OREGON STATE UNIVERSITY COMPUTER SCIENCE DEPARTMENT

Context-Free Grammars with Graph Control

CURTIS R. COOK

Department of Computer Science
Oregon State University

78-20-2

CONTEXT-FREE GRAMMARS WITH GRAPH CONTROL

Curtis R. Cook
Oregon State University

Abstract

Context-free grammars with graph control provide a general framework for the various types of context-free grammars with regulated rewriting. The vertices or edges of the directed graph are labeled with the productions of the grammar. The only strings in the language generated by the grammar are those whose derivations correspond to labeled paths in the graph. Inclusion relations among the various classes of context-free grammars with regulated rewriting such as programmed grammars (without failure fields), matrix grammars, periodically time-variant grammars, state grammars, and grammars with regular control are easily obtained and the graph provides insight into the nature of the restriction. Adding negative context to context-free grammars with graph control, we obtain a class of grammars equivalent to the context-free programmed grammars.

1. INTRODUCTION

Various types of context-free grammars with regulated rewriting have been developed with the goal of generating the family or a large subfamily of the context-sensitive languages. Some of these include programmed grammars [5], matrix grammar [1], periodically time-varying grammars [7], state grammars [2] and grammars with control sets [8]. Typically the regulated rewriting takes the form of limiting the set of production candidates at each step in a derivation. All of these context-free grammars with regulated rewriting given above have been shown to be equivalent.

Rozenberg and Salomaa [6] introduced graph control over the productions as a general framework for the various types of context-free grammars with regulated rewriting. The nodes of the graph are labeled with sets of productions. The only strings in the language generated by the grammar are those whose derivations correspond to labeled paths in the graph.

In Section 2 of this paper we show that labeling the arcs instead of the nodes, specifying a starting node, specifying a set of end nodes for each derivation path in the graph, labeling with a single production instead of a set of productions, or any combination of these modifications does not increase the generating capacity of a context-free grammar with graph control.

In Section 3 we show that the graph control makes the inclusion relations between the various types of regulated rewriting obvious. Also the structure of the graph provides insight into the nature of the restrictions and relations between the various restrictions.

We extend grammars with graph control to include negative context in Section 4. That is, the nodes are labeled with sets of symbols and the arcs with productions. A certain production is applicable to a string at a node if no symbol in the node label appears in the string and the production appears as a label on an arc emanating from the node. We show that the negative context

grammars with graph control are equivalent to programmed grammars with both success and failure fields.

2. DEFINITIONS AND EQUIVALENCE OF MODIFICATIONS

It is assumed that the reader is familiar with the standard notation and results of formal language theory [8] and basic graph theory [3].

A context-free grammar (CFG) is an ordered quadruple $G = (N,T,P,S)$ where

- N is a finite nonempty set of nonterminal symbols or variables,
- T is a finite nonempty set of terminal symbols,
- S is in N and is called the starting symbol, and
- P is a finite set of productions or rewriting rules of the form $A \to \alpha$, where A is N and α is a string over $N \cup T$.

We say that α directly generates β, denoted $\alpha \Rightarrow \beta$, if $\alpha = wAy$, $\beta = wxy$ and the rewriting rule $A \to x$ is in P. Let $\overset{*}{\Rightarrow}$ be the reflexive, transitive closure of $\Rightarrow$. The language generated by the CFG $G = (N,T,P,S)$ is the set $L(G) = \{w : S \overset{*}{\Rightarrow} w \text{ and } w \text{ in } T^*\}$.

A context-free grammar with graph control is an ordered pair (G,H) where

- $G = (N,T,P,S)$ is a context-free grammar, and
- H = is a directed graph whose nodes are labeled from Lab (P), the set of labels of productions in P.

Note: that there may be several copies of the same production with several labels.

We say that (α, i) directly generates (β, j), denoted $(\alpha, i) \Rightarrow (\beta, j)$, if and only if both of the following conditions are satisfied:

1. For some w,x,y and A, $\alpha = wAy$, $\beta = wxy$ and the production $A \to x$ has label i;

2. There is an arc from node i to node j in H.

Let $\Rightarrow^*$ denote the reflexive, transitive closure of $\Rightarrow$.

The language generated by (G,H), denoted L(G,H), is the set $\{w : (S,j) \Rightarrow^* (w,i)$ for some i and j and w in $T^*\}$.

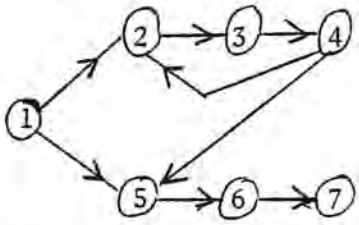
An example of a context-free grammar with graph control is given below. The language generated is not context-free.

Example 1.

G = ({S,A,B,C}, {1,2,3}, P, S)

where P :
① S → ABC
② A → 1A
③ B → 2B
④ C → 3C
⑤ A → 1
⑥ B → 2
⑦ C → 3

H =



$L(G,H) = \{1^n 2^n 3^n : n \geq 1\}$

Path ① ② ③ ④ ② ③ ④ ⑤ ⑥ ⑦ corresponds to derivation

$S \Rightarrow ABC \Rightarrow 1ABC \Rightarrow 1A2BC \Rightarrow 1A2B3C \Rightarrow 11A2B3C \Rightarrow 11A22B3C \Rightarrow$
$11A22B33C \Rightarrow 11122B33C \Rightarrow 11122233C \Rightarrow 111222333.$

Note that a context-free grammar (without graph control) is a context-free grammar with graph control where the graph is a complete digraph with a node for each production.

Rozenberg and Salomaa [6] labeled the nodes of the graph with sets of productions which they called tables. The two definitions

of graph control are equivalent since a node n labeled with $p_1,\ldots,p_m$ can be split into m nodes labeled $p_1,\ldots,p_m$ respectively, and the m nodes have the same incoming and outgoing arcs as n.

Lemma 1. For any context-free grammar with graph control whose nodes are labeled with sets of production there is an equivalent context-free grammar with graph control whose nodes are labeled with a single production.

In the remainder of this section we will show that several modifications of grammars with graph control do not increase the generating capacity. These modifications will greatly simplify the inclusion relations of the next section.

A context-free grammar with graph control and final nodes is an ordered triple (G,H,F) where

(G,H) is a context-free grammar with graph control and F is a subset of the nodes of H.

$L(G,H,F) = \{w \text{ in } T^* : (S,j) \Rightarrow^* (w,k) \text{ for node } j \text{ in } H \text{ and } k \text{ in } F\}$

Lemma 2. For every context-free grammar with graph control and final nodes there is an equivalent context-free grammar with graph control.

Proof:

Let (G,H,F) be a context-free grammar with graph control and final nodes. We will construct an equivalent grammar with graph control. Modify G by replacing each terminal symbol in the right side of each production in P with a special symbol and add productions that rewrite these special symbols as the terminal it replaced. Now modify the graph H by adding a complete digraph with nodes labeled with the added productions and for each arc of H leading to final node in F, add arcs to each of the nodes in the complete digraph.

Lemma 3. For every context-free grammar with graph control there is an equivalent context-free grammar with graph control and final nodes.

Proof:

Let (G,H) be a context-free grammar with graph control. The final nodes of H are those nodes with an incoming arc corresponding to a terminal production, i.e. the right side is a terminal string.

A context-free grammar with graph control and initial node is an ordered triple (G,H,i_0) where

(G,H) is a context-free grammar with graph control and i_0 is the label of a node in H.

$L(G,H,i_0) = \{w \text{ in } T^* : (S,i_0) \Rightarrow^* (w,k) \text{ for some node k in H}\}$.

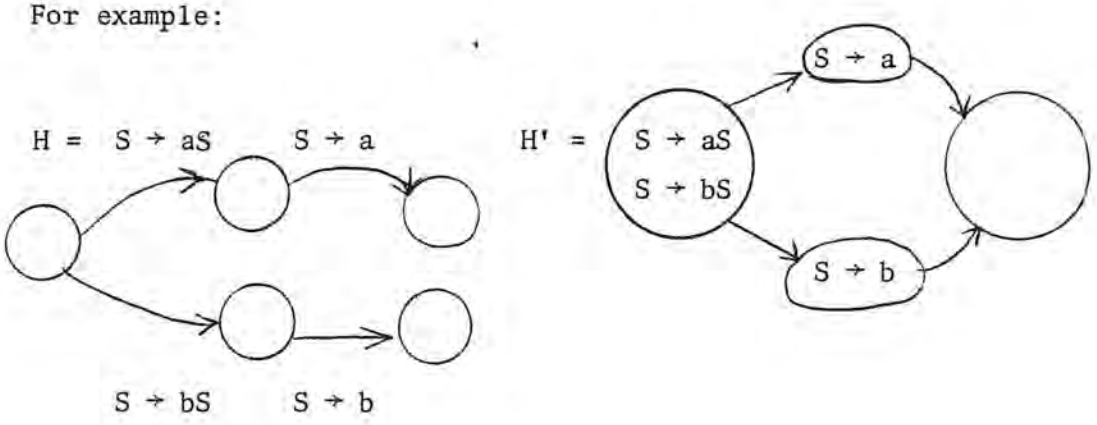
By augmenting a context-free grammar with a new start symbol S' and the production $S' \to S$ and assigning a unique label to this production, we can convert a context-free grammar with graph control into an equivalent one with initial node. By the same process we can convert a context-free grammar with graph control and initial node into an equivalent context-free grammar with graph control.

Lemma 4. L is generated by a context-free grammar with graph control if an only if L is generated by a context-free grammar with graph control and initial node.

Next consider labeling the edges instead of the nodes of the control graph. Then a string is generated by the context-free grammar with graph control if it is generated by the context-free grammar and the labels of productions in the derivation correspond to a path in the graph. Thus $(\alpha, i) \Rightarrow (\beta, j)$ if and only if $\alpha = xAy$, $\beta = xwy$ and there is an edge from node i to node j labeled $A \to w$. Let $\Rightarrow^*$ be the reflexive transitive closure of $\Rightarrow$.

It is straightforward to convert a context-free grammar with graph control and node labeling to one with edge labeling. For each node place that node's label on all arcs directed away from the node. Converting an edge labeled context-free grammar with graph control is not the reverse of the preceding construction.

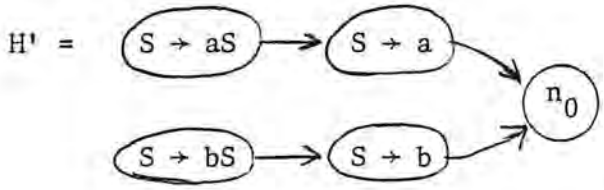
For example:



$L(G,H) = \{a,b,aa,bb\}$ $L(G,H') = \{a,b,aa,ab,ba,bb\}$

Given the edge labeled graph H, we want to form a node labeled graph H' from H such that there is a 1-1 correspondence between edge labeled paths in H and node labeled paths in H'. To do this we will form H', the line digraph of H. Graph H' will have a node for each edge in H with the same label. There is an arc from e = (i,j) to e' = (m,n) in H' whenever j = m; that is, there is an arc from e to e' in H' if the terminal node of e = initial node of e'. Because a path contains one more node than edge, we also have to add a node n_0 and an arc from each node to n_0.

Our example then becomes



Lemma 5. L is generated by a context-free grammar with graph control if and only if L is generated by a context-free grammar with graph control and edge labeling.

It should also be clear that combining several of the variants such as initial node and edge labeling does not increase the generative capacity of context-free grammars with graph control. In the following sections we will use the most convenient variant of context-free grammars with graph control.

3. INCLUSION RELATIONS

Graph control is a general framework for grammars with regulated rewriting. Programmed grammars [5], matrix grammar [1], periodically time-varying grammars [7], grammars with regular control [8] and state grammers [2] are examples of grammars with restrictions on the use of productions and are all equivalent with respect to generative power [8, 4]. Rozenberg and Salomaa [6] showed the equivalence of programmed grammars and grammars with graph control.

The purpose of this section is to show that all of the grammars given above are special cases of grammars with graph control and hence the inclusion relations are obvious. We will also see that the structure of the control graph gives insight into the nature of the restriction and the relation between the various restrictions.

A context-free programmed grammar is an ordered triple (G,g,f), where $G = (N,T,P,S)$ is a context-free grammar and g and f are mappings of Lab(P) into subsets of Lab(P). Mappings g and f are called the success and failure go-to fields respectively.

We say that $(\alpha, h_1) \Rightarrow (\beta, h_2)$ if and only if production $A \rightarrow w$ is labeled h_1 and either (1) $\alpha = xAy$, $\beta = xwy$ and h_2 in $g(h_1)$ or (2) A does not occur in α, $\beta = \alpha$ and h_2 in $f(h_1)$. Let $\Rightarrow^*$ denote the reflexive transitive closure of $\Rightarrow$.

A context-free programmed grammar with empty failure fields is the special case of a programmed grammar (G,g,f) where f is the empty mapping. Context-free programmed grammars with empty failure fields turn out to be identical to context-free grammars with graph control where the nodes of the control graph are labeled with single productions. Rozenberg and Salomaa [6] showed the equivalence between context-free programmed grammars and context-free grammars with graph controlled tables.

Theorem 1. A language L is generated by a context-free programmed grammar with empty failure fields if and only if L is generated by a context-free grammar with graph control.

A context-free matrix grammar is an ordered pair (G,M) where G is a context-free grammar and M is a finite set of finite nonempty sequences of context-free productions. We say that $\alpha \underset{M}{\Rightarrow} \beta$ if for some matrix $m = (A_1 \to w_1, \ldots, A_n \to w_n)$ in M, β is obtained from α by applying the sequence of productions in m to α in order beginning with $A_1 \to w_1$ and ending with $A_n \to w_n$. Let $\overset{*}{\underset{M}{\Rightarrow}}$ be the reflexive transitive closure of $\underset{M}{\Rightarrow}$. The language generated by (G,M), $L(G,M) = \{w \text{ in } T^* : S \overset{*}{\underset{M}{\Rightarrow}} w\}$.

The control graph for a matrix grammar consists of a collection of circuits, one for each matrix. There is an arc from the node corresponding to the last production in each matrix to the nodes corresponding to the first production in each matrix. Hence the application of the sequence of productions in a matrix corresponds to a path from the first to the last nodes in a circuit.

Theorem 2. If L is generated by a context-free matrix grammar, then we can construct a context-free grammar with graph control that generates L.

A periodically time-varying context-free grammar is an ordered pair (G,f) where G = (N,T,P,S) is a context-free grammar and f is a periodic mapping of the set of natural numbers into the subsets of P, i.e. there exists a natural number $k \geq 1$ such that f(j+k) = f(j) for all j. k is called the period. We say that

$(\alpha, i) \underset{pv}{\Rightarrow} (\beta, j)$ and if and only if $j = i+1$, $\alpha = xAy$, $\beta = xwy$ and and $A \to w$ is in f(i). Let $\overset{*}{\underset{pv}{\Rightarrow}}$ be the reflexive transitive closure of $\underset{pv}{\Rightarrow}$. Note that the mapping f specifies which productions f(i) can be used at the i^{th} step in a derivation. The language generated by a periodically time-varying context-free grammar is $L(G,f) = \{w \text{ in } T^* : (S,1) \overset{*}{\underset{pv}{\Rightarrow}} (w,j) \text{ for some } j\}$.

The control graph for a periodically time-varying context-free grammar of period k is a circuit with k nodes labeled f(1),...,f(k) and initial node f(1).

Theorem 3. If L is generated by a periodically time-varying context-free grammar, then we can construct a context-free grammar with graph control that generates L.

A context-free grammar with regular control is an ordered pair (G,C) where G = (N,T,P,S) is a context-free grammar and C is a regular control language over Lab(P), i.e. words in C are strings over Lab(P), the set of labels for the productions in P. A control word over Lab(P) consists of the labels of productions applied in a derivation in the order of their application. The language generated by a context-free grammar with regular control, denoted L(G,C) is the set of words in L(G) with a control word in C.

The control graph for a context-free grammar with regular control consists of the finite automaton that accepts the control language.

Theorem 4. If L is generated by a context-free grammar with regular control, then we can construct a context-free grammar with graph control that generates L.

A state grammar is an ordered six tuple $G = (K,N,T,P,p_0,S)$ where N,T and S are variables, terminals and start symbol, respectively, K is a finite set of states, p_0 in K is the initial state and P is a finite set of state productions of the form $(p,A) \to (q,w)$, p,q in K, A in N and w in $(NUT)^*$.
We say that $(p, \alpha) \Rightarrow (q, \beta)$ if $\alpha = xAy$, $\beta = xwy$ and $(p,A) \to (q,w)$ is a state production in P. Let $\Rightarrow^*$ be the reflexive transitive closure of $\Rightarrow$. The language generated by a state grammar $G = (K,N,T,P,p_0,S)$ is the set

$$\{w \text{ in } T^* : (S,p_0) \Rightarrow^* (w,q) \text{ for some } q \text{ in } K\}.$$

The nodes of the control graph correspond to the states of the state grammar and there is an arc from node i to node j labeled $A \to w$ if there is a state production $(i,A) \to (j,w)$. Also node p_0 is the initial node.

Theorem 5. If L is generated by a state grammar, then we can construct a context-free grammar with graph control that generates L.

A state grammar with accepting states is an ordered seven tuple $G = (K,N,T,P,p_0,S,F)$ where (K,N,T,P,p_0,S) is a state grammar and F, the accepting states, is a subset of K. The language generated by a state grammar with accepting states $G = (K,N,T,P,p_0,S,F)$ is the set $\{w \text{ in } T^* : (S,p_0) \Rightarrow^* (w,q), q \text{ in } F\}$ It should be clear that the control graph for the state grammar with accepting states is identical to the control graph for the state grammar with certain nodes designated as final nodes.

Theorem 6. If L is generated by a state grammar with accepting states, then we can construct a context-free grammar with graph control that generates L.

Remark. The proofs and results in this section hold whether or not the context-free grammar is ε-free and whether or not productions are applied in the appearance checking sense [8]. Also the results of this section hold for other types of grammars such as regular, context-sensitive, phrase structure and linear.

4. NEGATIVE CONTEXT GRAMMARS

In this section we add negative context to context-free grammars with graph control and show that the resulting grammars are equivalent to context-free programmed grammars with both success and failure goto fields.

Negative or forbidding context was introduced by Van der Walt [9] and is another restriction that increases the generative capacity of context-free grammars. By negative or forbidding context we mean that a production cannot be applied to a string α if any one of a set of symbols appears in α. Each production in a negative context grammar has a negative context associated with it.

A negative context grammar with graph control is an ordered pair (G,H) where G= (N,T,P,S) is a context-free grammar and H is a graph whose nodes are labeled with subsets of $N \cup T$ and whose edges are labeled with productions in P.

We say that (α, i) directly generates (β, j) denoted $(\alpha, i) \underset{n}{\Rightarrow} (\beta, j)$ and if the following three conditions are satisfied:

1. For some w,x,y and A, $\alpha = xAy$, $\beta = xwy$ and $A \to w$ is in P;
2. Node i is labeled $\{A_1,\ldots,A_m\}$ and none of these symbols appear in α;
3. There is an arc from node i to node j labeled $A \to w$.

Let $\underset{n}{\Rightarrow}^*$ be the reflexive transitive closure of $\underset{n}{\Rightarrow}$. The language generated by (G,H) is the set L(G,H) =

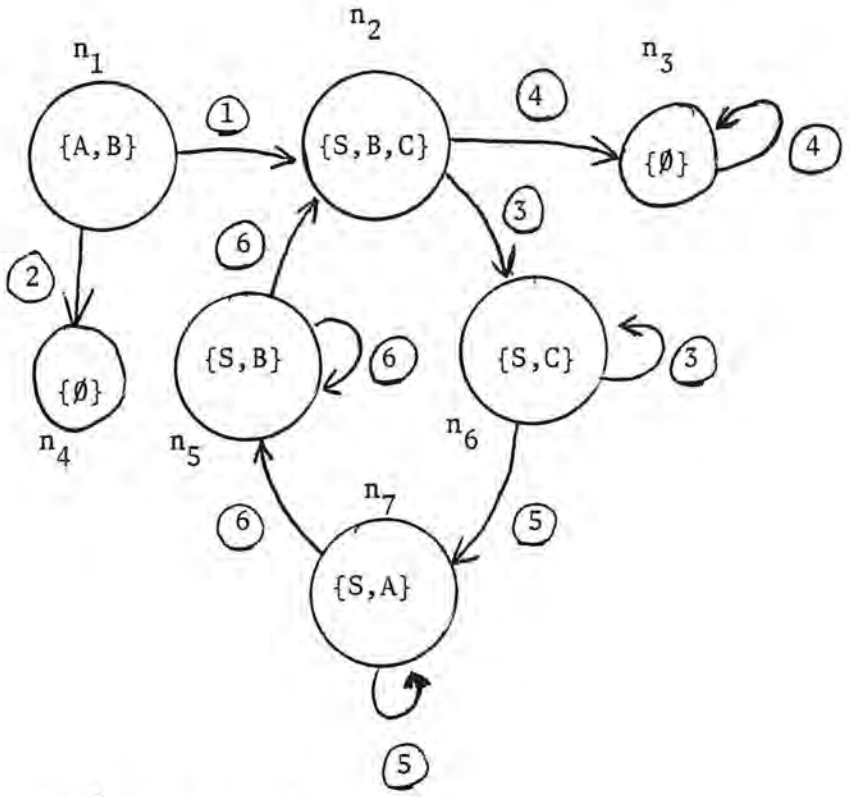
$$\{w \text{ in } T^* : (S,i) \underset{n}{\Rightarrow}^* (w,j) \text{ for some i and j}\}.$$

Example 2.

G = ({ S,A,B,C}, {a}, P,S)

where P: (1) S → AA
(2) S → a
(3) A → B
(4) A → a
(5) B → CC
(6) C → A

H =



$L(G,H) = \{a^{2^n} : n \geq 0\}$

Example 2 demonstrates that negative context grammars with graph control can check a string for the occurrence or nonoccurrence of symbols in a string. It also shows that the simultaneous application of the same production rule to all occurrences of a variable can be simulated by negative context grammars with graph control. The circuit n_2, n_6, n_7, n_5 causes each A to be rewritten as

a B, then each B as two C's and finally each C as an A. The same type of construction used in this example will be used in the proof of Theorem 7.

Just as we did for context-free grammars with graph control we can show that negative context grammars with graph control do not become more powerful by specifying an initial node in the graph or by specifying a certain set of nodes as the final nodes.

A negative context grammar with graph control and initial node is an ordered triple (G,H,N_0) where (G,H) is a negative context grammar with graph control and N_0 is a specific node in H called the starting node. The language generated by (G,H,N_0) is the set $L(G,H,N_0) = \{w \text{ in } T^* : (S,N_0) \overset{n}{\Rightarrow}{}^* (w,i) \text{ for some } \overline{\text{node } i}\}$.

Lemma 6. For every negative context grammar with graph control we can construct an equivalent negative context grammar with graph control and initial node.

A negative context grammar with graph control and final nodes is an ordered triple (G,H,F) where (G,H) is a negative context grammar with graph control and F is a subset of the nodes of H and is called the set of final nodes. The language generated by (G,H,F) is the set $L(G,H,F) = \{w \text{ in } T^* : (S,i) \overset{n}{\Rightarrow}{}^* (w,j),\ j \text{ in } F\}$.

Lemma 7. For every negative context grammar with graph control we can construct an equivalent negative context grammar with graph control and final nodes.

The proofs of both Lemmas are identical to the ones in Section 2.

In the remainder of this section we will give constructive proofs of the equivalence between negative context grammars with graph control and context-free programmed grammars.

Theorem 7. If L is generated by a context-free programmed grammar, then L is generated by a negative context grammar with graph control.

Proof:

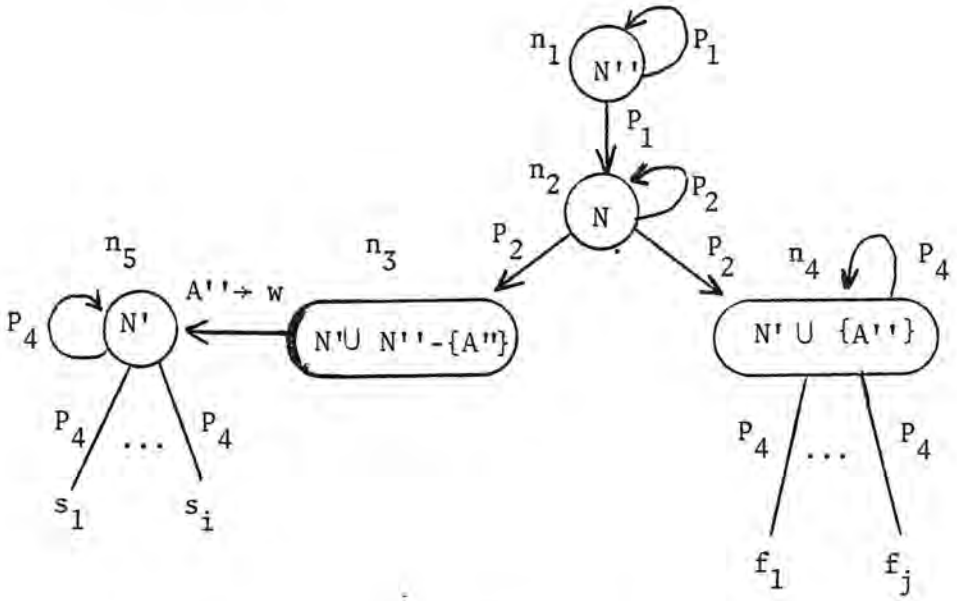
Let L be generated by (G,s,f) where G = (N,T,P,S). Let (G'H) be a negative context grammar with graph control where G' and H are constructed as follows:

1. $G' = (N_1, T, P', S)$ where

   $N_1 = N \cup N' \cup N''$, $N' = \{A' : A \text{ in } N\}$ and $N'' = \{A'' : A \text{ in } N\}$

   $P' = P_1 \cup P_2 \cup P_3 \cup P_4$, $P_1 = \{A \to A' : A \text{ in } N\}$, $P_2 = \{A' \to A'' : A \text{ in } N\}$, $P_3 = \{A'' \to w : A \to w \text{ in } P\}$, $P_4 = \{A'' \to A : A \in N\}$

2. For each production $A \to w$ with success field $\{s_1,\ldots,s_i\}$ and failure field $\{f_1,\ldots,f_j\}$ in P, the graph H has the following five node subgraph.



Nodes n_1 and n_2 force every B in N in the string to be rewritten first as an B' and then as an B''. If A'' is in the string, e.g. A appeared in the original string, then the branch from n_2 to n_3 will be taken; otherwise the branch from n_2 to n_4 will be taken. The branch from n_3 to n_5 is the application of the production $(A'' \to w)$. At node n_5 all B'' symbols are rewritten as B's and on the last such rewrite there is a branch to the first node of each subgraph corresponding to a production label in the success field. Note that the first node of each of these subgraphs is labeled N''.

At node n_4, again all B'' symbols are rewritten as B's and there is a branch to the first node of each subgraph corresponding to a production label in the failure field.

Theorem 8: If L is generated by a negative context grammar with graph control, then we can construct an equivalent context-free programmed grammar.

Proof:

Let (G,H) generate L where G = (N,T,P,S) and H = (V,E). Let (G',s,f) be a context-free programmed grammar constructed by modifying the productions of G in the following way. For each node n in H labeled $\{A_1,\ldots,A_k\}$ and productions $B_1 \rightarrow w_1,\ldots, B_m \rightarrow w_m$ as labels on the edges emanating from node n to nodes $n_1,\ldots,n_m$, respectively, the programmed grammar will contain the following rules:

Label	Rule	Success	Failure
n	$A_1 \rightarrow A_1$	$\emptyset$	r_2
r_2	$A_2 \rightarrow A_2$	$\emptyset$	r_3
.	. .	.	.
.	. .	.	.
.	. .	.	.
r_k	$A_k \rightarrow A_k$	$\emptyset$	$\{t_1,\ldots,t_m\}$
t_1	$B_1 \rightarrow w_1$	n_1	$\emptyset$
t_2	$B_2 \rightarrow w_2$	n_2	$\emptyset$
.	. .	.	.
.	. .	.	.
.	. .	.	.
t_m	$B_m \rightarrow w_m$	n_m	$\emptyset$

It should be clear from the construction that (G',s,f) simulates the application of a rule in (G,H) by first checking that none of the negative context symbols appear in the string and then applying the production if not. If one of the negative context symbols appears, then the success field is empty and the derivation halts.

It is important to note that for both of the previous two theorems the same constructive proof applies whether the grammar is ε-free (e.g. does not contain any productions of the form $A \rightarrow \varepsilon$) or not.

REFERENCES

1. Abraham, S. Some questions of phrase structure grammars, Computational Linguistics 4 (1965), 61-70.

2. Kasai, T. An hierarchy between context-free and context-sensitive languages, J. Comput. System Sci. 4 (1970), 492-508.

3. Harary, F. Graph Theory, Addison Wesley, Reading, Mass., 1969.

4. Moriya, E. Some remarks on state grammars and matrix grammars, Inform. Contr. 23 (1973), 48-57.

5. Rosenkrantz, D. J. Programmed grammars and classes of formal languages, J. Assoc. Comput. Mach. 16 (1969), 107-134.

6. Rozenberg, G. and A. Salomaa. Context-free grammars with graph controlled tables, J. Comput. System Sci. 13 (1976), 90-99.

7. Salomaa, A. Periodically time-variant context-free grammars, Inform. Contr. 17 (1970), 294 -

8. Salomaa, A. Formal Languages, Academic Press, New York, 1973.

9. Van der Walt, A. P. J. Random context languages, in Information Processing 71 (C. V. Freiman, ed.), North-Holland, Amsterdam, 1972, pp. 66-68.