INFORMATION AND CONTROL 17, 226–256 (1970)

Properties of Deterministic Top-Down Grammars

D. J. ROSENKRANTZ AND R. E. STEARNS

General Electric Company, Schenectady, New York 12301

The class of context-free grammars that can be deterministically parsed in a top down manner with a fixed amount of look-ahead is investigated. These grammars, called $LL(k)$ grammars where k is the amount of look-ahead are defined and a procedure is given for determining if a context-free grammar is $LL(k)$ for a given value of k. A procedure is given for eliminating the ϵ-rules from an $LL(k)$ grammar at the cost of increasing k by 1. There exist cases in which this increase is inevitable. A procedure is given for obtaining a deterministic push-down machine to recognize a given $LL(k)$ grammar and it is shown that the equivalence problem is decidable for $LL(k)$ grammars. Additional properties are also given.

INTRODUCTION

The class of context-free grammars that can be parsed in a top-down manner without backtrack is of interest because the parsing can be done quickly and the type of syntax directed transductions which can be performed over such grammars by a deterministic pushdown machine is fairly general (Lewis and Stearns (1968)). The object of this paper is to study these grammars.

More specifically, we study the $LL(k)$ grammars defined by Lewis and Stearns (1968). A number of decision procedures are given, including the testing of a grammar for the $LL(k)$ property and the testing of two $LL(k)$ grammars for equivalence. Methods are given for obtaining canonical forms which inherit the $LL(k)$ property. Some of the results were stated previously in Lewis and Stearns (1968) without proofs.

We represent a context-free grammar G by a four-tuple (T, N, P, S) where T is the finite terminal alphabet, N is the finite nonterminal alphabet, P is a finite set of symbols each of which represents a production that we write in the form $A \to w$ where A is in N and w in $(N \cup T)^*$, and S in N is the starting symbol.

For γ_1 and γ_2 in $(N \cup T)^*$, we write $\gamma_1 \to \gamma_2$ if and only if there exit φ_1 and φ_2 in $(N \cup T)^*$ and production $A \to \gamma$ in P such that $\gamma_1 = \varphi_1 A \varphi_2$ and $\gamma_2 = \varphi_1 \gamma \varphi_2$. We write $\gamma_1 \to_L \gamma_2$ if in addition φ_1 is in T^*. We let "$\Rightarrow$" represent the reflexive transitive completion of "$\to$" and "$\Rightarrow_L$" the reflexive transitive completion of "$\to_L$". Intuitively $\gamma_1 \Rightarrow \gamma_2$ means that γ_2 can be derived from γ_1 using productions in P and $\gamma_1 \Rightarrow_L \gamma_2$ means that γ_2 can be obtained from a left derivation.

For any γ in $(N \cup T)^*$, we let $L(\gamma) = \{w \text{ in } T^* \mid \gamma \Rightarrow w\}$. The language generated by G is $L(S)$. This language will sometimes be written as $L(G)$. If production p is $A \to \gamma$, we write $L_p(A) = L(\gamma)$.

For a given word w and nonnegative integer k, we define w/k to be w if the length of w is less than or equal to k and we define w/k to be the string consisting of the first k symbols of w if w has more than k symbols.

If R is a set of words, let

$$R/k = \{w/k \mid w \text{ in } R\}.$$

If A is a nonterminal, w is a word in $(N \cup T)^*$ and p is the name of a production in P, we write

$$A \Rightarrow w \quad (p)$$

if and only if w can be derived from A after first applying production p.

DEFINITION 1. Let k be >0. A grammar $G = (T, N, P, S)$ is said to be an $LL(k)$ grammar if and only if given:

1. a word w in T^*/k;
2. a nonterminal A in N;
3. a word w_1 in T^*;

there is at most one production p in P such that for some w_2 and w_3 in T^*,

4. $S \Rightarrow w_1 A w_3$;
5. $A \Rightarrow w_2 \ (p)$;
6. $(w_2 w_3)/k = w$.

Stated informally in terms of parsing, an $LL(k)$ grammar is a context-free grammar such that for any word in its language, each production in its derivation can be identified with certainty by inspecting the word from its beginning (left end) to the k-th symbol beyond the beginning of the production. Thus when a nonterminal is to be expanded during a top down parse, the portion of the input string which has been processed so far plus the next k input symbols determine which production must be used for the nonterminal. Thus the parse can proceed without backtrack. Conversely, w_1, A, and w constitute the only information available at that point in a left-to-right top-down parse.

Any context-free language that has a $LL(k)$ grammar can be recognized (top-down) by a (deterministic) push-down machine (Lewis and Stearns, 1968). The machine uses a predictive recognition scheme (Oettinger, 1961) in a manner that "uses" the grammar in the recognition process and can be said to "recognize" each production.

TEST FOR $LL(k)$

In this section, we give a construction which is basic to our $LL(k)$ test and then we give the test. In what follows, we use the standard mathematical notation $2T^{*/k}$ to represent the set of all subsets of T^*/k and $\times$ for the cross product of two sets. We use the term *structurally equivalent* as in Paull and Unger (1968) to mean that two grammars generate the same strings and the same trees (with the intermediate nodes unlabeled) for these strings. We use ϵ to represent the null string.

CONSTRUCTION 1. Given a grammar $G = (T, N, P, S)$ we begin the construction of a grammar $G' = (T, N', P', S')$ by letting

$$T'' = T \times 2T^{*/k},$$
$$N'' = N \times 2T^{*/k},$$
$$S'' = (S, \{\epsilon\}),$$
$$P'' = P \times 2T^{*/k},$$

where symbol (p, R) represents the production

$$(A, R) \rightarrow (A_n, R_n) \cdots (A_1, R_1)$$

where $A \rightarrow A_n \cdots A_1$ is the production p and R_{i+1} satisfies the condition

$$R_{i+1} = (L(A_i \cdots A_1)R)/k$$

for all $n > i \geqslant 0$. If $n = 0$, the right sides of the productions are understood to represent ϵ. Similarly, the condition for R_1 is understood to be:

$$R_1 = (L(\epsilon)R)/k = R.$$

This gives us a grammar $G'' = (T'', N'', P'', S'')$.

Remove all the symbols from N'' and all productions from P'' which cannot be used in deriving a terminal string from S''. Finally, replace each occurrance of terminal (a, R) by terminal a. Letting N' be the new nonterminal set, P' the new production set, and S' be the starting symbol S'', we obtain $G' = (T, N', P', S')$.

Two lemmas are now given which clarify the relation between G and G'.

LEMMA 1. *The G' of Construction* 1 *is a grammar structurally equivalent to the original grammar G.*

Proof. Given a derivation in G', a corresponding derivation in G is obtained by replacing each nonterminal (A, R) by A.

Given a derivation from S in G, a corresponding derivation from S'' in G'' (where G'' is defined in the construction) is obtained if, instead of applying production p to an instance of A, one applies (p, R) to the corresponding (A, R). Since all nonterminals used must, by their very use, be in N' and all productions used must be in P' (after replacing each (t, R) for t in T by t) we obtain a corresponding derivation in G'.

COROLLARY 1. $L_{(p,R)}((A, R)) = L_p(A)$ *for* (A, R) *in* N'.

Proof. As with S' and S, derivations from (A, R) and A can be made to correspond.

LEMMA 2. *Given G and G' as defined in Construction* 1, *then for all* (A, R) *in N' and φ and γ in* $(N' \cup T)^*$,

$$S' \underset{L}{\Rightarrow} \varphi(A, R)\gamma \qquad \textit{implies} \quad R = L(\gamma)/k.$$

Proof. We will prove the result by induction on the length of a leftmost derivation of an intermediate string. It certainly holds for the zero length derivation since the initial string is $(S, \{\epsilon\})$ and $\{\epsilon\} = L(\epsilon)/k$. Now suppose that it is true for all leftmost derivations of length $\leqslant n$. Consider a leftmost derivation of length $n + 1$.

$$S \underset{L}{\Rightarrow} w(A, R)\,\gamma_1 \underset{L}{\rightarrow} w(A_n\,, R_n) \cdots (A_1\,, R_1)\,\gamma_1\,.$$

The lemma certainly holds for occurrances of nonterminals in γ_1 since they were generated in a derivation of length $\leqslant n$. It also holds for nonterminals in w since there are none. Thus it remains to be shown that it holds for the new nonterminals $(A_i\,, R_i)$. From the induction hypothesis $R = L(\gamma_1)/k$.

Therefore, from Construction 1

$$\begin{aligned} R_{i+1} &= (L(A_i \cdots A_1)R)/k = (L(A_i \cdots A_1)(L(\gamma_1)/k))/k \\ &= L(A_i \cdots A_1\gamma_1)/k. \end{aligned}$$

Thus the lemma is true by induction.

COROLLARY 2. *The number* $|N'|$ *is bounded by* $|N|$ *times the number of sets of the form* $L(\gamma)/k$ *for* γ *in* $(N \cup T)^*$.

Although the intermediate set N'' in the construction has at least $|N| \cdot 2^{|T|^k}$ elements, the corollary indicates that the set N' may be much smaller. If for example G contained no ϵ productions, N' could not have more than $|N| \cdot |N \cup T \cup \{\epsilon\}|^k$ elements since the first k elements of string γ would determine R. Thus, a much more practical approach to deriving G' is to generate it directly from G without contructing all of G''.

We are now in a position to state the $LL(k)$ test.

Test. Given a grammar $G = (T, N, P, S)$ and given an integer k, construct the grammar G' of Construction 1. Then for each w in T^*/k and (A, R) in N', test to see if there is at most one p in P such that

$$w \text{ is in } (L_p(A)R)/k.$$

This last expression is equal to $((L_p(A)/k)R)/k$ which is computable since $L_p(A)/k$ is computable (for instance by rewriting the grammar so that all productions begin with a terminal symbol (Greibach 1965) and then trying all derivations of length k or less). If all w and (A, R) pass the test, then the grammar is $LL(k)$; otherwise it is not.

To prove that this test works, we need a lemma which characterizes $LL(k)$ grammars in terms of the entities computed in the test.

LEMMA 3. *A grammar* (T, N, P, S) *is* $LL(k)$ *if and only if for all* A *in* N, w *in* T^*/k, *and* $R \subseteq T^*/k$, *there exists at most one production* p *in* P *such that for some* w_1 *in* T^* *and* γ *in* $(N \cup T)^*$, *the following three conditions hold*:

$$S \underset{L}{\Rightarrow} w_1 A \gamma, \tag{1}$$

$$R = L(\gamma)/k, \tag{2}$$

$$w \textit{ is in } \quad (L_p(A)\, L(\gamma))/k. \tag{3}$$

Proof. Suppose that given A, w, and R, there are two productions p and p' satisfying (1), (2), and (3). Let w_1 in T^* and γ in $(N \cup T)^*$ satisfy conditions (1), (2), and (3) for production p and let w_1' in T^* and γ' in $(N \cup T)^*$ satisfy

$$S \underset{L}{\Rightarrow} w_1'A\gamma', \tag{1'}$$

$$R = L(\gamma')/k, \tag{2'}$$

$$w \text{ is in } (L_{p'}(A)\, L(\gamma'))/k. \tag{3'}$$

Because of (3), there exist w_2 and w_3 in T^* such that

$$A \Rightarrow w_2 \;\; (p), \tag{4}$$

$$\gamma \Rightarrow w_3 \;\; (S \Rightarrow w_1Aw_3), \tag{5}$$

$$w_2w_3/k = w. \tag{6}$$

Because of (3′), there exist w_2' and x in T^* such that

$$A \Rightarrow w_2' \;\; (p'), \tag{4'}$$

$$\gamma' \Rightarrow x, \tag{5'}$$

$$w_2'x/k = w. \tag{6'}$$

Since $L(\gamma)/k = L(\gamma')/k$ by (2) and (2′), x/k is in $L(\gamma)/k$ and must be the prefix of a string in $L(\gamma)$. Thus, there exists a w_3' in T^* such that $w_3'/k = x/k$ and

$$\gamma \Rightarrow w_3' \;(S \Rightarrow w_1Aw_3'). \tag{7}$$

But $w_3'/k = x/k$ and (6′) implies

$$w_2'w_2'/k = w_2'x/k = w. \tag{8}$$

Comparing relations (4), (5), (6), (4′), (7), and (8) with conditions 4 through 6 of Definition 1, we see that the $LL(k)$ condition is violated.

Suppose the $LL(k)$ condition is violated for some w in T^*/k, A in N, and w_1 in T^*. This means there exist distinct p and p' in P and w_2 , w_2', w_3 , w_3' in T^* such that

$$S \Rightarrow w_1Aw_3 \qquad \text{and} \qquad S \Rightarrow w_1Aw_3', \tag{9}$$

$$A \Rightarrow w_2 \;(p) \qquad \text{and} \qquad A \Rightarrow w_2' \;(p'), \tag{10}$$

$$(w_2w_3)/k = w = (w_2'w_3')/k. \tag{11}$$

Relations (9) imply that there exist γ and γ' such that

$$S \underset{L}{\Rightarrow} w_1 A\gamma \qquad \text{and} \qquad S \underset{L}{\Rightarrow} w_1 A\gamma' \tag{12}$$

$$\gamma \Rightarrow w_3 \qquad \text{and} \qquad \gamma' \Rightarrow w_3'. \tag{13}$$

Because of (12), there exist left-derivation sequences

$$S = \psi_0 \underset{L}{\rightarrow} \cdots \underset{L}{\rightarrow} \psi_n = w_1 A\gamma$$

and

$$S = \psi_0' \underset{L}{\rightarrow} \cdots \underset{L}{\rightarrow} \psi_m' = w_1 A\gamma'.$$

At least one of the following three possibilities must occur.

Case 1. There is an i, $0 \leqslant i \leqslant m$ such that $\psi_i' = \psi_n$.

Case 2. There is an i, $0 \leqslant i \leqslant n$ such that $\psi_i = \psi_m'$.

Case 3. There is an i, $0 < i \leqslant \min(n, m)$ such that

$$\psi_j = \psi_j' \qquad \text{for} \qquad 0 \leqslant j < i \qquad \text{and} \qquad \psi_i \neq \psi_i'.$$

We will show that in each case the condition of the lemma is violated.

Case 1 has three subcases, depending on what part of $A\gamma'$ is generated from γ.

Case 1A: $\gamma = \gamma'$. In this case, conditions (1), (2), and (3) of the lemma are clearly violated.

Case 1B: $A \Rightarrow \epsilon$ and $\gamma \Rightarrow_L A\gamma'$. In this case $\gamma \Rightarrow \gamma'$; therefore, w is in $(L_{p'}(A)\, L(\gamma))/k$ since $L(\gamma) \supset L(\gamma')$ and so conditions (1), (2), and (3) are immediately violated for $w_1 A\gamma$, $R = L(\gamma)$ and w because (3) is then satisfied for both p and p'.

Case 1C: $A \Rightarrow_L A\gamma_1$, $\gamma_1\gamma = \gamma'$ and $\gamma_1 \neq \epsilon$. If $L(\gamma_1) = \{\epsilon\}$, then $L(\gamma) = L(\gamma')$ and the argument of Case 1B applies. We cannot have $L(\gamma_1)$ empty because $\gamma_1\gamma \Rightarrow w_3'$. Therefore, there is a nonnull x in $L(\gamma_1)$.

Letting $\psi = \gamma_1^k\gamma$, we know from $A \Rightarrow_L A\gamma_1$ and from (12) that

$$S \underset{L}{\Rightarrow} w_1 A\psi \tag{14}$$

and we can define R by

$$R = L(\psi)/k. \tag{15}$$

Let p'' be a production such that $A \Rightarrow A\gamma_1(p'')$. Now it is evident that

$$(w_2 x^k)/k \in (L_p(A)\, L(\psi))/k \cap (L_{p''}(A)\, L(\psi))/k \tag{16}$$

and

$$(w_2' x^k)/k \in (L_{p'}(A)\, L(\psi))/k \cap (L_{p''}(A)\, L(\psi))/k. \tag{17}$$

But (14) and (15) together with (16) or (17) contradicts the conditions of the lemma since p'' cannot be equal to both p and p'.

Case 2 follows from Case 1 by symmetry.

To prove Case 3, let x in T^*, B in N, and ψ in $(N \cup T)^*$ be such that

$$S \underset{L}{\overset{*}{\Rightarrow}} \psi_{i-1} = \psi'_{i-1} = xB\psi. \tag{18}$$

Because $xB\psi$ is a step in a left derivation of $w_1 A\gamma$, there must be a y in T^* such that $xy = w_1$. Because $\psi_i \neq \psi_i'$, there are z_2, z_2', z_3, and z_3' in T^* and distinct productions q and q' such that

$$B \Rightarrow z_2 \ (q) \qquad \text{and} \qquad B \Rightarrow z_2' \ (q'),$$
$$z_3 \text{ and } z_3' \text{ are in } L(\psi),$$
$$z_2 z_3 = y w_2 w_3 \qquad \text{and} \qquad z_2' z_3' = y w_2' w_3'.$$

Let w' be defined by

$$w' = (yw)/k = (z_2 z_3)/k = (z_2' z_3')/k.$$

Clearly

$$w' \in (L_q(B)\, L(\psi))/k \cap (L_{q'}(B)\, L(\psi))/k. \tag{19}$$

Letting

$$R = L(\psi)/k \tag{20}$$

we have from (18), (19), and (20) a violation of the conditions of the lemma because q and q' are distinct. Thus the final case also leads to a contradiction and the lemma is proved.

The significance of Lemma 3 is that the choice of p can be obtained from a finite amount of information, namely A and $L(\gamma)/k$. The construction has given us a method of computing the $L(\gamma)/k$ as we go along. We are now ready to verify the test.

THEOREM 1. *Given a context-free grammar $G = (T, N, P, S)$ and given an integer k, one can decide if the grammar is $LL(k)$.*

Proof. We show that the test given earlier in this section works.

By Lemma 2, the nonterminals (A, R) of N' represent all the possible A in N and R in T^*/k such that $R = L(\gamma)/k$ and $S \Rightarrow_L w_1 A\gamma$ for some w_1 in T^* and γ in $(N \cup T)^*$. The test is therefore a test of whether the condition of Lemma 3 holds and is therefore an $LL(k)$ test.

We derive one more consequence of Lemma 3 for later use.

LEMMA 4. *An $LL(k)$ grammar is unambiguous.*

Proof. In order that grammar (T, N, P, S) be ambiguous, there must be w_1, w_2, and w_3 in T^*, A in N, γ in $(N \cup T)^*$, and distinct p and p' in P such that

$$S \underset{L}{\Rightarrow} w_1 A\gamma,$$

$$A \Rightarrow w_2 \ (p) \qquad \text{and} \qquad A \Rightarrow w_2 \ (p'),$$

$$\gamma \Rightarrow w_3 .$$

But letting $R = L(\gamma)/k$ and $w = (w_2 w_3)/k$, Lemma 3 tells us that the grammar is not $LL(k)$.

STRONG $LL(k)$ GRAMMARS

In this section we define the concept of a strong $LL(k)$ grammar. In terms of generative power, these grammars will be shown to be structurally equivalent to $LL(k)$ grammars. We consider these strong grammars more as a normal form rather than as a class for separate study.

For grammar $G = (T, N, P, S)$ and nonterminal A in N, let

$$R(A) = \{w \text{ in } T^* \mid S \Rightarrow w_1 A w \text{ for some } w_1 \text{ in } T^*\}.$$

For any positive integer k, let

$$R_k(A) = R(A)/k.$$

Now $R(A)$ is itself a context-free language with a grammar easily obtained from G. The set $R_k(A)$ is then certainly computable.

For fixed k, we wish to consider grammars which satisfy the property that for any A in N and w in T^*, there is at most one p such that $(L_p(A)\ R_k(A))/k$ contains w. We call these strong $LL(k)$ grammars. Formulating this concept without reference to R_k, we get the following:

DEFINITION 2. A grammar $G = (T, N, P, S)$ is said to be a *strong* $LL(k)$ grammar for some positive integer k if and only if given:

1. a word w in T^*/k;
2. a nonterminal A in N;

there is at most one production p in P such that for some w_1, w_2 and w_3 in T^*,

3. $S \Rightarrow w_1 A w_3$;
4. $A \Rightarrow w_2\ (p)$;
5. $(w_2 w_3)/k = w$.

The only difference between this definition and that of an $LL(k)$ grammar is the quantifier "for all w_1" has been moved within the scope of the "there exist at most one p." Thus, strong $LL(k)$ grammars are a special case of $LL(k)$. Intuitively, they are grammars where one can parse correctly knowing only that one is looking for a given nonterminal and knowing the next k inputs. The power of these grammars is expressed by the following:

THEOREM 2. *Given an $LL(k)$ grammar $G = (T, N, P, S)$, one can find a structurally equivalent strong $LL(k)$ grammar using Construction 1 of the previous section.*

Proof. We already know that the construction gives a structurally equivalent grammar (Lemma 1). To prove that it is strong, we first want to show that $R_k((A, R)) = R$ where (A, R) is a nonterminal in the derived grammar (T, N', P', S') expressed in the notation of the construction. Set $R_k((A, R))$ is the union of all $L(\psi)/k$ for all x in T^* and ψ in $(N' \cup T)^*$ such that $S' \Rightarrow_L x(A, R)\psi$. But by Lemma 2, these $L(\psi)/k$ are all equal to R and hence $R_k((A, R)) = R$.

Now consider an (A, R) in N' and w in T^*/k. If $S' \Rightarrow w_1(A, R)w_3$ for w_1 and w_3 in T^*, we know that

$$S' \underset{L}{\Rightarrow} w_1(A, R)\,\gamma'$$

for some γ' in $(N' \cup T^*)$. By Lemma 2, we know that

$$L(\gamma')/k = R = R_k((A, R)).$$

Letting γ be the string in $N \cup T$ corresponding to γ',

$$S \underset{L}{\Rightarrow} w_1 A \gamma$$

and

$$R = L(\gamma)/k.$$

Since G is an $LL(k)$ grammar, from Lemma 3 there is at most one production p such that w is in $(L_p(A)R)/k$. But there is a one-to-one correspondence between the productions for A in P and the productions for (A, R) in P'. Therefore, from Corollary 1, there is at most one production, $p' = (p, R)$, such that

$$w \text{ is in } (L_{p'}((A, R))\, R_k((A, R)))/k,$$

which we observed in an equivalent statement of the strong $LL(k)$ property. Thus, the theorem is proved.

Role of ϵ-Rules

A production is called an ϵ-rule if its right hand side is ϵ.

THEOREM 3. *Given an $LL(k)$ grammar $G = (T, N, P, S)$, an $LL(k + 1)$ grammar without ϵ-rules can be constructed which generates the language $L(G) - \{\epsilon\}$.*

Proof. We will obtain the desired grammar by rewriting G in two stages. For a given grammar $G' = (T', N', P', S')$, we will call an element A of $N' \cup T'$ *nullable* if $L(A) \supseteq \{\epsilon\}$ and call A *nonnullable* otherwise. In particular, this means that terminals are nonnullable. The first step is to rewrite G so that the first symbol on the right side of a non ϵ-rule is nonnullable. This will be done in such a way as to preserve the $LL(k)$ property. The new grammar will be obtained from the old by the advance substitution of ϵ-derivations into the various strings of leading nullable symbols that occur on the right side of productions in P.

This preserves the $LL(k)$ property because the look-ahead of k determines precisely which initial ϵ-derivations should be applied. Readers who are not interested in the details of this step may skip ahead to the description of the second step.

For each nullable symbol A of G, we will add a new nonterminal symbol A' to the nonterminal set; A' will have the property that $L(A') = L(A) - \{\epsilon\}$. Letting $G_1 = (T, N_1, P_1, S_1)$ be the grammar we are trying to construct, our new nonterminal set is described symbolically as

$$N_1 = N \cup \{A' \mid A \text{ is nullable in } G\}.$$

Each production of P can be expressed in the form:

$$A \to A_1 \cdots A_n B_1 \cdots B_m ,$$

where n and m are nonnegative integers ($n = 0$ is interpreted to mean $A_1 \cdots A_n = \epsilon$ and $m = 0$ to mean $B_1 \cdots B_m = \epsilon$) where the A_i for $1 \leqslant i \leqslant n$ are all nullable and B_1 is nonnullable in the case $m > 0$. For each such production we let P_1 contain the productions:

$$\begin{aligned} &A \to A_1' A_2 \cdots A_n B_1 \cdots B_m , \\ &A \to A_2' \cdots A_n B_1 \cdots B_m , \\ &A \to A_n' B_1 \cdots B_m , \\ &A \to B_1 \cdots B_m . \end{aligned}$$

Furthermore, if A is nullable, we let P_1 contain these same productions with A' on the left side instead of A. If, however, $m = 0$, the production $A' \to B_1 \cdots B_m$ (i.e., $A' \to \epsilon$) is omitted. The starting symbol S_1 is taken to be S if S is nonnullable and to be S' if S is nullable.

Each production in P_1 corresponds to a derivation in G. For example, $A \to A_2 \cdots A_n B_1 \cdots B_m$ corresponds to the production $A \to A_1 \cdots A_n B_1 \cdots B_m$ followed by the derivation of $A_1 = \epsilon$. Thus, a derivation in G_1 certainly corresponds to a derivation in G. Conversely, given a derivation tree in G, a derivation for G_1 is obtained by successively deleting all left most branches which result in ϵ and replacing leftmost occurrences of other nullable nonterminals by their nonnullable counterparts.

To verify that G_1 is $LL(k)$, suppose that we are given w_1 in T^*, A_0 in N_1, and w in T^*/k. A_0 is either the nullable or nonnullable version of some nonterminal A of N (i.e., $A_0 = A$ or A'). Now suppose that there are two productions of G_1 satisfying conditions 4, 5, and 6 of Definition 1. Because G is an $LL(k)$ grammar it has at most one production, $A \to A_1 \cdots A_n B_1 \cdots B_m$ satisfying conditions 4, 5, and 6, and the two productions of G_1 must have both been obtained from this production of G. The two productions can be written as

$$A_0 \to A_i' \cdots A_j \cdots A_n B_1 \cdots B_m$$

and

$$A_0 \to A_j' \cdots A_n B_1 \cdots B_m ,$$

where $1 \leqslant i < j \leqslant n + 1$ and $j = n + 1$ means that the second production is $A_0 \to B_1 \cdots B_m$.

Therefore in G_1

$$S \Rightarrow w_1 A_0 w_3\,, \qquad S \Rightarrow w_1 A_0 w_3',$$
$$A_0 \rightarrow A_i' \cdots A_j \cdots A_n B_1 \cdots B_m \Rightarrow w_2\,, \qquad A_0 \rightarrow A_j' \cdots A_n B_1 \cdots B_m \Rightarrow w_2',$$
$$(w_2 w_3)/k = w, \qquad (w_2' w_3')/k = w.$$

Letting x be the portion of w_2 derived from A_i', we note that x is nonnull since A' is a nonnullable symbol. Let $w_2 = xw_4$. Then in G

$$S \Rightarrow w_1 A w_3 \Rightarrow w_1 A_i \cdots A_n B_1 \cdots B_m w_3 \Rightarrow w_1 A_i w_4 w_3\,,$$
$$S \Rightarrow w A w_3' \Rightarrow w_1 A_i \cdots A_n B_1 \cdots B_m w_3 \Rightarrow w_1 A_i w_2' w_3',$$
$$A_i \Rightarrow x, \qquad A_i \Rightarrow \epsilon.$$

The leftmost derivations of x and ϵ from A_i must diverge. At the point just before the divergence, the intermediate string must begin with a nonterminal, C, since the intermediate string must be able to generate ϵ. Let this string be $C\nu$. Therefore at the point of divergence, there exist productions p and p' and some y and z satisfying $x = yz$ such that the following hold: $A_i \Rightarrow C\nu$, $C \Rightarrow y\ (p)$, $\nu \Rightarrow z$, $C \Rightarrow \epsilon\ (p')$, and $\nu \Rightarrow \epsilon$. Therefore, letting $w_5 = zw_4w_3$ and $w_5' = w_2'w_3'$ we have

$$S \Rightarrow w_1 C w_5\,, \qquad S \Rightarrow w_1 C w_5',$$
$$C \Rightarrow y\ (p), \qquad C \Rightarrow \epsilon\ (p'),$$
$$(yw_5)/k = w, \qquad (\epsilon w_5')/k = w.$$

This is a contradiction to the assumption that G is an $LL(k)$ grammar. Therefore, G_1 is an $LL(k)$ grammar.

We will now give a procedure for converting G_1 into an equivalent grammar G_2 without ϵ-rules. We will assume that each nonterminal of G_1 generates a nonnull terminal string. A nonterminal A which does not have this property is easily removed by deletion (if $L(A)$ is empty) or by substitution (if $L(A) = \{\epsilon\}$) without affecting the $LL(k)$ property. Let V_ϵ be the nullable symbols of G_1 and let V_1 be the nonnullable symbols. Let $V = V_1 V_\epsilon^*$. Any word γ in $V_1(N_1 \cup T)^*$ has a unique representation as a word in V^+ and we let $\alpha(\gamma)$ represent this word. For example, letting A represent symbols of V_ϵ and B represent symbols of V_1,

$$\alpha(B_1 B_2 A_3 A_4 B_5 A_6 B_7) = [B_1][B_2 A_3 A_4][B_5 A_6][B_7],$$

where the square brackets limit the words of V. Thus, the sequence of nullable nonterminals that can be generated in a leftmost derivation by G_1

are combined with the preceding nonnullable symbol. Elements of V that are strings of length one are considered to be elements of V_1 i.e., $[A] = A$ for A in V.

The overall plan of the construction is to have a left derivation $S_1 \Rightarrow_L \gamma$ in G_1 correspond to a left derivation $[S_1] \Rightarrow_L \alpha(\gamma)$ in G_2. Steps in the G_1 derivation which involve an ϵ-rule will be combined into a non-ϵ-step in order to avoid ϵ-rules for G_2. This approach involves a small discrepency in timing as a derivation such as

$$S_1 \underset{L}{\overset{*}{\Rightarrow}} b_1 b_2 A_1 B_2$$

(where b_1 and b_2 are in T, A_1 is in V_ϵ, and B_2 is in V_1) represents the situation after 2 (i.e., b_1 and b_2) plus the look-ahead inputs have been considered and

$$[S_1] \underset{L}{\overset{*}{\Rightarrow}} [b_1][b_2 A_1][B_2]$$

represents the situation where only 1 (i.e., $[b_1]$) plus the look-ahead inputs have been considered. Thus, to get the same information, the look-ahead for processing G_2 will need to be one larger than the look-ahead for processing G_1. In other words, when a decision as to which prodcution for $[b_2 A_1]$ should be used, the b_2 plus the next k input symbols may be needed to determine whether or not A_1 will be expanded into ϵ.

We now give the construction in more detail. Let V' be the set of elements of V which occur in some word $\alpha(\gamma)$ for some γ in $(N_1 \cup T)^*$ such that $S_1 \Rightarrow_L \gamma$. Any element in V' of the form $[BA_1 \cdots A_n]$ must have distinct A_i for otherwise G_1 would be ambiguous. (If A_i were repeated, there would be two derivations of $A_1 \cdots A_n \Rightarrow w_0$ where w_0 is a nonnull element of $L(A_i)$). We let T be the terminal set of G_2 and let $N_2 = V' - T$ be the nonterminal set. The starting symbol will be S_1 (sometimes written $[S_i]$). Finally, let the production set P_2 for G_2 be the set of productions determined by the following three rules:

Rule 1. If B in N_1 and γ in $V_\epsilon{}^*$ are such that $[B\gamma]$ is in V' and if $B \rightarrow \gamma_1$ is a production of P_1, then P_2 has the production:

$$[B\gamma] \rightarrow \alpha(\gamma_1\gamma).$$

Rule 2. If b in T, A in V_ϵ, and γ_1 and γ_2 in $V_\epsilon{}^*$ are such that $[b\gamma_1 A\gamma_2]$ is in V' and if $A \rightarrow \gamma$ is a non-ϵ-rule of P_1, then P_2 has the production

$$[b\gamma_1 A\gamma_2] \rightarrow [b]\ \alpha(\gamma\gamma_2).$$

Rule 3. If b in T and γ in V_ϵ^+ are such that $[b\gamma]$ is in V', then P_1 has production

$$[b\gamma] \to [b].$$

A production obtained from Rule 1 is used in G_2 whenever the corresponding rule is used in P_1. A production obtained from Rule 2 is used when $\gamma_1 \Rightarrow \epsilon$ followed by $A \to \gamma$ is applied. In this manner, left derivations in G_1 and G_2 are made to correspond to each other and the equivalence of G_1 and G_2 obtained.

To see that G_2 is $LL(k+1)$, assume that we are given w_1 in T^*, w in $T^*/k+1$, and a nonterminal of G_2. If the nonterminal is of the form $[B\gamma]$ where B is in N_1, then the only production of P_2 that satisfies conditions 4, 5, and 6 of Definition 1 for G_2 is the one obtained via Rule 1 from the production of P_1 that satisfies conditions 4, 5, and 6 of Definition 1 for w, B, and w_1. If the nonterminal has the form $[bA_1 \cdots A_n]$ as in Rules 2 and 3, then w must have the form bw_2 for w_2 in T^*/k. If it does not have this form, no rule can be applied. If w does have this form, then w_1b and w_2 determine which of the leading A_i must be eliminated with ϵ-derivations and (if all A_i are not so eliminated) which non-ϵ-rule to apply to the next A_i. For assume that there are two productions in G_2 satisfying conditions 4, 5, and 6 of Definition 1 for $k+1$. The two productions can be written as

$$[bA_1 \cdots A_n] \to b\alpha(\gamma_i \cdots A_j \cdots A_n)$$

and

$$[bA_1 \cdots A_n] \to b\alpha(\gamma_j \cdots A_n),$$

where $1 \leqslant i < j \leqslant n+1$, $j = n+1$ means that the second production is $[bA_1 \cdots A_n] \to b$, $A_i \to \gamma_i$ is a production of G_1, and if $j \neq n+1$, $A_j \to \gamma_j$ is a production of G_1. Therefore, in G_2 $S \Rightarrow w_1[bA_1 \cdots A_n]w_3$, $S \Rightarrow w_1[bA_1 \cdots A_n]w_3'$, $[bA_1 \cdots A_n] \to b\alpha(\gamma_i \cdots A_j \cdots A_n) \Rightarrow bw_2$, $[bA_1 \cdots A_n] \to b\alpha(\gamma_j \cdots A_n) \Rightarrow bw_2'$, and

$$(bw_2w_3)/(k+1) = (bw_2'w_3')/(k+1).$$

Then in G_1

$$S \Rightarrow w_1bA_i \cdots A_nw_3$$

and

$$A_i \cdots A_n \underset{L}{\to} \gamma_i \cdots A_n \Rightarrow w_2 .$$

Let x be the portion of w_2 generated from γ_i and let y be the remainder.

Furthermore, in G_1:

$$S \Rightarrow w_1 b A_i \cdots A_n w_3',$$
$$A_i \Rightarrow \epsilon,$$
$$A_{i+1} \cdots A_n \Rightarrow w_2'.$$

Let p be the production $A \to \gamma_i$ and p' be the first production used in the derivation $A_i \Rightarrow \epsilon$. Since γ_i begins with a nonnullable symbol, $p \neq p'$. Now in G_1

$$S \Rightarrow w_1 b A_i y w_3, \qquad S \Rightarrow w_1 b A_i w_2' w_3',$$
$$A_i \Rightarrow x \ (p), \qquad A_i \Rightarrow \epsilon \ (p'),$$
$$x y w_3/k = w_2' w_3'/k.$$

But since G_1 is $LL(k)$, this is a contradiction, and G_2 is $LL(k+1)$, thereby proving the theorem.

A nonterminal symbol, A, is said to be *left recursive* if and only if $L(A)$ is non empty and there is a nontrivial (trivial meaning zero length) derivation of the relation $A \Rightarrow Aw$ for some w in T^*.

LEMMA 5. *An $LL(k)$ grammar G can have no left recursive nonterminals.*

Proof. Assume that an $LL(k)$ grammar has a left recursive symbol. Then for some nonterminal A, $A \Rightarrow Ay$ (p) and $A \Rightarrow x$ (p') where x and y are in T^*, and p and p' are different productions. Because G is unambiguous, $y \neq \epsilon$. Furthermore, $S \Rightarrow uAv$ for some u and v. Now consider the derivations

$$S \Rightarrow uAv \Rightarrow uAy^k v \Rightarrow uxy^k v$$

and

$$S \Rightarrow uAv \Rightarrow uAy^k v \Rightarrow uAy^{k+1} v \Rightarrow uxy^{k+1} v.$$

Thus

$$S \Rightarrow uAy^k v, \qquad A \Rightarrow xy \ (p), \qquad A \Rightarrow x \ (p'),$$

and

$$(xy^{k+1}v)/k = (xy^k v)/k.$$

Therefore, since the grammar is $LL(k)$, $p = p'$, and there cannot be a left recursive nonterminal.

A grammar is said to be in *Greibach normal form* (Greibach (1965)) if the right side of every production begins with a terminal symbol.

THEOREM 4. *Given an $LL(k)$ grammar without ϵ-rules, another $LL(k)$ grammar in Greibach normal form can be obtained for the same language.*

Proof. For nonterminals A and B let $>$ be the transitive relation defined by $A > B$ if $A \Rightarrow B\varphi$ for some φ. From Lemma 5 it cannot be true that $A > A$; therefore, the nonterminals can be arranged in a linear order $A_1, \ldots, A_n$ such that for $i \leqslant j$, it is not true that $A_i > A_j$. The grammar can now be rewritten in n steps. In the i-th step, each occurrence of A_i as the first symbol on the right side of a production is replaced by all the productions for A_i (each of which begins with a terminal symbol).

In order to see that the new grammar is $LL(k)$, assume for purposes of induction that the grammar before the ith step is $LL(k)$. Assume that for the grammar after the ith step there is a w, B, w_1, satisfying conditions 1, 2, and 3 of Definition 1 and productions p and p' such that

$$S \Rightarrow w_1Bw_3, \qquad S \Rightarrow w_1Bw_3',$$
$$B \Rightarrow w_2 \ (p), \qquad B \Rightarrow w_2' \ (p'),$$
$$(w_2w_3)/k = (w_2'w_3')/k = w.$$

If p and p' were both productions of the previous grammar, they would violate its $LL(k)$ property. If p was obtained from the production $B \rightarrow A_i\nu_1$ and p' was a previous production, then $B \rightarrow A_i\nu_1$ and p' would have violated the $LL(k)$ property. If p was obtained from $B \rightarrow A_i\nu_1$ and p' from $B \rightarrow A_i\nu_2$, then these two productions would have violated the $LL(k)$ property. If p was obtained from $B \rightarrow A_i\nu_1$ and $A_i \rightarrow \psi_1$; p' from $B \rightarrow A_i\nu_1$ and $A_i \rightarrow \psi_2$; then $A_i \rightarrow \psi_1$ and $A_i \rightarrow \psi_2$ would have violated the $LL(k)$ property. Thus, the new grammar is $LL(k)$.

COROLLARY 3. *Given an $LL(k)$ grammar G with ϵ-rules, a strong $LL(k+1)$ grammar in Greibach normal form can be obtained for $L(G) - \{\epsilon\}$.*

Proof. From Theorem 3, an $LL(k+1)$ grammar without ϵ-rules can be obtained for $L(G) - \{\epsilon\}$, and from Theorem 4, an $LL(k+1)$ grammar in Greibach normal form can then be obtained. Construction 1 preserves this form and the result is strong $LL(k)$ by Theorem 2.

THEOREM 5. *Given an $LL(k+1)$ grammar without ϵ-rules for $k \geqslant 1$, there exists an $LL(k)$ grammar with ϵ-rules for the same language.*

Proof. From Theorem 4, the grammar can be rewritten so that it is in Greibach normal form, and is still $LL(k+1)$. This grammar will now be rewritten so that it is $LL(k)$ with ϵ-rules. If there is more than one production for nonterminal A with terminal symbol a as the first symbol on the right side of the production, then a new nonterminal, (A, a) will be introduced. Let the set of productions for A with a as the first symbol on the right side be

$$A \to aw_1 , A \to aw_2 ,..., A \to aw_n .$$

Then in the new grammar these productions will be replaced by

$$A \to a(A, a), (A, a) \to w_1 , (A, a) \to w_2 ,..., (A, a) \to w_n .$$

Note that if one of the original productions is $A \to a$, then the new grammar will contain the production $(A, a) \to \epsilon$.

Each of the productions in the new grammar for one of the original nonterminals begins with a distinct terminal symbol and therefore the next input symbol distinguishes between these productions. Since the next $k + 1$ input symbols distinguish between the original productions for A, the next k symbols after the a distinguish between the productions for (A, a) in the new grammar. Thus the new grammar is $LL(k)$.

The class of $LL(1)$ grammars in Greibach normal form are the simple grammars of Korenjak and Hopcroft (1966). For these grammars the right side of each production begins with a terminal symbol and each production for a nonterminal begins with a distinct terminal.

Canonical Pushdown Machines

We will assume throughout this section that $G = (T, N, P, S)$ is a strong $LL(k)$ grammar in Greibach normal form. We know from Corollary 4 that any $LL(k')$ grammar can be put into this form for some k satisfying $k \leqslant k' + 1$. We will describe a (deterministic) pushdown machine which recognizes $L(G)$.

The input set for the machine is $T \cup \{\vdash\}$ where $\vdash$ is an end of tape marker not in T. The tape alphabet is $N \cup T$. The machine is designed to accept sequences from the language followed by $k - 1$ end markers.

The machine rules will be written in the form $(A, w) \to \psi$ where A is a tape symbol, w is an input string of length k, and ψ is a string of tape symbols. If A is a terminal symbol, then w must begin with A and ψ must equal ϵ. The interpretation of the rule is that if A is the top stack symbol and w is the next k input symbols, then A is replaced by ψ.

A machine configuration is a pair (γ, x) where γ is a string of tape symbols and x a string of input symbols of length $\geqslant k - 1$. We write $(A \gamma, awy) \to$

$(\psi\gamma, wy)$, corresponding to a move of the machine, if $(A, aw) \to \psi$ is a rule of the machine. We write $(\gamma, x) \Rightarrow (\psi, y)$ if there is a sequence of moves such that

$$(\gamma, x) = (\varphi_0, z_0) \to (\varphi_1, z_1) \to \cdots \to (\varphi_n, z_n) = (\psi, y).$$

The pushdown tape will be initialized with the starting symbol S. The language accepted by the machine is the set of strings x in T^* such that $(S, x\vdash^{k-1}) \Rightarrow (\epsilon, \vdash^{k-1})$.

We will now describe how to obtain the set of machine moves from the grammar G. For A in N an w in $T^*\vdash^{k-1}/k$, there is at most one production p such that w is in $(L_p(A)\ R_k(A)\vdash^k)/k$. If there is such a production, it is of the form $A \to a\gamma$, and the corresponding machine rule is $(A, w) \to \gamma$. If there is no such production, there is no corresponding machine rule; the machine would halt and reject the input sequence if it arrived in a configuration calling for such a rule. For each a in T and w of length k beginning with a, the machine has the rule $(a, w) \to \epsilon$.

It is important for later proofs to observe that the machine has no ϵ-moves. It accepts an input string if and only if it reaches the end marker with an empty stack. The machine has only one state, so state information does not appear in our formalism. The look-ahead feature, however, gives the machine move power than a one-state machine without look-ahead.

Each of our machines has a equivalent pushdown machine without look-ahead. The finite state control of this equivalent machine has enough memory to store an input string of length $k - 1$ and perform such obvious tasks as reading in the first $k - 1$ input symbols. It operates in a manner such that its tape after $i + k - 1$ inputs has the same tape as our machine after i inputs. It need not read beyond the first end marker as it could instead complete its recognition with $k - 2$ ϵ-moves. There are of course many multi-state pushdown machines without look-ahead that have no equivalent one-state pushdown machine with look-ahead. Thus, our pushdown notation is not sufficient for describing all pushdown recognizers, but its simplicity makes it convenient for describing $LL(k)$ recognition.

The machine operates in close correspondence to a leftmost derivation. Let w_i be the string of the first i symbols of the input string and y_i be the remainder of the input string. If $(S, w_i y_i) \Rightarrow (\gamma_i, y_i)$, then $S \Rightarrow_L w_i\gamma_i$ and either $w_i\gamma_i \to_L w_{i+1}\gamma_{i+1}$ (if γ_i begins with a nonterminal) or $w_i\gamma_i = w_{i+1}\gamma_{i+1}$ (if γ_i begins with a terminal). To prove that the machine works, we need a lemma to the effect that the pushdown tape contains the needed information.

LEMMA 6. *Let M be the canonical pushdown machine for a strong $LL(k)$ grammar $G = (T, N, P, S)$ in Greibach normal form. If $(S, w_1w) \Rightarrow (\gamma, w)$ for w of length $k - 1$, then*

(i) $S \rightarrow_L w_1\gamma$;

(ii) *For all* w_3 *in* T^* *such that* $S \Rightarrow w_1w_3$ *and* $(w_3\vdash^k)/k - 1 = w$, w_3 *is in* $L(\gamma)$.

Proof. It is evident from the construction that $S \Rightarrow_L w_1\gamma$ and all that needs to be shown is that γ generates all the w_3 satisfying the conditions of (ii). Assume that w_3 satisfies $S \Rightarrow w_1w_3$ and $w_3\vdash^k/k - 1 = w$ but not $\gamma \Rightarrow w_3$. There must be a w_1' in T^*, A in N, and γ' in $(N \cup T)^*$ such that $S \Rightarrow_L w_1'A\gamma'$ is the last configuration before the left derivations of $S \Rightarrow_L w_1\gamma$ and $S \Rightarrow_L w_1w_3$ diverge. If one of the steps in the leftmost derivation $S \Rightarrow_L w_1\gamma$ is $yA\varphi \rightarrow ya\psi\varphi$ for production $A \rightarrow a\psi$, then ya must be a prefix of w_1 because the machine makes this substitution only after the input a is read and our hypothesis is that the machine has only read the word w_1. Therefore, the only intermediate string in the derivation of $w_1\gamma$ with prefix w_1 is $w_1\gamma$ itself. Therefore, if w_1' has the form w_1x, it must be true that $w_1'A\gamma'$ is in fact the string $w_1\gamma$. But this is impossible as we have assumed that γ cannot generate w_3. Since w_1 cannot therefore be a prefix of w_1', it follows that w_1' is a proper prefix of w_1 and w_1 may be assumed to have the form $w_1'y$ for some nonnull y. But since $(yw)/k$ has k symbols, the choice of production to apply at $w_1A\gamma'$ consistent with $(yw)/k$ is unique by the $LL(k)$ property, contrary to the assumption that the derivations differ. Thus the lemma is proved by contradiction.

When one applies Construction 1 to an $LL(k)$ grammar to make it strong and then applies the construction of the machine just given, one obtains a construction which is essentially the same as that given in Appendix I of Lewis and Stearns (1968).

THEOREM 6. *The canonical pushdown machine for a strong* $LL(k)$ *grammar in Greibach normal form recognizes the language generated by that grammar.*

Proof. It is evident from the construction that any word accepted by the machine has a grammatical derivation. It is also evident that the machine will not stop if there is a method of continuing the left derivation (as discussed prior to Lemma 6) in a manner consistent with the lookahead. Lemma 6 assures us that it is always possible to continue the left derivation process represented by the machine configuration. Thus, given a word x in the language, input word $x\vdash^{k-1}$ must be processed to completion and the resulting machine configuration must have a tape γ such that $\gamma \Rightarrow \epsilon$. Since G has no ϵ-rules, the only such tape is ϵ and hence $x\vdash^{k-1}$ is accepted by the machine. Thus the lemma is proved.

It should be pointed out that pushdown machines of the type described above can recognize languages that cannot be generated by an $LL(k)$ grammar for any k. For instance the language $\{a^nb^n + a^nc^n \mid n \geqslant 1\}$, which it will be shown has no $LL(k)$ grammar, can be recognized by the pushdown machine described as follows. The machine operates on the basis of an input word of length 2. Combinations of stack symbol and w not shown below result in rejection of the input string. The initial stack symbol is S.

$$\begin{aligned}
&(S, aa) \to SA,\\
&(S, ab) \to A,\\
&(S, ac) \to A,\\
&(A, bb) \to \epsilon,\\
&(A, cc) \to \epsilon,\\
&(A, b\vdash) \to \epsilon,\\
&(A, c\vdash) \to \epsilon.
\end{aligned}$$

Assume now that $\{a^nb^n\} \cup \{a^nc^n\}$ can be generated by an $LL(k)$ grammar. First rewrite the grammar in Greibach normal form. Now note that for each k, $S \Rightarrow_L a^{n-k}\gamma_n$, $\gamma_n \Rightarrow a^kb^n$, and $\gamma_n \Rightarrow a^kc^n$. Furthermore, for $n_1 \neq n_2$, $\gamma_{n_1} \neq \gamma_{n_2}$, or else the grammar could have a derivation of the form $S \Rightarrow a^{n_1-k}\gamma_{n_1} \Rightarrow a^{n_1}b^{n_2}$. Thus for some value of n, the length of γ_n is $> k + 2$. Furthermore in the derivations $\gamma_n \Rightarrow a^kb^n$ and $\gamma_n \Rightarrow a^kc^n$, since the grammar has no ϵ-rules, at most the first k symbols of γ_n can be expanded into a's. Thus each of the last two symbols of γ_n can be expanded into both b's and c's. Thus $\gamma_n \Rightarrow a^kb^{m_1}c^{m_2}$, and the language cannot be generated by any $LL(k)$ grammar.

Hierarchy of $LL(k)$ Languages

In this section it will be shown that for every $k \geqslant 1$, the class of languages generated by $LL(k)$ grammars is properly contained within the class generated by $LL(k + 1)$ grammars.

First, for each $k \geqslant 1$, consider the language $\{a^n(b^kd + b + cc)^n \mid n \geqslant 1\}$. Here "+" denotes the "or" operation. This language can be generated by the following strong $LL(k)$ grammar with ϵ-rules. It is $LL(k)$ because $b^{k-1}d$ is not in $R_k(B)$.

$$\begin{aligned}
&S \to aDA,\\
&D \to aDA,\\
&D \to \epsilon,\\
&A \to cc,\\
&A \to bB,\\
&B \to \epsilon,\\
&B \to b^{k-1}d.
\end{aligned}$$

However, this language cannot be generated by an $LL(k)$ grammar without ϵ-rules.

LEMMA 7. *There exists no $LL(k)$ grammar without ϵ-rules for the language $\{a^n(b^k d + b + cc)^n \mid n \geqslant 1\}$ where $k \geqslant 1$.*

Proof. Assume that there exists such a grammar. We may assume that it is a strong $LL(k)$ grammar $G = (N, T, P, S)$ in Greibach normal form and we let M be the canonical push-down machine associated with this grammar. There must be an integer n_0 such that $(S, a^{n_0+k-1}) \Rightarrow (\nu, a^{k-1})$ where $\mid \nu \mid \geqslant 2k - 1$. (This is because M must distinguish between all pairs a^{n_1} and a^{n_2} for $n_1 \neq n_2$). Thus ν has the form $\gamma_1 Z \gamma_2$ where γ_1 and γ_2 in $(N \cup T)^*$ satisfy $\mid \gamma_1 \mid \geqslant k - 1$ and $\mid \gamma_2 \mid \geqslant k - 1$ and Z is an element of $N \cup T$.

By Lemma 6, $S \Rightarrow_L a^{n_0}\gamma_1 Z \gamma_2$. Also by Lemma 6, $\gamma_1 Z \gamma_2 \Rightarrow a^{k-1}b^{n_0+k-1}$ and $\gamma_1 Z \gamma_2 \Rightarrow a^{k-1}c^{2(n_0+k-1)}$. Since G has no ϵ-rules and since $\mid \gamma_1 \mid \geqslant k - 1$, there must exist four numbers n_1, n_2, n_3, and m such that

$$\gamma_1 \Rightarrow a^{k-1}b^{n_1},$$

$$Z \Rightarrow b^{n_2} \qquad \text{and} \qquad Z \Rightarrow c^m,$$

$$\gamma_2 \Rightarrow b^{n_3} \qquad \text{and} \qquad n_1 + n_2 + n_3 = n_0 + k - 1.$$

Since $\mid \gamma_2 \mid \geqslant k - 1$ and G has no ϵ-rules, we know also that $n_3 \geqslant k - 1$ and $n_2 \geqslant 1$.

By Lemma 6,

$$(S, a^{n_0+k-1}b^{n_1+n_2+k-1}) \Rightarrow (\gamma_2, b^{k-1})$$

since $S \Rightarrow_L a^{n_0+k-1}b^{n_1+n_2}\gamma_2$, $\gamma_2 \Rightarrow b^{n_3}$ and $b^{n_3}/k - 1 = b^{k-1}$. Since furthermore $b^{k-1}db^{n_3}/k - 1 = b^{k-1}$ and $a^{n_0+k-1}b^{n_1+n_2}b^{k-1}db^{n_3}$ is in the language, it follows by Lemma 6 that $\gamma_2 \Rightarrow b^{k-1}db^{n_3}$.

Having obtained the relations $\gamma_1 \Rightarrow a^{k-1}b^{n_1}$, $Z \Rightarrow c^m$, and $\gamma_2 \Rightarrow b^{k-1}db^{n_3}$, we can conclude

$$S \Rightarrow a^{n_0+k-1}b^{n_1}c^m b^{k-1}\, db^{n_3},$$

but this string is clearly not in the language since it contains a d not prefixed by b^k. Therefore, the language cannot be obtained by an $LL(k)$ grammar without ϵ-rules.

THEOREM 7. *For every $k \geqslant 1$ the class of languages generated by $LL(k)$ grammars without ϵ-rules is properly contained within the class of languages generated by $LL(k + 1)$ grammars without ϵ-rules.*

Proof. For each $k \geqslant 1$, the language $\{a^n(b^k d + b + cc)^n \mid n \geqslant 1\}$ cannot be generated by an $LL(k)$ grammar without ϵ-rules, and since it is $LL(k)$, Theorem 3 implies that it can be generated by an $LL(k+1)$ grammar without ϵ-rules.

This theorem shows the existence of a hierarchy of languages generated by $LL(k)$ grammars without ϵ-rules. The smallest class in the hierarchy is that of the simple languages of Korenjak and Hopcroft (1966). By virtue of Theorem 5, the other classes correspond to classes in the hierarchy of languages generated by unrestricted $LL(k)$ grammars. The existence of this latter hierarchy was first demonstrated by Kurki–Suonio (1969).

Decidability of the Equivalence Problem

In this section it will be shown that it is decidable if two $LL(k)$ grammars generate the same language. We will begin with some definitions.

Let $G = (T, N, P, S)$ be a context-free grammar and γ be a string in $(N \cup T)^*$. We define $\tau(\gamma)$, the *thickness* of γ, as the length of the shortest terminal string that can be generated from γ, i.e., $\tau(\gamma) = \min\{n \mid \text{there exists } x$ in T^* such that $\gamma \Rightarrow x$ and $n = |x|\}$. Note that $\tau(\gamma_1\gamma_2) = \tau(\gamma_1) + \tau(\gamma_2)$.

For w in $T^*\vdash^*$ and γ in $(N \cup T)^*$, let $S(\gamma, w) = \{x \mid \gamma \Rightarrow x, x \text{ is in } T^* \text{ and } x\vdash^i = wy \text{ for some } i \geqslant 0 \text{ and } y \text{ in } T^*\}$. If $S(\gamma, w)$ is not empty, let

$$\tau_w(\gamma) = \min\{n \mid n = |x| \text{ for } x \text{ in } S(\gamma, w)\}.$$

LEMMA 8. *If grammar G is in Greibach normal form, t is the maximal thickness of the right sides of productions in P, w in $T^*\vdash^*$ is of length m, γ is in $(N \cup T)^*$, and $S(\gamma, w)$ is nonempty, then*

$$\tau(\gamma) \leqslant \tau_w(\gamma) \leqslant \tau(\gamma) + m(t-1).$$

Proof. Clearly $\tau_w(\gamma) \geqslant \tau(\gamma)$. Since G is in Greibach normal form, it requires the application of at most m productions to convert $\gamma\vdash^i$ into a string of the form $w\psi$. Each of these productions replaces a nonterminal (whose thickness is at least 1) by the right side of a production (whose thickness is at most t), thereby increasing the thickness of the intermediate string by at most $t - 1$. Thus $\tau_w(\gamma) \leqslant \tau(\gamma) + m(t-1)$.

THEOREM 8. *It is decidable whether or not two $LL(k)$ grammars generate the same language.*

Proof. For purposes of this proof, we will call strong $LL(k)$ grammar $G = (T, N, P, S)$ *super strong* if (1) all its productions have the form $A \rightarrow a\varphi$

for some A in N, a in T, and φ in N^*, and (2) there exists a function $f : N \to 2^{T^{*}\vdash^{k}/k}$ such that for all w in $T^*\vdash^k/k$, w_1 in T^* and $A\gamma$ in N^* satisfying $S \Rightarrow_L w_1 A\gamma$, $S(A\gamma, w)$ is nonempty if and only if w is in $f(A)$. If an $LL(k)$ grammar satisfies condition 1, it can be made to satisfy condition 2 by applying Construction 1. The function f is given simply by $f((A, R)) = L(A)\, R\vdash^k/k$ where (A, R) is a nonterminal expressed in the notation of the construction. An $LL(k)$ grammar satisfying condition 1 is easily obtained from a $LL(k)$ Greibach normal form grammar by introducing nonterminals A_a and productions $A_a \to a$ for a in T and then replacing occurrences of a by A_a where required to obtain the form expressed by 1.

The canonical push-down machine for a super strong $LL(k)$ grammar will stop and reject if and only if it discovers a tape γ and look-ahead word w such that $S(\gamma, w)$ is empty.

To prove the theorem, we need only give an equivalence test for two superstrong $LL(k)$ grammars $G_1 = (T_1, N_1, P_1, S_1)$ and $G_2 = (T_2, N_2, P_2, S_2)$. We let M_1 and M_2 be the corresponding canonical deterministic pushdown machines. We will describe a third deterministic pushdown machine, M_3 with the property that $L(G_1) = L(G_2)$ if and only if M_3 accepts the set of all strings.

M_3 will attempt to simultaneously simulate M_1 and M_2 by having a two track pushdown tape, one track for each tape of the simulated machine. For a symbol which can appear on the tape of M_1 or M_2 that has thickness τ, M_3 will have a "symbol" that can occupy τ squares on the corresponding track of its tape. Assume that after reading in w_1 (followed by look-ahead word w of length $k - 1$), M_1 has ν on its tape, M_2 has μ on its tape, and neither machine is in the rejecting state. Then $\tau(\nu)$ will be the size (number of squares) occupied by the string on the first track of M_3 that corresponds to ν, and $\tau(\mu)$ will be the size of the string on the second track of M_3 that corresponds to μ.

The key observation is that the lengths $\tau(\nu)$ and $\tau(\mu)$ will differ by at most $2(k - 1)(t - 1)$ whenever $L(G_1) = L(G_2)$, where t is the maximum thickness of the right sides of the productions in $P_1 \cup P_2$.

To verify this last observation, we note that $L(G_1) = L(G_2)$ implies that $\tau_w(\nu) = \tau_w(\mu)$, for if to the contrary $\tau_w(\nu) > \tau_w(\mu)$, the minimum continuation of ν acceptable by M_1 would not be acceptable to M_2. Since the length of w is $k - 1$, from Lemma 8

$$\tau_w(\nu) - (k - 1)(t - 1) \leqslant \tau(\nu) \leqslant \tau_w(\nu)$$

and

$$\tau_w(\mu) - (k - 1)(t - 1) \leqslant \tau(\mu) \leqslant \tau_w(\mu).$$

From the fact that $\tau_w(\nu) = \tau_w(\mu)$, it therefore follows that $|\tau(\nu) - \tau(\mu)| \leqslant 2(k-1)(t-1)$.

M_3 will now be described in greater detail. It has a two track tape as described above, but the top $2(k-1)(t-1)+1$ cells of the tape are kept in the finite state control unit. Thus if the difference in thickness of the two simulated tapes is less than this amount, M_3 has access to the top of both tracks of the simulating tape. M_3 simulates both M_1 and M_2 as long as neither one has entered the rejecting state and the difference in thickness between the two tapes being simulated is $\leqslant 2(k-1)(t-1)$.

Machine M_3 is designed to accept all input sequences until one of the following three things occur:

1. The difference in thickness between the two tapes become greater than $2(k-1)(t-1)$;

2. An input sequence causes exactly one of the machines M_1 or M_2 to stop in a rejecting state;

3. One of the machines accepts a sequence and the other does not.

If M_3 rejects because of reason 1, we know that the machine with the shorter tape can accept a short sequence which the other machine cannot and hence $L(G_1) \neq L(G_2)$. If M_3 rejects because of the second reason, we know that the machine which stopped in a rejecting state cannot accept any continuation of the input sequence whereas the other machine can. The languages are obviously different if rejection is for the last reason. Conversely, if there is an input sequence in $L(G_1)$ which is not in $L(G_2)$, then the application of that sequence to M_3 will obviously cause M_3 to reject for one of these reasons.

The problem has now been reduced to deciding if deterministic pushdown machine M_3 rejects any sequences (or if the complement machine accepts any). This is a well-known decidable question (Ginsburg and Greibach (1966)).

Properties of $LL(k)$ Grammars

In this section, various closure and undecidability properties of $LL(k)$ grammars will be given. These results are primarily of a negative nature. It will also be shown that every $LL(k)$ grammar is an $LR(k)$ grammar.

THEOREM 9. *If the finite union of disjoint $LL(k)$ languages is regular, then all the languages are regular.*

Proof. Let T be the combined terminal vocabulary of the languages. It is sufficient to prove the results for the regular set T^* since if the union is a regular set R, one can add the $LL(1)$ language $T^* - R$ to the given set of $LL(k)$ languages in order to get a disjoint union which gives T^*. Letting n be the number of languages in the union, we let M_i (with starting symbol S_i) for $1 \leqslant i \leqslant n$ represent the canonical pushdown machines for these languages. We will call a tape string, ν, *reachable* for a machine M_i if there exists a input string w_1x such that $(S_i, w_1x\vdash^{k-1}) \Rightarrow (\nu, x\vdash^{k-1}) \Rightarrow (\epsilon, \vdash^{k-1})$. To prove the theorem, we will derive an upper bound on the lengths of the reachable tape strings. Thus, each language can be recognized by a push-down machine whose tape can contain only a finite set of strings and must therefore be regular.

For each input sequence w_1 in T^* and w in $T^*\vdash^{k-1}/(k-1)$ let V be the set of pairs (i, ν_i) such that for machine M_i, $(S_i, w_1w) \Rightarrow (\nu_i, w)$ and for some z, $(\nu_i, wz) \Rightarrow (\epsilon, \vdash^{k-1})$. Let m be the integer and y the sequence in T^* such that $w = y\vdash^m$. Since every input sequence must be accepted by one of the machines, there is an (i, ν_i) in V such that for M_i, $(\nu_i, w\vdash^{k-m-1}) \Rightarrow (\epsilon, \vdash^{k-1})$. Since the canonical machines do not make ϵ-moves and accept only with empty stacks, ν_i cannot have more symbols than $|y|$. Therefore, $n_1 = k-1$ is a bound on the length of ν_i.

Now suppose that we have found a j element set V_j which is a proper subset of V and which has tape strings of maximum length n_j. Let m be the length of the shortest string z in T^* such that w is a prefix of $z\vdash^{k-1}$ and for some (p, ν_p) in $V - V_j$, $(\nu_p, z\vdash^{k-1}) \Rightarrow (\epsilon, \vdash^{k-1})$. Such a z exists because V_j is a proper subset of V, the languages are disjoint, and each configuration in V has a continuation of the input sequence that leads to acceptance. Let $n_{j+1} = \max(n_j, m)$. The length of ν_p must be less than or equal to m and therefore $V_{j+1} = V_j \cup \{(p, \nu_p)\}$ is a $j+1$ element subset of V with tapes of length n_{j+1} or less. We have defined by induction bounds n_i and sets V_i for all i such that $1 \leqslant i \leqslant |V|$. Thus if the n_i can be bounded independent of w_1, the tape lengths of all reachable configurations will be bounded and each machine will accept a regular set.

We have already observed that n_1 has a bound independent of w_1; namely, k. Suppose that a bound b_j has been found for all the possible n_j resulting from all choices of w and w_1. The possible b_{j+1} are determined from the possible V_j which can arise. But the V_j which arise have tape lengths of b_j or less and there are only a finite number of such V_j. Thus we may choose b_{j+1} to be the maximum n_{j+1} over the range of possible V_j. Thus the bounds b_i are established by induction and the theorem proved.

COROLLARY 4. *The complement of a nonregular $LL(k)$ language is never $LL(k)$.*

COROLLARY 5. *Theorem 9 generalizes to languages recognized by deterministic push-down machines which accept only with an empty stack and do not make ϵ-moves.*

THEOREM 10. *The $LL(k)$ languages are not closed under complementation, union, intersection, reversal, concatenation, or ϵ-free homomorphisms.*

Proof. Korenjak and Hopcroft (1966) give examples of simple languages whose closure under most of these operations produces non-$LL(k)$ languages.

Since the language $L_1 = \{a^m b^n \mid 1 \leqslant m < n\}$ is an $LL(1)$ language which is not finite state, its complement, L_2, is not $LL(k)$ by Corollary 4. The language $L_3 = \{a^n b^n + a^n c^n \mid n \geqslant 1\}$ was shown to be a non-$LL(k)$ language in an earlier section. However, L_3 is the union of two $LL(1)$ languages, $L_4 = \{a^n b^n \mid n \geqslant 1\}$ and $L_5 = \{a^n c^n \mid n \geqslant 1\}$. The language $L_6 = \{a^n (b + c)^n \mid n \geqslant 1\}$ and $L_7 = \{a^n b^m + a^n c^m \mid n, m \geqslant 1\}$ are two $LL(1)$ languages, whose intersection is L_3, which is not $LL(k)$. $L_8 = \{b^n a^n + c^n a^n \mid n \geqslant 1\}$ is an $LL(1)$ language whose reversal is L_3, and is not $LL(k)$.

From Theorem 9, the language $L_9 = \{a^m b^n \mid 1 \leqslant n \leqslant m\}$ is not an $LL(k)$ language since its union with the $LL(1)$ language L_1 is the finite state language $\{a^m b^n \mid m, n \geqslant 1\}$. However, L_9 is the concatenation of a^* and $\{a^n b^n \mid n \geqslant 1\}$, each of which is an $LL(1)$ language. Therefore the set of $LL(k)$ languages is not closed under concatenation.

The language $L_{10} = \{da^m b^{m+1} + ea^m c^{m+1} \mid m \geqslant 0\}$ is an $LL(1)$ language, but its image under the homomorphism that converts d and e to a while leaving a, b, and c unchanged is the non-$LL(k)$ language L_3.

Some undecidability properties will now be given.

THEOREM 11. *Given a context-free grammar, it is undecidable whether or not there exists a k such that the grammar is $LL(k)$.*

Proof. Consider a Turing machine M with some initial finite string on its tape. An instantaneous description (Davis (1958)) for the Turing machine is a string that indicates the current tape contents, internal state, and position of the head on the tape. The states will be denoted as q_i and the tape symbols as a_j, with a_0 representing the blank symbol.

Let c be a symbol that cannot appear in an instantaneous description and for a string x let x^r denote the reversal of x. An $LL(3)$ grammar G_1 with sentence symbol S_1 will be given whose sentences are of the form $p_0\, c\, p_1^r\, c\, p_2 \cdots c\, p_{2i-1}^r\, c\, p_{2i}\, c \cdots c\, p_{2n}$ where p_0 is the initial instantaneous description and for each i between 1 and n, p_{2i} is the instantaneous description that results from instantaneous description p_{2i-1} by one move of the machine. The grammar has the following productions.

$S_1 \to p_0 B,$	
$B \to \epsilon,$	
$B \to cDB,$	
$D \to A,$	
$D \to a_j q_i C a_j q_n a_0$	if (q_i, a_j, R, q_n) is a quadruple of M,
$A \to a_i A a_i$	for all a_i,
$A \to a_j q_i C q_n a_k$	if (q_i, a_j, a_k, q_n) is a quadruple,
$A \to a_m a_j q_i C a_j q_n a_m$	if (q_i, a_j, R, q_n) is a quadruple,
$A \to a_j q_i a_m C q_n a_m a_j$	if (q_i, a_j, L, q_n) is a quadruple,
$A \to a_j q_i c q_n a_0 a_j$	if (q_i, a_j, L, q_n) is a quadruple,
$C \to a_i C a_i$	for all a_i,
$C \to c.$	

Another $LL(3)$ grammar G_2 with sentence symbol S_2 will be given whose sentences are of the form $q_0\, c\, q_1^r\, c\, q_2 \cdots c\, q_{2i}\, c\, q_{2i+1}^r\, c \cdots c\, q_{2n-1}^r\, c$ where for each i between 0 and $n-1$, q_{2i+1} is the instantaneous description that results from instantaneous description q_{2i} by one move of the machine. G_2 has the following productions:

$S_2 \to BcS_2,$	
$S_2 \to \epsilon,$	
$B \to A,$	
$B \to q_i a_j C a_j a_0 q_n$	if (q_i, a_j, L, q_n) is a quadruple,
$A \to a_i A a_i$	for all a_i,
$A \to q_i a_j C a_k q_n$	if (q_i, a_j, a_k, q_n) is a quadruple,
$A \to a_m q_i a_j C a_j a_m q_n$	if (q_i, a_j, L, q_n) is a quadruple,
$A \to q_i a_j a_m C a_m q_n a_j$	if (q_i, a_j, R, q_n) is a quadruple,
$A \to q_i a_j c a_0 q_n a_j$	if (q_i, a_j, R, q_n) is a quadruple,
$C \to a_i C a_i$	for all a_i,
$C \to c.$	

Now let G_3 be the grammar with sentence symbol S_3 whose productions include all the productions of G_1 and G_2 plus the two new ones $S_3 \to S_1$ and $S_3 \to S_2$.

A string of the form

$$r_0 \, c \, r_1^r \, c \, r_2 \, c \cdots c \, r_{2n}$$

is a prefix of strings in both $L(G_1)$ and $L(G_2)$ if and only if $r_0, r_1, r_2, \ldots, r_n$ is the sequence of instantaneous descriptions produced by the Turing machine when it starts with p_0. Therefore, if the machine does not halt, then there are arbitrarily long sequences that are prefixes of sentences in $L(G_1)$ and $L(G_2)$, i.e., for each k there is a word w in T^* (composed from successive instantaneous descriptions of M) such that $|w| = k$ and for some w_1 and w_2 in T^*, $S_3 \Rightarrow ww_1$ beginning with the production $S_3 \to S_1$ and $S_3 \Rightarrow ww_2$ beginning with the production $S_3 \to S_2$. Thus, if the machine does not halt, there does not exist a k such that G_3 is $LL(k)$.

On the other hand, if the machine does halt, then there is a bound on the length of any prefix of strings in both $L(G_1)$ and $L(G_2)$ namely the length of the series of instantaneous descriptions that lead to the halting condition. Therefore, by looking ahead this amount it is always possible to choose between productions $S_3 \to S_1$ and $S_3 \to S_2$. Any subsequent choice between productions can be made on the basis of the next three input symbols. Therefore, if the machine halts, G_3 is an $LL(k)$ grammar for some k.

Since G_3 is $LL(k)$ if and only if the machine halts, which is undecidable, it is undecidable if there exists a k such that G_3 is $LL(k)$.

Although given a general context free grammar, it is undecidable if there exists a k such that it is $LL(k)$, given an $LR(k)$ grammar, the problem is decidable.

THEOREM 12. *Given an $LR(k)$ grammar of known k, it is decidable if there exists a k' such that the grammar is $LL(k')$.*

Proof. The problem of computing the look-ahead required to determine which production to apply is very similar to the problem in Lewis and Stearns (1968) of computing the "distinction index" of two occurrences of a nonterminal and it is a fairly straightforward exercise to reduce the first problem to the second. As there is little insight to be gained in repeating the relevant definitions and techniques from Lewis and Stearns (1968), we omit further detail.

THEOREM 13. *It is undecidable whether or not an arbitrary context-free grammar generates an $LL(k)$ language, even for a fixed k.*

Proof. The proof of the corresponding theorem (Korenjak and Hopcroft (1966)) for simple grammars is valid for $LL(k)$ grammars.

However, given an arbitrary context free grammar, it is decidable (Paull and Unger (1968)) if there exists an $LL(1)$ grammar without ϵ-rules that is structurally equivalent to the original one.

We will now show that every $LL(k)$ grammar is also an $LR(k)$ grammar (Knuth (1965)). We will use the definition of $LR(k)$ grammars that appears in Lewis and Stearns (1968).

A grammar is called $LR(k)$ if it is unambiguous and for all w_1, w_2, w_3, w_3' in T^* and A in N, $S \Rightarrow w_1 A w_3$, $A \Rightarrow w_2$, $S \Rightarrow w_1 w_2 w_3'$, and $w_3/k = w_3'/k$ imply that $S \Rightarrow w_1 A w_3'$.

THEOREM 14. *Every $LL(k)$ grammar is also an $LR(k)$ grammar.*

Proof. From Lemma 4, every $LL(k)$ grammar is unambiguous. Now assume that for an $LL(k)$ grammar there is a w_1, w_2, w_3, w_3' in T^* and A in N such that $S \Rightarrow w_1 A w_3$, $A \Rightarrow w_2$, $S \Rightarrow w_1 w_2 w_3'$ and $w_3/k = w_3'/k$. If $w_3' = w_3$, then $S \Rightarrow w_1 A w_3'$. If not, let $xB\nu$ where x is in T^* and B in N be the last intermediate string before the leftmost derivations of $w_1 w_2 w_3$ and $w_1 w_2 w_3'$ diverge. Then for w_4, w_4', w_5, w_5' in T^*, and productions p and p' in P (with $p \neq p'$).

$$
\begin{array}{rlrl}
S & \underset{L}{\Rightarrow} xB\nu, & & \\
B & \Rightarrow w_4\ (p), & B & \Rightarrow w_4'\ (p'), \\
\nu & \Rightarrow w_5, & \nu & \Rightarrow w_5', \\
xw_4w_5 & = w_1w_2w_3, & xw_4'w_5' & = w_1w_2w_3'.
\end{array}
$$

If $w_1 w_2 = xy$ for some y in T^*, then $w_4 w_5/k = w_4' w_5'/k$ and the $LL(k)$ property is violated. Therefore, $x = w_1 w_2 y$ for some y in TT^*, and the leftmost derivations do not diverge until after the generation of w_2 from A. Hence $S \Rightarrow w_1 A w_3'$, and the grammar is $LR(k)$.

RECEIVED: July 22, 1969; REVISED: January, 1970

ACKNOWLEDGMENT

The authors are grateful to P. M. Lewis for valuable discussion and encouragement.

REFERENCES

DAVIS, M. (1958), "Computability and Unsolvability," McGraw-Hill, New York.

GINSBURG, S., AND GREIBACH, S. (1966), Deterministic context-free languages, *Information and Control* **9**, 620–648.

Greibach, S. (1965), A new normal-form theorem for context-free phrase structure grammars, *J. ACM* **12**, 42–52.

Korenjak, A. J., and Hopcroft, J. E. (1966), Simple deterministic languages, *IEEE Conf. Rec. 7th Ann. Symp. on Switching and Automata Theory*, IEEE Pub. No. 16-C-40, 36–46.

Knuth, D. E. (1965), On the translation of languages from left to right, *Information and Control* **8**, 607–639.

Kurki-Suonio, R. (1969), Notes on top down languages, *BIT*, **9**, 225-238.

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

Oettinger, A. (1961), Automatic syntactic analysis and the pushdown store, *in* "Structure of Language and Its Mathematical Concepts," pp. 104–129, *Proc. 12th Symp. Appl. Math.*, Amer. Math. Soc., Providence, R. I.

Paull, M. C., and Unger, S. H. (1968), Structural equivalence of *LL-k* grammars, *IEEE Conf. Rec. 9th Ann. Symp. Switching and Automata Theory*, IEEE Pub. No. 68-C-50-C, 176–186.