Check for updates

PROPERTIES OF DETERMINISTIC TOP DOWN GRAMMARS

D.J. Rosenkrantz & R.E. Stearns

General Electric Research & Development Center
Schenectady, N.Y.

Abstract

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 first defined and a procedure is given for determining if a context free grammar is LL(k) for a given value of k. It is shown that ϵ-rules can be eliminated from an LL(k) grammar, at the cost of increasing the value of k by one, and a description is given of a canonical pushdown machine for recognizing LL(k) languages. It is shown that for each value of k there are LL(k+1) languages that are not LL(k) languages. 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 [L&S]. The object of this paper is to study these grammars.

More specifically, we study the LL(k) grammars defined by Lewis and Stearns in [L&S]. 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 canonic forms which inherit the LL(k) property. Some of the results were stated previously in [L&S] 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 non-terminal alphabet, P is a finite set of symbols each of which represents a production that we write in the form $A \rightarrow w$ 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 $\gamma_1 \rightarrow \gamma_2$ if and only if there exit φ_1 and φ_2 in $(N \cup T)^*$ and production $A \rightarrow \gamma$ in P such that $\gamma_1 = \varphi_1 A \varphi_2$ and $\gamma_2 = \varphi_1 \gamma \varphi_2$. We write $\gamma_1 \rightarrow_L \gamma_2$ if in addition φ_1 is in T^*. We let "$\Rightarrow$" represent the transitive completion of "$\rightarrow$" and "$\Rightarrow_L$" the "reflexive" transitive completion of "$\rightarrow_L$". 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 $A \rightarrow \gamma$, we write $L_p(A) = L(\gamma)$.

For a given word w and non-negative 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 non-terminal, w is a word in $(N \cup T)^*$ and p 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: A grammar $G = (T,N,P,S)$ is said to be an LL(k) grammar for some positive integer k if and only if given

1) a word w in T^* such that $|w| \le k$;
2) a non-terminal 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) $A \Rightarrow w_2 \quad (p)$;
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 [L&S]. The machine uses a predictive recognition scheme [OET] in a manner that "uses" the grammar in the recognition process and can be said to "recognize" each production. An LL(k) grammar is always an LR(k) grammar as defined in [KN].

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 $2^{T^*/k}$ to represent the set of all subsets of T^*/k. We use the term structurally equivalent as in [P&U] to mean that two grammars generate the same strings and the same trees (with the intermediate nodes unlabeled) for these strings.

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

$$T'' = T \times 2^{T^*/k}$$

$$N'' = N \times 2^{T^*/k}$$

$$S'' = (S, \{\epsilon\})$$

$$P'' = P \times 2^{T^*/k}$$

where symbol (p,R) represents the production

$$(A,R) \to (A_n,R_n) \ldots (A_1,R_1)$$

where $A \to A_n \ldots A_1$ is the production p and R_{i+1} satisfies

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

for all $n > i \geq 0$. If $n = 0$, the right-hand sides of the productions are understood to represent ϵ. The condition for R_1 reduces to

$$R_1 = R/k = R.$$

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 occurrence 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 $(N' \cup T)^*$,

$$S' \Rightarrow_L \varphi(A,R)\gamma \text{ implies } R = L(\gamma)/k.$$

Proof: We will prove the result by induction on the length of a leftmost derivation. 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 some $S \Rightarrow_L w(A,R)\gamma_1$ for w in T* and let (p,R) be a production which applies to (A,R) from which we obtain

$$S \Rightarrow_L w(A,R)\gamma_1 \rightarrow_L w(A_n,R_n) \ldots (A_1,R_1)\gamma_1.$$

The lemma certainly holds for occurrences of nonterminals in γ_1 since the string following them is the same as before. It also holds for nonterminals in w since there are none. Thus, it remains to be shown that it holds for the nonterminals (A_i,R_i). But this is true by construction since

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

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 constructing 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 certainly computable. 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 couple of lemmas relating the LL(k) property with left derivations.

Lemma 3: If G = (T,N,P,S) is an LL(k) grammar, then for all w_1 in T*, A in N, and w in T*/k, there exists a γ in $(N \cup T)^*$ such that for all w_2 and w_3 in T*, the three relations

1) $S \Rightarrow w_1Aw_3$

2) $A \Rightarrow w_2$

3) $(w_2w_3)/k = w$

imply the two relations

4) $S \Rightarrow_L w_1A\gamma$

5) $\gamma \Rightarrow w_3$.

Furthermore, if 1, 2, and 3 are satisfied for some w_2 and w_3, then there is only one γ satisfying 4 and 5.

Proof: Suppose that w_2 and w_3 do exist which satisfy 1, 2, and 3. Because $w_1w_2w_3$ has a derivation which includes w_1Aw_3, there must be a γ in $(N \cup T)^*$ such that

$S \Rightarrow_L w_1 A\gamma$ and $\gamma \Rightarrow w_3$ (which are conditions 4 and 5). Now consider another pair w_2' and w_3' satisfying 1, 2, and 3. There must also be a γ' in $(N \cup T)^*$ such that $S \Rightarrow_L w_1 A\gamma'$ and $\gamma' \Rightarrow w_3'$. If $\gamma \neq \gamma'$, then the two left derivation sequences must differ. Let $\bar{w}_1 B\varphi$ for $\bar{w}_1$ in T^*, B in N, and φ in $(N \cup T)^*$ be the last word for which the two strings are the same. Word $\bar{w}_1$ must of course be a prefix of w_1 since $w_1 A$ is the beginning of the words being derived. Symbolically, we now have

6) $S \Rightarrow_L \bar{w}_1 B\varphi$

and

7) $w_1 = \bar{w}_1 v$ for some v in T^*.

Since $\bar{w}_1 B\varphi$ is the departure point in the two derivations, there are two productions $B \rightarrow \psi$ and $B \rightarrow \psi'$ such that

8) $\psi\varphi \Rightarrow v w_2 w_3$

and

9) $\psi'\varphi \Rightarrow v w_2' w_3'$.

We will now show a violation of the LL(k) definition between the two productions for B. Because of 8, there are $\bar{w}_2$ and $\bar{w}_3$ in T^* such that

10) $B \rightarrow \psi \Rightarrow \bar{w}_2$

11) $\varphi \Rightarrow \bar{w}_3$,

and $\bar{w}_2 \bar{w}_3 = v w_2 w_3$.

Similarly by 9, there are $\bar{w}_2'$ and $\bar{w}_3'$ in T^* such that

12) $B \rightarrow \psi' \Rightarrow \bar{w}_2'$,

13) $\varphi \Rightarrow \bar{w}_3'$,

and $\bar{w}_2' \bar{w}_3' = v\bar{w}_2' \bar{w}_3'$. Because of 3,

$$(\bar{w}_2 \bar{w}_3)/k = (v w_2 w_3)/k = (v w_2' w_3')/k$$
$$= (\bar{w}_2' \bar{w}_3')/k.$$

letting

14) $\bar{w} = (\bar{w}_2 \bar{w}_3)/k = (\bar{w}_2' \bar{w}_3')/k$,

relations 6, 10, 11, 12, 13, and 14 violate the LL(k) definition as we have two productions $B \rightarrow \psi$ and $B \rightarrow \psi'$ which satisfy conditions 4, 5, and 6 of the definition. Therefore, we conclude that $\gamma = \gamma'$ and then 4 and 5 hold for all choices of w_2 and w_3.

The last statement of the lemma follows as a special case of the above by taking $w_2' = w_2$ and $w_3' = w_3$.

Lemma 4: A grammar (T,N,P,S) is LL(k) if and only if for each w_1 and w in T^*, A in N and γ in $(N \cup T)^*$ such that

$$S \Rightarrow_L w_1 A\gamma,$$

there exist at most one production p in P such that

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

Proof: Suppose there are two such productions. Then each production has a w_2 and w_3 in T^* such that $A \Rightarrow w_2(p)$ and $\gamma \Rightarrow w_3$ (hence $S \Rightarrow w_1 A w_3$) which is a violation of the LL(k) definition.

Conversely, assume the LL(k) definition is violated by some w_1, w_2, w_3, w_2', w_3', w in T^* and A in N and distinct p and p' in P. Letting $w_1' B\gamma$ be the first point where the left derivations differ, $w_1 w = w_1' u$ for some u. Thus, there are q and q' such that u/k is in

$$(L_q(B)L(\gamma))/k \cap (L_{q'}(B)L(\gamma))/k.$$

Thus, the lemma is proven.

Corollary 3: An LL(k) grammar is unambiguous.

Proof: The lemma says that in each step of a left derivation, there is at most one production which will enable one to reach the desired terminal string.

The significance of Lemma 4 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 4 holds and is therefore an LL(k) test.

Strong LL(k) Grammars

In this section we define the concept of a strong LL(k) grammar. The power of 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

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

For 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 position integer k if and only if given

1. a word w in T^* such that $|w| \le k$;
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. $S \Rightarrow w_1 Aw_3$;
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 input. 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). If $S' \Rightarrow w_1 (A,R) w_3$ for w_1 and w_3 in T^*, we know that $S' \Rightarrow_L w_1 (A,R)\gamma$ for some γ in $(N' \cup T)^*$ such that $\gamma \Rightarrow w_3$. For each w in T^*/k, we know from Lemma 4 that there is at most one p such that w is in $(L_p((A,R))L(\gamma))/k$.

By the last statement in Lemma 3, we know that γ is independent of w_1 and hence $R = R_k((A,R))$ which we know is equal to $L(\gamma)/k$ by Lemma 2. Hence, there is at most one p such that

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

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

Role of ϵ-Rules

We use ϵ to represent the null or length zero string. 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 rewritting 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 non-nullable otherwise. In particular, this means that terminals are non-nullable. The first step is to rewrite G so that the first symbol on the righthand side of a non-ϵ-rule is non-nullable. 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 righthand 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:

$$A \to A_1 \dots A_n B_1 \dots B_m$$

where $A_1 \dots, A_n$ are nullable, B_1 is non-nullable if m>0, m and n are non-negative integers, and where the case n=0 is interpreted to mean that $A_1 \dots A_n = \epsilon$ and the case m=0 to mean $B_1 \dots B_n = \epsilon$. For each such production we let P_1 contain the productions.

$$A \to A_1' A_2 \dots A_n B_1 \dots B_m$$

$$A \to A_2' \dots A_n B_1 \dots B_n$$

$$A \to A_n' B_1 \dots B_m$$

$$A \to B_1 \dots B_m .$$

Furthermore, if A is nullable, we let P_1 contain these same productions with A' on the lefthand side instead of A. If, however, m=0, the production $A' \to B_1 \dots B_m$ (i.e. $A' \to \epsilon$) is omitted.

The starting symbol S_1 is taken to be S if S is non-nullable and to be S' if S is nullable.

Each production in P_1 corresponds to a derivation in G. For example, $A \to A_2 \dots A_n B_1 \dots B_m$ corresponds to the production $A \to A_1 \dots A_n B_1 \dots B_m$ followed by the derivation of $A_1 \Rightarrow \epsilon$. 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 leftmost branches which result in ϵ and replacing leftmost occurrences of other nullable nonterminals by their non-nullable counterparts.

To verify that G_1 is LL(k), suppose that we are given w_1 in T*, A_o in N_1, and w in T*/k satisfying conditions 1, 2, and 3 of Definition 1. The corresponding symbols in G determine a production $A \to A_1 \dots A_n B_1 \dots B_m$ as described above (where A_o = A or A'), and it is clear under the correspondence of derivations that any production in P_1 satisfying 4 and 5 of Definition 1 must be one of the productions obtained from this. Furthermore, it is possible to determine which of the leading A_i must go into ϵ and which is the first symbol to not go into ϵ as the various ϵ-derivations which do this are determined without further lookahead. This information then determines the one possible production derived from $A \to A_1 \dots A_n B_1 \dots B_m$ that satisfies Definition 1.

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 non-null 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^ϵ be the non-nullable 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,

$$\alpha(B_1B_2A_3A_4B_5A_6A_7) = [B_1]\ [B_2A_3A_4][B_5A_6][B_7]$$

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 non-nullable 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

$$S_1 \Rightarrow_L b_1b_2A_1B_2$$

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

$$[S_1] \Rightarrow_L [b_1][b_2A_1][B_2]$$

represents the situation where only 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 production for $[b_2A_1]$ 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 $[BA_1 \ldots A_n]$ must have distinct A_i for otherwise G_1 would be ambiguous. (If A_i were repeated, there would be two derivations of $A_1 \ldots A_n \Rightarrow w_o$ where w_o is a non-null 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 $[S_1]$). Finally, let the production set P_2 for G_2 be the set of productions determined by the following three rules:

<u>Rule 1</u>: If B in N_1 and γ in V_ϵ^* 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:

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

<u>Rule 2</u>: If b in T, A in V_ϵ, and γ_1 and γ_2 in V_ϵ^* are such that $[b\gamma_1A\gamma_2]$ is in V' and if $A \rightarrow \gamma$ is a non-ϵ-rule of P_1, then P_2 has the production

$$[b\gamma_1A\gamma_2] \rightarrow [b]\ \alpha(\gamma\gamma_2).$$

<u>Rule 3</u>: If b in T and w in V_ϵ^+ are such that $[b\gamma]$ is in V', then P_1 has production

$$[b\gamma] \rightarrow [b].$$

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 $A \rightarrow \gamma$ 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 satisfying conditions 1, 2, and 3 of Definition 1. If the nonterminal is of the form $[B\gamma]$ where B is in N_1, then the only production which can be applied is the one obtained via Rule 1 from the production of P_1 which can be applied to B. If the nonterminal has the form $[bA_1 \ldots A_n]$ 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. Thus, the grammar is LL(k+1) and the theorem is proven.

A nonterminal symbol, A, is said to be <u>left recursive</u> if and only if $A \Rightarrow Aw$ for some w in T* and $L(A) \neq \emptyset$.

<u>Lemma 5</u>: An LL(k) grammar G can have no left recursive nonterminals.

<u>Proof</u>: 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 $$S \Rightarrow uAy^k v,\ A \Rightarrow xy\ (p),\ A \Rightarrow x\ (p'),$$

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 [GR] if the righthand side of every production begins with a terminal symbol.

<u>Theorem 4</u>: Given an LL(k) grammar without ϵ-rules, another LL(k) grammar in Greibach normal form can be obtained for the same language.

<u>Proof</u>: 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 $i \leq j$, 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 hand side of a production is replaced by all the productions for A_i (each of which begins with a terminal symbol). The rewritten grammar is LL(k) since if two new productions for a nonterminal cannot be distinguished by the next k input symbols, then there would be two corresponding productions of the original grammar which could not be distinguished by the next k input symbols.

<u>Corollary 4</u>: Given an LL(k) grammar G with ϵ-rules, a strong LL(k+1) grammar in Greibach normal form can be obtained for $L(G) - \{\epsilon\}$.

<u>Proof</u>: 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.

<u>Theorem 5</u>: Given an LL(k+1) grammar without ϵ-rules for $k \geq 1$, there exists an LL(k) grammar with ϵ-rules for the same language.

<u>Proof</u>: 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 hand 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 righthand side be

$$A \rightarrow aw_1$$
$$A \rightarrow aw_2$$
$$\vdots$$
$$A \rightarrow aw_n$$

Then in the new grammar these productions will be replaced by:

$$A \rightarrow a(A,a)$$
$$(A,a) \rightarrow w_1$$
$$(A,a) \rightarrow w_2$$
$$\vdots$$
$$(A,a) \rightarrow w_n$$

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

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 [K&H].

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 $k \leq k' + 1$. 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 machine is designed to accept sequences from the language followed by k-1 end markers.

It is important for later proofs to observe that the machine has no ϵ-moves and stops after reading in k-1 end markers. The pushdown control could, of course, be redesigned so as not to read beyond the first end marker, and the machine would terminate its recognition with k-2 ϵ-moves. When k=1, there is no need for the end marker.

The finite state control has enough memory to store an input string of length k-1 and to perform such obvious tasks as reading in the first k-1 inputs. After the (r+k-1)-th input symbol has been processed, the word stored in the finite state control is the string consisting of the (r+1)-th input through the (r+k-1)-th input.

Initially, the machine reads the first k-1 input symbols and stores them as the word in the finite control. The pushdown tape is initialized with the starting symbol S. The tape alphabet will be $N \cup T$.

If the machine has not stopped after r+k-1 inputs have been processed, the machine reads in the (r+k)-th input, uses the top tape symbol and the word w formed by the (r+1)-th through the (r+k)-th input symbols to manipulate the stack as described below, and stores the (r+k+2)-th through the (r+k)-th symbols in the control unit. The stack manipulation consists of two cases.

Case 1: If the top tape symbol is a nonterminal A, then there is at most one production p such that w is in $(L_p(A)R_k(A)\vdash^k)/k$. If there is no such production, the machine stops and rejects the sequence. If it has such a production, it is of the form $A \rightarrow a\gamma$ where γ is in $(N \cup T)^*$, and the machine replaces A by γ.

Case 2: If the top stack symbol is a terminal, then it is popped off if it matches the first symbol of w; otherwise the machine stops and the sequence is rejected.

When the machine reaches a configuration with no tape symbols, the machine stops and accepts the sequence if and only if the control word (i.e. inputs r+1 to r+k-1) is $\vdash^{k-1}$.

The machine operates in close correspondence to a leftmost derivation in the grammar. This relationship is described by the following:

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 M reaches a configuration with tape γ in $(N \cup T)^*$ and look-ahead word w of length k-1 after input sequence w_1w has been processed, then

1) $S \Rightarrow_L w_1\gamma$

2) For all w_3 in T^* such that $S \Rightarrow w_1w_3$ and $(w_3 \vdash^k)/k\text{-}1 = w$, w_3 satisfies the relation $\gamma \Rightarrow w_3$.

Proof: The proof is similar in spirit to the proof of Lemma 3 so we present the logic more briefly. 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 2). Assume that w_3 satisfies $S \Rightarrow w_1w_3$ and $w_3\vdash^k/k\text{-}1 = w$ but not $\gamma \not\Rightarrow w_3$. 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. Word

w_1' cannot have the form w_1x because the only such configuration in the left derivation of $w_1\gamma$ is $w_1\gamma$ itself and we are assuming w_3 cannot be derived from γ. Therefore, there is an input word x of length k such that w_1x is a prefix of both w_1w and $w_1w_3\vdash^k$. But since x has k symbols, the choice of production to apply at $w_1A\gamma'$ consistent with x 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 [L&S].

This lemma says in effect that the pushdown tape always has the necessary information to recognize any legitimate extensions. The case $w_3 = \epsilon$ implies $\gamma = \epsilon$ so the machine does recognize all words in the language. Since it obviously only accepts words in the language, we have proven:

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

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 $\{a^nb^n\} \cup \{a^nc^n\}$, which it will be shown has no LL(k) grammar, can be recognized by the pushdown machine described as follows. This machine operates on the basis of a word of length 2 and its operation is described by a set of rules of the form $(A,w) \to \gamma$, which mean that if A is the top stack symbol and w is in the stored input string, then A is replaced by γ and another input symbol is read in. Combinations of stack symbol and w not shown below result in rejection of the input string. The initial stack symbol is S.

$(S,aa) \to SA$

$(S,ab) \to A$

$(S,ac) \to A$

$(A,bb) \to \epsilon$

$(A,cc) \to \epsilon$

$(A,b\vdash) \to \epsilon$

$(A,c\vdash) \to \epsilon$

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 $k \geq 1$, the class of languages generated by LL(k) grammars is properly contained within the class generated by LL(k+1) grammars.

First, for each $k \geq 1$, consider the language $\{a^n(b^kd+b+cc)^n \mid n \geq 1\}$. Here "+" denotes the "or" operation. This language can be generated by the following LL(k) grammar with ϵ-rules.

$S \to aSA$

$S \to aA$

$A \to cc$

$A \to bB$

$B \to \epsilon$

$B \to b^{k-1}d$

However, this language cannot be generated by an LL(k) grammar with ϵ-rules.

Lemma 7: There exists no LL(k) grammar without ϵ-rules for the language $\{a^n(b^kd+b+cc)^n \mid n \geq 1\}$ where $k \geq 1$.

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 pushdown machine associated with this grammar. There must be an integer n_o such that the input sequence a^{n_o+k-1} causes the machine to have a pushdown tape with at least $2k-1$ symbols. (This is because M must distinguish all pairs a^{n_1} and a^{n_2} for $n_1 \neq n_2$). Thus, the resulting tape has the form $\gamma_1 Z \gamma_2$ where γ_1 and γ_2 in $(N \cup T)^*$ satisfy $|\gamma_1| \geq k-1$ and $|\gamma_2| \geq k-1$ and Z is an element of $N \cup T$.

By Lemma 6, $S \Rightarrow_L a^{n_o}\gamma_1 Z \gamma_2$. Also by Lemma 6,

$$\gamma_1 Z \gamma_2 \Rightarrow a^{k-1}b^{n_o+k-1} \text{ and } \gamma_1 Z \gamma_2 \Rightarrow a^{k-1}c^{2(n_o+k+1)}.$$

Since G has no ϵ-rules and since $|\gamma_1| \geq k-1$, there must exist four numbers n_1, n_2, n_3, and m such that

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

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

$$\gamma_2 \Rightarrow b^{n_3}$$

and $n_1+n_2+n_3 = n_o+k-1$

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

By Lemma 6, input sequence

$$a^{n_o+k-1}b^{n_1+n_2+k-1}$$

results in the tape γ_2 since

$S \Rightarrow_L a^{n_o+k-1}b^{n_1+n_2}\gamma_2$, $\gamma_2 \Rightarrow b^{n_3}$ and $b^{n_3}/k-1 = b^{k-1}$. Since furthermore $b^{k-1}db^{n_3}/k-1 = b^{k-1}$ and $a^{n_o+k-1}b^{n_1+n_2}b^{k-1}db^{n_3}$ 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

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

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 $k \geq 1$ 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 $k \geq 1$, the language

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

cannot be generated by an LL(k) grammar without ϵ-rules. The smallest class in the hierarchy is that of the simple languages. 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 observed by Dr. Kurki-Suomio.

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. $\tau(\gamma) = \min\{n \mid \text{there exists } x \text{ such that } \gamma \Rightarrow x \text{ and } n = |x|\}$. 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 $S(\gamma,w) = \{x \mid \gamma \Rightarrow x$ and $x\vdash^i \Rightarrow wy$ for some $i \geq 0$ and y 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 maximum thickness of the righthand sides of productions in P, w in $T^*\vdash^*$ is of length m, γ is in $(N \cup T)^*$, and $S(\gamma,w)$ is non-empty, then

$$\tau(\gamma) \leq t_w(\gamma) \leq \tau(\gamma) + m(t-1)$$

Proof: Clearly $\tau_w(\gamma) \geq \tau(\gamma)$. 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 righthand side of a production (whose thickness is at most t), thereby increasing the thickness of the intermediate string by t-1. Thus,

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

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 exist a function

$$f: N \rightarrow 2^{T^*\vdash^k/k}$$ such that for all w in $T^*\vdash^k/k$, w_1 in T^* and $A\gamma$ in N^* satisfying $S \Rightarrow_L w_1A\gamma$, $S(A\gamma,w)$ 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 $f((A,R)) = R\vdash^k/k$ 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 $A_a \rightarrow a$ for a in T and then replacing occurrences of a by A_a where required to obtain the form expressed by 1.

The canonical pushdown machine for a super strong LL(k) grammar will stop and reject if and only if it discovers a tape γ, look-ahead word w, and input a such that $S(wa,\gamma) = \varphi$.

To prove the theorem, we need only give an equivalence test for two super strong 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) occupies 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 $\tau(\nu)$ 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 righthand 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. From the fact that $\tau_w(\nu) = \tau_w(\mu)$, it follows from Lemma 8 that $|\tau(\nu) - \tau(\mu)| \leq 2(k-1)(t-1)$.

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 $\leq 2(k-1)(t-1)$.

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 [G&G].

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.

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 for $1 \leq i \leq n$ represent the canonical pushdown machines for these languages. We will call a machine configuration viable if there exists a sequence of inputs which causes the machine to enter an accepting configuration. To prove the theorem, we will derive an upper bound on the tape langths of the viable configurations that can occur when input sequences are applied to the starting configurations of the M_i. Thus the languages will be shown to be recognized by a finite set of configurations and hence be regular.

Suppose that input sequence w_1 in T* is applied to each machine M_i causing it to enter configuration C_i and let V be the subset of these configurations which are viable. Since a sequence of k end markers must be accepted by one of the C_i and since the canonical M_i do not make ϵ-moves and accept only with empty stacks, this C_i will be in V and will have tape length ℓ_1 which is less than or equal to k. Now suppose that we have found a j element set V_j which is a proper subset of V and which has configurations with tape lengths of maximum size ℓ_j. Let ℓ_{j+1} be the shortest sequence in $T^*\vdash^k$ which does not cause a configuration in V_j to enter an accepting configuration. (Such a sequence exists because V_j is a proper subset of V and the languages are disjoint). Some configuration C_i in $V-V_j$ must accept this shortest sequence and must therefore have tape of length less than or equal to ℓ_{j+1} and therefore $V_{j+1} = V_j \cup \{C_i\}$ is a j+1 element subset of V with tapes of length ℓ_{j+1} or less. We have defined by induction bounds ℓ_i and sets V_i for all i such that $1 \le i \le |V|$. Thus if the ℓ_i can be bounded independent of w_1, the tape lengths of all reachable viable configurations will be bounded and each machine will accept a regular set.

We have already observed that ℓ_1 has a bound independent of w_1; namely k. Suppose that a bound b_j has been found for all the possible ℓ_j resulting from all choices of 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 ℓ_{j+1} over the range of possible V_j. Thus bounds b_i are established by induction and the theorem proved.

Corollary 5: The complement of a non-regular LL(k) language is never LL(k).

Corollary 6: Theorem 9 generalizes to languages recognized by deterministic pushdown 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 [K&H] give examples of simple languages whose closure under most of these operations produces non-LL(k) languages.

Since the language $L_1 = \{a^m b^n \mid 1 < m < n\}$ is an LL(1) language which is not finite state, its complement, L_2, is not LL(k) by Corollary 5. The language $L_3 = \{a^n b^n + a^n c^n \mid n \ge 1\}$ was shown to be a non-LL(k) language in an earlier section. However, L_3 is the union of two LL(1) languages, $L_4 = \{a^n b^n \mid n \ge 1\}$ and $L_5 = a^n c^n \mid n \ge 1\}$. The languages $L_6 = \{a^n (b + b)^n \mid n \ge 1\}$ and $L_7 = \{a^n b^m \cup a^n c^m \mid n,m \ge 1\}$ are two LL(1) languages, whose intersection is L_3, which is not LL(k).

$L_8 = \{b^n a^n + c^n a^n \mid n \ge 1\}$ is an LL(1) language whose reversal is L_3, and is not LL(k).

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

The language $L_{10} = \{da^m b^{m+1} + ea^m c^{m+1} \mid m \ge 0\}$ 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] for the Turing machine is a string that indicates the current tape contents, internal state, and position of the head on the tape.

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(1) grammar G_1 with sentence symbol S_1 can be obtained whose sentences are of the form $p_o\ c\ p_1^r\ c\ p_2 \ldots c\ p_{2i-1}^r\ c\ p_{2i}\ c \ldots c\ p_{2n}$ where p_o 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.

Another LL(1) grammar G_2 with sentence symbol S_2 can be written such that its sentences are of the form $q_o\ c\ q_1^r\ c\ q_2 \ldots c\ q_{2i}\ c\ q_{2i+1}^r\ c \ldots c\ q_{2n}$ 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.

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

$$r_o\ c\ r_1^r\ c\ r_2\ c\ \dots\ c\ r_{2n}$$

is a prefix of strings in both $L(G_1)$ and $L(G_2)$ if and only if $r_o, r_1, r_2, \dots, r_n$ is the sequence of instantaneous descriptions produced by the Turing machine when it starts with p_o. Therefore, if the machine does not halt, then there are arbitrarily long sequences that are prefixes of sentences in $L(G_1)$ and $L(G_2)$, no choice can be made between productions $S_3 \to S_1$ and $S_3 \to S_2$ by looking ahead a finite amount, and G_3 is not an LL(k) grammar for any 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 input symbol. 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 decidable if there exists a k such that it is LL(k), given an LR(k) grammar, the problem is undecidable.

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 [L&S] 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 [L&S], 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 [K&H] for simple grammars is valid for LL(k) grammars.

However, given an arbitrary context free grammar, it is decidable [P&U] if there exists an LL(1) grammar without ϵ-rules that is structurally equivalent to the original one.

Acknowledgment

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

References

[Davis] - Davis, M., Computability and Unsolvability, McGraw-Hill, New York, 1958.

[G&G] - Ginsburg, S. and Greibach, S., "Deterministic Context-Free Languages," Information & Control, 9,6 (December 1966) 620-648.

[GR] - Greibach, S., "A New Normal-Form Theorem for Context-Free Phrase Structure Grammars," J. ACM, 12, 1 (Jan. 1965) 42-52.

[K&H] - Korenjak, A.J., and Hopcroft, J.E., "Simple Deterministic Languages," IEEE Conference Record of Seventh Annual Symposium on Switching and Automata Theory, IEEE Pub. No. 16-C-40, Oct. 1966, pp. 36-46.

[KN] - Knuth, D.E., "On the Translation Of Languages from Left to Right," Information & Control, 8,6

(December 1965) 607-639.

[L&S] - Lewis, P.M. II, and Stearns, R.E., "Syntax-directed Transductions," J. ACM 15, 3 (July 1968), 465-488.

[OETT]- Oettinger, A. Automatic Syntactic Analysis and the Pushdown Store, Jacobson, R. (Ed.), Structure of Language and Its Mathematical Concepts, Proc. 12th Symposium in Appl. Math., Amer. Math. Soc., Providence, R. I., 1961, pp. 104-129.

[P&U] - Paull, M.C., and Unger, S.H., "Structural Equivalence of LL-k Grammars," IEEE Conference Record of Ninth Annual Symposium on Switching and Automata Theory, IEEE Pub. No. 68-C-50-C, Oct. 1968, pp. 176-186.