Iteration Theorems for Deterministic Families of Languages

Michael A. Harrison*
Computer Science Division
University of California
Berkeley, CA 94720

December 4, 1985

Abstract

In this paper, we consider the problem of finding iteration theorems for various subfamilies of deterministic languages. Because deterministic languages are constrained in their generation, it is not possible to merely "pump substrings" as in the general context free case. We lay out, in detail, a collection of techniques for proving theorems of this type for deterministic context free languages.

*Sponsored by the National Science Foundation under Grant MCS-8311781 and by the Defense Advanced Research Projects Agency (DoD), Arpa Order No. 4871, monitored by Naval Electronics Systems Command, under Contract No. N00039-84-C-0089.

1 Introduction

In research into the theory of languages, one of the most important questions is whether a given language L belongs to family $\mathcal{X}$ of languages. If one is working with very rich families of languages such as context sensitive languages, recursive sets, etc, there are powerful techniques like diagonalization which can be employed. At the opposite extreme is the family $\mathcal{R}$ of regular sets. For $\mathcal{R}$, one can succeed without much formal machinery. Completely elementary techniques such as congruence relations, simple pumping methods, and closure properties all work for $\mathcal{R}$.

The most important of all families of formal languages is the collection $\mathcal{C}$ of context free languages. The first systematic way of obtaining non-context free languages was the "Pumping Lemma" of [4]. (This result will be stated shortly.) The result nicely captured essential properties of derivations. The inadequacies of the result were improved subsequently in what was called the "Iteration Theorem."

As the field developed, the significant applications were to deterministic context free languages which could be parsed in linear time. Many of the definitions of these languages were difficult to handle as they involved quantification over infinite families of derivations. It took quite some time to figure out *systematic* techniques for showing non membership in these language classes. The purpose of this paper is to expose the techniques used to obtain iteration theorems for various subfamilies of deterministic context free languages. By focusing on the techniques involved, we hope to be able to apply them to certain open problems in these areas.

We are now ready to state the original "pumping lemma" originally proven in [4].

Theorem 1.1 *Let $G = (V, \Sigma, P, S)$ be a context free grammar. It is possible to determine two natural numbers p and q such that every sentence z of $L(G)$ with $\lg(z) > p$ admits a decomposition of the form $z = xuwvy$ where $u = \Lambda$ or $v = \Lambda$, $\lg(uwv) < q$, and all strings $z_k = xu^kwv^ky \in L(G)$ for all $k > 1$.*

While this lemma was useful in proving a number of results, it could not handle a number of cases. The set L is a concrete example.

$$L = \{a^*bc\} \cup \{a^pba^nca^n \mid p \text{ prime}, n > 0\}$$

L is not context free yet the conclusion of the pumping lemma hold for L. An additional practical difficulty with the Pumping Lemma is the large number of cases that have to be dealt with in working with the pumping lemma on specific languages.

This practical difficulty was eliminated by adding the concept of "positions" to the pumping lemma to give what is commonly called the Iteration Theorem or Ogden's lemma. In [18], Ogden credits the idea of position to Dana Scott. We now set the stage for a careful statement of this theorem.

Definition 1.1 *Let $w \in \Sigma^*$. Any sequence $\varphi = (v_1, \ldots, v_n) \in (\Sigma^*)^n$ such that $w = v_1 \cdots v_n$ is called a factorization of w. Any integer i, $1 \leq i \leq \lg(w)$ is called a position in w.*

Let K be a set of positions in w. Any factorization φ induces a "partition" of K which we write as

$$K/\varphi = (K_1, \ldots, K_n)$$

where for each $i, 1 \leq i \leq n$

$$K_i = \{k \in K \mid \lg(v_1 \cdots v_{i-1}) < k \leq \lg(v_1 \cdots v_i)\}.$$

Note that some K_i may be empty so this is not a true partition.

Example 1.1 *Let $w = a^p b^p c^p$.*

$$\varphi = (a^p, b^p, c^p)$$

and

$$K = \{p+1, \ldots, 2p\}$$

then $K_1 = K_3 = \emptyset$ and $K_2 = K$.

We can now state the iteration theorem for context free languages.

Theorem 1.2 *(The Iteration Theorem) Let L be a context free language. There exists a number $p(L)$ such that for each $w \in L$ and any set K of positions in w, if $|K| > p(L)$ then there is a factorization $\varphi = (v_1, \ldots, v_5)$ of w such that*

1. *for each $q \geq 0$, $v_1 v_2^q v_3 v_4^q v_5 \in L$.*
2. *if $K/\varphi = \{K_1, ..., K_5\}$ then*
   (a) *either $K_1, K_2, K_3 \neq \emptyset$ or $K_3, K_4, K_5 \neq \emptyset$ and*
   (b) *$|K_2 \cup K_3 \cup K_4| < p(L)$.*

Part (2) implies that $K_3 = \emptyset$ and either K_2 or $K_4 = \emptyset$. Thus either $v_2 = \Lambda$ or $v_4 = \Lambda$. Condition (2)(b), $|K_2 \cup K_3 \cup K_4| < p(L)$, is necessary to prohibit trivial factorizations in which $w = v_1v_2v_3v_4$ or $w = v_2v_3v_4v_5$.

The reader is referred to [10] for a proof. The proof of the Iteration Theorem requires only a little extra combinatorial overhead as compared to the proof of the Pumping Lemma. The benefits are real as the work involved in showing sets to be non language is reduced and more importantly, new problems can now be solved that were previously open. Note that the example immediately following the statement of the Pumping Lemma can be dealt with by the Iteration Theorem.

Note that the Pumping Lemma now becomes a corollary of the Iteration Theorem by simply choosing $K = \{1, ..., \lg(w)\}$. Moreover, the result extends to sequential forms directly.

There are two principal applications of the Iteration Theorem in the full context free case. The most common is determining that certain sets are not context free. There are mathematical implications such as showing the inherent ambiguity of specific language without the algebraic machinery of Parikh's theorem. [19]

It is interesting to note that the idea of distinguished positions can be generalized naturally to having both "included" and "excluded" positions. If one does so, then the set K becomes two sets I and E of included positions and excluded positions respectively. The corresponding theorem from [3] follows.

Theorem 1.3 *Let L be a context free language. There exists a number $p(L)$ such that for each $w \in L$ and any sets I and E of positions in w, if $|I| > (p(L))^{(1+|E|)}$ then there is a factorization $\varphi = (v_1, v_2, v_3, v_4, v_5)$ of w such that*

1. *for each $q \geq 0$, $v_1 v_2{}^q v_3 v_4{}^q v_5 \in L$.*
2. *if $I/\varphi = I_1 \ldots I_5)$ and $E/\varphi = (E_1 \ldots E_5)$ then*

(a) *either* $I_1, I_2, I_3 \neq \emptyset$ *and* $E_1, E_2, E_3 = \emptyset$ *or* $I_3, I_4, I_5 \neq \emptyset$ *and* $E_3, E_4, E_5 = \emptyset$ *and*

(b) $|I_2 \cup I_3 \cup I_4| \leq (P(L))^{(1+|E_2|\cup|E_3|\cup|E_4|}$

Proof. The argument is not substantially different than the one in Chapter 11 of [10].

Consider the set

$$L = \{w \in \{a,b\}^* \mid \text{if } w = ab^q \text{ for some } q \text{ then } q \text{ is prime}\}$$

It is not difficult to see that L satisfies Ogden's lemma but L is not context free and does not satisfy Theorem 1.3.

2 Deterministic Languages

In [13], the family of strict deterministic grammars was introduced. The idea is to find a family of grammars which capture the essence of the notion of determinism in a pushdown automata. While the languages generated by this family are on the full family of deterministic context free languages, they do possess the right mathematical properties. This will all be clearer after we deal with the precise definitions.

Notation. Let $\alpha \in V^*$ and $n \geq 0$. ${}^{(n)}\alpha$ denotes the prefix of α of length $\min\{n, \lg(\alpha)\}$. The notation $\alpha^{(n)}$ is used for the suffix of α of length $\min\{n, \lg(\alpha)\}$.

Definition **2.1** *Let $G = (V, \Sigma, P, S)$ be a context free grammar and let π be a partition of the set V of terminal and nonterminal letters of G. Such a partition π is called strict if*

1. $\Sigma \in \pi$, and

2. For any $A, A' \in N$ and $\alpha, \beta, \beta' \in V^*$ if $A \rightarrow \alpha\beta$ and $A' \rightarrow \alpha\beta'$ are in P and $A \equiv A' \pmod{\pi}$ then either (i) both $\beta, \beta' \neq \Lambda$ and ${}^{(1)}\beta \equiv {}^{(1)}\beta' \pmod{\pi}$ 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 $[A]_\pi = \{A' \in V \mid A' \equiv A \pmod{\pi}\}$.

It is worth considering a simple example of a grammar for parenthesized arithmetic expressions which use the operator symbols + and *. The "natural grammar" (cf. [10]) is not strict deterministic but the following one is

Example 2.1 *Let $G_2 = (V, \Sigma, P, S)$ be a context free grammar where*

$$\begin{aligned} \Sigma &= \{a, +, *\} \\ V - \Sigma &= \{S, E, T_1, T_2, F_1, F_2, F_3\} \\ \pi &= \{\Sigma, \{S\}, \{E\}, \{T_1, T_2\}, \{F_1, F_2, F_3\}\} \end{aligned}$$

The rules are:

$$\begin{aligned} S &\rightarrow (E \\ E &\rightarrow T_1 E \mid T_2 \\ T_1 &\rightarrow F_1 T_1 \mid F_2 \\ T_2 &\rightarrow F_1 T_2 \mid F_3 \\ F_1 &\rightarrow (E* \mid a* \\ F_2 &\rightarrow (E+ \mid a+ \\ F_3 &\rightarrow (E) \mid a) \end{aligned}$$

Definition 2.2 *Any grammar $G = (V, \Sigma, P, S)$ is called strict deterministic if there exists a strict partition π of V. A language L is called a strict deterministic language if $L = L(G)$ for some strict deterministic grammar G.*

From the definition, a strict deterministic grammar cannot have rule of the following type:

$$A \rightarrow \alpha\beta$$
$$A' \rightarrow \alpha$$

with $A \equiv A' \pmod{\pi}$ and $\beta \neq \Lambda$.

Definition 2.3 *A set $L \subseteq \Sigma^*$ is prefix-free if $uv \in L$ and $u \in L$ imply $v = \Lambda$.*

Every strict deterministic language is prefix free.

Lemma 2.1 *Let $G = (V, \Sigma, P, S)$ be a strict deterministic context free grammar and let π be a partition of the set V. For any $A, a' \in V - \Sigma$, $w, u \in \Sigma^*$, if $A \equiv A' \pmod{\pi}$, $A \Rightarrow^* w$, and $A' \Rightarrow^* wu$, then $u = \Lambda$.*

Proof. The argument is identical to the proof of Theorem 2.2 of [13].

Definition 2.4 *Let $G = (V, \Sigma, P, S)$ be a context free grammar. For each $\alpha \in V^*$, define $L(\alpha) = \{w \in \Sigma^* \mid \alpha \Rightarrow^* w\}$.*

Lemma 2.2 *Let $G = (V, \Sigma, P, S)$ be a strict deterministic grammar. For each $\alpha \in V^*$, $L(\alpha)$ is a prefix-free set.*

Proof. Use induction on $\lg(\alpha)$ and apply Lemma 2.1 to each variable in α.

No reduced strict deterministic grammar can be left-recursive (i.e. for no $A \in N$ and $\alpha \in V*$, does $A \Rightarrow^* A\alpha$). Given these facts, which are proven in [11], the following result is not surprising.

Theorem 2.1 *Any strict deterministic grammar is equivalent to a strict deterministic grammar in Greibach normal form.*

Certainly, the most natural way to study deterministic languages might seem to be through their automata. Indeed that has been the historic way. The intuition that this gives is helpful but there have been problems with this approach. Deterministic languages are less robust than context-free languages and not every choice for a definition of a pushdown automaton leads to the same family of languages. For deterministic pushdown automata, the question of moving on an empty pushdown store is such an issue. Cf. [17] for an example of of an unfortunate choice. We will refer to the definition of [10] which is the most general one. Using that definition, a general deterministic pushdown automaton can accept families of languages in three different ways, namely

1. by final state and empty (pushdown) store, or

2. by final state and leaving any one work symbol on the pushdown, or

3. by final state and ignoring the pushdown.

We shall call these families Δ_2,Δ_1, and Δ_0 respectively. More formally, and using the notation of [10], if $A = (Q, \Sigma, \Delta, \Gamma, \delta, q_0, Z_0, F)$ is a deterministic pushdown automaton then

$$T(A,K) = \{w \in \Sigma^* \mid (q_0, w, Z_0) \overset{*}{\vdash} (q, \Lambda, \alpha) \text{ for some } q \in F, \alpha \in K\}$$

then

$$T_0(A) = T(A, \Gamma^*)$$
$$T_1(A) = T(A, \Gamma)$$
$$T_2(A) = T(A, \{\Lambda\})$$

Finally, for $i = 0, 1, 2$

$$\Delta_i = \{T_i(A) \mid A \text{ is a deterministic pushdown automaton}\}$$

Thus Δ_0 is the family of deterministic context free languages. Δ_2 is the collection of strict deterministic context free languages. Δ_1 is an interesting intermediate family which contains all the regular sets. More details about these families may be found in [10] and [8]. It is true that

$$\Delta_2 \subseteq \Delta_1 \subseteq \Delta_0$$

and all the inclusions are proper.

For example, $\{\Lambda, a\}$ is in $\Delta_1 - \Delta_0$ while

$$L = \{a^n b^n \mid n \geq 1\} \cup a^*$$

is in Δ_0 but not Δ_1.

The relationship between Δ_2 and Δ_0 is easy to express.

Theorem 2.2 *L is a strict deterministic context free language if and only if L is a prefix-free and is a deterministic context free language.*

2.1 Left Parts of Trees

The key to giving a rigorous proof for deterministic iteration theorems is to use trees, not machines. This was a problem in [18] where a deterministic iteration theorem was first expressed but not proved rigorously. It turns out to be easy to do with strict deterministic grammars. There are two key concepts involved and we can now discuss the main one, namely the left-part of a grammatical tree. We use the notation of [10]

Definition 2.5 *Let T be a derivation tree of some grammar G. For any $n \geq 0$ we define ${}^{(n)}T$, the left n-part of T (or the left part where n is understood) as follows. Let $(x_1, \ldots, x_m)$ be the sequence of all terminal*[1] *nodes in T (from the left to the right), i.e., $\{x_1, \ldots, x_m\} = \lambda^{-1}(\Sigma)$ and $x_1 \leftharpoondown^{+} x_2 \leftharpoondown^{+} \cdots \leftharpoondown^{+} x_m$. Then*

$$\begin{aligned} {}^{(n)}T &= \{x \in T \mid x \overset{*}{\rhd} \overset{*}{\leftharpoondown} x_n\} \text{ if } n \leq m, \text{ and} \\ {}^{(n)}T &= T \text{ if } n > m. \end{aligned}$$

We consider ${}^{(n)}T$ to be a tree under the same relations $\rhd$, $\leftharpoondown$ and labeling λ as T.

Thus, for instance, the shaded area on Figure 2.1 (including the path to x) is the left n-part of T if x is the n^{th} terminal node in T. Note that, in general, ${}^{(n)}T$ may not be a derivation tree.

As immediate consequences of Definition 2.5, we have

Fact 2.1 *${}^{(n)}T = {}^{(n+1)}T$ if and only if ${}^{(n)}T = T$.*

Fact 2.2 *If ${}^{(n)}T = {}^{(n)}T'$ then ${}^{(j)}T = {}^{(j)}T'$ for any $j \leq n$.*

Our principal result about the structure of derivation trees of a strict deterministic grammar can be informally stated as follows. A reduced grammar G has a strict partition π if and only if, given any $V \in \pi$, then each prefix u of any string $w = uv \in \Sigma^*$ generated by some symbol $A \in V_i$ uniquely determines the left partial subtree of tree T (cf. 2.1 T corresponds to a derivation $A \Rightarrow^* w$ up to the path to the terminal node (x) labeled by

[1]Note that the Λ-nodes are not being indexed. We use the two partial orders presented in [10] for defining trees.

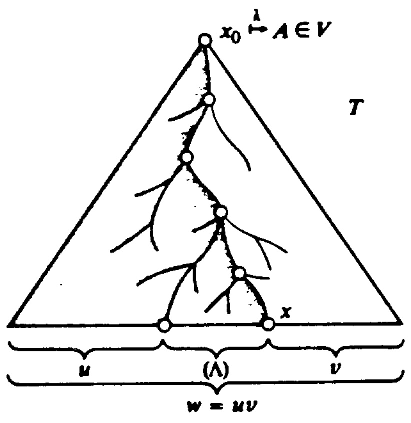


Figure 1: The left part of a tree

the first letter of v. We allow the labels of nodes on this particular path to be determined modulo the partition π (in particular, the equivalence class containing x is Σ. In the case when $v = \Lambda$, the complete tree is specified uniquely (and then it cannot be a proper subtree of any other tree with an equivalent root label).

For the formal statement of the theorem we first introduce the "left part property" of a set of derivation trees.

Definition 2.6 *Let $\mathcal{T} \subseteq \mathcal{T}_G$, a set of derivation trees for some grammar $G = (V, \Sigma, P, S)$, and let π be an arbitrary partition on V, not necessarily strict. $\mathcal{T}$ satisfies the left-part property with respect to π if and only if for any $n \geq 0$ and $T, T' \in \mathcal{T}$ if*

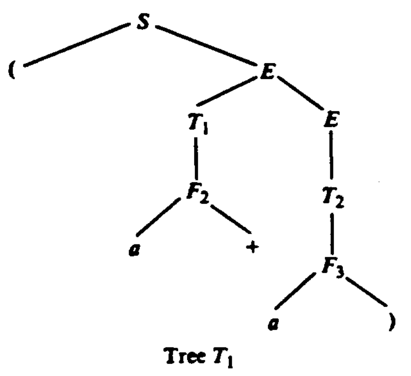
$$\mathbf{rt}(T) = \mathbf{rt}(T') \pmod{\pi} \textit{ and } {}^{(n)}\mathbf{fr}(T) =^{(n)} \mathbf{fr}(T')$$

then

$$ {}^{(n+1)}T =^{(n+1)} T' \tag{1}$$

and, moreover, if $x \mapsto x'$ is the structural isomorphism ${}^{(n+1)}T \mapsto {}^{(n+1)}T'$ then for every x in ${}^{(n+1)}T$,

$$\begin{aligned} \lambda(x) = \lambda(x') \quad & \textit{if} \quad x \overset{+}{\leftharpoonup} y \textit{ for some } y \in {}^{(n+1)}T \\ & \textit{or if} \quad {}^{(n+1)}T =^{(n)} T \end{aligned}$$



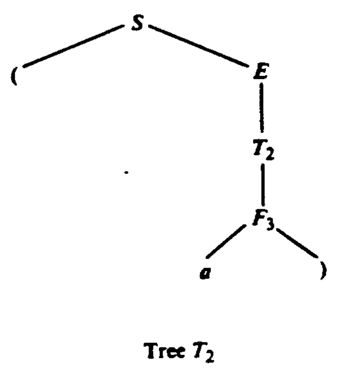


Figure 2: Two Trees from Grammar G_2

and

$$\lambda(x) = \lambda(x') \pmod{\pi} \textit{ otherwise.} \tag{2}$$

Note that the condition "$x \leftarrow^+ y$ for some $y \in {}^{(n+1)}T$" in (2) is equivalent to "x is not on the rightmost path in ${}^{(n+1)}T$."

An example of this kind of mapping can be obtained by considering grammar G_2 from Section 2. Let us consider the trees shown in Figure 2.1. If we let $n = 2$, we see that $\mathrm{rt}(T_1) = S \equiv S = \mathrm{rt}(T_2)$ and ${}^{(2)}\mathrm{fr}(T_1) = (a =^{(2)} \mathrm{fr}(T_2)$. Thus ${}^{(3)}T_1 \cong {}^{(3)}T_2$ and the path from S to $+$ in T_1 is "equivalent modulo π" to the path from S to) in T_2.

Note that the left-part property is a global property of trees in distinction to the strictness of a grammar, which is a local property of derivation trees (being a property of their elementary subtrees).

Theorem 2.3 *(The Left-Part Theorem). Let $G = (V, \Sigma, P, S)$ be a reduced grammar and let π be a partition on V such that $\Sigma \in \pi$. Then π is strict in G if and only if the set $\mathcal{T}_G$ of all derivation trees of G satisfies the left-part property with respect to π.*

Proof. (The "if" direction). Let G and π be as in the assumption of the theorem and assume $\mathcal{T}_G$ satisfies the left-part property with respect to π. We shall show the π is strict.

Let A, $A' \in N$, let $A \to \alpha\beta$ and $A' \to \alpha\beta'$ be in P and assume $A \equiv A'$ (mod π). Since G is reduced we have $\alpha \Rightarrow^* w_\alpha$, $\beta \Rightarrow^* w_\beta$ and $\beta' \Rightarrow^* w_{\beta'}$ for some w_α, w_β, $w_{\beta'} \in \Sigma^*$. Let us consider two derivation trees T, T' corresponding to derivations $A \Rightarrow \alpha\beta \Rightarrow *w_\alpha w_\beta$ and $A' \Rightarrow \alpha\beta' \Rightarrow *w_\alpha w'_\beta$ respectively. Let $n = \lg(w_\alpha)$. Then $\mathbf{rt}(T) = A \equiv A' = \mathbf{rt}(T')$ and ${}^{(n)}\mathbf{fr}(T) = {}^{(n)}\mathbf{fr}(T')$. Thus ${}^{(n+1)}T \cong^{(n+1)} T'$ by the left-part property of T_G. Let $x \mapsto x'$ be the structural isomorphism from ${}^{(n+1)}T$ to ${}^{(n+1)}T'$. We distinguish two cases:

Case 1: $w_\beta \neq \Lambda$. Let x be the $(n+1)^{st}$ terminal node of T (labeled by ${}^{(1)}w_\beta$) and y the $(\lg(\alpha)+1)^{st}$ node among the immediate descendants of the root in T, counted from the left (i.e., $\lambda(y) = {}^{(1)}\beta$). Clearly y is in ${}^{(n+1)}T$ and by the structural isomorphism, y' is in ${}^{(n+1)}T'$, $\lambda(y') = {}^{(1)}\beta'$. Then ${}^{(1)}\beta \equiv^{(1)} \beta'$ by (2) or by (3) (depending on whether $y \leftharpoondown^{+} x$ or not).

Case 2: $w_\beta = \Lambda$. Then ${}^{(n+1)}T = {}^{(n)}T$ and by (2), $\lambda(x) = \lambda(x')$ for all x in ${}^{(n+1)}T$. Thus ${}^{(n+1)}T =^{(n+1)} T'$. Then also ${}^{(n)}T =^{(n)} T'$ by Fact 2 and using Fact 1, we conclude $T = T'$. Hence $A = A'$ and $\beta' = \beta = \Lambda$.

(The "only if" direction). Let G be a reduced grammar with strict partition π. Let $T, T' \in T_G$ be two derivation trees such that

$$\mathbf{rt}(T) \equiv \mathbf{rt}(T') \pmod{\pi} \tag{3}$$

and

$${}^{(n)}\mathbf{fr}(T) =^{(n)} \mathbf{fr}(T') \tag{4}$$

for some $n \geq 0$. To show that ${}^{(n+1)}T$ and ${}^{(n+1)}T'$ satisfy (1), (2), and (3), one proceeds by induction on the height h of the larger one of the two trees. The rather straightforward details are in [11].

The following two corollaries are immediate consequences of the Left-Part Theorem.

Corollary 2.1 *Any strict deterministic grammar is unambiguous.*

Corollary 2.2 *If π is a strict partition on G then for every $U \in \pi$ the set $\{w \in \Sigma \mid A \Rightarrow^* w$ for some $A \in U\}$ is prefix-free.*

Thus, in particular, $L(G)$ is prefix-free, which we already know.

2.2 A Strict Deterministic Iteration Theorem

Now that we have a left part theorem for strict deterministic languages, we are ready to prove an iteration theorem for the family. In order to do this, we need the second key ingredient for proving iteration theorems. This is a collection of (trivial) facts about cross sections of derivation trees. Depending on the grammar class, different types of results are needed. These little results are used, together with the isomorphism of the left part theorem, to deduce key facts about the derivations which give more detailed information about how and what to "pump."

Theorem 2.4 *(Iteration Theorem for Δ_2)*

Let L be a strict deterministic context free language and $L \subseteq \Sigma^*$. There exists a number $p(L)$ such that for each $w \in L$ and any set K of positions in w, if $|K| > p(L)$ then there is a factorization $\varphi = (v_1, \ldots, v_5)$ of w such that

1. *$v_2 \neq \Lambda$;*
2. *for each $n, m \geq 0$, $u \in \Sigma^*$,*
$$v_1 v_2^{n+m} v_3 v_4^n v_5 \in L \text{ if and only if } v_1 v_2^m v_3 u \in L.$$
3. *if $K/\varphi = \{K_1, ..., K_5\}$ then*
   (a) *either $K_1, K_2, K_3 = \emptyset$ or $K_3, K_4, K_5 = \emptyset$ and*
   (b) *$|K_2 \cup K_3 \cup K_4| < p(L)$.*

Corollary 2.3 *For each $n \geq 0$*

$$v_1 v_2^n v_3 v_4^n v_5 \in L.$$

Proof. The argument mirrors the proof of Theorem 6.2.1 of [10]. Let $G = (V, \Sigma, P, S)$ be a strict deterministic grammar generating L. Choose $p(L)$ as in the proof of Theorem 6.2.1 of [10] and follow that proof. That is, find $\varphi = (v_1, \ldots, v_5)$ satisfying the requirements of the proof.

Property (3) follows from the iteration theorem for context free languages. To show property (1), suppose $v_2 = \Lambda$. Then we would have

$$A \overset{+}{\Rightarrow} A v_4$$

But left recursion is impossible in a strict deterministic grammar. Thus (3) has been satisfied.

To establish property (2), let $m, n \geq 0$. By the iteration theorem for context free languages

$$S \overset{*}{\Rightarrow} v_1 v_2^m A v_4^m v_5 \overset{*}{\Rightarrow} v_1 v_2^{m+n} v_3 v_4^{n+m} v_5 = w_1 v_4^m v_5 \tag{5}$$

where $w_1 = v_1 v_2^{m+n} v_3 v_4^n \in \Sigma^*$. Assume now

$$S \overset{*}{\Rightarrow} v_1 v_2^{m+n} v_3 v_4^n u = w_1 u\,. \tag{6}$$

Let T, T' be two derivation trees corresponding to (5) and (6) respectively. Let $k = \lg(w_1)$. By the left part theorem,

$$^{(k+1)}T \cong {}^{(k+1)}T'\,.$$

Let x be the node in T which corresponds to the indicated occurrence of A in sentential form $v_1 v_2^m A v_4^m v_5$. Clearly x is left of the terminal node $(k+1)$ in T (labeled with $^{(1)}v_4$). By the left-part property (2), the corresponding node in T' is also labeled with A. Therefore the following derivation corresponds to T':

$$S \overset{*}{\Rightarrow} v_1 v_2^m A u \overset{*}{\Rightarrow} w_1 u\,.$$

Since $A \Rightarrow^+ v_2^{n'} v_3 v_4^{n'}$ for any $n' \geq 0$ we have

$$v_1 v_2^{m+n'} v_3 v_4^{n'} u \in L \text{ ,for any } n' \geq 0\,.$$

In particular, taking $n' = 0$ we obtain the "only if" part of property (2); taking $n = 0$ in (5) and n' arbitrary we obtain the "if" direction of (2).

2.3 The Full Deterministic Case

With the strict deterministic case behind us, doing the full deterministic case is quite easy. The trick is to use the "endmarking map"

$$L \mapsto L\$$$

which makes the correspondence very clear.

Theorem 2.5 *(Iteration Theorem for Δ_0)*

Let $L \subseteq \Sigma^*$ be a deterministic context free language. There exists a number $p'(L)$ such that for each $w \in L$ and any set K of positions in w, if $|K| > p'(L)$ then there is a factorization $\varphi = (v_1, \ldots, v_5)$ of w such that

1. $v_2 \neq \Lambda$;

2. *for each* $n \geq 0$

$$v_1 v_2^n v_3 v_4^n v_5 \in L.$$

3. *if* $K/\varphi = \{K_1, \ldots, K_5\}$ *then*

(a) *either* $K_1, K_2, K_3 = \emptyset$ *or* $K_3, K_4, K_5 = \emptyset$ *and*

(b) $|K_2 \cup K_3 \cup K_4| < p(L)$.

4. *if* $v_5 \neq \Lambda$ *then for each* $m, n \geq 0$, $u \in \Sigma^*$

$$v_1 v_2^{m+n} v_3 v_4^n u \in L \text{ if and only if } v_1 v_2^m v_3^m u \in L$$

Proof. Much of the previous proof and all of the proof of Iteration Theorem, Cf. [10], still apply here. Thus 3 follows. Also 2 follows easily. To complete the proof, let $L \in \Delta_0$. Then $L\$ \in \Delta_2$ and is subject to the Iteration Theorem for strict deterministic languages. Define $p'(L) = p(L\$)$ and let $w \in L$ with a set of positions K where $|K| \geq p'(L)$. Consider the factoization $\varphi = (v_1, v_2, v_3, v_4, v_5')$ of $w\$$ obtained from Theorem 2.4 applied to $L\$$. Since \$ does not occupy any position from K, there are only two possibilities where it can occur: either $v_3 \in \Sigma^+\$$ and $v_4 = v_5' = \Lambda$ or $v_3 \in \Sigma^+$ and $v_5' = v_5\$$ for some $v_5 \in \Sigma^*$. In either event, (1) holds. Assume $v_5 \neq \Lambda$. We can restrict ourselves to the second possibility. Take $\varphi = (v_1, v_2, v_3, v_4, v_5)$ as the factorization of w. The result is now simple for any $n, m \geq 0$ and $u \in \Sigma^*$ we have

$$\begin{aligned} v_1 v_2^{n+m} v_3 v_4^n v_5 \in L \quad &\text{if and only if} \quad v_1 v_2^{n+m} v_3 v_4^n v_5\$ \in L\$ \\ &\text{if and only if} \quad v_1 v_2^m v_3 u\$ \in L\$ \\ &\text{if and only if} \quad v_1 v_2^m v_3 u \in L \end{aligned}$$

This satisfies (4) and completes the proof.

To illustrate the importance of the last theorem, some applications will be given. Using the standard techniques developed for the classical iteration theorem, it is easy to show that

$$L = \{a^n b^n \mid n \geq 1\} \cup \{a^n b^{2n} \mid n \geq 1\}$$

is not a deterministic context free language.

It is shown in [11] that

$$L = \{ww^T \mid w \in \{a,b\}^*\}$$

is not in any finite union of deterministic context free languages.

3 A Hierarchy Result

It turns out that there exists a natural hierarchy of strict deterministic languages. Here we approach this from the point of view of grammars.

Definition 3.1 *For any strict partition π on a given context free grammar $G = (V, \Sigma, P, S)$, define*

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

Thus $\|\pi\|$ is the cardinality of the largest non- block of π. It is possible to relate $\|\pi\|$ to the number of states of a deterministic pushdown automaton accepting the language generated by such a grammar. The pertinent result from [13] is the following:

Theorem 3.1 *Let L be any language and let $n \geq 1$. Then $L = L(G)$ for some strict deterministic grammar with partition π such that $\|\pi\| = n$ if and only if $L = T_2(A)$ for some deterministic pushdown automaton A with n states.*

Let us recall that every strict deterministic grammar has a unique partition π_0 which is the minimal element in the semi-lattice of all strict partitions of G. Also, $\|\pi_0\| \leq \|\pi\|$ for any other strict partition of G. This suggests the following definition.

Definition 3.2 *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:

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

The trick in obtaining an Iteration Theorem for the languages in this hierarchy, is to consider a number of derivations at the same time. The left part theorems focus the trees to be "similar modulo π." By a shoe-box argument, we can convert "equivalence" to equality.

We can now give the main result of this section.

Theorem 3.2 *Let L be a strict deterministic language of degree n. There exists an integer p such that, for each $w \in L$ and each set K of p or more distinguished positions in w, there is a factorization $\varphi = (w_1, w_2, w_3, w_4, w_5)$ of w such that*

1. $w_2 \neq \Lambda$,
2. *if* $K/\varphi = \{K_1, \ldots K_5\}$ *then*
   (a) *either* $K_1, K_2, K_3 \neq \emptyset$ *or* $K_3, K_4, K_5 \neq \emptyset$,
   (b) $|K_1 \cup K_3 \cup K_4| \leq p$,
3. *for each* $k, m \geq 0$, *and* $u \in \Sigma^*$, $w_1 w_2{}^{k+m} w_3 w_4{}^k u \in L$ *if and only if* $w_1 w_2{}^m w_3 u \in L$,
4. *for each* $u_1, \ldots, u_{n+1} \in \Sigma^*$, *if* $w_1 w_2{}^{n_i} u_i \in L$ *for each* $i = 1, \ldots n+1$, *where each* $n_i \geq n$, *there exist* $1 \leq i < j \leq n+1$, $1 \leq r \leq n_i$, $1 \leq r' \leq n_j$, *and factorizations* $\xi = (v, x, y, z)$ *and* $\xi' = (v', x', y', z')$ *of* u_i *and* u_j, *respectively, such that*
   (a) *for all* $m \geq 0$, *the following are all in* L:

$$w_1 w_2{}^{(n_i - r) + mr} v x y^m z, \qquad w_1 w_2{}^{(n_j - r') + mr'} v' x' y'^m z',$$
$$w_1 w_2{}^{(n_i - r) + mr} v' x y^m z \text{ and }, \quad w_1 w_2{}^{(n_j - r') + mr'} v x' y'^m z'.$$

   (b) *none of* w_3, v, v' *is a proper prefix of any of* w_3, v, v'.

Proof. Let $G = (V, \Sigma, P, S)$ be a reduced strict deterministic grammar of degree n such that $L = L(G)$ and let π be a strict partition of V such that $\|\pi\| = n$. The proof of Theorem 4.2 (in [8]) shows that there exists an integer p such that, for each $w \in L$ and each set K of p or more distinguished positions in w, there is a factorization $\varphi = (w_1, w_2, w_3, w_4, w_5)$ of w such that parts, 1, 2, and 3 hold and such that, for some $A \in V - \Sigma$,

$$S \overset{*}{\Rightarrow} w_1 A w_5 \overset{+}{\Rightarrow} w_1 w_2 A w_4 w_5 \overset{+}{\Rightarrow} w_1 w_2 w_3 w_4 w_5 = w \tag{7}$$

Thus, to complete the proof, we need only show that φ satisfies part 4 of the theorem.

Assume that $w_1 w_2{}^{n_i} u_i \in L$ for $i = 1, \ldots n+1$, where $u_1, \ldots u_{n+1} \in \Sigma^*$, and each $n_i \geq n$. For each $i = 1, \ldots, n+1$, let T_i' be a derivation tree corresponding to $S \Rightarrow^* w_1 w_2{}^{n_i} u_i$. Hence, $\mathrm{rt}(T_i') = S$ and $\mathrm{fr}(T_i') = w_1 w_2{}^{n_i} u_i$.

From (7) we obtain the derivation

$$\begin{aligned} S \quad &\Rightarrow^* \quad w_1 A w_5 \overset{+}{\Rightarrow} w_1 w_2 A w_4 w_5 \overset{+}{\Rightarrow} w_1 w_2^2 A w_4^2 w_5 \overset{+}{\Rightarrow} \cdots && (8) \\ &\Rightarrow^+ \quad w_1 w_2^{n_i} A w_4^{n_i} w_5 \overset{+}{\Rightarrow} w_1 w_2^{n_i} w_3 w_4^{n_i} w_5 && (9) \end{aligned}$$

for each i. Let T_i be a derivation tree corresponding to (9). For $j = 0, \ldots, n_i$, let x_j^i be the node of T_i labeled by A in the cross-section (CS) of T_i labeled by $w_1 w_2^j A w_4^j w_5$. Clearly $x_0^i \rhd^+ x_1^i \rhd^+ \cdots \rhd^+ x_{n_i}^i$. Let $k_i = \lg(w_1 w_2^{n_i})$. and let $y_{k_i+1}^i$ be the leaf of T_i which is labeled by the (k_i+1) st symbol in $w_1 w_2^{n_i} w_3 w_4^{n_i} w_5$ (such a node exists since $K_3 \neq \emptyset$). Then, for $i = 1, \ldots, n+1$, $\mathrm{rt}(T_i) = \mathrm{rt}(T_i') = S$ and ${}^{(k_i)}\mathrm{fr}(T_i) = {}^{(k_i)}\mathrm{fr}(T_i') = w_1 w_2{}^{n_i}$. Therefore, by Theorem 2.3, there exist maps $h_1, \ldots h_{n+1}$ such that, for $i = 1, \ldots n+1$,

1. ${}^{[k_i+1]}T_i \cong {}^{[k_i+1]}T_i'$ under h_i

2. $\lambda(x) = \lambda(h_i(x))$ for all $x \in {}^{[k_i+1]}T_i$ such that $x \rhd^+ y$ holds for some $y \in {}^{[k_i+1]}T_i$, and

3. $\lambda(x) \equiv \lambda(h_i(x)) \pmod{\pi}$ for all $x \in {}^{[k_i+1]}T_i$.

Since w_3 contains a distinguished position, it is nonempty; hence $y_{k_i+1}^i$ is labeled by the first symbol in w_3, so $x_{n_i}^i \rhd^+ y_{k_i+1}^i$. Thus,

$$x_0^i \overset{+}{\rhd} x_1^i \overset{+}{\rhd} \cdots \overset{+}{\rhd} x_{n_i}^i \overset{+}{\rhd} y_{k_i+1}^i$$

so $x_0^i, \ldots, x_{n_i}^i \in {}^{[k_i+1]}T_i$. Let $z_j^i = h_i(x_j^i)$ for $i = 1, \ldots, n+1$, $j = 0, \ldots, n_i$. By (a), $z_0^i \rhd^+ z_1^i \rhd^+ \cdots \rhd^+ z_{n_i}^i$. By (c), $\lambda(x_{n_i}^i) \equiv \lambda(z_{n_i}^i) \pmod{\pi}$ for $i = 1, \ldots, n+1$. Since $\|\pi\| = n$, and $\lambda(x_{n_i}^i) = A$ for all i, there exist i, j, where $1 \leq i < j \leq n+1$, such that $\lambda(z_{n_i}^i) = \lambda(z_{n_j}^j)$ For the remainder of this proof, i and j are fixed at these values. Let $B = \lambda(z_{n_i}^i) = \lambda(z_{n_j}^j)$.

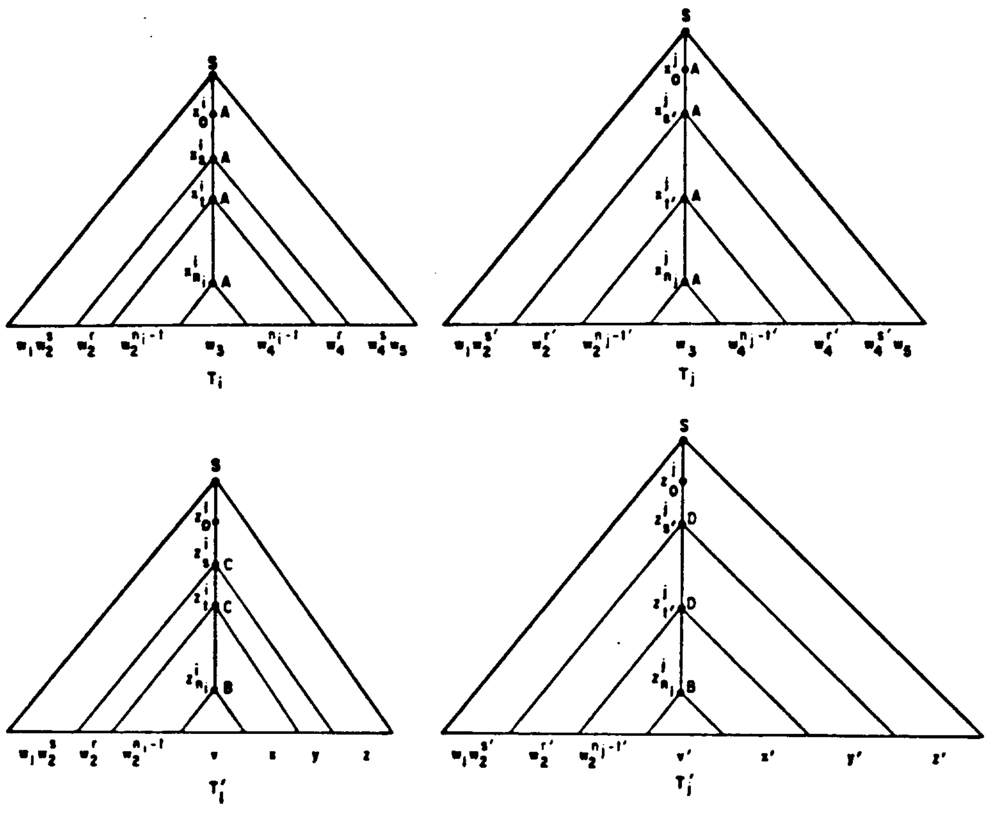


Figure 3: Derivation Trees T_i, T_j, T_i', and T_j'

Also, for $q = 0, \ldots, n_i$, $\lambda(z^i_q) = \lambda(x^i_q) = A$ so each $\lambda(z^i_q)$ is in the same equivalence class as A. Since $\|\pi\| = n$, there are at most n elements in this equivalence class, so since $n_i \geq n$, there exist $0 \leq s < t \leq n_i$ such that $\lambda(z^i_s) = \lambda(z^i_t)$. By a symmetrical argument, there exist $0 \leq s' < t' \leq n_j$ such that $\lambda(z^j_{s'}) = \lambda(x^j_{t'})$. We fix the values s,t, s', and t' for the remainder of the proof. Let $C = \lambda(z^i_s) = \lambda(z^i_t)$ and $D = \lambda(z^j_{s'}) = \lambda(z^j_{t'})$. The trees T_i, T_j, T_i', and T_j' now appear as in Figure 3.

Let η_1, η_2, η_3 be the cross sections of T_i in which only x^i_s, x^i_t, x^n_i, respectively, are internal nodes. Then

$$
\begin{aligned}
\lambda(\eta_1) = &\quad w_1 w_2^s A w_4^s w_5 \\
\lambda(\eta_2) = &\quad w_1 w_2^t A w_4^t w_5,\ t = s + r, \text{ and}
\end{aligned}
$$

$$\lambda(\eta_3) = \quad w_1 w_2^{n_i} A w_4^{n_i} w_5$$

by definition of x_s^i, x_t^i, $x_{n_i}^i$. Similarly, let η_1', η_2', η_3' be the cross sections of T_i' in which only $z_s^i, z_t^i, z_{n_i}^i$, respectively, are internal nodes.

We have already seen that $x_s^i, x_t^i, x_{n_i}^i \in {}^{[k_i+1]}T_i$. Hence, by (b), each node to the left of z_s^i (respectively $z_t^i, z_{n_i}^i$) in η_1' (respectively η_2', η_3'). Therefore, for some $x, y, z \in \Sigma^*$, we have

$$\begin{aligned}
\lambda(\eta_1') &= \quad w_1 w_2^s C z \\
\lambda(\eta_2') &= \quad w_1 w_2^t C y z \text{ and} \\
\lambda(\eta_3') &= \quad w_1 w_2^{n_i} B x y z.
\end{aligned}$$

Let v be the frontier of the tree rooted at $z_{n_i}^i$ and let $r = t - s$ (hence $1 \leq r \leq n_i$). From η_1', η_2', η_3' we obtain the following derivation:

$$\begin{aligned}
S \overset{*}{\Rightarrow} w_1 w_2^s C z \overset{*}{\Rightarrow} w_1 w_2^s w_2^r C y z \quad &\Rightarrow^+ \\
w_1 w_2^s w_2^r w^{n_i-(s+r)} B x y z \quad &\Rightarrow^+ \quad w_1 w_2^s w_2^r w^{n_i-(s+r)} v x y z \qquad (10)
\end{aligned}$$

Thus, $\xi = (v, x, y, z)$ is a factorization of u_i. Also, from (10) we see that

$$w_1 w^{(n_i-r)+mr} v x y^m z \in L$$

for all $m \geq 0$, which satisfies part of 4(a).

The arguments of the last two paragraphs apply if we use T_j' instead of T_i'. Hence, there exist $1 \leq r' \leq n_j, v', x', y', z' \in \Sigma^*$ such that $\xi' = (v', x', y', z')$ is a factorization of u_j and

$$\begin{aligned}
S \overset{*}{\Rightarrow} \quad w_1 w_2^{s'} D z' \Rightarrow^+ w_1 w_2^{s'} w_2^{r'} D y' z' \Rightarrow^+ \qquad (11) \\
w_1 w_2^{s'} w_2^{r'} w^{n_j-(s'+r')} B x' y' z' \Rightarrow^+ \qquad (12) \\
w_1 w_2^{s'} w_2^{r'} w^{n_j-(s'+r')} v' x' y' z' \qquad (13)
\end{aligned}$$

Again, from (13) we have that

$$w_1 w_2^{(n_i-r')+mr'} v' x' y'^m z' \in L$$

for all $m \geq 0$.

By substituting the last part of (13), i.e., $B \Rightarrow^* v'$, into (10), we see that

$$w_1 w_2^{(n_i - r) + mr} v x y^m z \in L$$

for all $m \geq 0$.

Similarly, by substituting $B \Rightarrow^* v$ into (13), it is clear that

$$w_1 w^{(n_i - r') + mr'} v x' y'^m z' \in L$$

for all $m \geq 0$. Thus, 4(a) holds.

Since $A \Rightarrow^* w_3$, $B \Rightarrow^* v$, $B \Rightarrow^* v'$, and $A \equiv B \pmod{\pi}$, none of w_3, v, v' is a proper prefix of any of w_3, v, v', by Lemma 2.2. This establishes 4(ii), completing the proof of Theorem 3.2.

Definition 3.3 *For $n \geq 1$, let L_n denote the context-free language*

$$\{a^m b^k a^m b^k \mid 1 \leq m, 1 \leq k \leq n\}.$$

In [7], a hierarchy of strict deterministic languages by degree is established by proving that, for $n > 1$, L_n is not a strict deterministic language of degree $n-1$ (or less). The proof there is quite complicated but elementary. Using Theorem 3.2, we give a short proof of the same result.

Theorem 3.3 *For all $n > 1$, L_n is not strict deterministic of degree $n-1$.*

Proof. Assume for the sake of contradiction that L_n is strict deterministic of degree $n-1$. Let p be the constant of Theorem 3.2. Let $w = a^p b^n a^p b^n$ and let the leftmost block of p a's be distinguished. By invoking Theorem 3.2, we obtain a factorization $\varphi = (w_1, w_2, w_3, w_4, w_5)$ of w such that parts (1) through (4) hold. In order to satisfy (1), (2), and (3), we must have $w_1 = a^s$, $w_2 = a^t$, $w_3 \in a^{p-(s+t)} b^n a^*$, $w_4 = a^t$, and $w_5 \in a^* b^n$, for some $s, t \geq 1$.

Now let

$$u_i = a^{p-(s+t)} b^i a^{p+(n-2)t} b^i$$

for $1 \leq i \leq n$. Clearly $w_1 w_2^{n-1} u_i \in L$ for $1 \leq i \leq n$, so by part (4) of the theorem, there exist $1 \leq i < j \leq n$, $1 \leq r$, $r' \leq n-1$, and factorizations $\xi = (v, x, y, z)$ and $\xi' = (v', x', y', z')$ of u_i and u_j respectively, such that (4a) and (4b) hold. Since v is a prefix of u_i and v' is a prefix of u_j, and, by (4b) neither v nor v' is a proper prefix of w_3, it must be the case that $v \in a^p - (s+t)b^i a^+$ and $v' \in a^{p-(s+t)} b^j a^+$.

Observe that, by (4a), $w_1 w_2^{(n-1-r)+mr} v x y^m z \in L$ for all $m \geq 0$. Since $w_2 \neq \Lambda$ and $r \geq 1$, this implies that $y \in a^+$. Similarly, we must have $y' \in a^+$. Thus, neither v nor v' can include the entire block of $p + (n-2)t$ a's in u_i or u_j, respectively. By 4(a), with $m = 1$, $w_1 w_2^{n-1} v' x y z \in L$. However, since $w_1 w_2^{n-1} v' x y z \in a^* b^j a^* b^i$ and $i \neq j$, this is impossible. Therefore, L_n is not strict deterministic of degree $n-1$.

4 The LL Languages

Our next definition combines some special notations which are commonly used in studying parsing. First we give a classical definition.

Definition 4.1 *A context free grammar $G = (V, \Sigma, P, S)$ is $LL(k)$ if for any $A \in N; w, x, y \in \Sigma^*$; $\gamma, \beta, \beta' \in V^*$; and any two derivations*

$$S \overset{*}{\underset{L}{\Rightarrow}} wA\gamma \underset{L}{\Rightarrow} w\beta\gamma \overset{*}{\underset{L}{\Rightarrow}} wx$$

$$S \overset{*}{\underset{L}{\Rightarrow}} wA\gamma \underset{L}{\Rightarrow} w\beta'\gamma \overset{*}{\underset{L}{\Rightarrow}} wy$$

for which

$$^{(k)}x = ^{(k)}y$$

we necessarily have $\beta = \beta'$. A language is $LL(k)$ if it is generated by some $LL(k)$ grammar.

The following easy theorem is little more than a restatement of the previous definition.

Theorem 4.1 *Let $G = (V, \Sigma, P, S)$ be a reduced context free grammar. G is an $LL(k)$ grammar if and only if for any $A \in N; w, x, y \in \Sigma^*$; $\gamma, \beta, \beta' \in V^*$; and any two derivations*

$$S \overset{n}{\underset{L}{\Rightarrow}} wA\gamma \underset{L}{\Rightarrow} w\beta\gamma \overset{*}{\underset{L}{\Rightarrow}} wx$$

$$S \overset{n}{\underset{L}{\Rightarrow}} wA\gamma \underset{L}{\Rightarrow} w\beta'\gamma \overset{*}{\underset{L}{\Rightarrow}} wy$$

for which

$$^{(k)}x =^{(k)} y$$

we necessarily have $\beta = \beta'$.

We shall now state three easy but useful properties of LL grammars.

Theorem 4.2 *Let* $G = (V, \Sigma, P, S)$ *be an LL(k) grammar. Then G is unambiguous.*

Theorem 4.3 *Let* $G = (V, \Sigma, P, S)$ *be a reduced LL(k) grammar. Then G has no left recursive variables.*

The following result asserts that we may, in fact, ignore γ.

Theorem 4.4 *Let* $G = (V, \Sigma, P, S)$ *be a context free grammar. G is an LL(k) grammar if and only if for each* $w, x, y \in \Sigma^*$; $A \in N$; $\beta, \beta', \gamma, \gamma' \in V^*$, *and any two derivations*

$$S \underset{L}{\overset{*}{\Rightarrow}} wA\gamma \underset{L}{\Rightarrow} w\beta\gamma \underset{L}{\overset{*}{\Rightarrow}} wx$$

$$S \underset{L}{\overset{*}{\Rightarrow}} wA\gamma' \underset{L}{\Rightarrow} w\beta'\gamma' \underset{L}{\overset{*}{\Rightarrow}} wy$$

if $^{(k)}x =^{(k)} y$ *then* $\beta = \beta'$.

4.1 An LL Left Part Theorem

In order to work with the *LL* languages, it is essential to find a "left part theorem." This turns out to be possible but the concepts change in an interesting way.

Definition 4.2 *Let T be a grammatical tree, and let* $m = \mathrm{lg}(\mathrm{fr}(T))$. *Let* $(y_1, \ldots, y_m)$ *be a left-to-right sequence of all terminal nodes in T. For any n,* $1 \leq n \leq m$, *define the tree*

$$\begin{aligned} {}^{\{n\}}T &= \{x \in T \mid x \overset{*}{\rightharpoondown} \overset{*}{\triangleright} y_n\} \\ &\cup \{x \in T \mid \text{ there exists } b \in T \text{ such that } b \overset{*}{\triangleright} y_n\,,\ b \sqcap^+ x\}. \end{aligned}$$

Also, let ${}^{\{0\}}T$ *be the empty tree, and let* ${}^{\{n\}}T = T$ *if* $n > m$. *We also call* ${}^{\{n\}}T$ *a left n-part.*

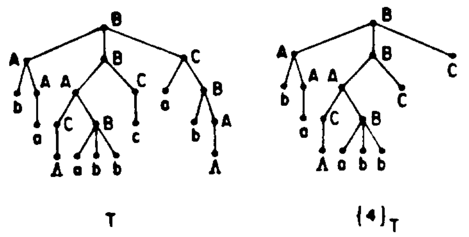


Figure 4: A tree T and ${}^{\{4\}}T$

${}^{\{n\}}T$ contains all nodes in ${}^{[n]}T$, and in addition contains all immediate descendants of nodes on the path from $\mathrm{rt}(T)$ to y_n. Figure 4.1 shows a grammatical tree T and ${}^{\{4\}}T$.

The first preliminary result is analogous to a result about LR grammars from [8].

Theorem 4.5 *(The Extended LL(k) Theorem) Let* $G = (V, \Sigma, P, S)$ *be an* $LL(k)$ *grammar. For any* $A \in N$; $w, x, y \in \Sigma^*$; *and* $\varphi \in V^*$, *if*

$$\begin{array}{lllll} S & \Rightarrow_L^* & wA\gamma & \Rightarrow_L^* & wx \\ S & \Rightarrow_L^* & & & wy \\ {}^{(k)}x & = & {}^{(k)}y & & \end{array}$$

then

$$S \Rightarrow_L^* wA\gamma \Rightarrow_L^* wy.$$

Proof. The argument is an easy proof by contradiction which is omitted.

Now let us begin to motivate the kind of left part theorem which is needed for the LL grammars. Let G be an $LL(k)$ grammar and suppose that wx and wy are in $L(G)$ with ${}^{(k)}x ={}^{(k)} y$. Let T^{wx} and T^{wy} denote the derivation trees. One essential part of this notion is that the portion of these trees which are filled in at the time that the last symbol of w is exposed in the leftmost derivation of wx and wy will be the same.

Let us state the full result.

Theorem 4.6 *(The LL(k) Left Part Theorem) A reduced context free grammar* $G = (V, \Sigma, P, S)$ *is* $LL(k)$ *if and only if the following conditions hold for all* $n \geq 0$: *If* T *and* T' *are grammatical trees over* G *such that*

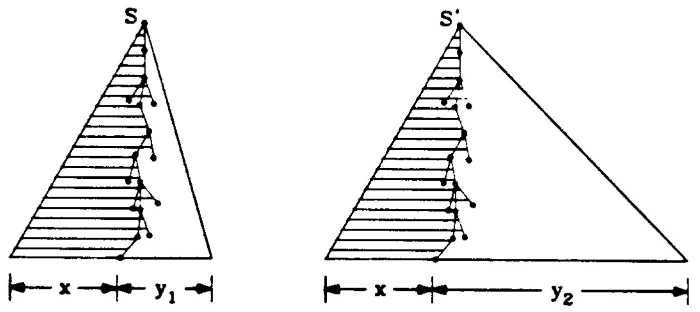


Figure 5: Example of the LL Left Part Theorem

1. $\mathrm{rt}(T) = \mathrm{rt}(T')$

2. ${}^{(n+k)}\mathrm{fr}(T) = {}^{(n+k)}\mathrm{fr}(T')$

then ${}^{\{n+1\}}T = {}^{\{n+1\}}T'$.

Before dealing with the proof, let us examine Figure 4.1. The left $\{\lg(x) + 1\}$-parts of these trees for xy, and xyz are shaded. These left parts have been filled in when all of $x({}^{(1)}y_1)$ and $x({}^{(1)}y_2)$ were exposed. If the grammar is $LL(k)$ and ${}^{(k)}y_1 = {}^{(k)}y_2$ then these leftparts are identical.

The detailed proof of this theorem is in [5], but we will discuss it briefly. The forward direction is a straightforward and tedious induction. The reverse direction, usually done by contradiction, involves a careful analysis of the leftmost sections of derivation trees. It is these properties which can be used with the $LL(k)$ property to derive a contradiction.

4.2 The First LL Iteration Theorem

The following notation is convenient for repeated concatenation.

Definition 4.3 *Let* $u_i \in \Sigma^*$, $1 \leq i \leq r$, *for some alphabet* Σ. *Then*

$$\Pi_{i=1}^{r}(u_i) = u_1 u_2 \cdots u_{r--1} u_r$$

We are now ready to proceed to our first Iteration Theorem for the LL languages.

Theorem 4.7 *Let L be an LL(k) language. There exists an integer p such that, for each $w \in L$ and each set K of p or more distinguished positions in w, there is a factorization $\varphi = (w_1, w_2, w_3, w_4, w_5)$ of w such that*

1. $w_2 \neq \Lambda$,
2. *if* $K/\varphi = \{K_1, \dots K_5\}$ *then*
   (a) *either* $K_1, K_2, K_3 \neq \emptyset$ *or* $K_3, K_4, K_5 \neq \emptyset$,
   (b) $|K_1 \cup K_3 \cup K_4| \leq p$,
3. (a) *Let* $n = \lg(w_1 w_2)$ *and let* $w' \in L$ *such that* ${}^{(n+k)}w = {}^{(n+k)}w'$ *Then there is a factorization* $(w_1, w_2, w_3', w_4', w_5')$ *of* w *such that*
   i. $w_1 w_2{}^n w_3 (\prod_{i=1}^n u_i) w_5$
   ii. $w_1 w_2{}^n w_3' (\prod_{i=1}^n u_i) w_5$
   iii. $w_1 w_2{}^n w_3 (\prod_{i=1}^n u_i) w_5'$
   iv. $w_1 w_2{}^n w_3' (\prod_{i=1}^n u_i) w_5'$

   are all in L *for all* $n \geq 0$ *and for all strings* $\prod_{i=1}^n (u_i)$ *in which* $u_i = w_4$ *or* $u_i = w_4'$ *for any* $i, 1 \leq i \leq n$.

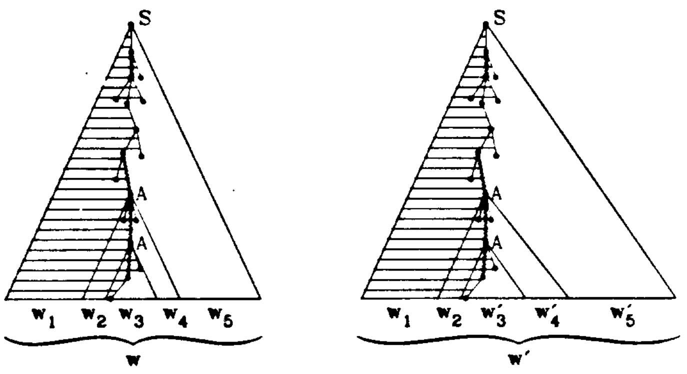
   (b) *Moreover if* $\prod_{i=1}^n (\overline{u}_i)$ *is a concatenation of words* $\overline{u}_i \in \{w_4, w_4'\}$ *such that*
$$\prod_{i=1}^{n} (u_i) = \prod_{i=1}^{n} (\overline{u}_i)$$
   then $u_i = \overline{u}_i$ *for all* $i, 1 \leq i \leq n$.

Proof. Let $G = (V, \Sigma, P, S)$ be an arbitrary reduced $LL(k)$ grammar generating L. The standard methods establish the existence of an integer p such that for any string w in L in which p or more positions K are distinguished, there is a factorization $\varphi = (w_1, w_2, w_3, w_4, w_5)$ of w such that (2) holds and for some variable $A \in N$ for which $A \Rightarrow^+ w_2 A w_4$ we have

$$S \overset{*}{\Rightarrow} w_1 A w_5 \overset{*}{\Rightarrow} w_1 w_2^r A w_4^r w_5 \overset{+}{\Rightarrow} w_1 w_2^r w_3 w_4^r w_5$$

for all non-negative integers r. Since no $LL(k)$ grammar is left recursive (1) holds. To complete our proof we must show that φ satisfies (3) as well.

We shall just sketch the intuitive idea of the remainder of the proof. The reader is referred to [5] for the details.



S
A
A
w_1 w_2 w_3 w_4 w_5
w
S
A
A
w_1 w_2 w_3' w_4' w_5'
w'

Figure 6: Derivation trees for w and w'

The actual idea is to use the second left part theorem as shown in Figure 4.2.

In these trees for w and w', the left $(\lg(w_1w_2) + 1)$ parts are shaded. Since G is $LL(k)$ and since

$$^{(k+\lg(w_1w_2))}(w_1w_2w_3w_4w_5) =^{(k+\lg(w_1w_2))} k(w_1w_2w_3'w_4'w_5')$$

the respective left parts are isomorphic. Note that the two nodes labeled A in $^{\{n+1\}}T$ must appear in the same position in $^{\{n+1\}}T'$.

The isomorphism between $^{\{n+1\}}T$ and $^{\{n+1\}}T'$ forces a correspondence between certain of the strings by an argument similar to that used in the strict deterministic case. But a straightforward application of the $LL(k)$ condition derives the restriction on the concatenation. That is, it is easy to see from T that we have derivations.

$$S \Rightarrow_L^* w_1 A\beta \tag{14}$$
$$A \Rightarrow_L^* w_2 A\alpha \tag{15}$$
$$A \Rightarrow_L^* w_3 \tag{16}$$
$$\alpha \Rightarrow_L^* w_4 \tag{17}$$
$$\beta \Rightarrow_L^* w_5 \tag{18}$$

and from T' the derivations

$$S \Rightarrow_L^* w_1 A\beta \tag{19}$$
$$A \Rightarrow_L^* w_2 A\alpha \tag{20}$$
$$A \Rightarrow_L^* w_3' \tag{21}$$
$$\alpha \Rightarrow_L^* w_4' \tag{22}$$
$$\beta \Rightarrow_L^* w_5' \tag{23}$$

for some terminal strings w_3', w_4' and w_5' such that $w_1 w_2 w_3' w_4' w_5' = w$. By suitably combining these derivations we can obtain any of the strings specified in (3a). For example, to obtain strings of the form

$$w_1 w_2^r w_3 \Pi_{i=1}^r (u_i) w_5$$

we begin with (9), followed by r applications of (10), followed by (11), followed by a suitable mixture of (12) and (17), and finish with (13).

Next we establish (3b). If $w_4 = w_4'$, then (3b) follows trivially. Therefore, assume that $w_4 \neq w_4'$, so that (12) and (17) are distinct leftmost derivations, neither of which is a prefix of the other. For the sake of simplicity we restrict our attention now to strings of type (i). Let R be the set

$$\{(9)\}\{(10)\}^r\{(11)\}\{(12)+(17)\}^r\{(13)\}$$

Notice that a string in R uniquely specifies the leftmost derivation of a type (i) word in L. In particular, let π_i, $1 \leq i \leq r$, be defined by

$$\pi_i = (12) \text{ if } u_i = w_4$$
$$\pi_i = (17) \text{ if } u_i = w_4'$$

Then given a string of type (i), which determines a sequence π_i,

$$\{(9)\}\{(10)\}^r\{(11)\}\Pi_{i=1}^r\{\pi_i\}\{(13)\}$$

is a leftmost derivation of the word. If there exist *two* catenations

$$\Pi_{i=1}^r(u_i) \text{ and } \Pi_{i=1}^r(\overline{u_i})$$

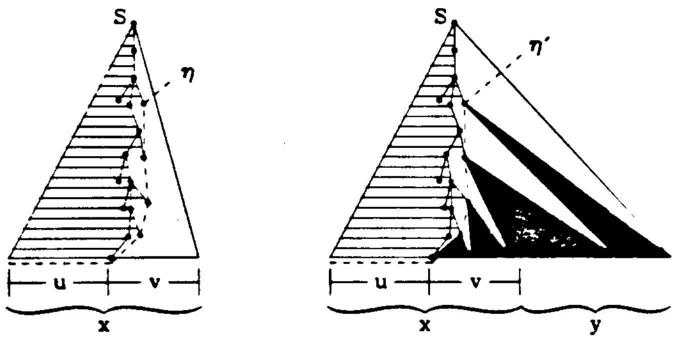


Figure 7: The Motivation for The Second LL Iteration Theorem.

and corresponding sequences π_i and $\overline{\pi_i}$ such that

$$\Pi_{i=1}^{r}(u_i) = \Pi_{i=1}^{r}(\overline{u_i})$$

and for which $u_i \neq \overline{u_i}$, for some i in the range $1 \leq i \leq r$, so that $\pi_i \neq \overline{\pi_i}$, then there are two distinct strings in R, representing two distinct leftmost derivations of the same string in L. But then G is an ambiguous grammar, which cannot be the case since G is $LL(k)$. Hence (3b) follows for a string of type (i).

We can extend (3b) to strings of type (ii), (iii) and (iv) by analogous arguments. The details are omitted.

4.3 The Second LL Iteration Theorem

The intuition behind our second LL iteration theorem is different. (Refer to Figure 4.3. Suppose that uv and uvy, $\lg(v) = k$, are strings in some language L generated by a Λ-free $LL(k)$ grammar G. Leftmost derivations of uv and uvy must proceed identically at least until all of u has been exposed; That is the meaning of the Extended $LL(k)$ Theorem. After exposing the rightmost terminal of u in a leftmost derivation of either uv of uvy there can be no more than k variables remaining in the left sentential form since G is Λ-free and $\lg(v) = k$. Judicious use of this fact, together with the Left Part Theorem and the argument of the First Iteration Theorem, is sufficient for our purposes.

We will need the following result from [21].

Theorem 4.8 *Given an $LL(k)$ grammar $G = (V, \Sigma, P, S)$ we can construct an $LL(k{+}1)$ grammar $G' = (V', \Sigma, P', S')$ such that $L(G') = L(G)$ and G' is Λ-free unless $\Lambda \in L(G)$, in which case P' contains the single Λ-rule $S' \to \Lambda$ and S' does not appear in the right-hand side of any rule in P'.*

Proof. Using the arguments found in [22], pages 236-241 , we may obtain a Λ-free $LL(k+1)$ grammar $G'' = (V'', \Sigma, P'', S'')$ generating $L(G) - \{\Lambda\}$. If $\Lambda \notin L(G)$ then set $G' = G''$.

Suppose, however, that $L(G)$ contains Λ. Then we form a new grammar G' whose start symbol is S' and whose rules are the rules of G'' together with $S' \to S'' \mid \Lambda$, where S' is a new variable not in $V'' - \Sigma$. It is trivial to prove that G' is also $LL(k+1)$ and generates exactly $L(G)$.

We are now in a position to state the second LL Iteration Theorem.

Theorem 4.9 *(The Second LL Iteration Theorem) Let L be an $LL(k-1)$ language, $k \geq 1$. There exists an integer p such that for any two distinct strings x and xy in L, if $\lg(x) \geq k$ and p or more positions in y are distinguished, then there is a factorization $\varphi = (w_1, w_2, w_3, w_4, w_5)$ of xy such that (1) - (3) of the First LL Iteration Theorem hold and $\lg(w_1) \geq \lg(x) - k$.*

Proof. In view of Theorem 4.5 we may assume that L is generated by some $LL(k)$ grammar $G = (V, \Sigma, P, S)$ which is Λ-free, except possibly for an $S \to \Lambda$ rule, in which case S does not appear in any right-hand side.

For any variable A let $G_A = (V, \Sigma, P, A)$ be the context free grammar obtained from G by changing the start symbol to A, let p_A be the constant obtained from the First LL Iteration Theorem for the language $L(G_A)$ (which is also $LL(k)$ — see Theorem 1.8 of [5]), and let

$$\begin{aligned} p' &= \max\{p_A \mid A \in V - \Sigma\} \\ p &= kp' + 1 \end{aligned}$$

Suppose that x and xy are strings belonging to L, where $\lg(x) \geq k$ and p or more positions are distinguished in y. Let us write $x = uv$, where $\lg(u) = n$ and $\lg(v) = k$, and let T and T' be derivation trees for uv and uvy. (See

Figure 4.3) Let [2] $\eta = \text{leaves}({}^{(n+1)}T)$ and $\eta' = \text{leaves}({}^{(n+1)}T')$.

Since ${}^{(n+k)}x ={}^{(n+k)} (xy) = x$, it follows from the Left Part Theorem that ${}^{(n+1)}T ={}^{(n+1)} T'$, whence η and η' are isomorphic and $\lambda(\eta) = \lambda(\eta')$. It follows from Theorem 2.17 that η and η' are left canonical cross sections of T and T', respectively. Consequently we may write

$$S \underset{L}{\overset{*}{\Rightarrow}} u\gamma = \lambda(\eta) \underset{L}{\overset{*}{\Rightarrow}} uv$$
$$S \underset{L}{\overset{*}{\Rightarrow}} u\gamma = \lambda(\eta') \underset{L}{\overset{*}{\Rightarrow}} uvy$$

for some γ in V^* (fact 2.8 of [5]). Since $\lg(v) = k \geq 1$ these derivations involve no Λ-rules. It follows that $\lg(\gamma) \leq k$ since $\lg(v) = k$ and $\gamma \Rightarrow^*{}_L v$.

Now write

$$\gamma = X_1X_2\cdots X_s \quad (s \leq k)$$

Let $(z_1, z_2, \cdots, z_s)$ be the factorization of vy such that $X_i \Rightarrow^*{}_L z_i$ for each i, $1 \leq i \leq s$. Suppose that there are p' or fewer distinguished positions in each z_i. Then there are at most $sp' \leq kp' < ip$ distinguished positions in vy, which is not the case. Hence some particular z_i contains more than $p' \geq p_{X_i}$ distinguished positions. Now the string z_i belongs to the language $L(G_{X_i})$, which (as we noted above) is an $LL(k)$ language. Also, we have distinguished p_{X_i} or more positions in this string. It follows from the First Iteration Theorem that there is a factorization $(\sigma_1, \sigma_2\sigma_3, \sigma_4, \sigma_5)$ of z_i such that (1) – (2) of Theorem 4.7 hold with respect to $L(G_{X_i})$ and for some variable B we have $B \Rightarrow^+ \sigma_2 B \sigma_4$ and

$$X_i \overset{*}{\Rightarrow} \sigma_1 B \sigma_5 \overset{*}{\Rightarrow} \sigma_1\sigma_2^r B \sigma_4^r \sigma_5 \overset{+}{\Rightarrow} \sigma_1\sigma_2^r\sigma_3\sigma_4^r\sigma_5$$

in G_{X_i}. From this it follows that the factorization

$$(uz_1\cdots z_{i-1}\sigma_1, \sigma_2, \sigma_3, \sigma_4, \sigma_5 z_{i+1}\cdots z_s) = (w_1, w_2, w_3, w_4, w_5)$$

satisfies (1) –(2) with respect to L. Since u is necessarily a prefix of w_1 it is clear that $\lg(w_1) \geq \lg(x) - k$. If we let

$$n = \lg(uz_1\cdots z_{i-1}\sigma_1\sigma_2)$$

and consider any string w' in L such that ${}^{(n+k)}w' ={}^{(n+k)} w$, the argument used to deduce (3) in Theorem 4.7 may be used to deduce property (3) here, and the proof is complete.

[2] If T' is a subset of a tree T with leaves $x_1, \ldots, x_k$ in left to right order, i.e. $x_1 \leftarrow \cdots \leftarrow x_k$ then we write $\text{leaves}(T') = (x_1, \ldots, x_k)$.

4.4 Some Applications of LL Iteration Theorems

Of course, our iteration theorems have the natural applications of showing specific languages which are not *LL*. For example one can show that (Cf. [5])

$$L_1 = \{a^n b^n \mid n \geq 1\} \cup \{a^n c^n \mid n \geq 1\}$$

is *LR* but not *LL*. The proof is a standard factorization argument. Another example is

$$L_2 = \{a^n 0 b^n \mid n \geq 1\} \cup \{a^n 1 b^{2n} \mid n \geq 1\}$$

which is also in $LR - LL$.

The Second Iteration Theorem is by its very nature not applicable to *LL* languages which are prefix-free. Thus Theorem 4.9 could not be used to prove any that any of $L_1, L_2, or L_3$ are not LL. It is not known , however, whether there are languages which satisfy the First Iteration Theorem but which the Second Iteration Theorem can show are not *LL*, nor is it known whether one can always establish that a language fails to be *LL* *via* Theorem 4.7 when that is the case.

It is shown in [5] that

$$L_3 = \{a^m b^n \mid m \geq n \geq 0\}$$

is not an *LL* language. L_3 abstracts the fatal difficulty, insofar as $LL(k)$ grammars are concerned, with the infamous dangling-ELSE introduced by the original ALGOL report [16] and eliminated in the revised report [17]). Constructs such as

IF <bexp> THEN IF <bexp> THEN <stmt> ELSE <stmt>

in which the ELSE-clause might plausibly belong to either IF-THEN are allowed in PL/I [21] and Pascal [12]. The ambiguity is customarily resolved by associating an ELSE with the last previous unmatched THEN. It is claimed without proof in [1] that such constructs are not *LL*; applying the argument of Theorem 5.6 of [5] allows us to establish this rigorously. A direct proof such as ours is necessary since the family of *LL* languages is not closed under homomorphisms or *gsm* mappings.

Theorem 4.10 *The dangling* IF-THEN-ELSE *construct does not appear in any LL language.*

Since this construct is, however, easily handled by a recursive descent compiler operating without backup, it follows that the $LL(k)$ languages form a proper subset of the family of languages which can be compiled by this technique, and are therefore not a perfect model of this family.

Another application is quite interesting. Using the Left Part Theorem yields a rigorous and natural proof of the following theorem.

Theorem 4.11 *Every reduced LL(k) grammar is LR(k),* $k \geq 0$.

We refer the reader to [5] for a direct proof. The literature has some interesting but not rigorous proofs, eg. [2]. Brosgol [6] obtained a rigorous proof *via* $LR(k)$ grammar theory by embedding Λ-rules in the grammar, and Soisalon-Soininen has reportedly also obtained a rigorous proof [22].

5 Simple Languages

The simple languages, originally defined in [15], are important because this family is the first nontrivial class of languages for which the equivalence problem was known to be decidable.

We will need the following basic definitions.

Definition 5.1 *A context free grammar* $G = (V, \Sigma, P, S)$ *in Greibach normal form is said to be a* simple grammar *if for all* $A \in N$, $a \in \Sigma$, *and* α, $\beta \in V^*$,

$$A \rightarrow \alpha a \text{ and } A \rightarrow \alpha\beta \text{ in } P$$

imply $\alpha = \beta$. *A* simple language *is a language generated by an s-grammar.*

For example,

$$\begin{aligned} S &\rightarrow aSA \mid b \\ A &\rightarrow a \end{aligned}$$

is a simple grammar which generates the set $L(G) = \{a^n b a^n \mid n \geq 0\}$. On the other hand,

$$\begin{aligned} S &\rightarrow aAd \mid aBe \\ A &\rightarrow aAb \mid b \\ B &\rightarrow aBc \mid c \end{aligned}$$

is not a simple grammar. The language generated is

$$L = \{a^n b^n d \mid n \geq 1\} \cup \{a^n c^n e \mid n \geq 1\}.$$

It can be shown that L is not a simple language, although it is a realtime strict deterministic language. [12]

The following result gives some elementary facts about simple languages.

Theorem 5.1 *Let $L \subseteq \Sigma^*$. The following statements are equivalent.*

1. *L is simple.*
2. *L is strict deterministic of degree one.*
3. *L is accepted as $T_2(A)$ for some realtime deterministic pushdown automaton with one state or $L = \{\Lambda\}$*
4. *L is accepted as $T_2(B)$ for some deterministic pushdown automaton with one state.*

It has been shown that the family of simple deterministic languages coincides with the family of strict deterministic languages of degree 1 (except for $\{\Lambda\}$, which is not simple deterministic). Hence, Theorem 3.2 (with $n = 1$) can be used to show that a language is not simple deterministic. However, using the special properties of the simple deterministic languages, a stronger and more concise iteration theorem was established in [14]. We sketch the proof here as it illustrates the importance of our left-part results. We build on Theorem 4.6.

Theorem 5.2 *Let $G = (V, \Sigma, P, S)$ be a reduced context-free grammar in Greibach normal form. Then G is simple if and only if*

> *($*$) for any $n \geq 0$ and any grammatical trees T and T' over G, if $\mathrm{rt}(T) = \mathrm{rt}(T')$ and if ${}^{(n)}\mathrm{fr}(T) ={}^{(n)} \mathrm{fr}(T')$ then ${}^{\{n\}}T ={}^{\{n\}} T'$.*

Proof. Suppose that G is simple deterministic. Every simple deterministic grammar is $LL(1)$ [13], so by Theorem 4.6, for any $n \geq 0$ and any grammatical trees T, T', if $\mathrm{rt}(T) = \mathrm{rt}(T')$ and ${}^{(n+1)}\mathrm{fr}(T) ={}^{(n+1)} \mathrm{fr}(T')$, then ${}^{\{n+1\}}T ={}^{\{n+1\}} T'$. Since ${}^{\{0\}}T ={}^{\{0\}} T'$ for any T, T', we can replace $n + 1$ by n to get ($*$).

Conversely, suppose that (*) holds. Then, for any $n \geq 0$ and any grammatical trees T,T', if $\mathbf{rt}(T) = \mathbf{rt}(T')$ and ${}^{(n+1)}\mathbf{fr}(T) ={}^{(n+1)} \mathbf{fr}(T')$, then ${}^{(n+1)}T ={}^{(n+1)} T'$. Hence, by Theorem 4.6, G is $LL(1)$. By [21] G is simple deterministic, since G is $LL(1)$ and in Greibach normal form.

We now prove an iteration theorem for simple deterministic languages.

Theorem 5.3 *Let L be a simple deterministic language. There exists an integer p such that, for each $w \in L$ and each set K of p or more distinguished positions in w, there is a factorization $\varphi = (w_1, w_2, w_3, w_4, w_5)$ of w such that*

1. *$w_2 \neq \Lambda$,*
2. *if $K/\varphi = \{K_1, \ldots K_5\}$ then*
   (a) *either $K_1, K_2, K_3 \neq \emptyset$ or $K_3, K_4, K_5 \neq \emptyset$,*
   (b) *$|K_1 \cup K_3 \cup K_4| \leq p$,*
3. *for each $u \in \Sigma^*$, if $w_1 w_2 u \in L$, then there exists a factorization $\xi = (w_1, w_2, w_3', w_4', w_5')$ of $w_1 w_2 u$ such that*
   (a) *for each $n \geq 0$, for each $u_1, \ldots, u_n \in \{w_4, w_4'\}$, the following are all in L :*

$$w_1 w_2{}^n w_3 (\textstyle\prod_{i=1}^{n} u_i) w_5 \qquad w_1 w_2{}^n w_3' (\displaystyle\prod_{i=1}^{n} u_i) w_5$$

$$w_1 w_2{}^n w_3 (\textstyle\prod_{i=1}^{n} u_i) w_5' \text{ and } \quad w_1 w_2{}^n w_3' (\displaystyle\prod_{i=1}^{n} u_i) w_5'$$

   (b) *w_3 (respectively w_4, w_5) is not a proper prefix of w_3' (respectively w_4', w_5') and vice -versa.*

Our proof is similar to the proof of Theorem 4.6. Let $G = (V, \Sigma, P, S)$ be a reduced simple deterministic grammar such that $L = L(G)$. Thus, as we noted in Section 2, G is strict deterministic. The proof of Theorem 2.4 in [11] shows that there exists an integer p such that, for each $w \in L$ and each set K of p or more distinguished positions in w, there is a factorization $\varphi = (w_1, w_2, w_3, w_4, w_5)$ of w such that (1) and (2) of the theorem hold, and such that, for some $A \in V - \Sigma$,

$$S \overset{*}{\Rightarrow} w_1 A w_5 \overset{+}{\Rightarrow} w_1 w_2 A w_4 w_5 \overset{+}{\Rightarrow} w_1 w_2 w_3 w_4 w_5 = w \ . \tag{24}$$

We must now show that φ satisfies part 3 of the theorem. Let T be a derivation tree corresponding to (1). Let x (respectively. y) be the node of T labeled by A in the cross section of T labeled by w_1Aw_5 (respectively. $w_1w_2Aw_4w_5$). Clearly $x \rhd^* y$.

Suppose that $w_1w_2u \in L$ for some $u \in \Sigma^*$. Let T' be a derivation tree corresponding to $S \Rightarrow^* w_1w_2u$. Thus, $\mathrm{rt}(T') = S$ and $\mathrm{fr}(T') = w_1w_2u$. Let $k = \lg(w_1w_2)$. Since $\mathrm{rt}(T) = \mathrm{rt}(T') = S$ and ${}^{(k)}\mathrm{fr}(T) ={}^{(k)} \mathrm{fr}(T') = w_1w_2$, we have by Theorem 5.2 that ${}^{\{k\}}T ={}^{\{k\}} T'$. Let h be the isomorphism that maps nodes of ${}^{\{k\}}T$ to nodes of ${}^{\{k\}}T'$.

Let y_k (respectively. y_{k+1}) denote the leaf of T labeled by the kth (respectively. $(k+1)^{st}$) symbol in $w = w_1w_2w_3w_4w_5$. Since $w_2 \neq \Lambda$ by part 1 of the theorem, and $w_3 \neq \Lambda$ by part 2, y_k is labeled by the last symbol of w_2, and y_{k+1} is labeled by the first symbol of w_3. By the definition of x and y, we have that $x \rhd^* y_k$ and $y \rhd^* y_{k+1}$.

Since y_k and y_{k+1} are leaves of T, we have that $y_k \leftharpoondown^+ y_{k+1}$. Suppose that there exists a leaf $y' \in T$ such that $y_k \leftharpoondown^+ y' \leftharpoondown^+ y_{k+1}$. Since G is in Greibach normal form, y' is labeled by some $a \in \Sigma$. But then y_{k+1} cannot be labeled by the $(k+1)^{st}$ symbol in w, which is a contradiction. Therefore, $y_k \leftharpoondown y_{k+1}$.

By the definition of $\leftharpoondown$, there exist $z_1, z_2 \in T$ such that

$$y_k(\overset{*}{\rhd}{}_R^{-1})z_1 \sqcap z_2(\overset{*}{\rhd}{}_L)y_{k+1} .$$

Let z be the parent of z_1 and z_2 (see Figure 5). Since $y \rhd^* y_{k+1}$, either $y \rhd^* z$ or $z_2 \rhd^* y$. Suppose that $y \rhd^* z$. Then $y \rhd^+ z_1 \rhd^* y_k$, which is impossible since both y_k and y appear in the cross section of T labeled by $w_1w_2Aw_4w_5$. Thus, it must be the case that $z_2 \rhd^* y$.

Suppose that $z_2 \neq y$. Let z' be the leftmost immediate descendant of z_2. Since G is in Greibach normal form, z' is labeled by some $a \in \Sigma$. Since $z_2 \rhd^* y$ and $z_2 \neq y$, we have that $z' \rhd^* y$. However, $z' \neq y$ since $\lambda(y) = A$, and z' has no descendants, so it is not possible that $z' \rhd^* y$, which is a contradiction. Hence, $z_2 = y$. Since $z_1 \rhd^* y_k$ and $z_1 \sqcap y$, $y \in {}^{\{k\}}T$.

Since $x \rhd^* y_k$, $x \in{}^{\{k\}} T$. Thus, both x and y are in ${}^{\{k\}}T$, so we have that

$$A = \lambda(x) = \lambda(h(x)) \text{ and } A = \lambda(y) = \lambda(h(y)) .$$

Let η and θ be left cross sections of T in which the leftmost internal nodes are x and y, respectively. Such left cross sections must exist by Lemma 3.4

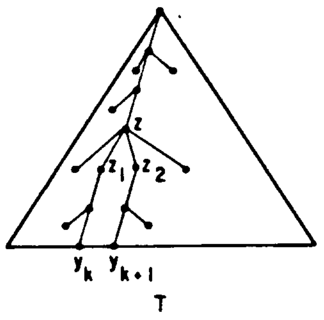


Figure 8: The Derivation tree T.

of [14]. From the definition of x and y, we see that

$$\begin{aligned}\lambda(\eta) &= w_1 A\beta \text{ and} \\ \lambda(\theta) &= w_1 w_2 A\alpha\beta\end{aligned}$$

for some α, $\beta \in V^*$.

Since x and y are internal nodes of T that belong to ${}^{\{k\}}T$, η and θ are left cross sections of ${}^{\{k\}}T$ (Lemma 3.6 of [14]). But ${}^{\{k\}}T =^{\{k\}} T'$, so $h(\eta)$ and $h(\theta)$ are left cross sections of ${}^{\{k\}}T'$, hence by Lemma 3.7 of [14], $h(\eta)$ and $h(\theta)$ are left cross sections of T'. Since ${}^{\{k\}}T =^{\{k\}} T'$,

$$\begin{aligned}\lambda(h(\eta)) &= \lambda(\eta) = w_1 A\beta \text{ and} \\ \lambda(h(\theta)) &= \lambda(\theta) = w_1 w_2 A\alpha\beta\,.\end{aligned}$$

Applying Lemma 3.5 of [14] to η and θ, we have

$$S \overset{*}{\Rightarrow} w_1 A\beta \overset{*}{\Rightarrow} w_1 w_2 A\alpha\beta \overset{*}{\Rightarrow} w_1 w_2 w_3 w_4 w_5 \tag{25}$$

(where $A \Rightarrow^* w_3$, $\alpha \Rightarrow^* w_4$, and $\beta \Rightarrow^* w_5$, by the definition of α ad β). Next, we apply Lemma 3.5 of [14] to $h(\eta)$ and $h(\theta)$ to get

$$S \overset{*}{\Rightarrow} w_1 A\beta \overset{*}{\Rightarrow} w_1 w_2 A\alpha\beta \overset{*}{\Rightarrow} w_1 w_2 u.$$

Let w_3', w_4', $w_5' \in \Sigma^*$ be such that $u = w_3' w_4' w_5'$, $A \Rightarrow^* w_3'$, $\alpha \Rightarrow^* w_4'$, and $\beta \Rightarrow^* w_5'$. (See Figure 5)

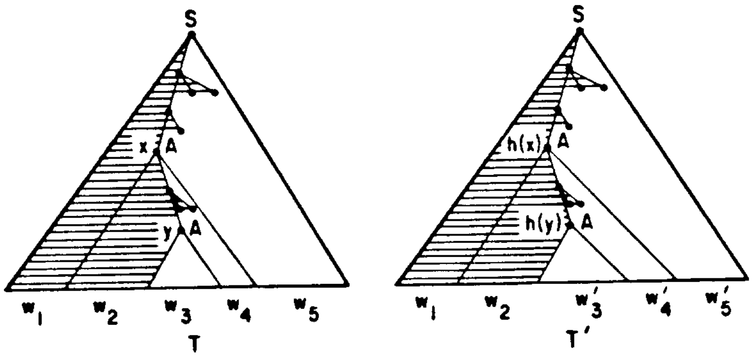


Figure 9: Derivation trees T and T'.

Setting $\xi = (w_1, w_2, w_3', w_4', w_5')$, we have that ξ is a factorization of w_1w_2u. Let n be any nonnegative integer. From (25) , we obtain the derivation

$$S \overset{*}{\Rightarrow} w_1A\beta \overset{*}{\Rightarrow} w_1w_2A\alpha\beta \overset{*}{\Rightarrow} w_1w_2^2A\alpha^2\beta \overset{*}{\Rightarrow} \cdots \overset{*}{\Rightarrow} w_1w_2^nA\alpha^n\beta.$$

A terminal string may now be derived by continuing with either

$$A \overset{*}{\Rightarrow} w_3 \text{ or } A \overset{*}{\Rightarrow} w_3',$$

then n applications of any combination of

$$\alpha \overset{*}{\Rightarrow} w_4 \text{ and } \alpha \overset{*}{\Rightarrow} w_4',$$

completing the derivation with either

$$\beta \overset{*}{\Rightarrow} w_5 \text{ or } \beta \overset{*}{\Rightarrow} w_5'.$$

Clearly any of the strings in part 3(i) of the theorem may be obtained in this manner, so ξ satisfies 3(i).

Since G is strict deterministic, by Lemma 2.2 each of $L(A)$, $L(\alpha)$ and $L(\beta)$ is a prefix-free set. Thus, since w_3, $w_3' \in L(A)$, w_3 is not a proper prefix of w_3' and vice-versa. Similarly, w_4 (respectively. w_5) is not a proper prefix of w_4' (respectively. w_5'), and vice-versa. Therefore, ξ satisfies part 3(ii), and the theorem is proved.

Theorem 5.3 resembles Theorem 4.7 for LL(k) languages in the case that $k = 1$. This is understandable, since every simple deterministic language is LL(1). There are two differences between the theorems, however. First, condition 3 in Theorem 5.3 requires only a string $w_1w_2u \in L$, while Theorem 4.7 requires that we have a string $w_1w_2u \in L$ such that ${}^{(1)}u ={}^{(1)} w_3$. Second, part 3(ii) in Theorem 5.3 is stronger than the corresponding condition in Beatty's theorem. In fact, part 3(ii) is very useful in practice, as we see in the following example.

Theorem 5.4 *The language* $L_1 = \{a^n(bd \cup b \cup c)^n\$ \mid n \geq 1\}$ *(where* (,), *and* $\cup$ *are metasymbols,* $\cup$ *denoting alternation) is not simple deterministic.*

We omit the proof which is rather technical and is available in [14].

The language L_1 above is a variation on the $LL(k)$ language

$$\{a^n(b^kd + b + cc)^n \mid n \geq 1\}$$

(where k is any fixed value greater than or equal to 1) which cannot be generated by an LL(k) grammar without Λ-rules, cf. [20] Since the class of simple deterministic languages is equal to the class of language generated by LL(1) grammars without Λ-rules [12], their result shows that

$$\{a^n(bd + b + cc)^n \mid n \geq 1\}$$

is not a simple deterministic language. This also follows trivially from the observation that

$$\{a^n(bd + b + cc)^n \mid n \geq 1\}$$

is not prefix-free. Thus, the added \$ is essential in Theorem 6.4. Note also that each of the alternates (bd, b, and c) in L_1 is necessary for L_1 to be nonsimple. An interesting exercise is to verify that the languages

$$\{a^n(bd + b)^n\$ \mid n \geq 1\},$$

$$\{a^n(bd + c)^n\$ \mid n \geq 1\},$$

and

$$\{a^n(b + c)^n\$ \mid n \geq 1\}$$

are all simple deterministic.

We have noted earlier that every simple deterministic language is both LL(1) and strict deterministic (hence prefix-free). The language L_1 is LL(1), since it is generated by the following LL(1) grammar:

$$\begin{aligned} S &\rightarrow aDA\$, \\ D &\rightarrow aDA \mid \Lambda, \\ A &\rightarrow bB \mid c, \\ B &\rightarrow d \mid \Lambda. \end{aligned}$$

Hence, L_1 is a prefix-free LL(1) language which is not simple deterministic. A simple theorem follows immediately.

Theorem 5.5 *The class of simple deterministic languages is properly included in the class of prefix-free LL(1) languages.*

6 Precedence Languages

The family of precedence language [23] plays an important role in the theory of parsing. Although the family is rather old, rigorous proofs that certain deterministic context free languages are not precedence languages have been obtained only recently [16]. Fischer [7] gave an intuitive argument but it was based on properties of a precedence parser but the parser had never been formally defined. It is possible to sketch the ideas of an iteration theorem for such a family here.

First we need the ideas of precedence relations.

Definition 6.1 *Let $G = (V, \Sigma, P, S)$ be a context free grammar and define the relations $<\!\cdot$, $\doteq$, and $\cdot\!>$ on V as follows:*

$$X <\!\cdot\, Y \quad \text{if there is } A \rightarrow \alpha X B \beta \text{ in } P \text{ such that } B \overset{+}{\Rightarrow} Y\gamma \tag{26}$$

$$X \doteq Y \quad \text{if there is } A \rightarrow \alpha X Y \beta \text{ in } P \tag{27}$$

$$X \cdot\!> a \quad \text{if } a \in \Sigma, A \rightarrow \alpha B Y \beta \text{ is in } P, B \overset{+}{\Rightarrow} \gamma X \text{ and } Y \overset{*}{\Rightarrow} a\delta \tag{28}$$

The idea of precedence parsing is to scan a string from left to right, computing precedence relations between pairs of adjacent characters as we go. When the sequence

$$<\!\cdot \doteq \cdots \doteq \cdot\!>$$

is found, a phrase has been detected. In order for this to work, we need to have a grammar which satisfies the following properties.

Definition 6.2 *A context free grammar $G = (V, \Sigma, P, S)$ is said to be a simple precedence grammar if*

1. *G is reduced,*
2. *G is Λ-free,*
3. *G is invertible,*[3]
4. *$<\!\cdot$, $\doteq$, and $\cdot\!>$ are pairwise disjoint, and*
5. *the derivation $S \Rightarrow^+ S$ is impossible.*

This is not the standard definition as given in [23] or [2]. Instead we use the equivalent definition of [24] which is simpler and cleaner.

There are several tricks to getting an iteration theorem for precedence languages. Since the parsing is to work as previously described, the left-to-right scan corresponds to a right-most derivations just as in the LR case [10]. This motivates the following definition:

Definition 6.3 *Let $G = (V, \Sigma, P, S)$ be a context free grammar and $w \in \Sigma^*$. Let p be the constant of the Pumping Lemma. A G-factorization $\varphi = (v_1, \ldots, v_5)$ of w is defined to satisfy the following properties.*

1. *There exists $A \in V - \Sigma$, $\alpha_1, \alpha_2 \in V^*$ such that*

$$\begin{aligned} S &\Rightarrow_R^* \alpha_1 A v_5 \\ A &\Rightarrow^* \alpha_2 A \alpha_4 \\ A &\Rightarrow^* v_3 \\ \alpha_2 &\Rightarrow_R^* v_2 \\ \alpha_1 &\Rightarrow_R^* v_1 \end{aligned}$$

[3]A context free grammar $G = (V, \Sigma, P, S)$ is *invertible* if $A \rightarrow \alpha$ and $B \rightarrow \alpha$ in P implies $A = B$.

2. For every $q \geq 0$, $v_1 v_2^q v_3 v_4^q v_5 \in L$

3. $lg(v_2 v_3 v_4) \leq p$ and $lg(v_2 v_4) > 0$.

These are just the conclusions of the usual Pumping Lemma with the refinement about the rightmost derivations noted.

In order to prove an iteration theorem for this family, one needs two additional results. The first is analogous to a technique used in the LR case. Cf. the Extended LR(k) Theorem in [10]. The proofs of these lemmas will be omitted.

Lemma 6.1 *(The Extended Simple Precedence Lemma) Let $G = (V, \Sigma, P, S)$ be a simple precedence grammar in which the following derivations exist:*

$$\$S\$ \Rightarrow_r^* \quad \$\alpha\delta w\$ \overset{n}{\underset{r}{\Rightarrow}} \$\alpha\beta w\$ \tag{29}$$

$$\$S\$ \Rightarrow_r^* \quad \$\alpha'\beta w'\$ \tag{30}$$

such that $w, w' \in \Sigma^*$; $\beta \in V^+$; $\alpha, \alpha' \in V^*$; $\delta \in V^+$, and $n \geq 0$. if (a) ${}^{(1)}w ={}^{(1)} w'$, (b) $\alpha^{(1)} = (\alpha')^{(1)}$, and (c) α' has only $<\!\cdot$ and $\doteq$ between any two adjacent characters, then derivation (30) is of the form

$$\$S\$ \overset{*}{\underset{r}{\Rightarrow}} \$\alpha'\delta w'\$ \overset{n}{\underset{r}{\Rightarrow}} \alpha'\beta w'\$.$$

the next lemma shows one of the techniques used so successfully, namely interleaving.

Lemma 6.2 *Let $G = (V, \Sigma, P, S)$ be a simple precedence grammar and let $\alpha_1, \alpha_2, \alpha_1', \alpha_2' \in V^*$, $a, a' \in V - \Sigma$, v_1, v_2, v_3, v_4, v_5, v_1', v_2', v_3', v_4', $v_5' \in \Sigma^*$. Suppose v_2, $v_2' \in x^+$ for some $x \in \Sigma^+$ and following two G-derivations exist:*

$$\$S\$ \Rightarrow_r^* \quad \$\alpha_1 a v_5\$, a \overset{+}{\underset{r}{\Rightarrow}} \alpha_2 a v_4, a \overset{+}{\underset{r}{\Rightarrow}} v_3, \alpha_1 \overset{*}{\underset{r}{\Rightarrow}} v_1, \alpha_2 \overset{*}{\underset{r}{\Rightarrow}} v_2 \tag{31}$$

$$\$S\$ \Rightarrow_r^* \quad \$\alpha_1' a' v_5'\$, a' \overset{+}{\underset{r}{\Rightarrow}} \alpha_2' a' v_4', a' \overset{+}{\underset{r}{\Rightarrow}} v_3', \alpha_1' \overset{*}{\underset{r}{\Rightarrow}} v_1', \alpha_2' \overset{*}{\underset{r}{\Rightarrow}} v_2' \tag{32}$$

Then, for every $k \geq 0$ there exists $z \in \Sigma^+$, where ${}^{(1)}z ={}^{(1)} x$, such that

$$\$S\$ \overset{*}{\underset{r}{\Rightarrow}} \$\alpha_1 \alpha_2^k \alpha_2' z\$ \overset{*}{\underset{r}{\Rightarrow}} \$\alpha_1 \alpha_2^k v_2' z\$$$

Now we are ready for the main result.

Theorem 6.1 *(The Iteration Theorem For Simple Precedence Languages.) Let L be a simple precedence language and let v and v' be two strings in L having G-factorizations $(v_1, v_2, v_3, v_4, v_5)$ and $(v_1', v_2', v_3', v_4', v_5')$, respectively which satisfy the following conditions:*

1. *$v_2, v_2' \in x^+$ for some $x \in \Sigma^+$, and*
2. *there exist $r \geq 0$ and $z \in \Sigma^+$ with ${}^{(1)}z = {}^{(1)}v_4'v_5'$ and $v_1v_2^r(v_2')^r v_3'v_4'z \in L$.*

Then for each $m \geq 0$, $v_1v_2^r(v_2')^{m+1}v_3'v_4'^{m}z \in L$.

When v, v' and the factorizations are fixed, we can write

$$w(r,m,z) = v_1v_2^r(v_2')^{m+1}v_3'(v_4')^m z$$

in these terms, the theorem states that $w(r,l,z)$ in L implies $w(r,m,z)$ in L for each $m \geq 0$.

Proof. Assume that the G-factorizations for v and v' induce the following derivations:

$$\$S\$ \Rightarrow_R^* \$\alpha_1Av_5\$,\quad A \overset{*}{\underset{R}{\Rightarrow}} \alpha_2Av_4, A \overset{*}{\underset{R}{\Rightarrow}} v_3, \alpha_1 \overset{*}{\underset{R}{\Rightarrow}} v_1, \alpha_2 \overset{*}{\underset{R}{\Rightarrow}} v_2 \tag{33}$$

$$\$S\$ \Rightarrow_R^* \$\alpha_1'A'v_5'\$,\quad A' \overset{*}{\underset{R}{\Rightarrow}} \alpha_2'A'v_4', A' \overset{*}{\underset{R}{\Rightarrow}} v_3', \alpha_1' \overset{*}{\underset{R}{\Rightarrow}} v_1', \alpha_2' \overset{*}{\underset{R}{\Rightarrow}} v_2' \tag{34}$$

Consider the following two derivations:

$$\$S\$ \overset{*}{\underset{R}{\Rightarrow}} \$\alpha_1\alpha_2^r v_2v_3v_4^{r+1}v_5\$ \overset{*}{\underset{R}{\Rightarrow}} v_1v_2^r v_2v_3v_4^{r+1}v_5\$ \tag{35}$$

$$\$S\$ \overset{*}{\underset{R}{\Rightarrow}} v_1v_2^r v_2'^2 v_3'v_4'z\$ \tag{36}$$

Note that (35) follows from derivation (33) while (36) follows from assumption (ii). Now apply Lemma 6.1 to derivations (35) and (36) with $\alpha = \alpha' = \Lambda$, $\beta = v_1v_2^r$, $\delta = \alpha_1\alpha_2^r$, $w = v_2v_3v_4{}^{r+1}v_5$, $w' = v_2'v_2'v_3'v_4'z$ and we conclude that derivation (36) is of the form:

$$\$S\$ \overset{*}{\underset{R}{\Rightarrow}} \$\alpha_1\alpha_2^r v_2'v_3'v_4'z\$ \overset{*}{\underset{R}{\Rightarrow}} v_1v_2^r v_2'^2 v_3'v_4'z\$ \tag{37}$$

On the other hand we can apply Lemma 6.2 to (33) and (34) and obtain the derivation

$$\$S\$ \overset{*}{\underset{R}{\Rightarrow}} \$\alpha_1\alpha_2^r\alpha_2'z_1\$ \overset{*}{\underset{R}{\Rightarrow}} \$\alpha_1\alpha_2^r v_2'z_1\$ \tag{38}$$

for some $z_1 \in \Sigma^+$, ${}^{(1)}z_1 = {}^{(1)} x$. Now we can apply Lemma 6.1 to derivation (38) and (37) choosing β to be v_2', $\delta = \alpha_2'$, $w = z_1$, $w' = v_2'v_3'v_4'z$, and $\alpha = \alpha' = \alpha_1\alpha_2^r$. Thus, derivation (37) can be written:

$$\$S\$ \overset{*}{\underset{R}{\Rightarrow}} \$\alpha_1\alpha_2^r\alpha_2'v_2'v_3'v_4'z\$ \overset{*}{\underset{R}{\Rightarrow}} \alpha_1\alpha_2^r v_2'v_2'v_3'v_4'z\$ \tag{39}$$

Next consider the following derivation:

$$\$S\$ \overset{*}{\underset{R}{\Rightarrow}} \$\alpha_1'\alpha_2'A'v_4'v_5'\$ \overset{*}{\underset{R}{\Rightarrow}} \$\alpha_1'\alpha_2'v_2'v_3'v_4'v_4'v_5'\$ \tag{40}$$

This follows from derivation (34).

Appealing to Lemma 6.1 again, with derivations (40) and (39), we can choose β to be $v_2'v_3'v_4'$, $\delta = A'$, $\alpha = \alpha_1'\alpha_2'$, $\alpha' = \alpha_1\alpha_2^r\alpha_2'$, $w = v_4'v_5'$ and $w' = z$. It follows that derivation (39) is of the form:

$$\$S\$ \overset{*}{\underset{R}{\Rightarrow}} \$\alpha_1\alpha_2^r\alpha_2'A'z\$ \overset{*}{\underset{R}{\Rightarrow}} \$\alpha_1\alpha_2^r\alpha_2'v_2'v_3'v_4'z\$$$

But $A' \Rightarrow_R^* v_2'A'v_4'$ from the derivation in (34). It follows that for every $m \geq 0$

$$\$S\$ \overset{*}{\underset{R}{\Rightarrow}} \$\alpha_1\alpha_2^r\alpha_2'A'z\$ \overset{*}{\underset{R}{\Rightarrow}} \$v_1v_2^rv_2'(v_2')^m v_3'(v_4')^m z\$$$

Hence for every $m \geq 0$

$$v_1v_2^r(v_2')^{m+1}v_3'(v_4')^m z \in L$$

This complete the proof.

The set L defined below is an example of a language which is not a simple precedence language.

$$L = \{a0^n1^n \mid n \geq 1\} \cup \{b0^n1^{2n} \mid n \geq 1\}$$

Other examples of a much more general nature can be found in [16].

7 Conclusions

We have sketched the proofs of the major iteration theorems known for the important subfamilies of the deterministic context free languages. Our major techniques were the discovery of the left part theorems, properties of cross sections, and interlacing subderivations. Of course, the shoe box principle is used in several different ways and is, together with elementary facts about trees, the basic combinational technique.

While these results are interesting, there still remain open problems. Perhaps the most elusive open problem is to get an iteration theorem for extended precedence languages. [9]

References

[1] Alfred V. Aho and Jeffrey D. Ullman. Deterministic parsing of ambiguous grammars. *Communications of the ACM*, 18:441–452, 1975.

[2] Alfred V. Aho and Jeffrey D. Ullman. *The Theory of Parsing, Translating, and Compiling.* Prentice-Hall Publishing Company, Englewood Cliffs, NJ, 1972 and 1973. Volumes I and II.

[3] Christopher Bader and Arnaldo Moura. A generalization of Ogden's lemma. *Journal of the Association for Computing Machinery*, 29(2):404–407, April 1982.

[4] Y. Bar-Hillel, M. Perles, and E. Shamir. On formal properties of simple phrase structure grammars. *Zeitschrift für Phonetik, Sprachwissenschaft und Kommunikationsforschung*, 14:143–172, 1961. Also available in *Language and Information*, Addison Wesley Publishing Company, Reading Mass., 1964.

[5] John C. Beatty. Two iteration theorems for LL(k) languages. *Theoretical Computer Science*, 12:193–228, 1980.

[6] Benjamin M. Brosgol. *Deterministic Translation Grammars.* PhD thesis, Harvard University, Cambridge, Mass., 1974.

[7] Michael J. Fischer. Some properties of precedence languages. In *Proc. of 1st ACM Symp. on Theory of Computing*, pages 181–190, acm, acm, New York, NY, 1969.

[8] Matthew M. Geller and Michael A. Harrison. On LR(k) grammars and languages. *Theoretical Computer Science*, 4(3):245–276, 1977.

[9] Susan L. Graham. Extended precedence languages, bounded right context languages and deterministic languages. In *IEEE Conference Record of the 11th Annual Symposium on Switching and Automata Theory*, pages 175–180, IEEE, IEEE, 1970.

[10] Michael A. Harrison. *Introduction to Formal Language Theory*. Addison Wesley Publishing Company, Reading, Mass., 1978.

[11] Michael A. Harrison and Ivan M. Havel. On the parsing of deterministic languages. *Journal of the Association for Computing Machinery*, 21:525–548, 1974.

[12] Michael A. Harrison and Ivan M. Havel. Real-time strict deterministic languages. *SIAM Journal of Computing*, 1:333–349, 1972.

[13] Michael A. Harrison and Ivan M. Havel. Strict deterministic grammars. *Journal of Computer and System Science*, 7:237–277, 1973.

[14] Kimberly N. King. Iteration theorems for families of strict deterministic languages. *Theoretical Computer Science*, 10:317–333, 1980.

[15] A. J. Korenjak and J. E. Hopcroft. Simple deterministic languages. In *IEEE Conf. Record of 7th Annual Symposium on Switching and Automata Theory*, pages 34–46, IEEE Computer Society, IEEE, 1966.

[16] Yael Krevner and Amiram Yehudai. An iteration theorem for simple precedence languages. *Journal of the Association for Computing Machinery*, 30:820–833, 1983.

[17] Harry R. Lewis and Christos H. Papadimitriou. *Elements of the Theory of Computation*. Prentice Hall, Englewood Cliffs, New Jersey, 1981.

[18] William Ogden. *Intercalation Theorems for Pushdown Store and Stack Languages*. PhD thesis, Stanford University, Stanford, California, 1968.

[19] Rohit J. Parikh. On context free languages. *Journal of the Association for Computing Machinery*, 13:570–581, 1966.

[20] Daniel J. Rosenkrantz and Richard E. Stearns. Properties of deterministic top-down grammars. *Information and Control*, 17:226–256, 1970.

[21] Daniel J. Rosenkrantz and Richard E. Stearns. Properties of deterministic top-down grammars. *Information and Control*, 17:226–256, 17.

[22] E. Soisalon-Soinen. *Characterization of LL(k) languages by restricted LR(k) grammars*. PhD thesis, University of Helsinki, 1979.

[23] Niklaus Wirth and Helmut Weber. Euler-a generalization of ALGOL and its formal definition. *Communications of the ACM*, 9:13–23, 1966. Part I.

[24] Amiram Yehudai. A new definition of simple precedence grammars. *BIT*, 19:282–284, 1979.