Theoretical Computer Science 12 (1980) 193–228


# TWO ITERATION THEOREMS FOR THE LL($k$) LANGUAGES*

John C. BEATTY
*Dept. of Computer Science, Univ. of Waterloo, Waterloo, Ontario N2L 3G1, Canada, and Graphics Group, Lawrence Livermore Laboratory, Livermore, California 94550, USA*

Communicated by M. Harrison
Received August 1978
Revised August 1979

**Abstract.** The structure of derivation trees over an LL($k$) grammar is explored and a property of these trees obtained which is shown to characterize the LL($k$) grammars. This characterization, called the LL($k$) Left Part Theorem, makes it possible to establish a pair of iteration theorems for the LL($k$) languages. These theorems provide a general and powerful method of showing that a language is not LL($k$) when that is the case. They thus provide for the first time a flexible tool with which to explore the structure of the LL($k$) languages and with which to discriminate between the LL($k$) and LR($k$) language classes.

Examples are given of LR($k$) languages which, for various reasons, fail to be LL($k$). Easy and rigorous proofs to this effect are given using our LL($k$) iteration theorems. In particular, it is proven that the dangling-ELSE construct allowed in PL/I and Pascal cannot be generated by any LL($k$) grammar. We also give a new and straightforward proof based on the LL($k$) Left Part Theorem that every LL($k$) grammar is LR($k$).

## 1. Introduction

The classical pumping lemma [3] and Ogden's lemma [20] are among the most powerful tools we possess for proving that languages are not context-free. Hence one goal of recent research has been to obtain analogous theorems for subclasses of the context-free languages. Thus Ogden [19] gives an iteration theorem for the deterministic context-free languages, Harrison and Havel have established an iteration theorem for

* Research supported by the U. S. Earth Resources and Development Administration under Contract No. W-7405-Eng-48 and by the National Research Council of Canada under grant A-3022.

the family of strict deterministic languages [11] which is also extendible to the deterministic context-free languages, and Boasson has established an iteration theorem for the one-counter languages [6]. More recently King has obtained iteration theorems for the simple deterministic languages and the strict deterministic languages of degree n [13]. Such results help elucidate the structure of languages belonging to these families, and provide us with a convenient means of distinguishing between context-free languages which are and are not of a given class.

We establish here a property of LL($k$) derivation trees which is analogous to the left part properties of strict deterministic grammars [11] and left part grammars [18]. We show that our property characterizes the LL($k$) grammars and use it to establish two iteration theorems for the LL($k$) languages. These theorems, in turn, enable us to prove simply and rigorously that a variety of LR($k$) languages are not LL($k$). In particular, the ALGOL-60 dangling IF-THEN-ELSE construct allowed in Pascal and PL/I cannot be generated by an LL($k$) grammar. We are also able to give a new and straightforward proof that every LL($k$) grammar is LR($k$).

The present paper is organized as follows. In the remainder of this section we recall various commonly known definitions and theorems which we will need. In section 2 we will semi-formalize the notion of a derivation tree, and in the spirit of [11] will establish various useful properties of such trees. Thus equipped we will proceed in section 3 to prove a left part theorem for LL($k$) grammars which enables us to establish our iteration theorems in section 4. Finally, in section 5 we present some applications of our work.

A *context-free grammar* (*cfg*) G is a 4-tuple $(N,\Sigma,P,S)$. N is a finite, non-empty set of *nonterminals*, or *variables*, denoted by upper case Roman characters such as A and B. $\Sigma$ is a finite, non-empty set of *terminals*, denoted by lower case Roman characters from the beginning of the alphabet, such as a and b. The *vocabulary* of G, written V, is $N \cup \Sigma$. P is a finite subset of $N \times V^*$; an element $(A,\alpha)$ of P is called a *production* or *rule* and is written $A \to \alpha$. S is a special variable called the *start* or *goal symbol*. For any variable A we call $A \to \Lambda$ a *$\Lambda$-rule*, where $\Lambda$ is the empty string, and say that G is *$\Lambda$-free* iff G contains no $\Lambda$-rules.

We write $\alpha \Rightarrow \beta$ (*$\alpha$ derives $\beta$ in one step*) iff there exists a variable $A \in N$ and strings $\gamma_1, \gamma_2, \delta \in V^*$ such that $\alpha = \gamma_1 A \gamma_2$, $\beta = \gamma_1 \delta \gamma_2$ and

$A \to \delta$ is in P. If $\gamma_1 \in \Sigma^*$ then we may write $\alpha \Rightarrow_L \beta$; $\Rightarrow^+$ and $\Rightarrow_L^+$ are the transitive closures of $\Rightarrow$ and $\Rightarrow_L$, while $\Rightarrow^*$ and $\Rightarrow_L^*$ are their reflexive transitive closures. If $\alpha \Rightarrow^* \beta$ then we say that $\alpha$ *derives* $\beta$. If $\alpha \Rightarrow_L^* \beta$ then the derivation is *leftmost*. By $\Rightarrow^n$ we mean a derivation of exactly n steps, for any $n \geq 0$, while $\Rightarrow_L^n$ denotes a leftmost derivation of exactly n steps. The relations $\Rightarrow_R$, $\Rightarrow_R^+$ and $\Rightarrow_R^*$, *etc.*, are similarly defined. If we use a Greek letter such as $\pi$ (for example: $\Rightarrow_L^\pi$) which is constrained to belong to $P^*$ then $\pi$ represents the sequence of rules (possibly null) by which the derivation proceeds.

We will say that an occurrence of the symbol $X \in \Sigma$ is *exposed* at the $(n+1)^{\underline{st}}$ step of the leftmost derivation

$$S \Rightarrow_L^n wA\gamma \Rightarrow_L w\beta\gamma$$

if X appears somewhere in $\beta\gamma$ and there are no variables anywhere to the left of X in $\beta\gamma$.

The *context-free language* (*cfl*) $\mathcal{L}(G)$ generated by G is exactly the set of terminal strings which can be derived from the start symbol S. Similarly, if $\alpha \in V^*$ then $\mathcal{L}(\alpha)$ is the set of terminal strings which can be derived from $\alpha$. The *left sentential forms* of G are exactly those strings of terminals and nonterminals which can be generated from S by a leftmost derivation.

G is said to be *unambiguous* if no string in $\mathcal{L}(G)$ has more than one distinct leftmost derivation. Otherwise G is said to be *ambiguous*.

A variable A of G is said to be *reduced* iff A derives at least one terminal string and itself appears in some string of terminals and nonterminals which can be derived from S. G is said to be *reduced* iff either the variables of G are all reduced or $P = \emptyset$.

A variable A of G is said to be *left recursive* iff $A \Rightarrow^+ A\beta$ for some string $\beta \in V^*$. G is left recursive iff some variable A of G is left recursive.

If w is a string and k a non-negative integer then $w/k$ is the first k symbols of w if $|w| > k$ and is w itself if $|w| \leq k$, where $|w|$ is the length of w. More generally, for a cfg $G = (N,\Sigma,P,S)$ we define

$$\mathit{first}_k(\beta) = \left\{ w \in \Sigma^* \;\middle|\; \begin{array}{l} (|w| \leq k \text{ and } \beta \Rightarrow^* w) \text{ or} \\ (|w| = k \text{ and } \beta \Rightarrow^* wy \text{ for some } y \in \Sigma^+) \end{array} \right\}$$

for any $\beta \in V^*$. $\mathit{first}_k$ is extended to sets in the usual way.

Next we review pertinent facts about LL($k$) grammars.

**Definition 1.1.** A cfg $G = (N,\Sigma,P,S)$ is *LL($k$)* iff for any $A \in N$; w, x, $y \in \Sigma^*$; $\beta$, $\beta'$, $\gamma \in V^*$; and any two derivations

$$S \Rightarrow_L^* wA\gamma \Rightarrow_L w\beta\gamma \Rightarrow_L^* wx$$
$$S \Rightarrow_L^* wA\gamma \Rightarrow_L w\beta'\gamma \Rightarrow_L^* wy$$

for which $x/k = y/k$ we necessarily have $\beta = \beta'$. A language is LL($k$) iff it is generated by an LL($k$) grammar.

The following results are well-known or easily proven [5]. They will be used subsequently and are stated here for convenience.

**Theorem 1.2.** [22] No LL($k$) grammar is ambiguous.

**Theorem 1.3.** [22] No LL($k$) grammar is left recursive.

**Theorem 1.4.** Let $G = (N,\Sigma,P,S)$ be a cfg. G is an LL($k$) grammar iff for any $A \in N$; w, x, $y \in \Sigma^*$; $\beta$, $\beta'$, $\gamma$, $\gamma' \in V^*$; and any two derivations

$$S \Rightarrow_L^* wA\gamma \Rightarrow_L w\beta\gamma \Rightarrow_L^* wx$$
$$S \Rightarrow_L^* wA\gamma' \Rightarrow_L w\beta'\gamma' \Rightarrow_L^* wy$$

for which $x/k = y/k$ we necessarily have $\beta = \beta'$.

Theorem 1.4 allows the right context $\gamma$ of A in the two derivations of definition 1.1 to differ. Definition 1.1 is taken from Aho and Ullman [2]; theorem 1.4 is actually the LL($k$) definition used by Rosenkrantz and Stearns [22].

**Theorem 1.5.** [2] Let $G = (N,\Sigma,P,S)$ be a cfg. G is an LL($k$) grammar iff given any $A \in N$, $w \in \Sigma^*$, and $\gamma \in V^*$ such that $S \Rightarrow_L^* wA\gamma$, we have

$$\mathit{first}_k(\beta\gamma) \cap \mathit{first}_k(\beta'\gamma) = \emptyset$$

for every distinct pair of rules $A \to \beta$ and $A \to \beta'$ in P.

**Theorem 1.6.** Let $G = (N,\Sigma,P,S)$ be a cfg. G is an LL($k$) grammar iff given

(1) $w \in \mathit{first}_k(\Sigma^*)$

(2) $x \in \Sigma^*$

(3) $A \in N$

then there exists at most one rule $A \to \beta$ in P such that

(4) $S \Rightarrow^* xAw_2$

(5) $A \Rightarrow \beta \Rightarrow^* w_1$

(6) $(w_1w_2)/k = w$

for any $w_1$, $w_2 \in \Sigma^*$.

This was the definition of LL($k$) grammars used by Lewis and Stearns [15].

The following special version of the LL($k$) definition will be useful in section 3.

**Theorem 1.7.** Let $G = (N,\Sigma,P,S)$ be a reduced cfg. G is an LL($k$) grammar iff for any $A \in N$; $w, x, y \in \Sigma^*$; $\beta, \beta', \gamma \in V^*$; and any two derivations

$$S \Rightarrow_L^n wA\gamma \Rightarrow_L w\beta\gamma \Rightarrow_L^* wx$$
$$S \Rightarrow_L^n wA\gamma \Rightarrow_L w\beta'\gamma \Rightarrow_L^* wy$$

for which $x/k = y/k$ we necessarily have $\beta = \beta'$. (Notice that $wA\gamma$ is derived in n steps in both derivations.)

**Proof:** A proof in the forward direction is trivial. To establish the reverse direction, suppose that G is not LL($k$), but that the existence of two such derivations necessarily forces $\beta = \beta'$. Since G is not LL($k$) it follows from theorem 1.5 that there exist strings $A \in N$; $w \in \Sigma^*$; $\beta, \beta'$, $\gamma \in V^*$; such that $S \Rightarrow_L^* wA\gamma$ and

$$\mathit{first}_k(\beta\gamma) \cap \mathit{first}_k(\beta'\gamma) \neq \emptyset \tag{1}$$

for some distinct pair of rules $A \to \beta$ and $A \to \beta'$ in P. Let x and y be strings in $\mathcal{L}(\beta\gamma)$ and $\mathcal{L}(\beta'\gamma)$, respectively, such that $x/k = y/k$ and suppose that S derives $wA\gamma$ leftmost in n steps. Then

$$S \Rightarrow_L^n wA\gamma \Rightarrow_L w\beta\gamma \Rightarrow_L^* wx$$
$$S \Rightarrow_L^n wA\gamma \Rightarrow_L w\beta'\gamma \Rightarrow_L^* wy$$

where $x/k = y/k$. By hypothesis we must have $\beta = \beta'$, which is a contradiction. Hence G must be LL($k$). ∎

**Theorem 1.8.** Let $G = (N,\Sigma,P,S)$ be a reduced LL($k$) grammar. Let $G_A = (N,\Sigma,P,A)$ be the grammar formed from G by changing the start symbol from S to A, for any variable A of G. Then $G_A$ is also an LL($k$) grammar.

**Proof:** Suppose that $G_A$ were not LL($k$). Then for some $x, y_1, y_2 \in \Sigma^*$; $\beta, \beta', \gamma \in V^*$; $B \in N$; there must exist two derivations

$$A \Rightarrow_L^* xB\gamma \Rightarrow_L x\beta\gamma \Rightarrow_L^* xy_1$$
$$A \Rightarrow_L^* xB\gamma \Rightarrow_L x\beta'\gamma \Rightarrow_L^* xy_2$$

in $G_A$ with $y_1/k = y_2/k$ and $\beta \neq \beta'$. But this is also a derivation in G. Since G is reduced, there also exists in G a derivation sequence $S \Rightarrow_L^* wA\delta$ for some $w \in \Sigma^*$ and $\delta \in V^*$. We obtain the following derivations in G:

$$S \stackrel{*}{\Rightarrow}_L wA\delta \stackrel{*}{\Rightarrow}_L wxB\gamma\delta \Rightarrow_L wx\beta\gamma\delta \stackrel{*}{\Rightarrow}_L wxy_1z$$
$$S \stackrel{*}{\Rightarrow}_L wA\delta \stackrel{*}{\Rightarrow}_L wxB\gamma\delta \Rightarrow_L wx\beta'\gamma\delta \stackrel{*}{\Rightarrow}_L wxy_2z$$

where z is any string derived from $\delta$. Recall that $y_1/k = y_2/k$. If $|y_1| < k$ or $|y_2| < k$ then we must have $y_1 = y_2$, in which case $(y_1z)/k = (y_2z)/k$. If both $y_1$ and $y_2$ are of length k or greater then again $(y_1z)/k = (y_2z)/k$. Since G is LL($k$), we must therefore have $\beta = \beta'$, which is a contradiction. Therefore $G_A$ must also be LL($k$). ∎

We also need to introduce LR($k$) grammars. We use the definition suggested by Geller and Harrison [10].

**Definition 1.9.** A cfg $G = (N,\Sigma,P,S)$ is *LR*($k$) for some $k \geq 0$ iff $S \stackrel{+}{\Rightarrow}_R S$ is impossible in G and for any w, w', x $\in \Sigma^*$; $\alpha$, $\alpha'$, $\beta$, $\beta'$ $\in V^*$; A, A' $\in$ N; and derivations

$$S \stackrel{*}{\Rightarrow}_R \alpha A w \Rightarrow_R \alpha\beta w$$
$$S \stackrel{*}{\Rightarrow}_R \alpha' A' x \Rightarrow_R \alpha'\beta' x = \alpha\beta w'$$

if $w/k = w'/k$ then $(A\to\beta,|\alpha\beta|) = (A'\to\beta',|\alpha'\beta'|)$.

## 2. Trees

Following Harrison and Havel [11] we semi-formally develop the notion of trees, particularly derivation trees, and their properties. Our presentation is a compromise between the demands of rigor and a desire not to sacrifice entirely comprehensibility and intuition. To this end we will occasionally make informal use of pictures.

For our purposes a *tree* $\mathcal{T}$ is a directed acyclic graph defined by a pair of sets $(\mathcal{V},\mathcal{E})$, where $\mathcal{V}$ is a set of *nodes* and $\mathcal{E}$ is a set of *edges* $(x,y) \in \mathcal{V}\times\mathcal{V}$, in which all nodes save one (the *root node* of $\mathcal{T}$, written $rtn(\mathcal{T})$) have exactly one entering edge; the root node has no entering edges. For example, the tree in figure 1 is defined by

$$\left( \{x_0,x_1,x_2,x_3\},\ \{ (x_0,x_1),\ (x_0,x_2),\ (x_2,x_3) \} \right)$$

The edges (x,y) in $\mathcal{E}$ define the *immediate descendency* relation $\Gamma$; x is a *parent* of y and y is a *child* of x. In figure 1 we have $x_0 \Gamma x_1$ but not $x_1 \Gamma x_2$. The reflexive transitive closure $\Gamma^*$ of $\Gamma$ is called the *descendancy* relation. There is a *path* from node x to node y iff $x \Gamma^* y$. Thus in figure 1 there is a path from $x_0$ to $x_3$ since $x_0 \Gamma^* x_3$, but no path from $x_3$ to $x_1$. If $rtn(\mathcal{T})\, \Gamma^i\, y$ then y is said to be at *depth* $i$ in $\mathcal{T}$. The *height* of $\mathcal{T}$ is the length of a longest path in $\mathcal{T}$; it is thus equal to the depth of a deepest node.

A node x is *internal* iff there exists a node y such that $x \Gamma y$. Otherwise x is a *leaf*, and has no children.

We will need a left to right ordering of the nodes in a tree. For this reason we assume that $\mathcal{E}$ is actually a sequence of edges so that we may define an additional relation $\sqcap$ on the nodes of a tree in the following way. If the r edges leaving an arbitrary node x are listed in $\mathcal{E}$ in the order $(x,y_1), \cdots, (x,y_r)$ then $y_1 \sqcap y_2 \sqcap \cdots \sqcap y_r$ and the edges will be drawn left to right according to this ordering, as in figure 2. Furthermore, if $p \Gamma y$ and there does not exist any node x such that $x \sqcap y$ then $p \Gamma_L y$ (y is a *leftmost child* of p). The relation $\Gamma_R$ is defined similarly. Finally, we write $x \mathrm{L} y$ iff $(x,y) \in (\Gamma_R^{-1})^* \sqcap (\Gamma_L)^*$, so that $x \mathrm{L} y$ iff there are no nodes between x and y. The reflexive transitive closure $\mathrm{L}^*$ of L then defines the notion of *left to right order* in $\mathcal{T}$. (The relations $\Gamma$ and L are identical to the relations represented by these symbols in Harrison and Havel [11].) If we list the leaves $\ell_1, \cdots, \ell_r$ of $\mathcal{T}$ in left to right order, which is to say that

$$\ell_1 \mathrm{L} \ell_2 \mathrm{L} \cdots \mathrm{L} \ell_r$$

then we obtain the left to right sequence of nodes

$$\mathit{leaves}(\mathcal{T}) = ( \ell_1, \ell_2, \cdots, \ell_r )$$

Let us adopt the convention that if we list the nodes in a subtree $\mathcal{T}'$ of $\mathcal{T}$ then edges between those nodes in $\mathcal{T}$ are implicitly the edges of $\mathcal{T}'$

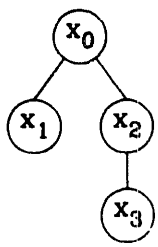



Fig. 1.

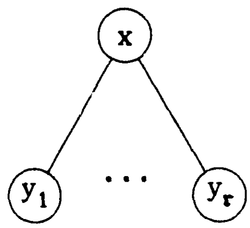



Fig. 2.

(the *induced subtree*). Then for any internal node x of the tree $\mathcal{T}$ the set $\{ y \in \mathcal{T} \mid x = y \text{ or } x \Gamma y \}$ defines the *elementary subtree of $\mathcal{T}$ with root $x$*. Also, if x is a node of $\mathcal{T}$ then we define $\mathcal{T}_x$ to be the largest induced subtree of $\mathcal{T}$ whose root is x. More precisely,

$$\mathcal{T}_x = \{ y \in \mathcal{T} \mid x \Gamma^* y \}$$

Since our trees represent context-free derivations we will want each node to represent a grammar symbol or, perhaps, $\Lambda$. Furthermore, it is often desirable to distinguish between a node and the symbol it represents since several nodes may represent the same grammar symbol. Hence we define a *labeled tree* to be a tree $\mathcal{T} = (\mathcal{V},\mathcal{E})$ together with a labeling function $\lambda$ from $\mathcal{V}$ into a finite set $\mathcal{L}$ of labels such that $\mathcal{V} \cap \mathcal{L} = \emptyset$. The labeling function $\lambda$ is then extended to sequences of nodes in the obvious way; for a sequence $(x_0, \cdots, x_n)$ of nodes we have $\lambda(x_0, \cdots, x_n) = \lambda(x_0) \cdots \lambda(x_n)$. Our labels will always be drawn from some set $V_\Lambda = V \cup \{\Lambda\}$, where V is the vocabulary of some cfg. Of particular interest are the *root label* and *frontier* of $\mathcal{T}$:

$$\mathit{rtl}(\mathcal{T}) = \lambda(\ \mathit{rtn}(\mathcal{T})\ )$$
$$\mathit{fr}(\mathcal{T}) = \lambda(\ \mathit{leaves}(\mathcal{T})\ )$$

Let $G = (N,\Sigma,P,S)$ be a context-free grammar, and let $\mathcal{T}$ be a labeled tree for which the labels are symbols from $V_\Lambda$. $\mathcal{T}$ is said to be a *grammatical tree* iff $\mathit{fr}(\mathcal{T}) \in \Sigma^*$ and either

$\mathcal{T}$ is a trivial tree consisting of a single labeled node

or

for every internal node x in $\mathcal{T}$, if $y_1, \cdots, y_r$ are all of x's children in left to right order then $\lambda(x) \to \lambda(y_1) \cdots \lambda(y_r)$ is a rule of G and $\lambda(y_i) = \Lambda$ is allowed only if $1 = i = r$.

Leaves which are labeled with terminals are referred to as *terminal nodes*. Leaves which are labeled with $\Lambda$ are called *$\Lambda$-nodes*. Observe that a node x is internal iff $\lambda(x) \in N$. A grammatical tree $\mathcal{T}$ is said to be a *derivation tree* iff $\mathit{rtl}(\mathcal{T}) = S$.

Figure 3, for example, displays a grammatical tree over the context-free grammar $S \to aSbS \mid \Lambda$. Occasionally we will omit the names of nodes in a grammatical tree, leaving only the labels, in which case the tree of figure 3 would appear as in figure 4.

The sentential forms which appear in a derivation are embedded in a natural way in the grammatical tree representing that derivation. We represent this embedding by means of *cross sections* (CS's) and *canonical cross sections*, which we define inductively for a tree $\mathcal{T}$ by the following:

(1) $\eta = (x_0)$, where $x_0 = \mathit{rtn}(\mathcal{T})$, is a cross section at level 0.

(2) Let $\eta = (x_1, \cdots, x_k, \cdots, x_m)$ be a cross section of level $l$ and let $x_k$ be an internal node of $\mathcal{T}$. If $y_1, \cdots, y_r$ are all the children of $x_k$ in left to right order then

$$\eta' = (x_1, \cdots, x_{k-1}, y_1, \cdots, y_r, x_{k+1}, \cdots, x_m)$$

is a cross section of level $l+1$.

$(x_0)$ is also said to be a *left canonical cross section* (*LCCS*) of $\mathcal{T}$. If $\eta$ is an LCCS of $\mathcal{T}$ and $x_k$, the node which is replaced, is the leftmost internal node of $\eta$, then $\eta'$ is also a left canonical cross section of $\mathcal{T}$. *Right canonical cross sections* (*RCCS's*) are defined analogously. For readability we may sometimes write ( $x_1$ $x_2$ $\cdots$ $x_m$ ) instead of $(x_1,x_2,\cdots,x_m)$.

For example, in the grammatical tree of figure 3 ( $x_1$ $x_5$ $x_3$ $x_4$ ) is an LCCS, ( $x_1$ $x_2$ $x_3$ $x_6$ ) is a CS but not an LCCS and ( $x_1$ $x_2$ $x_0$ $x_4$ ) is neither an LCCS nor a CS.

The following properties of cross sections are intuitive. Consequently we state them without proof, though in an order convenient for rigorous development. More detail may be found in [5].

**Fact 2.1.** Let $\eta = (x_1,\cdots,x_m)$ be a cross section of some tree $\mathcal{T}$. Then $x_i \mathrel{L} x_{i+1}$, $1 \leq i < m$.

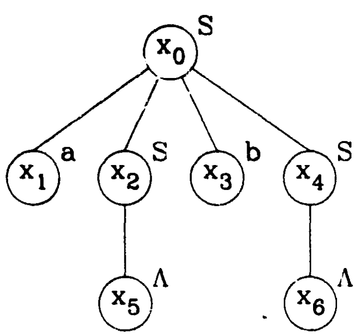



Fig. 3. A grammatical tree in which we distinguish nodes and labels.

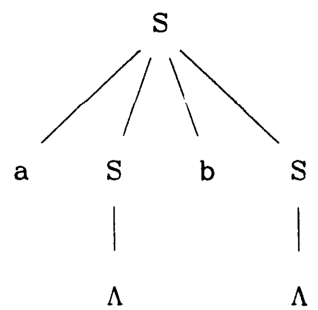



Fig. 4. A grammatical tree in which nodes and labels are not distinguished.

**Fact 2.2.** No node of any tree $\mathcal{T}$ appears more than once in any one cross section of $\mathcal{T}$.

**Fact 2.3.** [11] No two distinct LCCS's of a grammatical tree can be of the same level.

**Fact 2.4.** The level associated with any cross section is unique.

**Fact 2.5.** Let $\mathcal{T}$ be a tree and let $\mathfrak{n}$ be a node in $\mathcal{T}$. Then $\mathfrak{n}$ appears in at least one LCCS (respectively CS) of $\mathcal{T}$. Moreover, we may assume that there are no internal nodes to the left (respectively to the left and right) of $\mathfrak{n}$ in this cross section.

**Fact 2.6.** Let $\mathcal{T}$ be a tree. Then *leaves*$(\mathcal{T})$ is an LCCS of $\mathcal{T}$.

Next we delineate the relationship between cross sections and sentential forms. First we describe how to pass from cross sections to derivations.

**Fact 2.7.** Let $G = (N,\Sigma,P,S)$ be a cfg and let $\mathcal{T}$ be a grammatical tree over G. If $\eta$ is a cross section of $\mathcal{T}$ at level $\ell$ then $\mathit{rt}(\mathcal{T}) \Rightarrow^{\ell} \lambda(\eta)$.

We have a stronger result for canonical cross sections.

**Fact 2.8.** Let $G = (N,\Sigma,P,S)$ be a cfg and let $\mathcal{T}$ be a grammatical tree over G. If $\eta$ and $\eta'$ are LCCS's of level $\ell$ and $\ell+i$, for any $\ell$ and $i \geq 0$, then $\lambda(\eta) \Rightarrow^{i}_{L} \lambda(\eta')$. If $\eta$ and $\eta'$ are instead RCCS's then $\lambda(\eta) \Rightarrow^{i}_{R} \lambda(\eta')$.

This result does <u>not</u> hold for cross sections in general. In figure 5 the cross section

$$\eta = (\ x_1\ x_5\ x_{13}\ x_7\ x_8\ x_3\ x_4\ )$$

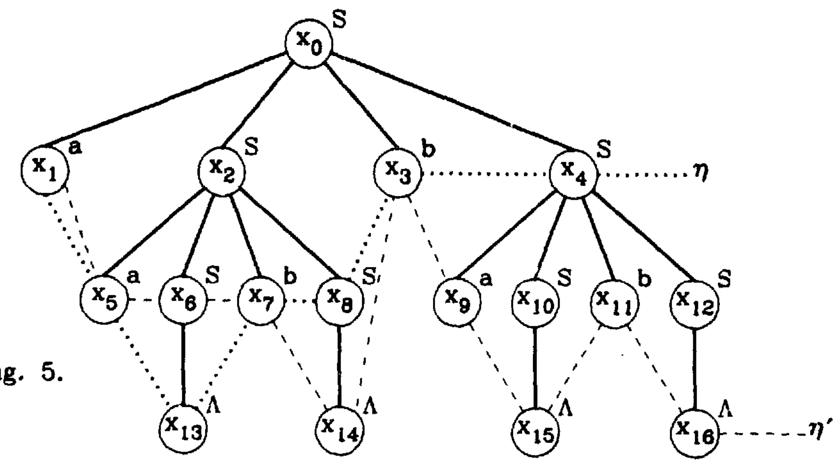



Fig. 5.

is at level 3 and the cross section

$$\eta' = (\ x_1\ x_5\ x_6\ x_7\ x_{14}\ x_3\ x_9\ x_{15}\ x_{11}\ x_{16}\ )$$

is at level 6, but $\lambda(\eta)$ = aabSbS cannot possibly derive $\lambda(\eta')$ = aaSbbab, the S in aaSbbab already having been erased in aabSbS.

**Fact 2.9.** [11] Let $\mathcal{T}$ be a derivation tree over some unambiguous cfg and let $\eta$ and $\theta$ be two LCCS's (or RCCS's) in $\mathcal{T}$. If $\lambda(\eta) = \lambda(\theta)$ then $\eta = \theta$.

We pass from derivations to cross sections *via* the next two results.

**Fact 2.10.** Let $G = (N,\Sigma,P,S)$ be a cfg and let $A \Rightarrow^{i} \alpha \Rightarrow^{*} w$, where A is a variable, $\alpha \in V^{*}$ and w is a string of terminals. Then there exists a grammatical tree $\mathcal{T}$ containing a cross section $\eta$ of level $i$ such that

$\mathit{rt}(\mathcal{T}) = A$, $\mathit{fr}(\mathcal{T}) = w$ and $\lambda(\eta) = \alpha$. Moreover, if the derivation is leftmost or rightmost then $\eta$ is respectively a left or right canonical cross section of $\mathcal{T}$.

If we are dealing with an unambiguous grammar then we can prove a stronger result.

**Fact 2.11.** Let $G = (N,\Sigma,P,S)$ be an unambiguous cfg and $\mathcal{T}$ a grammatical tree over G. If $\mathit{rt}(\mathcal{T}) \Rightarrow^{i} \alpha \Rightarrow^{*} \mathit{fr}(\mathcal{T})$, where $\alpha \in V^{*}$, then there exists a cross section $\eta$ at level $i$ in $\mathcal{T}$ such that $\lambda(\eta) = \alpha$. Moreover, if the derivation is leftmost or rightmost then $\eta$ is respectively a left or right canonical cross section of $\mathcal{T}$.

In developing our arguments we will need to disassemble and reassemble derivation trees and cross sections in a highly specialized manner. Hence we next define the tree fragments about which we will be speaking.

**Definition 2.12.** Let $\mathcal{T}$ be a grammatical tree such that $|\mathit{fr}(\mathcal{T})| = m$. Let $y_1, \cdots, y_m$ be a complete left to right sequence of the terminal nodes of $\mathcal{T}$. If n lies in the range $1 \leq n \leq m$ then

$$^{[n]}\mathcal{T} = \{ x \in \mathcal{T} \mid x L^{*} \Gamma^{*} y_n \}$$

$$^{\{n\}}\mathcal{T} = {}^{[n]}\mathcal{T} \cup \{ x \in \mathcal{T} \mid \exists\ b \in \mathcal{T} \text{ s.t. } \mathit{rtn}(\mathcal{T})\, \Gamma^{*} b\, \Gamma^{*} y_n \text{ and } b \sqcap^{+} x \}$$

$^{[0]}\mathcal{T} = {}^{\{0\}}\mathcal{T} = (\emptyset,\emptyset)$ and for $n > m$, $^{[n]}\mathcal{T} = {}^{\{n\}}\mathcal{T} = \mathcal{T}$. $^{[n]}\mathcal{T}$ is called a *left $[n]$-part* of $\mathcal{T}$ and $^{\{n\}}\mathcal{T}$ is called a *left $\{n\}$-part* of $\mathcal{T}$. Thus if $\mathbf{p}$ is the root-leaf path to the $n^{\underline{th}}$ terminal node (counting from the left), then $^{[n]}\mathcal{T}$ consists of those nodes which are on or left of $\mathbf{p}$, while $^{\{n\}}\mathcal{T}$ consists of those nodes of $\mathcal{T}$ which are left of $\mathbf{p}$, or on $\mathbf{p}$, or are right of $\mathbf{p}$ and have a parent on $\mathbf{p}$. For example, in figures 7 and 8 we see in bold the left [4]-part and left {4}-part of the tree in figure 6. (Our left []-parts correspond to the left parts defined by Harrison and Havel [11].)

Next we establish those properties of left parts which will be needed later.

**Theorem 2.13.** [11] Let $\eta$ be an RCCS of the grammatical tree $\mathcal{T}$ and let n be a positive integer. The restriction of $\eta$ to $^{[n]}\mathcal{T}$ is an RCCS of $^{[n]}\mathcal{T}$.

**Theorem 2.14.** Let $\eta$ be an LCCS of the grammatical tree $\mathcal{T}$ at level $l$ and let n be a positive integer. If $\eta$ contains an internal node of $^{\{n\}}\mathcal{T}$ then $\eta$ is an LCCS of level $l$ in $^{\{n\}}\mathcal{T}$ as well. (Refer to figures 9 and 10.)

**Proof:** The proof proceeds by means of an induction on $l$.

*Basis* ($l = 0$): Let $x_0 = rtn(\mathcal{T})$. We must have $\eta = (x_0)$, since this is the only LCCS of $\mathcal{T}$ having level 0. But then $\eta$ is, by definition, an LCCS of ${}^{\{n\}}\mathcal{T}$ for every $n \geq 1$.

*Induction Step*: We assume that the theorem is true for LCCS's of $\mathcal{T}$ having level $l$ or less and extend the theorem to LCCS's of $\mathcal{T}$ having level $l+1$. Let $\eta$ be such an LCCS of level $l+1$ in $\mathcal{T}$ and let $\theta$ be the LCCS of level $l$ in $\mathcal{T}$ from which it is obtained. Let

$$\theta = (\; z_1 \cdots z_{g-1} \; z_g \; z_{g+1} \cdots z_r \;)$$
$$\eta = (\; z_1 \cdots z_{g-1} \; x_1 \cdots x_s \; z_{g+1} \cdots z_r \;)$$

so that $z_g$ is the leftmost internal node of $\theta$ with respect to $\mathcal{T}$.

The leftmost internal node of $\eta$ with respect to $\mathcal{T}$ is an internal node of ${}^{\{n\}}\mathcal{T}$ as well since by hypothesis $\eta$ contains at least one internal node of ${}^{\{n\}}\mathcal{T}$, and by definition internal nodes of $\mathcal{T}$ which are left of such a node must be internal nodes of ${}^{\{n\}}\mathcal{T}$ also. It follows that if one of

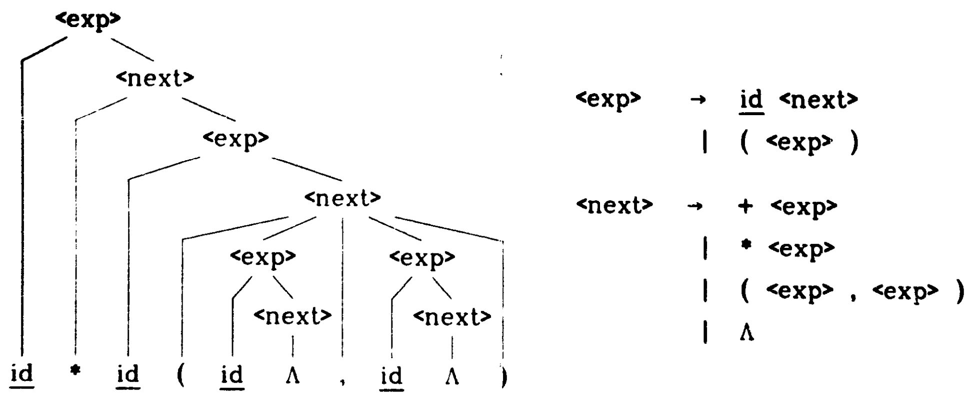



Fig. 6. The derivation tree $\mathcal{T}$ for id*id(id,id), over the indicated grammar.

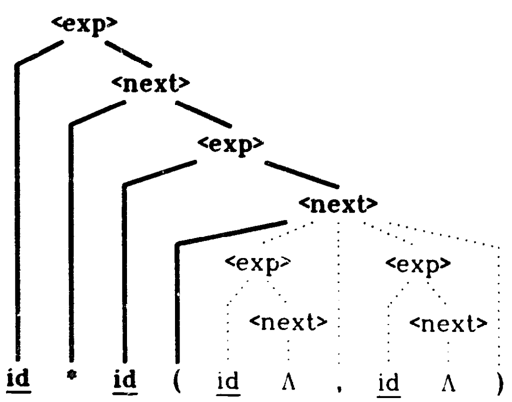



Fig. 7. The left part ${}^{[4]}\mathcal{T}$ of $\mathcal{T}$.

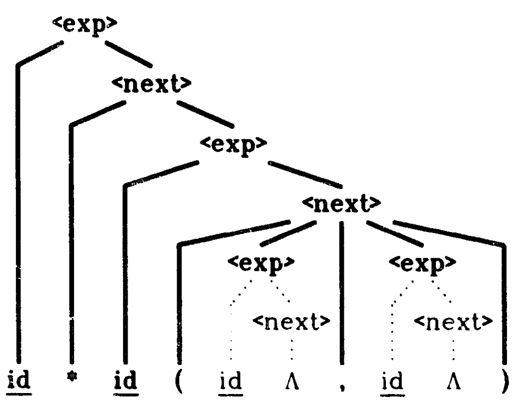



Fig. 8. The left part ${}^{\{4\}}\mathcal{T}$ of $\mathcal{T}$.

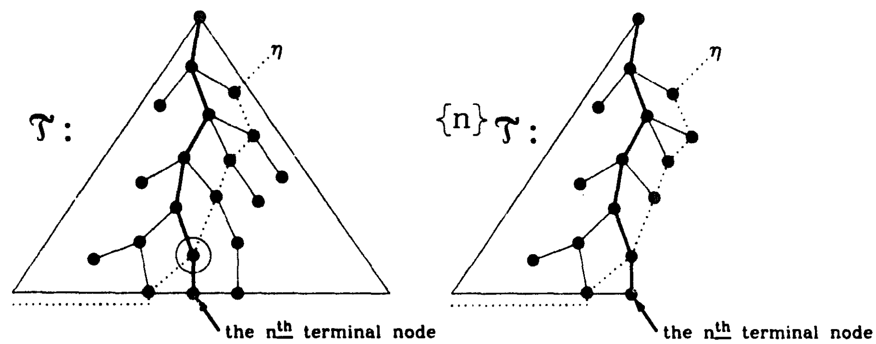



Fig. 9, illustrating theorem 2.14. The LCCS $\eta$ of $\mathcal{T}$ contains a node (circled above) which is internal to $^{\{n\}}\mathcal{T}$. Consequently $\eta$ is an LCCS of $^{\{n\}}\mathcal{T}$.

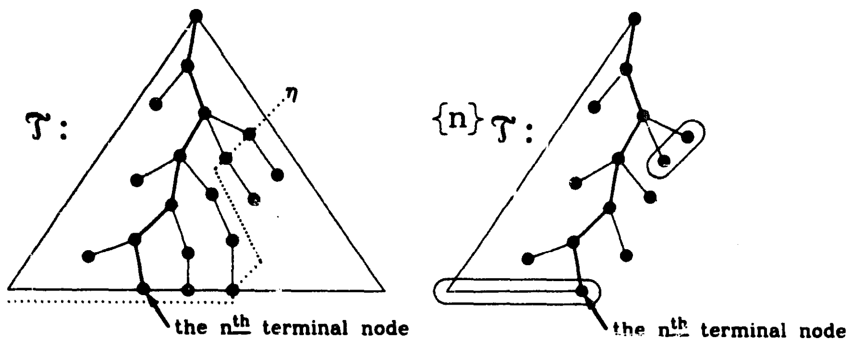



Fig. 10. The nodes of $\eta$ which belong to $^{\{n\}}\mathcal{T}$ are circled above right. None is an internal node of $^{\{n\}}\mathcal{T}$, and it is evident that $\eta$ is not an LCCS of $^{\{n\}}\mathcal{T}$.

$x_1, \cdots, x_s$ is the leftmost internal node of $\eta$ in $\mathcal{T}$ then its parent $z_g$ belongs by definition to $^{\{n\}}\mathcal{T}$. If the leftmost internal node of $\eta$ in $\mathcal{T}$ is instead one of $z_{g+1}, \cdots, z_r$ then since $z_g$ is left of that node in $\theta$ $z_g$ again must be an internal node of $^{\{n\}}\mathcal{T}$. In either case $\theta$ is an LCCS of $\mathcal{T}$ at level $l$ which contains the internal node $z_g$ of $^{\{n\}}\mathcal{T}$. It follows from the induction hypothesis that $\theta$ is an LCCS of $^{\{n\}}\mathcal{T}$ at level $l$. By definition, then, $\eta$ is an LCCS of $^{\{n\}}\mathcal{T}$ having level $l+1$, as desired. ∎

If $\eta$ does <u>not</u> contain an internal node of $^{\{n\}}\mathcal{T}$ then it need not be an LCCS of $^{\{n\}}\mathcal{T}$. Such a situation is depicted in figure 10.

**Theorem 2.15.** [11] Let $\mathcal{T}$ be a grammatical tree with respect to some cfg G, let n be a positive integer, and let $s = |fr(\mathcal{T})|$. Let $\eta = (x_1 \cdots x_k)$ be an RCCS in $^{[n]}\mathcal{T}$ and let $y_r, \cdots, y_s$ be all the leaves of $\mathcal{T}$ which are right of $x_k$; accordingly we assume $x_k \llcorner y_r \llcorner \cdots \llcorner y_s$. Then the sequence

$$\theta = (\ x_1 \cdots x_k\ y_r \cdots y_s\ )$$

is an RCCS of $\mathcal{T}$.

**Theorem 2.16.** Let $\mathcal{T}$ be a grammatical tree and n a positive integer. If $\eta$ is an LCCS of ${}^{\{n\}}\mathcal{T}$ then $\eta$ is an LCCS of $\mathcal{T}$ as well.

**Proof:** The proof is by induction on the level $l$ of $\eta$.

*Basis* ($l = 0$): It must be the case that $\eta$ is the root node, which is an LCCS of $\mathcal{T}$ by definition.

*Induction Step*: Assume that the theorem holds for all LCCS's of level $l$ or less. Let $\theta$ be an LCCS of ${}^{\{n\}}\mathcal{T}$ at level $l+1$ and let $\eta$ be the LCCS of ${}^{\{n\}}\mathcal{T}$ at level $l$ from which it is formed. By the induction hypothesis $\eta$ is an LCCS of $\mathcal{T}$. By definition, then, $\theta$ is an LCCS of $\mathcal{T}$. ∎

We will need the following special case of theorem 2.16.

**Theorem 2.17.** Let $\mathcal{T}$ a derivation tree and let n be a positive integer. Then $\mathit{leaves}({}^{\{n\}}\mathcal{T})$ is an LCCS of $\mathcal{T}$.

**Proof:** According to fact 2.6 $\mathit{leaves}({}^{\{n\}}\mathcal{T})$ is an LCCS of ${}^{\{n\}}\mathcal{T}$. It then follows from theorem 2.16 that $\mathit{leaves}({}^{\{n\}}\mathcal{T})$ is an LCCS of $\mathcal{T}$ as well. ∎

Finally, we will need to define what it means for trees, or parts of trees, to be equal.

**Definition 2.18.** Two labeled trees $\mathcal{T}$ and $\mathcal{T}'$ are said to be *structurally isomorphic*, written $\mathcal{T} \approx \mathcal{T}'$, iff there exists a bijection $\mathcal{T} \to \mathcal{T}' : x \to x'$ between the nodes of $\mathcal{T}$ and $\mathcal{T}'$ such that

- $x \ulcorner y$ iff $x' \ulcorner y'$
- $x \sqcap y$ iff $x' \sqcap y'$

(Note that we use the same symbols $\ulcorner$ and $\sqcap$ to represent the descendancy and left-right relations in both trees.) Intuitively, $\mathcal{T}$ and $\mathcal{T}'$ are identical except for labeling. If the structural isomorphism preserves labeling ($\lambda(x) = \lambda(x')$) then we say that the trees are *isomorphic* and write $\mathcal{T} = \mathcal{T}'$.

## 3. A Left Part Theorem

Our goal is to establish iteration theorems for the LL($k$) languages. Our first such theorem will be founded on an argument about derivation trees, and in particular on a characterization of derivation trees over LL($k$) grammars, which is our immediate goal. Our starting point is the following result, which is analogous to Geller's Extended LR($k$) Theorem [9].

**Theorem 3.1.** (The Extended LL($k$) Theorem). Let $G = (N,\Sigma,P,S)$ be an LL($k$) grammar. For any $A \in N$; $w, x, y \in \Sigma^*$; and $\gamma \in V^*$, if

$$(1)\ S \Rightarrow_L^{\pi} wA\gamma \Rightarrow_L^{*} wx$$
$$(2)\ S \Rightarrow_L^{*} \qquad wy$$
$$(3)\ x/k = y/k$$

then

$$(4)\ S \Rightarrow_L^{\pi} wA\gamma \Rightarrow_L^{*} wy$$

**Proof:** Assume for the sake of contradiction that (1), (2) and (3) hold, but not (4). Since the leftmost derivations of wx and wy have the initial left sentential form S in common, and (4) does <u>not</u> hold, derivations (1) and (2) diverge before reaching $wA\gamma$. Let $uB\delta$ be the last left sentential form they have in common (where $u \in \Sigma^*$, $B \in N$, and $\delta \in V^*$). Then for some $\sigma \in P^*$ and $v \in \Sigma^*$ such that $w = uv$ we have

$$S \Rightarrow_L^{\sigma} uB\delta \Rightarrow_L u\beta_1\delta \Rightarrow_L^{*} uvA\gamma \Rightarrow_L^{*} uvx = wx$$
$$S \Rightarrow_L^{\sigma} uB\delta \Rightarrow_L u\beta_2\delta \Rightarrow_L^{*} \qquad uvy = wy$$

for distinct rules $B \to \beta_1$ and $B \to \beta_2$ of G. Since $x/k = y/k$, we must have $(vx)/k = (vy)/k$. It follows that $\beta_1 = \beta_2$ since G is LL($k$), contradicting the assumption that $uB\delta$ is the last common sentential form, so that (4) must hold. ∎

This theorem describes a property of derivation trees as well as of derivations. Let wx and wy be strings in the language generated by an LL($k$) grammar G and suppose that $x/k = y/k$. Then the portions of the derivation trees $\mathcal{T}^{wx}$ and $\mathcal{T}^{wy}$ for wx and wy which have been filled in at the time the last symbol of w is exposed in leftmost derivations of wx and wy will be the same. Our left part theorem is a somewhat stronger formalization of this intuition. It is convenient to begin with the following preliminary result.

**Lemma 3.2.** Let $G = (N,\Sigma,P,S)$ be a reduced LL($k$) grammar and let $\mathcal{T}$ and $\mathcal{T}'$ be two grammatical trees over G such that $\mathit{rtl}(\mathcal{T}) = \mathit{rtl}(\mathcal{T}') = B$, where B is a variable, terminal or $\Lambda$. Let n be a non-negative integer. If for some variable A and terminal strings u, v and v′ such that $A \Rightarrow^* uBv$ and $A \Rightarrow^* uBv'$ we have $[\mathit{fr}(\mathcal{T})v]/(n+k) = [\mathit{fr}(\mathcal{T}')v']/(n+k)$ then ${}^{\{n+1\}}\mathcal{T} = {}^{\{n+1\}}\mathcal{T}'$.

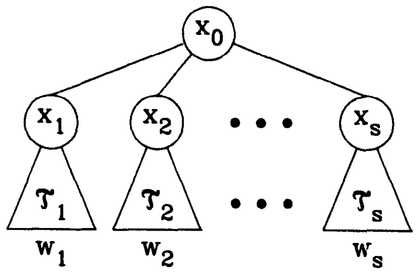
**Proof:** The proof proceeds by means of an induction on the height $h$ of the higher of the two trees $\mathcal{T}$ and $\mathcal{T}'$. Let $\mathit{rtn}(\mathcal{T}) = x_0$ and $\mathit{rtn}(\mathcal{T}') = x_0'$.

*Basis* ($k = 0$): Both $\mathcal{T}$ and $\mathcal{T}'$ consist of a single node. Suppose that $\lambda(x_0) = \lambda(x_0')$. Trivially we have $\mathcal{T} = \mathcal{T}'$, whence ${}^{\{n+1\}}\mathcal{T} = {}^{\{n+1\}}\mathcal{T}'$.

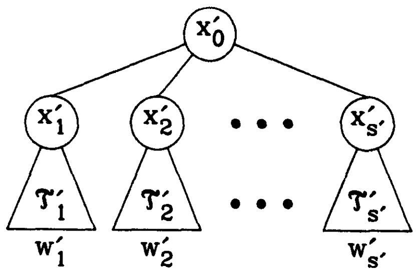
*Induction Step*: Assume that the lemma is true for trees of height $\leqslant k$, and call this assumption hypothesis H. We shall extend H to trees of height $\leqslant (k+1)$.

Without loss of generality assume that $\mathcal{T}$ has height $k+1$. Then $x_0$ is an internal node of $\mathcal{T}$ so that $B \in N$. Since $\lambda(x_0) = \lambda(x_0')$ and $\mathit{fr}(\mathcal{T}') \in \Sigma^*$ ($\mathcal{T}'$ is a grammatical tree) $x_0'$ must be an internal node of $\mathcal{T}'$.

Let $\mathcal{T}$ be the tree




and let $\mathcal{T}'$ be the tree




Our hypothesis is that

$$A \Rightarrow^* uBv$$
$$A \Rightarrow^* uBv'$$
$$\lambda(x_0) = \lambda(x_0') = B$$
$$[\mathit{fr}(\mathcal{T})v]/(n+k) = [\mathit{fr}(\mathcal{T}')v']/(n+k)$$

for some variable A and some $u, v, v' \in \Sigma^*$.

**Claim A.** The elementary subtrees rooted in $x_0$ and $x_0'$ are isomorphic. That is,

- $s = s'$
- $\lambda(x_i) = \lambda(x_i'), \qquad 1 \leqslant i \leqslant s$

**Proof of Claim A:** By definition $(x_1, \cdots, x_s)$ is a CS of $\mathcal{T}$ and $(x_1', \cdots, x_{s'}')$ is a CS of $\mathcal{T}'$. Hence by fact 2.7

$$\lambda(x_0) = B \Rightarrow \lambda(x_1 \cdots x_s) \Rightarrow^* w_1 \cdots w_s$$
$$\lambda(x_0') = B \Rightarrow \lambda(x_1' \cdots x_s') \Rightarrow^* w_1' \cdots w_{s'}'$$

Since $G_A$ is LL($k$) (theorem 1.8) and

$$(w_1 \cdots w_s v)/(n+k) = (w_1' \cdots w_{s'}' v')/(n+k)$$

it follows from theorem 1.6 that

$$\lambda(x_1 \cdots x_s) = \lambda(x'_1 \cdots x'_{s'})$$

and the claim is established. □

**Claim B.** If for some $l \leq s$ we have

(a) $\mathcal{T}_i = \mathcal{T}'_i, \qquad 1 \leq i < l$

(b) $|w_1 \cdots w_{l-1}| = |w'_1 \cdots w'_{l-1}| = m \leq n$

then for $n' = n-m$ we have ${}^{\{n'+1\}}\mathcal{T}_l = {}^{\{n'+1\}}\mathcal{T}'_l$.

**Proof of Claim B:** Observe that $\mathcal{T}_l$ and $\mathcal{T}'_l$ have height $\leq h$. If we can satisfy the conditions of hypothesis H then we will immediately obtain the desired result. If $l = s = 1$ and $\lambda(x_1) = \Lambda$ then the claim follows trivially. We may therefore assume that $x_l$ is not a $\Lambda$−node. From Claim A we know that $\lambda(x_l) = \lambda(x'_l)$. Let $C = \lambda(x_l)$. Since $x_l$ is not a $\Lambda$−node we have $C \in V$.

By assumption there exist derivations

$$A \Rightarrow^* uBv$$
$$A \Rightarrow^* uBv'$$

Since $\mathcal{T}$ and $\mathcal{T}'$ are grammatical trees there exist derivations

$$B \Rightarrow^* w_1 \cdots w_{l-1} C w_{l+1} \cdots w_s$$
$$B \Rightarrow^* w'_1 \cdots w'_{l-1} C w'_{l+1} \cdots w'_s$$

(facts 2.5 and 2.7) so that

$$A \Rightarrow^* uw_1 \cdots w_{l-1} C w_{l+1} \cdots w_s v$$
$$A \Rightarrow^* uw'_1 \cdots w'_{l-1} C w'_{l+1} \cdots w'_s v'$$

Since $w_i = w'_i$, $1 \leq i < l$, we may write

$$z = w_1 \cdots w_{l-1} = w'_1 \cdots w'_{l-1}$$

$$A \Rightarrow^* uzCw_{l+1} \cdots w_s v \tag{2}$$
$$A \Rightarrow^* uzCw'_{l+1} \cdots w'_s v' \tag{3}$$

It follows from (b) that $n' = n-m$ is a non-negative integer. Since

$$(w_1 \cdots w_s v)/(n+k) = (w'_1 \cdots w'_s v')/(n+k)$$

and $w_i = w'_i$, $1 \leq i < l$, we must have

$$(w_l \cdots w_s v)/(n'+k) = (w'_l \cdots w'_s v')/(n'+k)$$

or

$$[fr(\mathcal{T}_l) w_{l+1} \cdots w_s v]/(n'+k) = [fr(\mathcal{T}'_l) w'_{l+1} \cdots w'_s v']/(n'+k) \tag{4}$$

In view of (2), (3), (4), and the fact that $\mathcal{T}_l$ and $\mathcal{T}'_l$ have height at most $h$ we may invoke H to conclude that ${}^{\{n'+1\}}\mathcal{T}_l = {}^{\{n'+1\}}\mathcal{T}'_l$, as desired. □

**Claim C.** If for some $l \leq s$ no tree among $\mathcal{T}_1, \cdots, \mathcal{T}_l$ contains the $(n+1)^{\underline{st}}$ terminal node of $\mathcal{T}$ and no tree among $\mathcal{T}'_1, \cdots, \mathcal{T}'_l$ contains the $(n+1)^{\underline{st}}$ terminal node of $\mathcal{T}'$ then $\mathcal{T}_j = \mathcal{T}'_j$, for each $j$ in the range $1 \leq j \leq l$.

**Proof of Claim C:** The argument is an induction on $j$.

*Basis* ($j = 0$): Vacuous.

*Induction Step* ($j \geq 1$): Assume that the claim is true for indices $1, \cdots, (j-1)$. Then condition (a) of Claim B is satisfied for $l = j$. Since neither $\mathcal{T}_j$ nor $\mathcal{T}'_j$ contain the $(n+1)^{\underline{st}}$ terminal node of $\mathcal{T}$ and $\mathcal{T}'$, respectively, we have

$$|w_1 \cdots w_{j-1}| = |w'_1 \cdots w'_{j-1}| = m \leq n - |w_j|$$

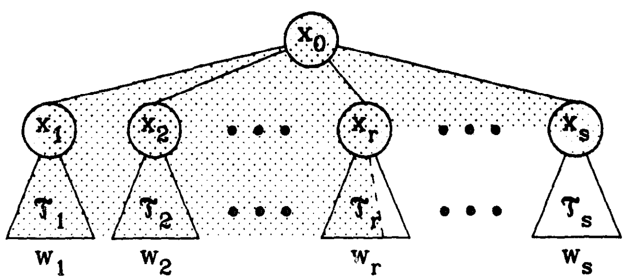
and, for $n' = n-m$,

$$n' \geq |w_j| \tag{5}$$

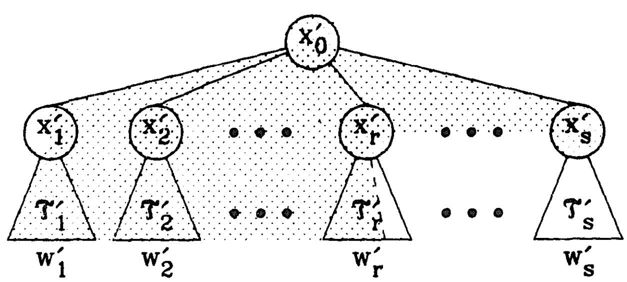
$$n' \geq |w'_j| \tag{6}$$

so that condition (b) of Claim B is satisfied and we may conclude that ${}^{\{n'+1\}}\mathcal{T}_j = {}^{\{n'+1\}}\mathcal{T}'_j$. In fact from (5) and (6) it follows that ${}^{\{n'+1\}}\mathcal{T}_j = \mathcal{T}_j$ and that ${}^{\{n'+1\}}\mathcal{T}'_j = \mathcal{T}'_j$, whence $\mathcal{T}_j = \mathcal{T}'_j$. □

Now let r be the least index such that at least one of $\mathcal{T}_r$ and $\mathcal{T}'_r$ contains the $(n+1)^{\underline{st}}$ terminal node of $\mathcal{T}$ and $\mathcal{T}'$, or $(s+1)$ if no such index exists. It follows from Claims B and C that there are isomorphisms $f_i$ establishing $\mathcal{T}_i = \mathcal{T}'_i$, $1 \leq i < r$, and (if $r \leq s$) an isomorphism $f_r$ establishing ${}^{\{n'+1\}}\mathcal{T}_r = {}^{\{n'+1\}}\mathcal{T}'_r$, where $m = |w_1 \cdots w_{r-1}| = |w'_1 \cdots w'_{r-1}|$ and $n' = n-m$. Now ${}^{\{n+1\}}\mathcal{T}$ is the shaded portion of




and ${}^{\{n+1\}}\mathcal{T}'$ is the shaded portion of



If we define the mapping $f$ by

- $f(x_0) = x'_0$
- $f(x_i) = x'_i,\ r+1 \leq i \leq s$
- $f(p) = f_i(p),\ 1 \leq i < r$, if p is a node of $\mathcal{T}_i$
- $f(p) = f_r(p)$ if p is a node of ${}^{\{n'+1\}}\mathcal{T}_r$ and $r \leq s$

then it follows easily from Claim A and the above argument that $f$ is a label-preserving structural isomorphism between ${}^{\{n+1\}}\mathcal{T}$ and ${}^{\{n+1\}}\mathcal{T}'$, so that ${}^{\{n+1\}}\mathcal{T} = {}^{\{n+1\}}\mathcal{T}'$ and the proof is complete. ∎

Lemma 3.2 is actually the forward direction of the Left Part Theorem, which we are now prepared to prove.

**Theorem 3.3.** (The LL($k$) Left Part Theorem) A reduced cfg G is LL($k$) iff the following condition holds for all $n \geq 0$: if $\mathcal{T}$ and $\mathcal{T}'$ are grammatical trees over G such that

(1) $rtl(\mathcal{T}) = rtl(\mathcal{T}')$

(2) $fr(\mathcal{T})/(n+k) = fr(\mathcal{T}')/(n+k)$

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

**Proof** ⇒: Lemma 3.2 suffices to establish the forward direction. Suppose that $G = (N,\Sigma,P,S)$ is a reduced LL($k$) grammar and that $\mathcal{T}$ and $\mathcal{T}'$ are any two grammatical trees over G such that

(1) $rtl(\mathcal{T}) = rtl(\mathcal{T}')$

(2) $fr(\mathcal{T})/(n+k) = fr(\mathcal{T}')/(n+k)$

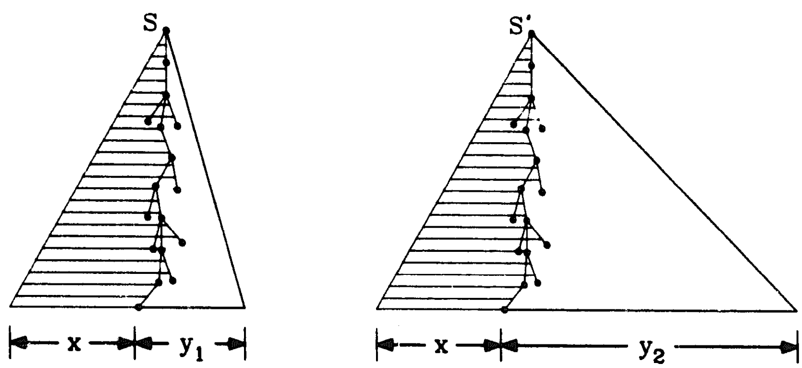



Fig. 11, illustrating the Left Part Theorem for LL languages. The left {|x|+1}–parts of derivation trees for $xy_1$ and $xy_2$ are shown shaded. These left parts are the portions of the respective trees which have been filled in at the time all of $x(y_1/1)$ and $x(y_2/1)$ have been exposed. If the grammar is LL(k) and $y_1/k = y_2/k$ then these left parts are necessarily identical.

Let $A = \mathit{rtl}(\mathcal{T}) = \mathit{rtl}(\mathcal{T}') = B$ and $u = v = v' = \Lambda$. For the derivations $A \Rightarrow^* uBv$ and $A \Rightarrow^* uBv'$ we use the trivial derivation $A \Rightarrow^* A$. Since $v = v' = \Lambda$,

$$[\mathit{fr}(\mathcal{T})v]/(n+k) = [\mathit{fr}(\mathcal{T}')v']/(n+k)$$

follows immediately from (2). We have now satisfied the hypothesis of lemma 3.2, and may therefore conclude that ${}^{\{n+1\}}\mathcal{T} = {}^{\{n+1\}}\mathcal{T}'$, as desired.

**Proof** $\Leftarrow$**:** Let $G = (N,\Sigma,P,S)$ be a reduced cfg with the property that if $\mathcal{T}$ and $\mathcal{T}'$ are any two grammatical trees over G such that

(1) $\mathit{rtl}(\mathcal{T}) = \mathit{rtl}(\mathcal{T}')$

(2) $\mathit{fr}(\mathcal{T})/(n+k) = \mathit{fr}(\mathcal{T}')/(n+k)$

then ${}^{\{n+1\}}\mathcal{T} = {}^{\{n+1\}}\mathcal{T}'$. We intend to show that G must necessarily be an LL($k$) grammar. For suppose that G is not LL($k$). In view of theorem 1.7 there must exist a pair of derivations

$$S \Rightarrow_L^* uA\beta \Rightarrow_L u\alpha\beta \Rightarrow_L^* uv$$

$$S \Rightarrow_L^* uA\beta \Rightarrow_L u\alpha'\beta \Rightarrow_L^* uv'$$

such that $v/k = v'/k$ and $\alpha \neq \alpha'$. Let $\mathcal{T}$ and $\mathcal{T}'$ be derivation trees over G for uv and uv', respectively, and let $n = |u|$ so that $(uv)/(n+k) = (uv')/(n+k)$. Since $\mathit{rtl}(\mathcal{T}) = S = \mathit{rtl}(\mathcal{T}')$, $\mathit{fr}(\mathcal{T}) = uv$, and $\mathit{fr}(\mathcal{T}') = uv'$ there exists by assumption an isomorphism $\mathbf{f}$ establishing ${}^{\{n+1\}}\mathcal{T} = {}^{\{n+1\}}\mathcal{T}'$. Let

$$\eta = (\ z_1 \cdots z_g \cdots z_r\ )$$

$$\eta' = (\ z'_1 \cdots z'_{g'} \cdots z'_{r'}\ )$$

be the unique LCCS's at level $l$ in $\mathcal{T}$ and $\mathcal{T}'$ (fact 2.3) having the label $uA\beta$, in which $z_g$ and $z'_{g'}$ are the leftmost internal nodes (so that they are labeled with A). Since $n = |u|$ and $u = \lambda(z_1, \cdots, z_{g-1})$ the $(n+1)^{\text{st}}$ terminal node of $\mathcal{T}$ is either one of the nodes $z_{g+1}, \cdots, z_r$ or is descended from one of the nodes $z_g, \cdots, z_r$. Similarly the $(n+1)^{\text{st}}$ terminal node of $\mathcal{T}'$ is either one of the nodes $z'_{g'+1}, \cdots, z'_{r'}$ or is descended from one of the nodes $z'_{g'}, \cdots, z'_{r'}$. Accordingly $\eta$ and $\eta'$ each contain an internal node of ${}^{\{n+1\}}\mathcal{T}$ and ${}^{\{n+1\}}\mathcal{T}'$ – $z_g$ and $z'_{g'}$, respectively. According to theorem 2.14 it follows that $\eta$ and $\eta'$ are LCCS's of ${}^{\{n+1\}}\mathcal{T}$ and ${}^{\{n+1\}}\mathcal{T}'$ at level $l$. Since $\mathbf{f}$ is an isomorphism it must be the case that $\mathbf{f}(\eta)$ is an LCCS of ${}^{\{n+1\}}\mathcal{T}'$ at level $l$. But $\eta'$ is also an LCCS of ${}^{\{n+1\}}\mathcal{T}'$ having level $l$. Since there can be at most one such LCCS (fact 2.3) we must have $\mathbf{f}(\eta) = \eta'$. It follows that $g = g'$ and $\mathbf{f}(z_g) = z'_{g'}$. Since $z_g$ and $z'_g$ are internal nodes of ${}^{\{n+1\}}\mathcal{T}$ and ${}^{\{n+1\}}\mathcal{T}'$, their children must belong to ${}^{\{n+1\}}\mathcal{T}$ and ${}^{\{n+1\}}\mathcal{T}'$, respectively, so that the elementary subtrees rooted in $z_g$ and $z'_{g'}$ are isomorphic. That is to say, if $x_1, \cdots, x_s$

are the children of $z_g$ and $x'_1, \cdots, x'_{s'}$ are the children of $z'_{g'}$ then $s = s'$ and

$$\lambda(\ x_1 \cdots x_s\ )\ =\ \lambda(\ x'_1 \cdots x'_{s'}\ )$$

But

$$\lambda(\ x_1 \cdots x_s\ )\ =\ \alpha$$
$$\lambda(\ x'_1 \cdots x'_{s'}\ )\ =\ \alpha'$$

so that $\alpha = \alpha'$, which we assumed was not the case. Consequently G must be LL($k$). ■

## 4. Iteration Theorems

Armed with the Left Part Theorem our intent is to establish some pumping properties of the LL($k$) languages. Roughly speaking, we will invoke the argument used in establishing Ogden's lemma to obtain the usual decomposition of the derivation tree for a string w belonging to an LL($k$) language L in which we have distinguished a sufficient number of positions. This induces the usual factorization of w as $w_1w_2w_3w_4w_5$. By looking at derivation trees for w and for any other string $w_1w_2u$ in L such that $(w_3w_4w_5)/k = u/k$, and applying the Left Part Theorem appropriately, we will obtain our first iteration theorem. We will need the following definitions.

**Definition 4.1.** Let $w \in \Sigma^*$ and let n be a positive integer. If $w_1 \cdots w_n = w$, where $w_i \in \Sigma^*$ for $1 \leq i \leq n$, then the sequence $(w_1, \cdots, w_n)$ is said to be a *factorization* of w.

**Definition 4.2.** Let $w \in \Sigma^*$. Suppose that $w = a_1a_2 \cdots a_n$, where each $a_i \in \Sigma$. Any index $i$, $1 \leq i \leq n$, is called a *position* in w. For example, the symbol occupying position 3 of the string aacbda is c. Next let $\mathcal{K}$ be any set of positions in a terminal string w. Any factorization $\varphi = (w_1, w_2, w_3, w_4, w_5)$ of w induces a natural "partition" $\mathcal{K}/\varphi$ of $\mathcal{K}$ into:

$$\mathcal{K}/\varphi = \{\ \mathcal{K}_1,\ \mathcal{K}_2,\ \mathcal{K}_3,\ \mathcal{K}_4,\ \mathcal{K}_5\ \}$$

where

$$\mathcal{K}_i\ =\ \{\ k \in \mathcal{K}\ \big|\ |w_1 \cdots w_{i-1}| < k \leq |w_1 \cdots w_i|\ \}$$

Thus $\mathcal{K}_i$ selects out of $\mathcal{K}$ those positions which appear in $w_i$. We call the elements of $\mathcal{K}$ *distinguished positions* (or *dp's*). The following notation will also be convenient.

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

$$\prod_{i=1}^{r}(u_i) = u_1u_2\cdots u_{r-1}u_r$$

We are now ready to proceed.

**Theorem 4.4.** (The First LL Iteration Theorem) Let L be an LL($k$) language. There exists an integer p such that given a string w in L and p or more distinguished positions $\mathcal{K}$ in w we may write

$$\varphi = (w_1,w_2,w_3,w_4,w_5)$$
$$\mathcal{K}/\varphi = \{ \mathcal{K}_1, \mathcal{K}_2, \mathcal{K}_3, \mathcal{K}_4, \mathcal{K}_5 \}$$

where

(1) $w_2 \neq \Lambda$

(2) a: Either $w_1$, $w_2$ and $w_3$ each contain dp's ($\mathcal{K}_1, \mathcal{K}_2, \mathcal{K}_3 \neq \emptyset$), or $w_3$, $w_4$ and $w_5$ each contain dp's ($\mathcal{K}_3, \mathcal{K}_4, \mathcal{K}_5 \neq \emptyset$),

b: and $w_2w_3w_4$ contains at most p dp's ($|\mathcal{K}_2 \cup \mathcal{K}_3 \cup \mathcal{K}_4| \le p$).

(3) a: Let $n = |w_1w_2|$ and suppose that $w'$ is any string in L such that $w'/(n+k) = w/(n+k)$. Then there is a factorization $(w_1,w_2,w_3',w_4',w_5')$ of $w'$ such that

(i) $w_1w_2^rw_3\prod_{i=1}^{r}(u_i)w_5$

(ii) $w_1w_2^rw_3'\prod_{i=1}^{r}(u_i)w_5$

(iii) $w_1w_2^rw_3\prod_{i=1}^{r}(u_i)w_5'$

(iv) $w_1w_2^rw_3'\prod_{i=1}^{r}(u_i)w_5'$

are in L for all $r \ge 0$ and for all strings $\prod_{i=1}^{r}(u_i)$ in which $u_i = w_4$ or $u_i = w_4'$, $1 \le i \le r$.

b: Furthermore, if $\prod_{i=1}^{r}(\bar{u}_i)$ is a catenation of words $\bar{u}_i \in \{w_4, w_4'\}$ such that

$$\prod_{i=1}^{r}(u_i) = \prod_{i=1}^{r}(\bar{u}_i)$$

then $u_i = \bar{u}_i$, $1 \le i \le r$.

**Proof:** Let $G = (N,\Sigma,P,S)$ be an arbitrary reduced LL($k$) grammar generating L. The methods used by Ogden [20] (or see Harrison and Havel [11]) suffice to establish the existence of an integer p such that for any string w in L in which p or more positions $\mathcal{K}$ 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 $A \Rightarrow^+ w_2Aw_4$ we have

$$S \Rightarrow^* w_1Aw_5 \Rightarrow^* w_1w_2^rAw_4^rw_5 \Rightarrow^+ w_1w_2^rw_3w_4^rw_5$$

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

Let $n = |w_1w_2|$ and consider any string $w'$ in L such that $w'/(n+k) = w/(n+k)$. Let $\mathcal{T}$ and $\mathcal{T}'$ be the derivation trees for w and w′, respectively. Since $w/(n+k) = w'/(n+k)$ we may invoke the Left Part Theorem to obtain ${}^{\{n+1\}}\mathcal{T} = {}^{\{n+1\}}\mathcal{T}'$. (Refer to figure 12.)

Consider $\mathcal{T}$. Let x and y be the internal nodes of $\mathcal{T}$ corresponding to the A's in $w_1Aw_5$ and $w_1w_2Aw_4w_5$. We know that $w_3 \neq \Lambda$ since $\mathcal{K}_3 \neq \emptyset$.

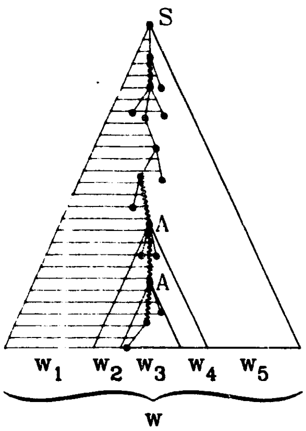



Fig. 12a: $\mathcal{T}$.

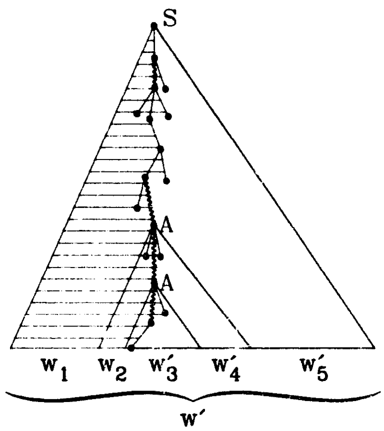



Fig. 12b: $\mathcal{T}'$.

Fig. 12. Derivation trees for w and w′, in which the left $\{|w_1w_2|+1\}$–parts are shaded. As a result of the fact that G is LL(k) and $(w_1w_2w_3w_4w_5)/(|w_1w_2|+k) = (w_1w_2w_3'w_4'w_5')/(|w_1w_2|+k)$ the left $\{|w_1w_2|+1\}$–parts are isomorphic. In particular, the two nodes labeled A in ${}^{\{n+1\}}\mathcal{T}$ must appear in the same position in ${}^{\{n+1\}}\mathcal{T}'$.

Therefore the subtree rooted in y has a terminal node among its leaves. The leftmost such terminal node $\mathfrak{n}$ is labeled with $w_3/1$ and is contained in ${}^{\{n+1\}}\mathcal{T}$; it is, in fact, the $(n+1)^{\underline{st}}$ terminal node of $\mathcal{T}$. Since the nodes x and y defined above lie on the root–leaf path to $\mathfrak{n}$ they also belong to ${}^{\{n+1\}}\mathcal{T}$. (They appear in figure 12a labeled by A). Let $\mathfrak{f}$ be the isomorphism of the Left Part Theorem. It follows that

$$A = \lambda(x) = \lambda(\mathfrak{f}(x))$$
$$A = \lambda(y) = \lambda(\mathfrak{f}(y))$$

Let $\eta$ and $\theta$ now be the unique LCCS's of $\mathcal{T}$ in which the leftmost internal nodes are x and y, respectively (fact 2.5). We may write

$$\eta = (\ a_1 \cdots a_{\underline{a}}\ x\ d_1 \cdots d_{\underline{d}}\ ) \tag{7}$$

$$\theta = (\ a_1 \cdots a_{\underline{a}}\ b_1 \cdots b_{\underline{b}}\ y\ c_1 \cdots c_{\underline{c}}\ d_1 \cdots d_{\underline{d}}\ ) \tag{8}$$

Since x and y are both internal nodes of ${}^{\{n\}}\mathcal{T}$, $\eta$ and $\theta$ are LCCS's of ${}^{\{n+1\}}\mathcal{T}$ as well (theorem 2.14). Since ${}^{\{n+1\}}\mathcal{T} = {}^{\{n+1\}}\mathcal{T}'$, $f(\eta)$ and $f(\theta)$ are LCCS's of ${}^{\{n+1\}}\mathcal{T}'$, and hence of $\mathcal{T}'$ (theorem 2.16). Again because ${}^{\{n+1\}}\mathcal{T} = {}^{\{n+1\}}\mathcal{T}'$ we may conclude that $\lambda(\eta) = \lambda(f(\eta))$ and $\lambda(\theta) = \lambda(f(\theta))$. In particular,

$$w_1 = \lambda(\ a_1 \cdots a_{\underline{a}}\ ) = \lambda(f(\ a_1 \cdots a_{\underline{a}}\ ))$$

$$w_2 = \lambda(\ b_1 \cdots b_{\underline{b}}\ ) = \lambda(f(\ b_1 \cdots b_{\underline{b}}\ ))$$

and for some $\alpha$, $\beta \in V^*$

$$\alpha = \lambda(\ c_1 \cdots c_{\underline{c}}\ ) = \lambda(f(\ c_1 \cdots c_{\underline{c}}\ ))$$

$$\beta = \lambda(\ d_1 \cdots d_{\underline{d}}\ ) = \lambda(f(\ d_1 \cdots d_{\underline{d}}\ ))$$

Now by invoking theorem 2.8 we obtain from $\mathcal{T}$ the derivations

$$S \quad \Rightarrow_L^* \lambda(a_1 \cdots a_{\underline{a}})\lambda(x)\lambda(d_1 \cdots d_{\underline{d}}) = w_1 A\beta \tag{9}$$

$$A = \lambda(x) \quad \Rightarrow_L^* \lambda(b_1 \cdots b_{\underline{b}})\lambda(y)\lambda(c_1 \cdots c_{\underline{c}}) = w_2 A\alpha \tag{10}$$

$$A = \lambda(y) \quad \Rightarrow_L^* w_3 \tag{11}$$

$$\alpha \quad \Rightarrow_L^* w_4 \tag{12}$$

$$\beta \quad \Rightarrow_L^* w_5 \tag{13}$$

and from $\mathcal{T}'$ the derivations

$$S \quad \Rightarrow_L^* \lambda(f(a_1 \cdots a_{\underline{a}}))\lambda(f(x))\lambda(f(d_1 \cdots d_{\underline{d}})) = w_1 A\beta \tag{14}$$

$$A = \lambda(f(x)) \quad \Rightarrow_L^* \lambda(f(b_1 \cdots b_{\underline{b}}))\lambda(f(y))\lambda(f(c_1 \cdots c_{\underline{c}})) = w_2 A\alpha \tag{15}$$

$$A = \lambda(f(y)) \quad \Rightarrow_L^* w_3' \tag{16}$$

$$\alpha \quad \Rightarrow_L^* w_4' \tag{17}$$

$$\beta \quad \Rightarrow_L^* w_5' \tag{18}$$

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

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

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*). (Season to taste.)

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 $p_i$, $1 \leq i \leq r$, be defined by

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

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

$$\{(9)\}\ \{(10)\}^r\ \{(11)\}\ \prod_{i=1}^{r}\{p_i\}\ \{(13)\}$$

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

$$\prod_{i=1}^{r}(u_i) \quad \text{and} \quad \prod_{i=1}^{r}(\bar{u}_i)$$

and corresponding sequences $p_i$ and $\bar{p}_i$ such that

$$\prod_{i=1}^{r}(u_i) = \prod_{i=1}^{r}(\bar{u}_i)$$

and for which $u_i \neq \bar{u}_i$, for some $i$ in the range $1 \leq i \leq r$, so that $p_i \neq \bar{p}_i$, 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. ■

Before proceeding with a formal development of a second pumping lemma for the LL($k$) languages, we sketch the intuition underlying our argument. (Refer to figure 13.) Suppose that uv and uvy, $|v| = k$, are strings in some language L generated by a $\Lambda$-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 or uvy there can be no more than $k$ variables remaining in the left sentential form since G is $\Lambda$-free and $|v| = k$. 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, which is due to Rosenkrantz and Stearns.

**Theorem 4.5.** Given an LL($k$) grammar $G = (N,\Sigma,P,S)$ we can construct an LL($k+1$) grammar $G' = (N',\Sigma,P',S')$ such that $\mathscr{L}(G') = \mathscr{L}(G)$ and $G'$ is $\Lambda$-free unless $\Lambda \in \mathscr{L}(G)$, in which case $G'$ contains the single $\Lambda$-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 Rosenkrantz and Stearns [22], pages 236–241 (or see Aho and Ullman [2], pages 674–681), we may obtain a $\Lambda$-free LL($k+1$) grammar $G'' = (N'',\Sigma,P'',S'')$ generating $\mathscr{L}(G) - \{\Lambda\}$. If $\Lambda \notin \mathscr{L}(G)$ then set $G' = G''$.
Suppose, however, that $\mathscr{L}(G)$ contains $\Lambda$. 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 $V''$. It is trivial to prove that $G'$ is also LL($k+1$) and generates exactly $\mathscr{L}(G)$. ∎

**Theorem 4.6.** (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 $|x| \geq k$ 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 $|w_1| \geq |x| - k$.

**Proof.** In view of theorem 4.5 we may assume that L is generated by some LL($k$) grammar $G = (N,\Sigma,P,S)$ which is $\Lambda$-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 $G_A = (N,\Sigma,P,A)$ be the cfg obtained from G by changing the start symbol to A, let $p_A$ be the constant obtained from the First Iteration Theorem for the language $\mathscr{L}(G_A)$ (which is also LL($k$) – see theorem 1.8), and let

$$p' = \max\{\, p_A \mid A \in N \,\}$$
$$p = kp' + 1$$

Suppose that x and xy are strings belonging to L, where $|x| \geq k$ and p or more positions are distinguished in y. Let us write x as uv, where $|u| = n$ and $|v| = k$, and let $\mathcal{T}$ and $\mathcal{T}'$ be derivation trees for uv and uvy. (See figure 13.) Let $\eta = \mathit{leaves}({}^{\{n+1\}}\mathcal{T})$ and $\eta' = \mathit{leaves}({}^{\{n+1\}}\mathcal{T}')$.

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

$$S \Rightarrow_L^* u\gamma = \lambda(\eta) \Rightarrow_L^* uv$$
$$S \Rightarrow_L^* u\gamma = \lambda(\eta') \Rightarrow_L^* uvy$$

for some $\gamma$ in $V^*$ (fact 2.8). Since $|v| = k \geq 1$ these derivations involve no $\Lambda$-rules. It follows that $|\gamma| \leq k$ since $|v| = k$ and $\gamma \overset{*}{\Rightarrow}_L v$.

Now write $\gamma$ as $X_1X_2 \cdots X_s$ $(s \leq k)$. Let $(z_1, z_2, \cdots, z_s)$ be the factorization of vy such that $X_i \overset{*}{\Rightarrow}_L z_i$, $1 \leq i \leq s$. Suppose that there are $p'$ or fewer dp's in each $z_i$. Then there are at most $sp' \leq kp' < p$ dp's in vy, which is not the

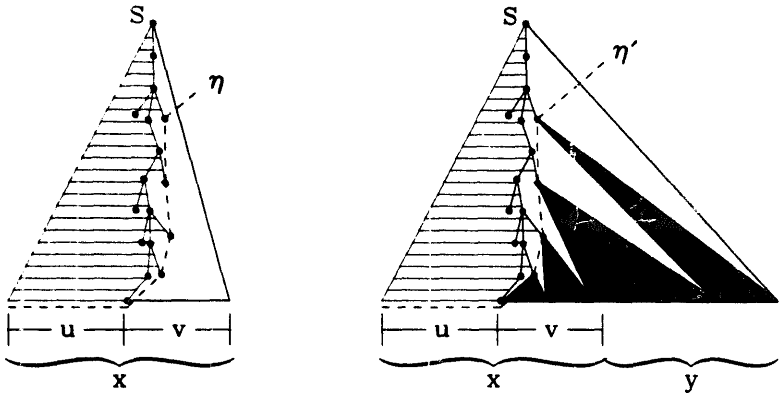



Fig. 13a. $\mathcal{T}$. Fig. 13b. $\mathcal{T}'$.

Fig. 13. The solidly shaded areas indicate the leaves descended from a particular internal node of $\mathcal{T}'$ which is a leaf of the left $\{|u|+1\}$-part of $\mathcal{T}'$. The dashed lines mark the frontier of the left $\{|u|+1\}$-parts for each tree. This is the left sentential form obtained at the time v/1 is exposed.

case. Hence some particular $z_i$ contains more than $p' \geq p_{X_i}$ dp's. Now the string $z_i$ belongs to the language $\mathcal{L}(G_{X_i})$, 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 4.4 hold with respect to $\mathcal{L}(G_{X_i})$ and for some variable B we have $B \overset{+}{\Rightarrow} \sigma_2 B \sigma_4$ 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 $|w_1| \geq |x| - k$. If we let

$$n = |uz_1 \cdots z_{i-1}\sigma_1\sigma_2|$$

and consider any string $w'$ in L such that $w'/(n+k) = w/(n+k)$, the argument used to deduce (3) in theorem 4.4 may be used to deduce property (3) here, and the proof is complete. ∎

## 5. Applications

We begin by showing that every LL($k$) grammar is LR($k$). This is not a new result; Brosgol [8] 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 [23]. It is more often argued intuitively from a consideration of LL($k$) and LR($k$) derivation trees that this result is obvious (see Aho and Ullman [2], for example). Using the LL($k$) Left Part Theorem we can now make the tree argument rigorous.

**Theorem 5.1.** Every reduced LL($k$) grammar is LR($k$), $k \geq 0$.

**Proof:** Let G be an arbitrary LL($k$) grammar. First of all, $S \Rightarrow_R^+ S$ is impossible since G is unambiguous. Hence if G is not LR($k$) then for some $w, w', x \in \Sigma^*$; $\alpha, \alpha', \beta, \beta' \in V^*$; $A, A' \in N$, there exist derivations

$$S \Rightarrow_R^* \alpha A w \Rightarrow_R \alpha\beta w$$
$$S \Rightarrow_R^* \alpha' A' x \Rightarrow_R \alpha'\beta' x = \alpha\beta w'$$

such that $w/k = w'/k$ and $(A \rightarrow \beta, |\alpha\beta|) \neq (A' \rightarrow \beta', |\alpha'\beta'|)$. If $k = 0$ then either $\mathscr{L}(G)$ is empty, in which case there are no derivations at all since G is reduced, or $\mathscr{L}(G)$ is a singleton set, in which case we have $\alpha\beta w = \alpha\beta w'$ and consequently $(A \rightarrow \beta, |\alpha\beta|) = (A' \rightarrow \beta', |\alpha'\beta'|)$ since both sentential forms must derive the same string and G is unambiguous. We need therefore only consider the case in which $k \geq 1$.

Let $z \in \mathscr{L}(\alpha\beta)$, let $\mathcal{T}$ be the derivation tree for zw, let $\mathcal{T}'$ be the derivation tree for zw', and let $n = |z|$. Since G is LL($k$) and $(zw)/(n+k) = (zw')/(n+k)$, we may apply the Left Part Theorem to obtain ${}^{\{n+1\}}\mathcal{T} = {}^{\{n+1\}}\mathcal{T}'$. Let f be the mapping which effects the isomorphism. Let $\eta = (u_1, \cdots, u_s)$ be the unique RCCS of $\mathcal{T}$ having the label $\alpha A w$ (theorems 1.2 and 2.9). Let $u_i$ be the node of $\eta$ labeled by the A explicitly shown in $\alpha A w$, and let

$$\theta = (\ u_1 \cdots u_{i-1}\ v_1 \cdots v_h\ u_{i+1} \cdots u_s\ )$$

be the RCCS formed from $\eta$ by expanding $u_i$, so that $\lambda(v_1 \cdots v_h) = \beta$ and $\lambda(\theta) = \alpha\beta w$. (Refer to figure 14a.) Let $a = w/1$ ($a \in \Sigma_\Lambda$). Since $w/k = w'/k$ and $k \geq 1$, we also have $a = w'/1$. Consider ${}^{[n+1]}\mathcal{T}$: $\mathit{fr}({}^{[n+1]}\mathcal{T}) = za$. Let $\chi = (u_1, \cdots, u_r)$ be the restriction of $\eta$ to ${}^{[n+1]}\mathcal{T}$ and recall that $\lambda(\eta) = \alpha A w$. If $a \in \Sigma$ then $i < r$, since the first n terminals are derived from $\alpha A$, and $u_i$ belongs to ${}^{[n+1]}\mathcal{T}$. If $a = \Lambda$ (because $w = \Lambda$) then

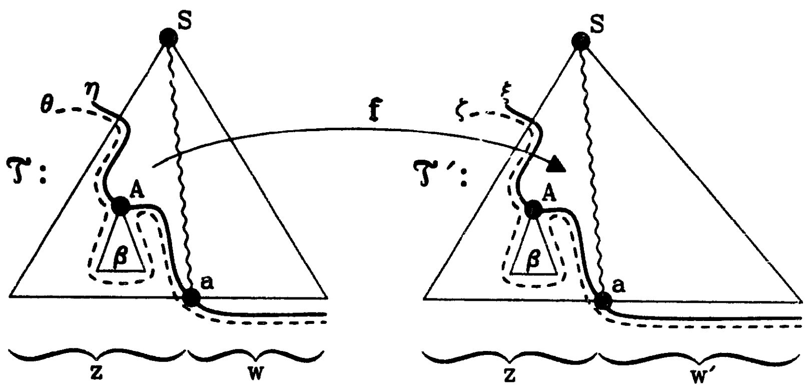



Figure 14a, illustrating the proof of theorem 5.1. In $\mathcal{T}$ we show $\eta$ and $\theta$, the unique RCCS's of $\mathcal{T}$ labeled $\alpha$Aw and $\alpha\beta$w. In $\mathcal{T}'$ we show RCCS's $\xi$ and $\zeta$, the extensions of $\chi'$ and $\psi'$ (see figure 14b below) to $\mathcal{T}'$ from ${}^{[n+1]}\mathcal{T}'$. The isomorphism f maps ${}^{[n+1]}\mathcal{T}$ onto ${}^{[n+1]}\mathcal{T}'$.

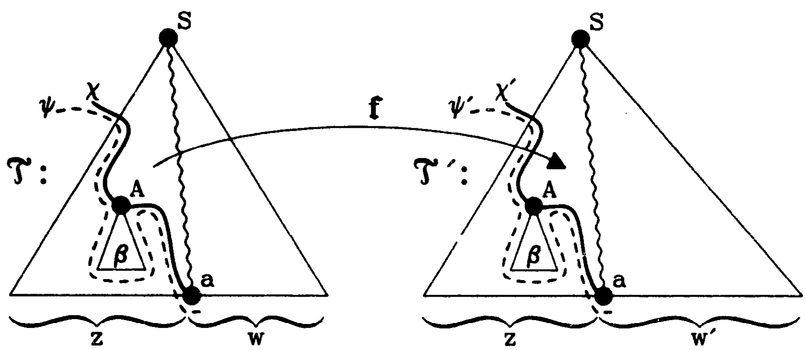



Figure 14b, illustrating the proof of theorem 5.1. In $\mathcal{T}$ we show the restrictions $\chi$ and $\psi$ of $\eta$ and $\theta$ to ${}^{[n+1]}\mathcal{T}$. In $\mathcal{T}'$ we show the isomorphic images $\chi'$ and $\psi'$ of $\chi$ and $\psi$ under f. Since ${}^{[n+1]}\mathcal{T} = {}^{[n+1]}\mathcal{T}'$ we have $\lambda(\chi) = \lambda(\chi') = \alpha$Aa and $\lambda(\psi) = \lambda(\psi') = \alpha\beta$a.

${}^{[n+1]}\mathcal{T} = \mathcal{T}$, so that $r = s$, $\chi = \eta$, and $\lambda(\chi) = \alpha\text{Aa} = \alpha\text{Aw} = \alpha\text{A}$. In either case $\lambda(\chi) = \alpha\text{Aa}$ $(i \leq r \leq s)$, so that $u_i$ appears in $\chi$. Next let

$$\psi = (\ u_1 \cdots u_{i-1}\ v_1 \cdots v_h\ u_{i+1} \cdots u_r\ )$$

be the restriction of $\theta$ to ${}^{[n+1]}\mathcal{T}$, so that $\lambda(\psi) = \alpha\beta\text{a}$. $\chi$ and $\psi$ are RCCS's of ${}^{[n+1]}\mathcal{T}$ (theorem 2.13), $\psi$ being obtained in one step from $\chi$ by rewriting $u_i$. Since ${}^{\{n+1\}}\mathcal{T} = {}^{\{n+1\}}\mathcal{T}'$ under f we must also have ${}^{[n+1]}\mathcal{T} = {}^{[n+1]}\mathcal{T}'$ under f. If we let $\chi' = f(\chi)$ and $\psi' = f(\psi)$ then

$$\lambda(\chi) = \lambda(\chi') = \alpha\text{Aa}$$
$$\lambda(\psi) = \lambda(\psi') = \alpha\beta\text{a}$$

and in view of the isomorphism $\chi'$ and $\psi'$ must be RCCS's of ${}^{[n+1]}\mathcal{T}'$, $\psi'$ being obtained in one step from $\chi'$ by rewriting $f(u_i)$. Now extend $\chi'$ to form an RCCS $\xi$ in $\mathcal{T}'$ by appending to $\chi'$ (in left-to-right order) all of the leaves of $\mathcal{T}'$ which are right of $f(u_r)$ (theorem 2.15), so that

$\lambda(\xi) = \alpha A w'$. Similarly extend $\psi'$ to obtain an RCCS $\zeta$ in $\mathcal{T}'$ such that $\lambda(\zeta) = \alpha\beta w'$. Since there are no internal nodes to the right of $u_i$ in $\eta$, there can be no internal nodes to the right of $u_i$ in $\chi$, and no internal nodes to the right of $f(u_i)$ in $\chi'$. Since $\xi$ is obtained from $\chi'$ by appending leaves, $f(u_i)$ is also the rightmost internal node of $\xi$. Hence $\zeta$ is an RCCS of $\mathcal{T}'$ which can be obtained from the RCCS $\xi$ of $\mathcal{T}'$ in one step by rewriting $f(u_i)$. We must have

$$rt\ell(\mathcal{T}) \Rightarrow_R^* \lambda(\xi) \Rightarrow_R \lambda(\zeta)$$

(fact 2.8). That is,

$$S \Rightarrow_R^* \alpha A w' \Rightarrow_R \alpha\beta w'$$

Since we also know that

$$S \Rightarrow_R^* \alpha' A' x \Rightarrow_R \alpha'\beta' x = \alpha\beta w'$$

and that G is unambiguous (theorem 1.2) it must be the case that $\alpha = \alpha'$, $\beta = \beta'$, and $A = A'$ so that $(A\to\beta,|\alpha\beta|) = (A'\to\beta',|\alpha'\beta'|)$ which is a contradiction. Hence G is, in fact, an LR($k$) grammar. ∎

It is necessary for the proof of theorem 5.1 that the grammar be reduced. For suppose that $(N,\Sigma,P,S)$ is a reduced LL($k$) grammar. If we add to G the rules $S \to A$ and $A \to A$ for some new variable A then it is easy to see from the definitions that G is still LL($k$) but not LR($k$). On the other hand, the presence in G of variables which cannot be derived from the start symbol does not effect the proof.

We next consider a number of results which follow easily from our iteration theorems. Theorems 5.2, 5.3, 5.4, 5.5 and 5.6 each illustrate a different way in which possessing the LL($k$) property restricts the form of strings in a language; each of the proofs illustrates a different way in which the iteration theorems may be used. We consider only languages which are LR($k$) since every LL($k$) language is LR($k$); if a language is not even LR($k$) then other tools already exist for demonstrating this which incidently demonstrate that the language also fails to be LL.

**Theorem 5.2.** The LR language $L_1 = \{\ a^n b^n,\ a^n c^n \mid n \geq 1\ \}$ is not LL.

**Proof:** (Figure 15.) Assume that $L_1$ is LL($k$) and let p be the constant obtained for $L_1$ from the First Iteration Theorem. Consider the string $w = a^p a^k b^{p+k}$ in which the first p a's are distinguished. From theorem 4.4 we obtain the usual factorization $\varphi = (w_1,w_2,w_3,w_4,w_5)$ of w. If $w_2$ or $w_4$ contained both a's and b's then in $w_1 w_2^2 w_3 w_4^2 w_5$ an a would follow a b, which cannot happen. Hence $w_2$, and similarly $w_4$, must consist

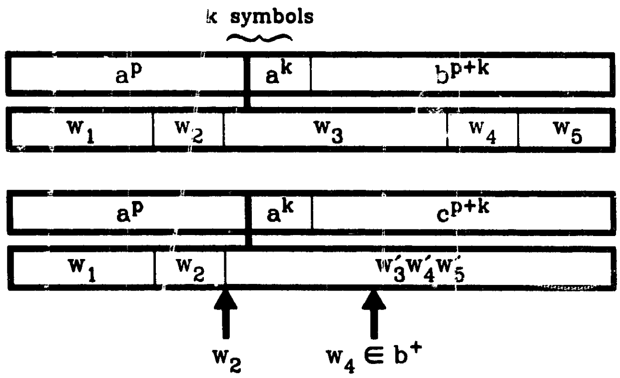



Fig.15. An application of Theorem 4.4 to the language $a^nb^n + a^nc^n$.

entirely of a's or of b's. Moreover, if $w_1w_2^2w_3w_4^2w_5$ is to contain an equal number of a's and b's then (since at least one of $w_2$ and $w_4$ is non-null) we must have $w_2 \in a^+$ and $w_4 \in b^+$. Also, $w_3$ must begin with at least k a's since $w_4$ does not contain any distinguished positions. Now consider $a^{p+k}c^{p+k}$, which we can write as $w_1w_2u$ for some $u \in a^ka^*c^+$. Note that $u/k = (w_3w_4w_5)/k = a^k$. It follows that for some $w_3'$, $w_4'$ and $w_5'$ we have $u = w_3'w_4'w_5'$ and $w_1w_2^2w_3'w_4'w_4w_5' \in L_1$. But $w_4 \in b^+$ and $w_3'w_4'w_5' \in a^+c^+$, and there are no strings containing both b's and c's in $L_1$. ∎

**Theorem 5.3.** The LR language $L_2 = \{\ a^n0b^n,\ a^n1b^{2n} \mid n \geq 1\ \}$ is not LL.

**Proof:** (Figure 16.) Assume that $L_2$ is LL($k$) and let p be the constant obtained for $L_2$ from the First Iteration Theorem. Consider the string $w = a^pa^k1b^{2(p+k)}$ in which the first p a's are distinguished. From theorem 4.4 we obtain a factorization $\varphi = (w_1,w_2,w_3,w_4,w_5)$ of w. Since $\varphi$ satisfies theorem 4.4 we must have $w_2 \in a^+$ and $w_4 \in b^+$, $2|w_2| = |w_4|$, and $w_3$ must begin with at least k a's. Now consider $a^{p+k}0b^{p+k}$, which may be written as $w_1w_2u$ for some $u \in a^ka^*0b^*$. Note that $u/k = (w_3w_4w_5)/k$. It follows from theorem 4.4 that for some $w_3'$, $w_4'$ and $w_5'$ we have $u = w_3'w_4'w_5'$, $|w_2| = |w_4'|$, and $w_1w_2^2w_3'w_4'w_4w_5' \in L_2$. Let $\#_a$ and $\#_b$ be the

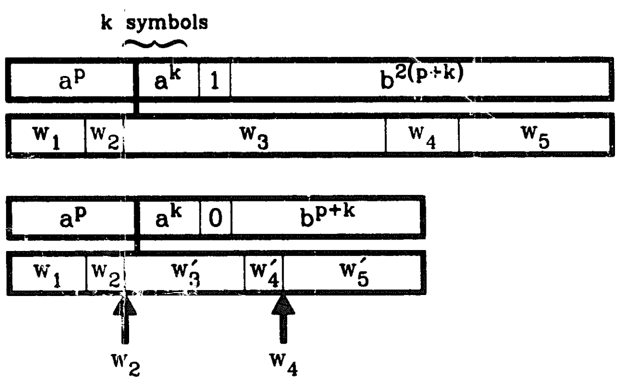



Fig. 16. An application of Theorem 4.4 to the language $a^n0b^n + a^n1b^{2n}$.

number of a′s and b′s in this string. Then $p+k+|w_2| = \#_a < p+k+2|w_2| = \#_b$, so that this string contains an illegal number of b′s (since $w_3'$ contains a 0) and cannot belong to $L_2$. ■

**Theorem 5.4.** The LR language $L_3 = \{ a^n da^n e,\ a^n fa^n g \mid n \geq 1 \}$ is not LL.

**Proof:** (Figure 17.) Assume that $L_3$ is LL($k$) and let p be the constant obtained for $L_3$ from the First Iteration Theorem. Consider the string $w = a^p a^k da^{p+k} e$ in which the first p a′s are distinguished. From theorem 4.4 we obtain a factorization $\varphi = (w_1,w_2,w_3,w_4,w_5)$ of w such that $w_2 \in a^+$, $w_4 \in a^+$ and $w_3 \in a^* da^*$. As usual we also have $(w_3w_4w_5)/k = a^k$. Now consider $a^{p+k} fa^{p+k} g$, which we may write as $w_1w_2u$ for some u. It is necessarily the case that $u/k = (w_3w_4w_5)/k$. It follows from theorem 4.4 that for some $w_3'$, $w_4'$ and $w_5'$ we have $u = w_3'w_4'w_5'$, $w_3' \in a^* fa^*$, $w_5'$ ends in g and $w_1w_2^n w_3 w_4^n w_5'$ is in $L_3$ for every $n \geq 0$. But these strings have the form $a^+ da^+ g$, and therefore cannot belong to $L_3$. ■

**Theorem 5.5.** The LR language $L_4 = \{ a^m b^{m+n} \mid m \geq 1,\ 0 \leq n \leq m \}$ is not LL.

**Proof:** Assume that $L_4$ is LL($k$) and let p be the constant obtained for $L_4$ from the First Iteration Theorem. Without loss of generality assume that $p \geq k$. Consider the string $a^p b^p$ in which the a′s are distinguished. From theorem 4.4 we obtain a factorization $\varphi = (w_1,w_2,w_3,w_4,w_5)$ of $a^p b^p$ such that $w_1w_2^n w_3 w_4^n w_5$ is in $L_4$ for every $n \geq 0$, from which it follows easily that $w_2$ must consist entirely of a′s and $w_4$ entirely of b′s. Furthermore, $|w_2| \leq |w_4|$, for otherwise we could obtain strings with more a′s than b′s for a suitably large value of n. In particular, $w_1w_3w_5$ is in $L_4$. Let $i = |w_2|$; we know that $i \geq 1$. If $w_4$ contains more than $i$ b′s then $w_1w_3w_5$ will contain more a′s than b′s, which is not allowed. Therefore $|w_2| = |w_4|$; we have $w_2 = a^i$ and $w_4 = b^i$.

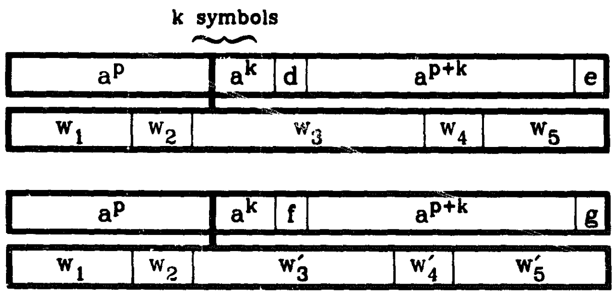



Fig. 17. An application of Theorem 4.4 to the language $a^n da^n e + a^n fa^n g$. If this language is LL then it must contain the strings $w_1w_2^n w_3 w_4^n w_5' \in a^+ da^+ g$, which it does not.

Now consider the string $a^p b^{2p}$. Since $w_1w_2 \in a^+$ and $p \geq k$ it must be the case that $a^p b^p/(|w_1w_2|+k) = a^p b^{2p}/(|w_1w_2|+k)$. Hence there is a factorization $(w_1,w_2,w_3',w_4',w_5')$ of $a^p b^{2p}$ such that $w_1w_2^n w_3' w_4'^n w_5'$ is in $L_4$ for every $n \geq 0$, so that $w_4' \in b^+$. In particular $w_1 w_3' w_5'$ belongs to $L_4$. Let $\#_a$ be the number of a's in $w_1 w_3' w_5'$. Define $\#_b$ similarly, and let $j = |w_4'|$. Since we must have $\#_b \leq 2\#_a$ we must have $(2p-j) \leq 2(p-i)$. It follows that $j \geq 2i > i$. Hence $w_4 \neq w_4'$. But $w_4 w_4' = w_4' w_4 = b^{i+j}$, which is a violation of condition (3b) of theorem 4.4. Hence $L_4$ cannot be LL. ∎

**Theorem 5.6.** The LR language $L_5 = \{ a^m b^n \mid m \geq n \geq 0 \}$ is not LL.

**Proof:** (Figure 18.) Suppose that $L_5$ is LL(k−1) for some k and let p be the constant obtained by applying the Second Iteration Theorem to $L_5$. Consider the two strings $a^{p+k}b^k$ and $a^{p+k}b^{p+k}$, and distinguish the final p b's in the latter string. According to the Second Iteration Theorem $a^{p+k}b^{p+k}$ has a factorization $(a^{p+k}w_1,w_2,w_3,w_4,w_5)$ such that

- $w_2 \neq \Lambda$
- $a^{p+k}w_1w_2^n w_3 w_4^n w_5 \in L_5$ for every $n \geq 0$

From this we can deduce that $w_2w_4 \in b^+$ so that for a sufficiently large value of n we can obtain a string with more b's than a's − a string which cannot belong to $L_5$. ∎

Note that it is possible to prove theorem 5.6 using the First Iteration Theorem and the technique applied in theorem 5.5.

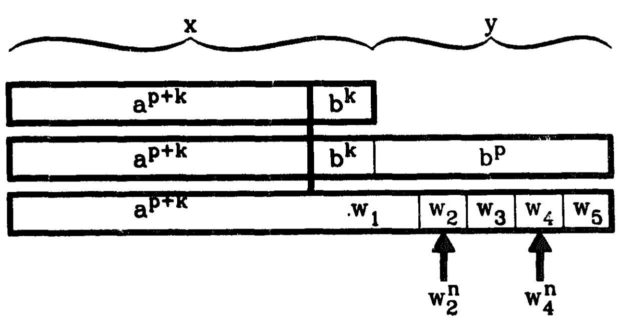



Fig. 18. An application of Theorem 4.6 to the language $a^m b^n$, $m \geq n \geq 0$. Because $a^{p+k}b^{p+k}$ is sufficiently longer than $a^{p+k}b^k$ a pumping must occur among the b's.

Using $L_5$ we easily obtain the following result.

**Theorem 5.7.** The LL languages are not closed under right quotient with a regular set.

**Proof:** It is easy to see that the language $a^n b^n$ is an LL language, and $b^*$ is obviously a regular set. However

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

is not an LL language, as we have just seen. ■

The Second Iteration Theorem is by its very nature not applicable to LL languages which are prefix-free, that is, to languages L for which $x \in L$ and $xy \in L$ imply $y = \Lambda$. Thus theorem 4.6 could not be used to prove any of theorems 5.2, 5.3 and 5.4. 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 4.4 when that is the case.

$L_1$ and $L_5$ are from Rosenkrantz and Stearns [22]. $L_2$ is taken from van Leeuwen [14]. $L_3$ is taken from Bordier and Saya [7]. $L_5$ 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 by Aho, Johnson and Ullman [1] that such constructs are not LL; applying the argument of theorem 5.6 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 [22].

**Theorem 5.8.** 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.

## Conclusions

Theorems 4.4 and 4.6 provide a powerful and reasonably general technique for establishing that languages are not LL($k$) when that is the case. Previous results of this kind ([7], [14] and [22]) have generally been based on more complicated and less satisfying *ad hoc* arguments.

We leave open the question of whether satisfying the conditions of theorem 4.4 is sufficient to ensure that a language is LL($k$), although we do not believe that to be the case. The task of characterizing a family of languages by means of an iteration theorem appears, in general, to be a difficult one. Although a number of iteration theorems have been established for several language classes, in only one case is the result known to be sufficient as well as necessary [24].

Finally, our arguments illustrate the advantages to be obtained from the careful analysis of derivation trees.

## Acknowledgements

A stronger version of theorem 4.4 is presented here than was reported in [4], and the author is indebted to Bill Ogden, who also suggested the proof of theorem 5.5, for the improvement. Theorem 4.6 was inspired by an observation of Jan van Leeuwen's [14]. The suggestions and observations of Kellogg Booth and especially Professor Michael Harrison are keenly appreciated. The author is also very grateful for Kimberly King's meticulous and invaluable refereeing.

## References

[1] A. V. Aho, S. C. Johnson and J. D. Ullman, Deterministic parsing of ambiguous grammars, *C. ACM* 18 (1975) 441–452.

[2] A. V. Aho and J. D. Ullman, *The Theory of Parsing, Translating, and Compiling*, Vols. I and II (Prentice-Hall, Englewood Cliffs, NJ, 1972 and 1973).

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

[4] J. C. Beatty, Iteration theorems for LL($k$) languages, Proceedings of the Ninth Annual Symposium on Theory of Computing, Boulder, Colorado (1977) 122–131.

[5] J. C. Beatty, Iteration theorems for the LL($k$) languages, Ph.D. Thesis, University of California, Berkeley, California (1977). Available as UCRL-52379 from the Technical Information Department, Lawrence Livermore Laboratory, Livermore, California.

[6] L. Boasson, Two iteration theorems for some families of languages, *J. Comput. System Sci.* 7 (1973) 583–596.

[7] J. Bordier and H. Saya, A necessary and sufficient condition for a power language to be LL($k$), *Computer Journal* 16 (1973) 351–356.

[8] B. M. Brosgol, Deterministic translation grammars, Ph. D. Thesis, Harvard University (1974).

[9] M. M. Geller, Compact parsers for deterministic languages, Ph. D. Thesis, University of California, Berkeley, California (1974).

[10] M. M. Geller and M. A. Harrison, On LR($k$) grammars and languages, *Theoretical Computer Science* 4 (1977) 245–276.

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

[12] K. Jensen and N. Wirth, PASCAL User Manual and Report, *Lecture notes in computer science* 18 (Springer–Verlag 1974).

[13] K. N. King, Iteration Theorems for Families of Strict Deterministic Languages, Technical Report UCB–CS–KK–78–01, University of California, Berkeley, California (1978).

[14] J. van Leeuwen, An elementary proof that a certain context–free language is not LL($k$), and a generalization, notes (1972).

[15] P. M. Lewis II and R. E. Stearns, Syntax–directed transduction, *J. ACM* 15 (1968) 465–488.

[16] P. Naur (ed.), Report on the algorithmic language ALGOL 60, *C. ACM* 3 (1960) 299–314.

[17] P. Naur (ed.), Revised report on the algorithmic language ALGOL 60, *C. ACM* 6 (1963) 1–17.

[18] A. Nijholt, A left part theorem for grammatical trees, *Discrete Mathematics* 25 (1979) 51–63.

[19] W. F. Ogden, Intercalation theorems for pushdown store and stack languages, Ph.D. Thesis, Stanford University, California (1968).

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

[21] PL/I language specifications, IBM document GY33–6003–2 (1970).

[22] D. J. Rosenkrantz and R. E. Stearns, Properties of deterministic top–down grammars, *Information and Control* 17 (1970) 226–256.

[23] E. Soisalon–Soinen, Characterization of LL($k$) languages by restricted LR($k$) grammars, Ph. D. Thesis, University of Helsinki.

[24] D. S. Wise, A strong pumping lemma for context–free languages, *Theoretical Computer Science* 3 (1976) 359–369.