KCS P524 524

PROCEEDINGS of the 1979 Conference on Information Sciences and Systems

Department of Electrical Engineering
The Johns Hopkins University
Baltimore, Maryland 21218

AN APPROACH TO THE ANALYSIS OF CONTEXT FREE LANGUAGES

Sudhir K. Arora, Lea Ginzberg and K.C. Smith
Department of Computer Science
University of Toronto
Toronto, Canada M5S 1A4

Abstract

This paper presents a fresh way to analyze context free languages. This leads to a more efficient algorithm to find semilinear set representation for a grammar. The new algorithm is by no means optimum and further work is needed to achieve this end. Several grammars have been tested on an implementation of the algorithm. Variations of the implementation can be directly applied to several other problems which are mentioned in this paper.

Introduction

This paper presents an algorithm to analyze context free languages and its computer implementation. While going through this work, it becomes increasingly clear that this way of analyzing context free languages can be applied directly to a wider set of problems.

Parikh showed [1] that every context free language (CFL) can be represented as a semilinear set of vectors in a $|V_T|$-Dimensional space where $|V_T|$ = No. of terminals in the language. However to obtain this semilinear set of vectors for any language one has to examine a grammar representing it in the following way.

(1) All possible trees in which the same variable may be repeated at most (t+2) times in a path where $t = |V_T| + |V_N|$ and $|V_N|$ = No. of variables and $|V_T|$ = No. of terminals. This gives the constant vectors of the semilinear set.

(2) All possible tree sections with a variable as the root and the same variable occurring only once in the result and no other variable in the result. Further any variable may repeat (t+2) times in any path where $t = |V_T| + |V_N|$.

It is obvious that as t increases this task becomes rapidly impossible. Our algorithm does this job as follows.

(1) To find the constant vectors, the number (t+2) is the upper bound on the number of times a variable may repeat in a path in any tree. However the algorithm does the job by examining a much lesser number of trees. In rare cases of course one has to go to the upper bound.

(2) To find the periods, the algorithm provides a more categoric result. It shows that the number of tree sections that need be examined is in fact independent of 't' and that any variable need occur at most twice in any path in any tree section that must be examined.

The algorithm has been implemented using ALGOL-W on IBM 370. The analyses and results for three grammars are presented. In the case of one grammar more details are presented and the algorithm is compared with Parikh's method [1]. Finally it is shown that this implementation in some ways is more general than [1].

Some Points

(1) A context free grammar (CFG), G is represented as, $G = (V_N, V_T, P, S)$ where V_N = set of variables, V_T = set of terminals, P = set of productions, S = start symbol.

(2) A derivation tree in G is any tree with the start symbol, S as the root and only terminals in the result.

(3) A tree in G is any tree with any variable as its root and both variables and terminals in its result.

(4) We do not distinguish between nodes in a tree and their labels. Both are referred to by the label.

(5) The occurrence of the same variable, B at different nodes in a tree is identified by B_1, B_2, etc.

(6) PERIOD & CONSTANT refer to a tree or the result of that tree, while 'period' and 'constant' refer to the Parikh vectors corresponding to PERIOD & CONSTANT.

(7) The Parikh vector of a variable is a zero vector, i.e., $P(A) = (0, 0 \ldots 0)$.

(8) L(G) is the language generated by the grammar G.

(9) The implementation can handle grammars with up to 26 variables (A to Z) and up to 10 terminals (0 to 9).

(10) The null word, ε, is treated like any other terminal (represented by 0) and is eliminated at the end of the program from the set of PERIODS.

(11) By reduced grammar, we mean a grammar from which all variables which cannot be reached from S or which do not terminate have been eliminated.

Definitions

N_1-Variables. All variables that do not repeat along any path in any derivation tree.

N_2 & N_3-Variables. All variables, A, s.t. if we generate all derivation trees in grammar (V_N, V_T, P, A) in which no variable is allowed to repeat along any path except under the following circumstances.

a) The variable A repeats once in any path.

b) Other variables in path AA, may occur twice in any path, one of which is in the path AA itself. Then if A repeats along only one path in any tree, it is an N_2-variable. If A repeats along more than one path in any tree, it is an N_3-variable. Any other variables may occur only once in any path.

PERIOD. Take the collection of derivation trees generated in N_2 & N_3 definition. In each tree cut the appropriate subtree to expose a single occurrence of A (A is the root also) at a time. The remaining tree part or its result containing all terminals and a single A is called a PERIOD or an A-PERIOD.

CROWNS. Take all trees with S as the root in which no variable repeats along a path and the result contains only terminals and at least one variable from ($N_2 \cup N_3$). These are called CROWNS. (Note if S belongs to ($N_2 \cup N_3$) then the tree with only one node, i.e. S is also a CROWN).

TERMINATING TREES. Take all trees with A as the root in which no variable repeats along a path and the result contains only terminals and A is any element of ($N_2 \cup N_3$). These are TERMINATING TREES.

CONSTANTS are derivation trees in the CFG which are obtained as follows:

(a) Take all derivation trees in which only N_1 variables occur. These are CONSTANTS.

(b) To every CROWN attach TERMINATING TREES to get all possible derivation trees. These are also CONSTANTS.

(c-i) Take the CONSTANTS obtained in (b). One at a time. Find all PERIODS which have their root variable occurring in this CONSTANT and also contain at least one variable in ($N_2 \cup N_3$) which does not occur in the CONSTANT.

(c-ii) Divide these PERIODS into sets $s_1, s_2, \ldots s_r$ such that every member of set s_i, $1 \leq i \leq r$ has the same new variables belonging to ($N_2 \cup N_3$) and not occurring in the CONSTANT.

Form sets $S_1, S_2 \ldots S_r$ such that $S_1 = \bigcup_{i=1}^{r} s_i$,

$S_2 = \bigcup_{i=1}^{r} \bigcup_{j=2}^{r} (s_i \times s_j)$ for $i < j$,

$S_3 = \bigcup_{i=1}^{r} \bigcup_{j=2}^{r} \bigcup_{k=3}^{r} (s_i \times s_j \times s_k)$ for $i < j < k$

and so on. Define the set $S = \bigcup_{i=1}^{r} S_i$. Each element of S is a set of PERIODS ranging from 1 to r in number. INSERT (defined later) these PERIODS in the CONSTANT, one element of S at a time to get a new CONSTANT each time.

(d) Take the CONSTANTS generated in step (c); one at a time. Find all PERIODS which have at least one variable in ($N_2 \cup N_3$) not occurring in the CONSTANT and their root variable is one of the new variables in ($N_2 \cup N_3$) introduced into the CONSTANT in step (c). Repeat step (c-ii) for these PERIODS to get new CONSTANTS.

(e) Carry on this process till no new CONSTANTS can be got.

Operations

INSERTION Take a path in a derivation tree in which some variable, B occurs and B belongs to ($N_2 \cup N_3$). We cut the subtree at B, attach a B-PERIOD at the exposed B and then reattach the cut subtree to the only B occurring in the result of the B-PERIOD. The B-PERIOD is said to be INSERTED at the B occurring in the original derivation tree.

DISSECTION is an operation on a derivation tree in a grammer. It is defined as follows: If we have a tree in a CFG, G s.t. a variable B repeats in a path, let the two occurrences of B be called B_1 and B_2 where B_1 is closer to the root. We cut the tree at B_1 and B_2, remove the tree part between B_1 and B_2 and attach the subtree at B_2 to the node at B_1. This is defined as DISSECTION between the nodes B_1 and B_2.

TRANSPLANTATION Take any derivation tree in which a variable B occurs more than once in a path. DISSECT the tree part between any two occurrences of B in the same path and INSERT it at another occurrences of B which is different from the location from where the tree part has been DISSECTED. This operation is called TRANSPLANTATION.

LEMMA I: - When the operations of INSERTION, DISSECTION or TRANSPLANTATION are done on a derivation tree, the resulting tree is another derivation tree in the same grammar.

PROOF: - This is obvious from the definitions.

LEMMA II: - The Parikh Mapping of a string W, Derived in a CFG, G is unaltered under the operation of TRANSPLANTATION.

PROOF: - Consider the derivation tree of W and let B be a variable in it which repeats along a path. In addition B occurs elsewhere in the tree also. Let these occurrences of B have labels B_1, B_2, B_3 as shown in Fig. 1. Let the subtrees at B_1, B_2 and B_3 derive words, W_1, W_2, and W_3 respectively. Then

$$W = xW_1yW_3z$$

$$P(W) = P(x) + P(W_1) + P(y) + P(W_3) + P(z)$$

Also

$$W_1 = x'W_2y'$$

$$P(W_1) = P(x') + P(W_2) + P(y')$$

$$P(W) = P(x) + P(x') + P(W_2) + P(y') + P(y) + P(W_3) + P(z)$$

Now we carry out TRANSPLANTATION as follows: Remove the tree part between B_1 and B_2 and INSERT it at the B_3 location. The tree after TRANSPLANTATION looks as shown in Fig. 2 which generates a new word W'. We show that $P(W') = P(W)$

$$W' = xW_2yW_1'z = xW_2yx'W_3y'z$$

Hence,

$$P(W') = P(x) + P(W_2) + P(y) + P(x') + P(W_3) + P(y') + P(z) = P(W)$$

LEMMA III: - Let T_1 be a derivation tree with the result W_1 and let T_2 be the derivation tree after INSERTION of a period in T_1 and let the result of T_2 be W_2. Let the PERIOD inserted be an A-PERIOD, p^A. Then we can write $P(W_2) = P(W_1) + P(p^A)$.

PROOF: - Consider the derivation tree of W_1 shown in Fig. 3-2. A is the variable at which the A-PERIOD, p^A, shown in Fig. 3-3, is to be INSERTED. The result of the A-subtree in Fig. 3-2 is W_A. Hence $W_1 = xW_Ay$ where x and y are strings of terminals, $P(W_1) = P(x) + P(W_A) + P(y)$. Consider the A-PERIOD shown in Fig. 3-3. The result contains only one A, and x_1 is the string of terminals to the left of A and y_1 is the string of terminals to the right of A. Hence

$$p^A = x_1Ay_1 \text{ and } P(p^A) = P(x_1) + P(A) + P(y_1) = P(x_1) + P(y_1)$$

Now consider T_2 shown in Fig. 3-1. From the figure, it is obvious,

$$W_2 = xx_1W_Ay_1y$$

$$P(W_2) = P(x) + P(x_1) + P(W_A) + P(y_1) + P(y) = P(W_1) + P(p^A)$$

LEMMA IV: - Let T_1 be a derivation tree with the result W_1 and let T_2 be a derivation tree after DISSECTION of T_1 at nodes B_1 and B_2 and let the result of T_2 be W_2. Let the tree part between B_1 and B_2 be denoted by q^B. Then we can write $P(W_2) = P(W_1) - P(q^B)$.

PROOF: - Consider the derivation tree of W_1 in Fig. 4-1. B_1 and B_2 occur in a path. The subtree at B_2 derives a word W_B. The subtree at B_1 derives a word xW_By. Hence,

$W_1 = x_1xW_Byy_1$ where x_1 and y_1 are strings of terminals

$$P(W_1) = P(x_1) + P(x) + P(W_B) + P(y) + P(y_1)$$

Consider the derivation tree of W_2 and the tree part q^B shown in Figs. 4-2 and 4.3.

$$W_2 = x_1WBy_1$$

$$q^B = xBy$$

$$\therefore \quad P(q^B) = P(x) + P(y)$$

and $P(W_2) = P(x_1) + P(W_B) + P(y_1) = P(W_1) - P(q^B)$.

THE ALGORITHM: The algorithm for finding the Parikh Mapping of any CFG is given.

(1) Identify N_1, N_2 and N_3 variables.

(2) Enumerate all trees and find the PERIODS as outlined in this paper in the definition of a PERIOD.

(3) Enumerate all the CONSTANTS as outlined in the definition of a CONSTANT. This is done after finding the CROWNS and the TERMINATING TREES.

(4) Find the Parikh Mapping of the CONSTANTS and the PERIODS to obtain the semilinear set, X.

(5) We define X as a set of points in a $|V_T|$-dimensional space $N_1 \times N_2 \times \ldots \times N_{|V_T|}$ where $|V_T|$ = No. of terminals.

N_i = set of positive integers where $1 \le i \le |V_T|$

$$X = \bigcup_{j=1}^{n} [P(W_j) + \sum_{A \text{ in } NW_j} \sum_{i=1}^{r(A)} k_i \times P(p_i^A)] \quad (1)$$

where W_j = one of n possible CONSTANTS,

$P(W_j)$ = Parikh mapping of W_j,

NW_j = set of variables in $(N_2 \cup N_3)$ which occur in the derivation tree of W_j.

p_i^A = one of r(A) possible A-PERIODS

$P(p_i^A)$ = Parikh mapping of p_i^A

k_i = An element of set N_i

X = A semilinear set with $P(W_j)$ as constants and $P(p_i^A)$ as periods.

PROOF: - We prove for any CFG, G.

(1) If W belongs to L(G) then P(W) belongs to X, i.e. $P(L) \leq X$.

(2) If x belongs to X then there exists a W belonging to L(G) s.t. P(W) = x, i.e. $X \leq P(L)$.

PART 1. Let G be a CFG and let W be a string of terminals s.t., W belongs to L(G).

We show that P(W) can be put in the form of X, i.e. P(W) = x where x belongs to X. Consider the derivation tree of 'W'. We show that it is pospossible to apply DISSECTION operation to this derivation tree repeatedly till we are left with a derivation tree which is a CONSTANT, W_j' and a number of tree parts which are PERIODS, $^jp_i^A$. Then since the Parikh mapping of 'W' is unaltered under TRANSPLANTATION it is obvious that as long as the derivation tree for the constant, W_j', has all the variables, A belonging to $(N_2 \cup N_3)^j$ that occur in the derivation tree of 'W' we will always be able to put back together this CONSTANT W_j' and the PERIODS p_i^A by using INSERTION to get another word W_1 whose Parikh mapping is the same as 'W', i.e., $P(W) = P(W_j') + \sum_{A \text{ in } N'} \sum_{i=1}^{r(A)} k_i P(p_i^A)$.

N' = All variables A s.t. at least one A-PERIOD has been DISSECTED from the derivation tree of 'W'. (2)

Now the CONSTANT W_j' that we obtain from the above described procedure may not always contain in its derivation tree all the variables, A belonging to $(N_2 \cup N_3)$ that occur in the derivation tree of 'W'. We show that it is possible to obtain from W_j' another CONSTANT W_j such that the above condition is satisfied. Hence

$$P(W) = P(W_j) + \sum_{A \text{ in } NW_j} \sum_{i=1}^{r(A)} k_i P(p_i^A) = x \text{ belonging to } X$$

We proceed as follows. Start with the root, S in the derivation tree of 'W'. We apply the following procedure only to those subtrees, the root variable of which repeats in some path (possibly more than one) in that subtree. It may be that S itself repeats in some path - then we apply the procedure to the whole derivation tree. So without loss of generality, we assume that S does not repeat.

(1) Proceed from S along each path till we come to the first variable that repeats in its own subtree or a terminal as shown in Fig. 5. In this figure, C and D do not repeat while A_1 and B repeat in their own subtree. The subtree at B has not been shown. Lower case alphabets are terminals.

(2) The subtrees at A_1 and B can be considered the same way and DISSECTED the same way. So we consider one of them, say, A_1-subtree. We follow the convention that A_1, A_2 - are different occurrences of A in the derivation tree. Let the root of this subtree be A_1.

(3) From A_1 trace a path to the next occurrence of A in the subtree. (Any one occurrence if there are more than one) and call it A_2, as shown in Fig. 5. Assume some variable, B repeats in this path, say, B_1 and B_2. We can DISSECT between the nodes B_1 and B_2. By repeated application of DISSECT operation we arrive at a tree in which no variable repeats in the path A_1 and A_2 as shown by Fig. 6. Further we may have several tree parts like B_1 B_2 which can be treated like A_1 A_2 separately. The only difference is that A_2 has a subtree attached to it while B_2 is an exposed node.

(4) The tree part A_1 A_2 may have other paths starting from variables in path A_1 A_2 (excluding A_2), say, from A_1 and C as shown in Fig. 6-4. We follow these paths and their branches till we come to terminals or variables, that repeat in their own subtree. In the Fig. 6-4, B_3 and B_4 do not repeat in their own subtrees while C_1, D_1, A_3, E_1 and F_1 repeat in their subtrees and a and b are terminals. Hence we have a tree part within this A_1 subtree s.t. the result of this tree part (call it s-tree) contains either terminals or variables that repeat in their own subtree. This is represented as shown in Fig. 7-1 where the variables that repeat in their own subtree (excluding A_2) are shown on the periphery and the original A_1 A_2 path is diagramatically shown as the symbol '↯'.

(5) Each of these variables that repeat in their own subtree can be treated just like the variable A_1 and each of them will give rise to similar s-trees within the A_1 subtree. This is as shown in Fig. 7-2.

(6) Now we consider variables A_4, A_2, F_2, E_2, D_2 and C_2. If they repeat in their own subtree they will give rise to other s-trees. If they do not then we follow all paths starting from them till we reach variables that do repeat in their own subtrees or terminals. Thus we will get a fresh crop of s-trees within the A_1 subtree. This is as shown in Fig. 7-3. We continue this process till all possible s-trees have been identified, i.e. in the A_4, F_4, D_4, C_4, F_2, E_2, D_2 and C_2 subtrees no variable repeats. These subtrees are by definition TERMINATING TREES.

(7) Now the s-trees have the following properties;

(a) In s-tree A_1 A_2, no variable repeats in path A_1 A_2.

(b) Any variable occurs at most twice in a path, one occurrence of which is in path A_1 A_2. This can be seen from Fig. 6-4.

In the path A_1C no variable repeats and in paths starting at C to b, E_1 and F_1 no variable repeats. So a variable can occur at most once in A_1C and once in any path starting from C. This argument can be extended to all paths.

(8) There are a finite number of s-trees in the A_1 subtree. We start with those s-trees from which no other s-tree originates, say F_3 and F_4 s-tree in Fig. 7-3. (This is always possible because of finite number of s-trees). We DISSECT at nodes F_3 F_4. The tree part F_3 F_4 is a PERIOD. We DISSECT all such periods.

(9) In general we consider the s-tree A_1 A_2 after all the periods have been DISSECTED. It looks as shown in Fig. 7-4. It has TERMINATING TREES attached to variables at its periphery and possibly some extra variables such as F_3, E_3, D_3 and C_3 at A_2. To those extra variables TERMINATING TREES are attached. We can show the following:

(a) The subtree at A_2 in Fig. 7-4 is also a TERMINATING TREE. This is because F_3, E_3, D_3, C_3 and A_2 do not repeat in their respective subtrees. (Otherwise s-trees could be formed). Also F_3, D_3 and C_3 subtrees are TERMINATING TREES - hence along any path no variable repeats in them. So in A_2 subtree no variable repeats along any path which is the definition of a TERMINATING TREE.

(b) If we DISSECT A_1 A_2 then A_1 A_2 tree part is a PERIOD. This follows from the fact that no variable in paths A_1 A_3, A_1 C_1, A_1 D_1, A_1E_1 and A_1 F_1 can repeat in A_3, C_1, D_1, E_1, F_1 terminating trees for otherwise s-trees could be formed. So the A_1 A_2 tree part still satisfies the conditions in step (7) and hence it is a PERIOD.

(10) Similarly we can reduce the tree parts like B_1 B_2, removed in steps (3), (5) and (6), to PERIODS.

(11) Similarly all other subtrees of the derivation tree of 'W'; can be treated like the A_1 subtree. The end result is a set of period p_i^A and a derivation tree as shown in Fig. 8 where A_1 and B have TERMINATING TREES attached to them. This derivation tree has no variable repeating in any path (otherwise an s-tree could be formed). If we remove the TERMINATING TREES at A_1 and B the remaining tree part is a CROWN. Hence the derivation tree derives a CONSTANT, W_j'. So we can express the Parikh mapping of 'W' in the form shown in equation (2).

Now we show how to obtain, W_j from W_j'. We do this by the following procedure.

(1) Consider the derivation tree of W_j'. No variable repeats in any path in this tree. Hence if we cut the tree at any set of variables belonging to ($N_2 \cup N_3$) and remove the subtrees the remaining portion is a CROWN. List out all the variables belonging to ($N_2 \cup N_3$) and occurring in the derivation tree of W_j'. Let them be (A_1, B_1, ...).

(2) Consider all the A_1-PERIODS, B_1-PERIODS,... and separate out those PERIODS that have at least one variable occurring in them which belongs to ($N_2 \cup N_3$) and does not occur in the derivation tree of W_j'. Divide these PERIODS into sets $s_1, s_2, \ldots s_r$ such that every member of set s_i, $1 \leq i \leq r$ has the same new variables belonging to ($N_2 \cup N_3$) and not occurring in the derivation tree of W_j'. Form the set $S_p = (s_1 \times s_2 \times s_r)$. Each member of S_p is a set of PERIODS which collectively contain all the above mentioned new variables. Also each member of S_p contains the same periods as some member of $\bigcup_{i=1}^{r} S_i$ defined in the definition of a CONSTANT. INSERT all the PERIODS in one element of S_p into the derivation tree of W_j'. Then by the definition of a CONSTANT the new derivation tree W_j'' is either a CONSTANT or its Parikh mapping is the same as of a CONSTANT. Let the new variables introduced in W_j'' be (P_1,Q_1...).

(3) Repeat step (2) for P_1-PERIODS, Q_1-PERIODS, ... to get a new CONSTANT.

(4) Repeat the procedure in steps (2) and (3) as many times as necessary till no new variables can be introduced to the CONSTANT. Call this final CONSTANT, W_j.

We can show that all variables belonging to ($N_2 \cup N_3$) and occurring in the derivation tree of W, also occur in the CONSTANT W_j. Suppose there is a variable, A which belongs to ($N_2 \cup N_3$) and occurs in the derivation tree of W but not in the derivation tree of W_j. Locate a PERIOD in which A occurs. (Always possible because A is not in W_j; so it must be in one of the PERIODS). If the root of this PERIOD (say, B) occurs in W_j, then obviously it should have normally been covered by the step (4) in the above procedure. Now A may be the root of a Period and A does not occur in W_j. Then find a PERIOD which contains A and has a different variable (say, C) as its root. Check if C occurs in W_j. If not, find a PERIOD in which C occurs and a different variable (say, D) as its root. Repeat the process till we find a PERIOD whose root (say, D) occurs in W_j. (This is always possible because all these variables and PERIODS were present in the original derivation tree of W). Now this D-PERIOD introduces a new variable to W_j. So it should have been considered in step (4) of the above procedure. So C does occur in W_j. But C-PERIOD contains A which is not in W_j. So it should have been considered in step (4). So A occurs in W_j. Hence all variables belonging to $(N_2 \cup N_3)$ and occurring in the derivation tree of W also occur in the constant, W_j. Hence we can express the Parikh mapping of W in the form shown in equation (1). So if W belongs to L(G) then P(W) = x belonging to X, i.e. $P(L) \leq X$.

PART 2. We show that for every x in X there exists a W belonging to L(G) s.t. P(W) = x.

Let $x = P(W_j) + k_1P(p_1^A) + k_2P(p_2^B) + \ldots$

Take the CONSTANT W_j and INSERT PERIOD p_1^A in it to get another word W_1.

Now $P(W_1) = P(W_j) + P(p_1^A)$ --- Lemma III.

Again INSERT p_1^A in the derivation tree of W_1 to get W_2.

$$P(W_2) = P(W_j) + 2P(p_1^A) \text{ --- Lemma III.}$$

Repeat the process k_1 times for p_1^A then k_2 times for p_2^B and so on to get a word W, s.t.

$$P(W) = P(W_j) + k_1P(p_1^A) + k_2P(p_2^B) + \ldots = x.$$

Since W has a derivation tree in the grammar G; W is in L(G). Hence,

$$X \leq P(L).$$

From part (1) and part (2) of the proof X = P(L), i.e. the set X obtained by the algorithm represents the Parikh mapping of the CFL.

The Results

Test runs for two grammars are presented. The analysis part of the test run shows the extent to which this implementation is unoptimized. In the case of example 1, we compare our algorithm to Parikh's method [1]. This comparison is presented in Note 1 while Note 2 points out some ways in which our algorithm is more general than Parikh's method.

Note 1: In the example 1, using Parikh's method we will have to examine Z_1 trees for CONSTANTS and Z_2 tree sections for PERIODS where,

$$Z_1 > 2 + (2^3) + (2^3)^2 + (2^3)^3 + (2^3)^4 + (2^3)^5 + (2^3)^6 + (2^3)^7$$

$$Z_1 > Z_1 + 2 + 2^2 + 2^3 + 2^4 + 2^5 + 2^6 + 2^7 + 2 + 2^2 + 2^3 + 2^4 + 2^5 + 2^6 + 2^7 .$$

These bounds are obtained as follows. Starting with S, we can generate two derivation trees in which A occurs only once in any path. Taking each of these two trees we can generate another tree in which A occurs twice in some path and there are a maximum of three such paths in the tree. So we can have a minimum of 2^3 derivation trees in which A occurs twice in some path.

Repeating this process we will have a minimum of $(2^3)^2$ trees in which A occurs thrice in any path and so on. Hence the lower bound for Z_1. Similarly for Z_2, we find the lower bound for the number of trees in which A, B or C is the root and A, B or C respectively is repeated in the paths of the tree.

By our algorithm, for example 1,

Total number of CONSTANTS generated = 72

Total number of PERIODS generated = 78

In addition in our algorithm we have,

Number of CROWNS generated = 2

Number of TERMINATING TREES generated = 8

Thus our algorithm improves the efficiency by several orders of magnitude. However an optimized algorithm could possibly generate only 7 CONSTANTS and 10 PERIODS for example 1 which is the minimum number.

Note 2: Our algorithm is in some sense more general than Parikh's method. i) It can handle rules of the form $A \rightarrow A$. ii) It can handle the null word.

In some cases, our algorithm comes close to optimum as shown by example 2.

Conclusion

A fresh way to analyze context free grammars have been presented. This leads to a more efficient algorithm to find a semilinear set representation of any context free grammar. The algorithm, although it improves on the earlier method [1] by several orders of magnitude, is, however, not optimized as is seen by the results of the implementation. This approach can be applied to the question of ambiguity in context free grammars. Many solvable and unsolvable problems about this area are presented in [2,3,4].

This approach and in particular, variations of the implementation can be used as algorithms for some of these problems.

References

[1] Parikh, R.J., Language Generating Devices, M.I.T. Res. Lab., Electron Quart. Prog., Dept. 60, 1961, pp. 199-212.

[2] Hopcroft, J.E., Ullman, J.D., Formal Languages and Their Relation to Automata, Addison-Wesley Publishing Co., 1969.

[3] Chomsky, N., Schutzenberger, M.P., The Algebraic Theory of Context Free Languages, Computer Progrmaming and Formal Systems, North Holland, Amsterdam, 1963, pp. 118-161.

[4] Ginsburg, S., Ullman, J., Ambiguity in Context Free Languages, JACM, 13:1, 1966, pp. 62-88.

FIG 1

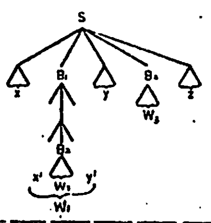


FIG 2

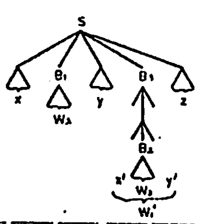


FIG 3

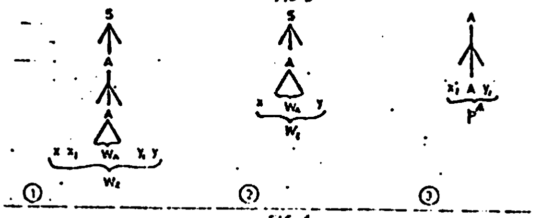


FIG 4

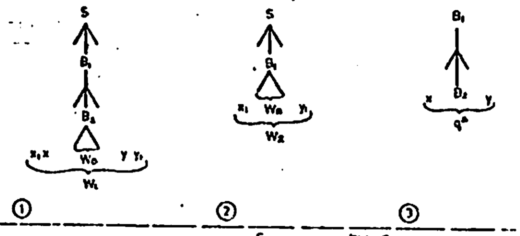


FIG 5

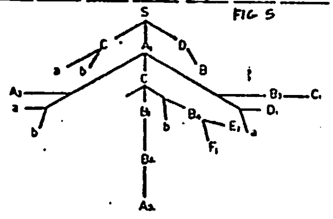


FIG 6

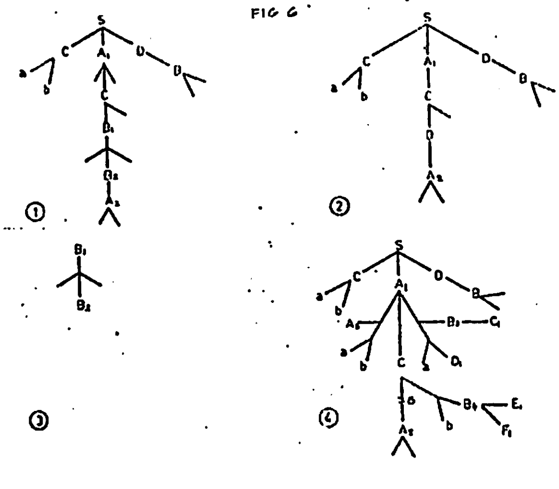


FIG 7

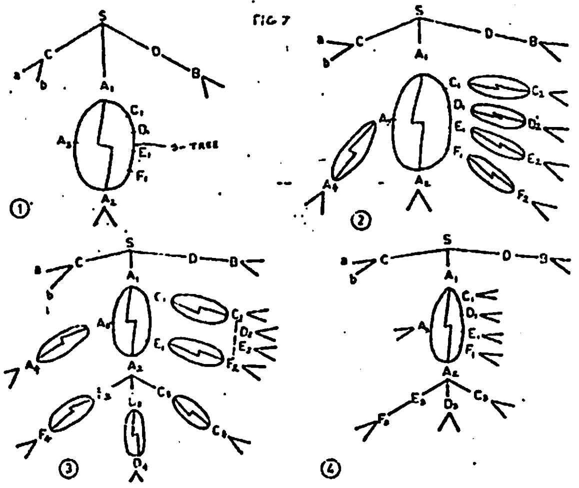
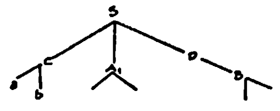


FIG 8



Example 1

```
GIVEN GRAMMAR
-------------
A -> 10C2
A -> 1
B -> 11A22
B -> 2
C -> 1A21A2
C -> 12
S -> 1A2

REDUCED GRAMMAR
---------------
A -> 10C2
A -> 1
B -> 11A22
B -> 2
C -> 1A21A2
C -> 12
S -> 1A2


SEMILINEAR SETS
***************
ORDER OF TERMINALS  =   (1,2)
CONSTANT           PERIODS
--------           -------
(2,1)              (4,4) (5,7) (7,5) (9,11)
(11,12)            (4,4) (5,7) (7,5) (9,11) (11,9)
(10,9)             (4,4) (5,7) (7,5) (9,11) (11,9)
(9,6)              (4,4) (5,7) (7,5) (9,11) (11,9)
(6,5)              (4,4) (5,7) (7,5) (9,11) (11,9)
(7,9)              (4,4) (5,7) (7,5) (9,11) (11,9)
(3,4)              (4,4) (5,7) (7,5) (9,11) (11,9)


A N A L Y S I S
---------------
MIN NO. OF A PERIODS   =   4
MIN NO. OF B PERIODS   =   2
MIN NO. OF C PERIODS   =   4
MIN NO. OF S PERIODS   =   0

MINIMUM TOTAL PERIODS  =  10
TOTAL PERIODS GENERATED=  78


MIN NO. OF CONSTANTS                  =   7
TOTAL NO. OF CONSTANTS GENERATED      =  72

MIN NO. OF SEMILINEAR SETS            =   7
TOTAL SEMILINEAR SETS GENERATED       =  72

NO. OF CHAINS GENERATED               =   2
NO. OF TERMINATING TREES GENERATED=   8

001.10 SECONDS IN EXECUTION
```

Example 2

```
GIVEN GRAMMAR
-------------
A -> 12
A -> 1A2
B -> 2
B -> 11B
C -> D
C -> 12
C -> 11C22
D -> C
S -> 1C2
S -> 1AB2

REDUCED GRAMMAR
---------------
A -> 12
A -> 1A2
B -> 2
B -> 11B
C -> D
C -> 12
C -> 11C22
D -> C
S -> 1C2
S -> 1AB2



SEMILINEAR SETS
***************
ORDER OF TERMINALS  =   (1,2)
CONSTANT           PERIODS
--------           -------
(2,2)              (2,2)
(2,3)              (1,1) (2,0)


A N A L Y S I S
---------------
MIN NO. OF A PERIODS    =    1
MIN NO. OF B PERIODS    =    1
MIN NO. OF C PERIODS    =    1
MIN NO. OF D PERIODS    =    0
MIN NO. OF S PERIODS    =    0

MINIMUM TOTAL PERIODS   =    3
TOTAL PERIODS GENERATED=     3


MIN NO. OF CONSTANTS                      =    2
TOTAL NO. OF CONSTANTS GENERATED          =    5

MIN NO. OF SEMILINEAR SETS                =    2
TOTAL SEMILINEAR SETS GENERATED           =    5

NO. OF CROWNS GENERATED                   =    4
NO. OF TERMINATING TREES GENERATED=            6

000.14 SECONDS IN EXECUTION
```

Example 3

```
GIVEN GRAMMAR
-------------
A -> 1S3
B -> 0
B -> 2B33
C -> S
C -> 12
C -> C
D -> 3
D -> 3F
D -> 1D2D2
F -> 12F
S -> 1D3
S -> 1AB2C3
S -> 0

REDUCED GRAMMAR
---------------
A -> 1S3
B -> 0
B -> 2B33
C -> S
C -> 12
C -> C
D -> 3
D -> 1D2D2
S -> 1D3
S -> 1AB2C3
S -> 0



SEMILINEAR SETS
***************
ORDER OF TERMINALS  =   (1,2,3)
CONSTANT           PERIODS
--------           -------
(0,0,0)            (2,1,2) (3,1,4) (3,2,2)
(3,1,4)            (0,1,2) (1,2,2) (2,1,2) (3,1,4) (3,2,2)
(5,2,6)            (0,1,2) (1,2,2) (2,1,2) (3,1,4) (3,2,2)
(6,3,6)            (0,1,2) (1,2,2) (2,1,2) (3,1,4) (3,2,2)
(2,1,2)            (0,1,2) (2,1,2) (3,1,4) (3,2,2)
(3,2,2)            (0,1,2) (2,1,2) (3,1,4) (3,2,2)
(1,0,2)            (1,2,2) (2,1,2) (3,1,4) (3,2,2)
(4,1,6)            (0,1,2) (1,2,2) (2,1,2) (3,1,4) (3,2,2)
(4,2,4)            (0,1,2) (1,2,2) (2,1,2) (3,1,4) (3,2,2)
A N A L Y S I S
---------------
MIN NO. OF A PERIODS    =    3
MIN NO. OF B PERIODS    =    1
MIN NO. OF C PERIODS    =    2
MIN NO. OF D PERIODS    =    1
MIN NO. OF S PERIODS    =    3

MINIMUM TOTAL PERIODS   =   10
TOTAL PERIODS GENERATED=    26


MIN NO. OF CONSTANTS                      =    9
TOTAL NO. OF CONSTANTS GENERATED          =  152

MIN NO. OF SEMILINEAR SETS                =    9
TOTAL SEMILINEAR SETS GENERATED           =  152

NO. OF CROWNS GENERATED                   =    2
NO. OF TERMINATING TREES GENERATED=            9

006.23 SECONDS IN EXECUTION
```