No PAGE #36

THE STRUCTURE-PRESERVING CONSTRAINT AND THE UNIVERSAL BASE HYPOTHESIS

by

DIANE KRAVIF

S.B., Massachusetts Institute of Technology

(1969)

SUBMITTED IN
PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE
DEGREE OF MASTER OF SCIENCE

at the

MASSACHUSETTS INSTITUTE OF
TECHNOLOGY

May, 1971

Signature of Author .....
Department of Foreign Literatures
and Linguistics, May 14, 1971

Certified by .....
Thesis Supervisor

Accepted by .....
Chairman, Departmental Committee
on Graduate Students

THE STRUCTURE-PRESERVING CONSTRAINT AND THE UNIVERSAL BASE HYPOTHESIS

by

Diane Kravif

Submitted to the Department of Foreign Literatures and Linguistics on May 14, 1971, in partial fulfillment of the requirements for the degree of Master of Science.

ABSTRACT

One measure of the adequacy of the theory of grammar as developed by transformational grammarians is the weak generative power of the transformational grammars allowed by the theory. Previous work, mainly that of Peters and Ritchie, has shown that the generative power of transformational grammars is equal to that of Turing machines, even if we impose the requirement that all transformational grammars must have the same base component (the "universal base hypothesis"). These results are a consequence of the tremendous power of cyclically applied transformations.

Emonds has proposed a linguistically (as opposed to mathematically) motivated constraint on transformations, namely that they be either structure-preserving or root transformations, or minor movement rules (the "structure-preserving constraint"). In this paper, we examine the generative power of transformational grammars obeying both the universal base hypothesis and the structure-preserving constraint. We show that imposing both these restrictions simultaneously does not restrict the generative power of transformational grammars.

Formally, our theorem is that there exists a phrase structure grammar $\mathcal{P}$ such that given an r.e. language L over the alphabet $\{a_1,\ldots a_n\}$, there exists a set of transformations $\mathcal{T}$ all of whose members obey the structure-preserving constraint, such that if $\mathcal{G} = (\mathcal{P},\mathcal{T})$ and the terminal alphabet of $\mathcal{G}$ is $\{a_1,\ldots a_n\}$, then $L(\mathcal{G}) = L$.

An outline of the proof is as follows: By a result of Peters and Ritchie, L is an r.e. language over $\{a_1,\ldots a_n\}$ if and only if L is generated by a transformational grammar G = (P,T), where P is a given universal base and T is a set of ten given transformations. So it is sufficient to construct a grammar $\mathcal{G}$ as specified above and to show that $L(\mathcal{G}) = L(G)$. We present such a grammar and prove that $L(G) \subseteq L(\mathcal{G})$ by mimicking Peters and Ritchie's derivation of a sentence of L(G) in $\mathcal{G}$. We prove the inclusion in the other direction by showing that if any phrase marker other than the one we start from in the derivation in the first half of the proof yields a sentence in $L(\mathcal{G})$, that sentence is also in L(G).

Thesis Supervisor: G.H.Matthews Title: Professor of Modern Languages

ACKNOWLEDGEMENTS

I would like to thank G.H.Matthews for the weekly advice and exhortation which I received in writing this paper, and Stanley Peters for his kind encouragement to a would-be mathematical linguist.

TABLE OF CONTENTS

Section 0: While generative grammars of particular languages express how natural languages differ from each other, the theory of grammar specifies how natural languages differ from all other languages. One measure of the adequacy of the theory of grammar in performing this task is the scope of the class of languages generated by transformational grammars the theory allows. The work of Peters and Ritchie (1969a) and Kimball (1967) showed that any recursively enumerable language can be generated by a context sensitive- or context free-based transformational grammar, i.e. that the generative power of transformational grammars is at least (and in fact at most) equal to that of Turing machines. The theory of grammar developed up to 1967, when these results were made known, was inadequate to differentiate the natural languages from other recursively enumerable languages, at least along the dimension of generative power. Clearly, the theory of grammar had to impose more restrictions on transformational grammars before considerations of generative power could constitute evidence for or against any particular theory.

The universal base hypothesis states that all natural languages utilize the same base component in their transformational grammars. Peters and Ritchie (1969b) investigated the universal base hypothesis and showed that there exists a base such that every recursively enumerable language is generated by a transformational grammar with this base. And Peters (1970;38) writes, "The reason for

this state of affairs is not hard to find. It is the enormous power and flexibility that exists in grammatical transformations when they are iterated by cyclic application. ...[This power] can be used to remove all differences in the base components of grammars for every natural language." So independent of the correctness of the universal base hypothesis, new constraints were needed on transformations.

In 1969, Emonds proposed that transformations be constrained to meet certain requirements. His structure-preserving constraint was empirically motivated, but it was clear that he intended it to have a bearing on this issue, for instance when he wrote (1970;33) in the introduction to his thesis, "The narrowing of the notion 'possible transformational rule' that emerges from this study is considerable." However, building on the work of Peters, Ritchie, Kimball, and Emonds, I showed first that any recursively enumerable language can be generated by a context sensitive-based transformational grammar all of whose transformations obey the structure-preserving constraint, and next that any r.e. language can be generated by a context free-based transformational grammar all of whose transformations obey the structure-preserving constraint (Kravif 1970, 1971).

Since the universal base hypothesis does not restrict the weak generative power of transformational grammars because transformations are too powerful, and since Emonds' independently proposed constraint is not strong enough by

itself to do the job, it is a natural step to see if combining the constraint and the universal base hypothesis has any effect on the generative power of transformational grammars. In fact, this is the question we investigate in this paper. The answer, unfortunately, is in the tradition of those discussed above. The theorem we prove is that there exists a base such that every recursively enumerable language is generated by a transformational grammar with that base, all of whose transformations obey the structure-preserving constraint.

In section 1 we present the structure-preserving constraint and define the new concepts Emonds introduced in connection with it. Empirical motivation for the constraint is not discussed here, and we refer the interested reader to Emonds (1970). In section 2 we recapitulate the theorem of Peters and Ritchie (1969b) mentioned above. Section 3 contains a proof of the existence of a universal base such that every r.e. language is generated by a transformational grammar with that base which obeys the structure-preserving constraint. A comparison of the grammar of section 3 with Peters and Ritchie's grammar in section 2 will show that the proof of our theorem is heavily dependent on the work of Peters and Ritchie.

Section 1: Emonds proposed his structure-preserving constraint in his thesis (1970); an earlier version appeared in his 1969 paper. Before we state the constraint, we will

review the three major notions Emonds developed in his discussions, namely the notions of structure-preserving transformation, root transformation, and minor movement rule.

Structure-preserving rules essentially move or insert a constituent X into a position in the phrase-marker "where a node X is already provided for by the phrase structure rules." (1970;37) A structure-preserving deletion rule "specifies the location of a non-empty node in trees and removes the material it dominates, leaving an empty node" (1970;38). In short, every deletion is structure-preserving.[1] To make precise the notion of insertion, Emonds allows a node to remain empty during a transformational derivation, as long as it dominates a terminal element at some point in the derivation. An insertion transformation inserts material into an empty space which already bears the proper label.

To formalize this notion, we introduce a new symbol Δ, a member of the terminal vocabulary of the phrase structure base but not of the terminal vocabulary of the transformational language, to act as a place holder. Since Δ is not in the terminal vocabulary of the transformational language, its presence in a surface string z is sufficient to filter out the phrase marker from which z was derived as a possible deep structure for the language, reflecting the fact that we do not allow empty nodes to occur in surface structures.

As formalized by Peters and Ritchie (1969a), a transformation consists of four types of elementary

transformations, namely deletions, substitutions, and left and right adjunctions. In this framework it is impossible to insert material not already located in the phrase marker into the phrase marker, although such insertion has traditionally been allowed, and has been exploited in such transformations as Complementizer Insertion and There-Insertion. However, recent work (e.g. Bresnan 1970 and Bowers1970) suggests that such insertions may not be necessary to describe natural languages, and we will not use them here.

We now define structure-preserving elementary substitutions in the obvious way, and we allow such an elementary transformation to substitute a Δ that is an X for other material that is an X in case we want to fill an "empty" X node later in the derivation. A structure-preserving elementary deletion is just a deletion.[2] We do not need to define structure-preserving elementary adjunctions since we think of a structure-preserving movement rule of a string z which is an X to another node X as a substitution of z for Δ dominated by X and a deletion of the original z. Although we do not allow insertions of new material into the phrase marker, if necessary Emonds' structure-preserving insertions can be handled as structure-preserving substitutions of new material for a Δ.

Emonds (1970;8) defines the root of a tree[3] as the highest S in the tree, an S immediately dominated by the highest S, or "the reported S in direct discourse." A root

transformation, then, is "one in which any constituents moved, inserted, or copied are immediately dominated by a root in the derived structure." (1970;10) We will not be concerned with root transformations in this paper, as it turns out, so we will say no more about them here.

A minor movement rule is a transformation which moves a constituent B over a single adjacent constituent C, where the two nodes are "mutually in construction". Furthermore, B is neither a "lexical category node" nor a "phrase node" (i.e., as we will see below, B is a "function category node") and C is not a function category node or S.

B and C are mutually in construction whenever either the node immediately dominating B, but not B itself, dominates C, or vice versa. This definition is due to Emonds, who based it on Klima's notion of "in construction with".

The nodes occurring in the phrase structure rules which never appear to the left of an → are the preterminal nodes; the others are non-preterminal. (The preterminal nodes are then subject to lexical insertion). Emonds partitions the set of nodes occurring in the phrase structure rules into three parts, the phrase nodes, the lexical category nodes, and the function category nodes. The phrase nodes are those non-preterminal nodes under which unlimited recursion can occur. The theory of grammar specifies that S is a phrase node, but whether there are any other phrase nodes is currently in dispute, with NP the leading candidate. We will only need one phrase node, S, in the phrase structure

grammar outlined in section 3 (see, however, footnote 5). The lexical category nodes are the heads of phrase nodes. The only restriction on the choice of lexical category nodes is that a phrase node can immediately dominate at most one lexical category node. Finally, the function category nodes are all the remaining nodes.

In formalizing the notion of minor movement rule, we define only the elementary minor movement substitution, since we regard a minor movement rule as two simultaneous elementary substitutions, one of B for C and one of C for B.[4]

We can now state the structure-preserving constraint:

(1) The only non-structure-preserving transformations are root transformations and minor movement rules. (1970;207)

Section 2: Peters and Ritchie (1969b) showed that there exists a base such that every recursively enumerable language is generated by a transformational grammar with this base and a set of transformations. In fact, the transformations vary very little from language to language; one depends on the particular instructions of a Turing machine that generates the r.e. language and most of them refer to the number of states of the machine. Peters and Ritchie's universal base is simple, consisting only of the rules in (2),

(2) $S \rightarrow S\#$

$S \rightarrow a_1 \ldots a_n b\#$

where $\{a_1, \ldots a_n\}$ is the terminal vocabulary of the

transformational grammar and $\{b,\#\}$ are additional terminal symbols of the phrase structure grammar. Throughout this paper the symbol # is not to be understood as the usual sentence boundary, but as a special marker whose use will become apparent as we look at some transformational derivations. The sentence boundary normally appears in the axiom of the phrase structure grammar, flanking the symbol S; to avoid confusion with the # used here, we will omit it.

Given an r.e. language generated by a Turing machine Z, Peters and Ritchies's ten transformations are as follows:

T_1 $a_1 \ldots a_n - e - \ldots - e - b - \#$

$$\begin{array}{cccccccc} 1 & & 2 & \ldots & r+1 & r+2 & r+3 & \Rightarrow \\ 1 & & r+3 & \ldots & r+3 & r+2 & r+3 & \end{array}$$

where e is the empty symbol (not in the vocabularies of the phrase structure grammar, but in the metalanguage) and r is the number of states of Z.

T_2 $a_1 \ldots a_n - \#^r - b - \#$

$$\begin{array}{cccccc} 1 & & 2 & 3 & 4 & \Rightarrow \\ 2+1 & & 0 & 3 & 4+3 & \end{array}$$

T_3 $[_S\# - \#^{r-1}a_1 \ldots a_n - b - \begin{Bmatrix} e \\ \# \end{Bmatrix} - \begin{Bmatrix} \# \\ e \end{Bmatrix} - (\#\#)b]_S - X - \#$

$$\begin{array}{cccccccccc} 1 & & 2 & & 3 & 4 & 5 & 6 & 7 & 8 & \Rightarrow \\ 1 & & 2 & & 3 & 4 & 4+5 & 6 & 3+7 & 0 & \end{array}$$

Condition: 1 is not an S.

T_4 $[_S\#-\#^{r-1}a_1 \ldots a_{i-1}-a_i-a_{i+1} \ldots a_n b-\begin{Bmatrix} e \\ \# \end{Bmatrix} - \begin{Bmatrix} \# \\ e \end{Bmatrix} \#b]_S - e - X - \#$

$$\begin{array}{ccccccccccc} 1 & & 2 & & 3 & & 4 & 5 & 6 & 7 & 8 & 9 & \Rightarrow \\ 1 & & 2 & & 3 & & 4 & 5 & 5+6 & 5 & 3+8 & 0 & \end{array}$$

Condition: 1 is not an S.

T_5 $[_S\#^r a_1 \ldots a_n b\#\#\# - \# - (\#\#)b]_S - b\# - Y - \#$

$$\begin{array}{cccccc} 1 & 2 & 3 & 4 & 5 & 6 \Rightarrow \\ 1+2 & 2 & 3 & 4 & 5 & 6 \end{array}$$

T_6 $[_S\#^r a_1 \ldots a_n b\#\# - \#\# - (\#)b]_S - Y - \#\begin{Bmatrix} a_1 \\ \vdots \\ a_n \\ b \end{Bmatrix} b\#$

$$\begin{array}{ccccc} 1 & 2 & 3 & 4 & 5 \Rightarrow \\ 1+2 & 2 & 3 & 4 & 5 \end{array}$$

T_7 See next page.

T_8 $[_S\#^r a_1 \ldots a_n b\#^6 - \# - b]_S - X - \begin{Bmatrix} a_1 \\ \vdots \\ a_n \\ b \end{Bmatrix} - \#^j - \begin{Bmatrix} a_1 \\ \vdots \\ a_n \\ b \end{Bmatrix} - Y - b\#$

$$\begin{array}{ccccccccc} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \Rightarrow \\ 0 & 2 & 0 & 2+4 & 5 & 0 & 7 & 8 & 0 \end{array}$$

where $1 \leq j \leq r$

T_9 $[_S\#]_S - X - \# - \begin{Bmatrix} b \\ e \end{Bmatrix} - \begin{Bmatrix} e \\ \begin{Bmatrix} a_1 \\ \vdots \\ a_n \end{Bmatrix} \end{Bmatrix} - Y - \#$

$$\begin{array}{ccccccc} 1 & 2 & 3 & 4 & 5 & 6 & 7 \Rightarrow \\ 1 & 2 & 0 & 0 & 5+3 & 6 & 0 \end{array}$$

Condition: $6 \neq e$

T_{10} $[_S\#]_S - X - \# - \begin{Bmatrix} b \\ e \end{Bmatrix} - \begin{Bmatrix} e \\ \begin{Bmatrix} a_1 \\ \vdots \\ a_n \end{Bmatrix} \end{Bmatrix} - \#$

$$\begin{array}{ccccccc} 1 & 2 & 3 & 4 & 5 & 6 \Rightarrow \\ 0 & 2 & 0 & 0 & 5 & 6 \end{array}$$

It is helpful to think of each phrase marker (3)

(3) $[_S \ldots [_S [_S a_1 \ldots a_n b\#]_S \#]_S \ldots \#]_S$

generated by the base as containing a copy of the alphabet

T_7 $[_S\#^q - U_1 - U_2 - U_3 - (\#(\#(\#))\#^4 b]_S$ - X - U_4 - e - U_5 - $\#^p$ - e - U_6 - U_7 - U_8 - Y - b - #

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
1	2	3	4	5	6	7	13	0	1	9	12	0	3	15	16	0

where the value of U_1 through U_8 is determined by the instructions of Z, as follows:

i) If the instruction is a print instruction, (q_p,c,d,q_q), then

$U_1 = \#^{r-q}a_1 \ldots a_{i-1}$ $\quad$ $U_4 \in \{a_1,\ldots a_n,b\}$

$U_2 = a_i$ (= d) $\quad$ $U_5 = U_6 = U_7 = e$

$U_3 = a_{i+1} \ldots a_n b$ $\quad$ $U_8 = c$

ii) If the instruction is a move right, (q_p,c,R,q_q), then

$U_1 = \#^{r-q}a_1 \ldots a_n b$ $\quad$ $U_5 = U_6 = e$

$U_2 = U_3 = e$ $\quad$ $U_7 = c$

$U_4 \in \{a_1,\ldots a_n,b\}$ $\quad$ $U_8 = e$

iii) If the instruction is a move left, (q_p,c,L,q_q), then

$U_1 = \#^{r-q}a_1 \ldots a_n b$ $\quad$ $U_5 \in \{a_1,\ldots a_n,b\}$

$U_2 = U_3 = e$ $\quad$ $U_6 = c$

$U_4 = e$ $\quad$ $U_7 = U_8 = e$

$\{a_1,\ldots a_n,b,\#\}$ (in the innermost subsentence) followed by a copy of the tape of Turing machine Z. Essentially, we start with just the alphabet. T_3 and T_4 construct the copy of the tape; after they have applied, we have a tape containing $b^u\#yb^v$, where y is the input to Z and u and v are respectively the number of blank squares to the left and right of y used in Z's computation of an output, given y as input. The # marks the position of the reading head, reading the leftmost symbol of y, when the machine is in its initial state q_1. In general, throughout the simulated computation the number of #'s in this part of the phrase marker is the index of the state Z is in; the #'s in the innermost subsentence and before each $]_S$ trigger and inhibit the transformations at the proper time. T_7 contains codings of the instructions of Z. Applied on successive cycles, it simulates the action of Z on the tape, while T_5 and T_6 mark whether the leftmost and rightmost ends of the tape have been reached. When T_7 can no longer apply, i.e. when Z has halted, T_8 deletes the copy of the alphabet, T_9 eliminates blanks within the output string, and T_{10} deletes all extraneous material, leaving only the consolidated output string appearing on Z's tape, a member of the language generated by Z.

In detail, the transformational derivation on a phrase marker of form (3) runs as follows:

First cycle: T_1 inserts r copies of the symbol # in the innermost subsentence, T_2 moves them to the beginning of the

alphabet and writes a b after the original # symbol.

Next v-1 cycles: T_3 writes a b on the copy of the Turing machine tape, using the first analysis in its structural description.

v+1st cycle: T_3 writes the final b to the right of the future input on the tape and signals it is done by inserting another # between the b's in the innermost S, using the second analysis in its structural description. T_4 writes one of the a's on the tape immediately to the left of the leftmost b, using the first analysis of its structural description.

Next $\ell(y)$-2 cycles, where $\ell(y)$ is the number of symbols in the input y: T_4 writes one of the a's on the tape immediately to the left of the leftmost symbol, using the first analysis in its structural description.

v+$\ell(y)$th cycle: T_4 writes one of the a's on the tape immediately to the left of the leftmost symbol, a # to the left of this, and signals it is done by inserting another # between the b's in the innermost S, using the second analysis of its structural description.

Next u-1 cycles: T_3 writes a b on the tape immediately to the left of the leftmost symbol, using the first analysis of its structural description.

v+$\ell(y)$+uth cycle: T_3 writes a b on the tape immediately to the left of the leftmost symbol and signals it is done by inserting another # between the b's in the innermost S, using the second analysis in its structural description.

T_7 simulates the first action of Z on input y.

Next q-1 cycles, where q = the number of steps Z takes before halting, given input y: T_7 simulates the next action of Z, on the altered tape and possibly in a new state. During cycles $v+\ell(y)+u$ through $v+\ell(y)+u+q-1$, T_5 and T_6 also apply, inserting their # markers between the two b's of the innermost subsentence to indicate the leftmost and rightmost squares of tape needed in the computation have been reached.

$v+\ell(y)+u+q$th cycle: T_8 erases the copy of the alphabet in the innermost subsentence, leaving only one #. It also writes a # at the leftmost end of the tape and erases the # markers representing the index of the state Z halted in.

Next $u+v+\ell(y)-1$ cycles: T_9 passes the inserted # over each symbol in turn, erasing b's and leaving a_i's unchanged.

$2(u+v+\ell(y))+q$th cycle: T_{10}, with the inserted # contemplating the rightmost symbol on the tape, deletes this symbol if it is a blank, and deletes everything but the consolidated string of a_i's which is the output of Z.

Peters and Ritchie (1969b) has a proof that this grammar generates exactly the language computed by Z.

Section 3: Let $\mathcal{P}$ be the context free phrase structure grammar with the following rules:

(4) $S \rightarrow S\#$

$S \rightarrow SS$

$S \rightarrow \Delta M+C\#$ where $M+ = M^* - \{e\}$

$M \rightarrow \Delta$

$M \rightarrow \#$

$C \rightarrow D+$ where $D+ = D^* - \{e\}$

The preterminal symbols of this grammar are $\{\#, \Delta, D\}$. Lexical insertions are all of the form $D \rightarrow a_i$, $1 \leq i \leq n$, and $D \rightarrow b$. We leave it to the reader to verify that it is consistent with the criteria of section 1 to partition the set of nodes of this phrase structure grammar into the following three parts:[5]

(5) Phrase nodes = $\{S\}$

Lexical category nodes = $\{C, D\}$

Function category nodes = $\{\#, \Delta, M\}$

The third S-expansion rule and the C-expansion rule collapse the rules $S \rightarrow \Delta M^n C\#$ and $C \rightarrow D^n$ for all $n \geq 1$ into schemata, just like the purported rule of English $S \rightarrow \left\{\begin{matrix}\text{and} \\ \text{or}\end{matrix}\right\} SS+$. Notice, finally, that $\mathcal{P}$ has only one recursive symbol, S (subject to the remark in footnote 5). We can now state our theorem.

Theorem For every recursively enumerable language L over the alphabet $\{a_1, \ldots a_n\}$, there exists a set of transformations $\mathcal{T}$ all of whose members obey the structure-preserving constraint such that if $\mathcal{G} = (\mathcal{P}, \mathcal{T})$ and the terminal alphabet of $\mathcal{G}$ is $\{a_1, \ldots a_n\}$, then $L(\mathcal{G}) = L$.

Proof By a result of Peters and Ritchie (1969b), L is an r.e. language over $\{a_1, \ldots a_n\}$ if and only if L is generated by a transformational grammar G = (P,T) where P (a universal base) has the rules given in (2) and $T = \{T_1, \ldots T_{10}\}$ of section 2. So it is sufficient to show that there exists a set of transformations $\mathcal{T}$ obeying the structure-preserving

constraint such that if $\mathcal{G} = (\mathcal{P}, \mathcal{T})$ then $L(\mathcal{G}1) = L(G)$. Suppose L(G) is generated by Turing machine Z, in the sense that $L(G) = \{z \mid z \in \{a_1, \ldots a_n\}^*$ and there exists a $y \in \{a_1 \ldots a_n\}^*$ such that Z halts given input y and yields z as output$\}$. Suppose further that Z has r states and s instructions, of which s_1 are print instructions, s_2 are move left instructions, and s_3 are move right instructions.

Let $\mathcal{T}$ be the set of the following $s_1+(r+2)(s_2+s_3)+9$ transformations:

$\mathcal{T}_1$ $[_S\Delta\#^{r+4}a_1 \ldots a_n\text{-}b\text{-}\#]_S[_S\Delta X\text{-}[_D Y]_D\text{-}Z]_S\text{-}\#$

1 2 3 4 5 6 ⇒
1 2 3 2 5 0

Condition: $D \in X$, $Y \neq b$, no substring of X or Z is an S.

$\mathcal{T}_2$ $[_S\Delta\#^{r+3}\text{-}\#\text{-}a_1 \ldots a_n\text{-}b\text{-}\#\text{-}]_S[_S\Delta X\text{-}[_D Y]_D\text{-}Z]_S\text{-}\#$

1 2 3 4 5 6 7 8 9 ⇒
1 0 3 4 0 6 4 8 0

Condition: No substring of X or Z is an S.

$\mathcal{T}_3$ $[_S\Delta\#^{r+2}\begin{Bmatrix}\# \\ e\end{Bmatrix}\text{-}a_1 \ldots a_{i-1}\text{-}a_i\text{-}a_{i+1} \ldots a_n b]_S[_S\Delta X\text{-}b\text{-}Y\text{-}Z]_S\text{-}\#$

1 2 3 4 5 6 7 8 9 ⇒
1 0 3 4 5 4 7 8 0

Condition: $b \notin Y$, $a_i \notin Z$ for all $1 \leq i \leq n$, no substring of X or Z is an S.

$\mathcal{T}_4$ $[_S\Delta\#^{r+1}\text{-}\#\text{-}a_1 \ldots a_n b]_S\text{-}[_S\Delta\text{-}[_M\Delta]^r_M\text{-}[_M\Delta]_M\text{-}b\text{-}X]_S\#$

1 2 3 4 5 6 7 8 ⇒
1 0 3 4 5 7 2 8

Condition: 4 is not an M, no substring of X is an S.

$\mathcal{T}_5$ $\left[_S\Delta\#^{r+1}-\#-a_1\ldots a_n b\right]_S-\left[_S\Delta-\left[_M\Delta\right]^r_M-\left[_M\Delta\right]_M-\left\{\begin{matrix}a_1\\ \vdots\\ a_n\end{matrix}\right\}-X\right]_S\#$

1	2	3	4	5	6	7	8 $\Rightarrow$
1	0	3	4	5	2	7	8

Condition: 4 is not an M, no substring of X is an S.

$\mathcal{T}_6$ $\left[_S\Delta\#^{r+1}a_1\ldots a_n b\right]_S-\left[_S\Delta-X-\left[_M\Delta\right]_M-b-Y\right]_S-\#$

1		2	3	4	5	6	7 $\Rightarrow$
1		2	3	5	4	6	0

Condition: 2 is not an M, no substring of X or Y is an S.

Let $\{I_m \mid 1 \leq m \leq s\}$ be the set of instructions of Z. For each print instruction $I_j = (q_p, c, d, q_q)$ add $\mathcal{T}_{7j}$. For each move left (i.e. move reading head left) instruction $I_k = (q_p, c, L, q_q)$ add $\mathcal{T}_{8k1}, \ldots \mathcal{T}_{8kr+2}$. For each move right instruction $I_\ell = (q_p, c, R, q_q)$ add $\mathcal{T}_{9\ell 1}, \ldots \mathcal{T}_{9\ell r+2}$. (See next page.)

$\mathcal{T}_{7j}$ $[_S\Delta\text{-}\overbrace{\#\text{-}\cdots\text{-}\#}^{q}\text{-}U_1\{{}^{e}_{U_2}\text{-}{}^{a_i\text{-}U_3}_{b\ \ e}\}]_S[_S\Delta X\text{-}[_M\Delta]_M\text{-}\Delta\text{-}\cdots\text{-}\Delta\text{-}\overbrace{\{{}^{\Delta\text{-}\cdots\text{-}\Delta\text{-}\overbrace{\Delta\text{-}\cdots\text{-}\Delta}^{q}}_{\#\text{-}\cdots\text{-}\#\text{-}\#\text{-}\cdots\text{-}\#}\}\text{-}\#\cdots\text{-}\#}^{r}\text{-}\{{}^{a_1}_{\vdots}{}^{\vdots}_{a_n}\}\text{-}Y]_S\text{-}\#$

1 2...q+1 q+2 q+3 q+4 q+5 q+6 ... r+5 r+6 ... q+r+5 q+r+6 q+r+7 q+r+8 ⇒

1 2...q+1 q+2 q+3 q+4 q+5 q+6 q+5.q+5 2 ... q+1 q+3 q+r+7 0

where $U_1 = \#^{r-q+1}a_1\ldots a_{i-1}$, $U_2 = a_i\ldots a_n$, $U_3 = a_{i+1}\ldots b$,

q+r+6 = c, and q+3 = d

Condition: $M \notin X$, no substring of X or Y is an S.

$\mathcal{T}_{8k1}$ $[_S\Delta\text{-}\overbrace{\#\text{-}\cdots\text{-}\#}^{q}\text{-}U_1]_S[_S\Delta X\text{-}[_D Y]_D\text{-}[_M\Delta]_M\text{-}\Delta\text{-}\cdots\text{-}\Delta\text{-}\overbrace{\{{}^{\Delta\text{-}\cdots\text{-}\Delta\text{-}\overbrace{\Delta\text{-}\cdots\text{-}\Delta}^{q}}_{\#\text{-}\cdots\text{-}\#\text{-}\#\text{-}\cdots\text{-}\#}\}\text{-}\#\cdots\text{-}\#}^{r}\text{-}\{{}^{a_1}_{\vdots}{}^{\vdots}_{a_n}\}\text{-}Z]_S\#$

1 2 ... q+1 q+2 q+3 q+4 q+5 ... r+4 r+5 ... q+r+4 q+r+5 q+r+6 ⇒

1 2 ⋯ q+1 q+2 q+4 q+3 q+5 ... q+4.q+4 2 ... q+1 q+r+5 q+r+6

where $U_1 = \#^{r-q+1}a_1\ldots a_n b$ and q+r+5 = c

Condition: No substring of X or Z is an S.

$\mathcal{T}_{8k2}$ $[_S\Delta\#^{r+1}a_1\ldots a_n b]_S[_S\Delta XM\text{-}[_D Y]_D\text{-}[_M Z]_M\text{-}M^{r-1}\text{-}\{{}^{a_1}_{\vdots}{}^{\vdots}_{a_n}\}\text{-}W]_S\#$

1 2 3 4 5 6 ⇒

1 3 2 4 5 6

$\mathcal{T}_{8k2}$ (continued)

where 5 = c

Condition: $M \notin X$, no substring of X or W is an S.

$$\mathcal{T}_{8k3}[_S\Delta\#^{r+1}a_1\ldots a_n b]_S[_S\Delta X M^2 - [_D Y]_D - [_M Z]_M - M^{r-2} - \begin{Bmatrix} a_1 \\ \vdots \\ a_n \\ b^n \end{Bmatrix} - W]_S\#$$

1		2	3	4	5	6 $\Rightarrow$
1		3	2	4	5	6

where 5 = c

Condition: $M \notin X$, no substring of X or W is an S.

etc.

$$\mathcal{T}_{8kr+1}[_S\Delta\#^{r+1}a_1\ldots a_n b]_S[_S\Delta X M^r - [_D Y]_D - [_M Z]_M - \begin{Bmatrix} a_1 \\ \vdots \\ a_n \\ b^n \end{Bmatrix} - W]_S - \#$$

1		2	3	4	5	6 $\Rightarrow$
1		3	2	4	5	0

where 4 = c

Condition: $M \notin X$, no substring of X or W is an S.

$$\mathcal{T}_{8kr+2}[_S\Delta\#^{r+1}a_1\ldots a_n b]_S - [_S\Delta - \Delta^{r-p+1}\#^p - \begin{Bmatrix} a_1 \\ \vdots \\ a_n \\ b^n \end{Bmatrix} - Y]_S\#$$

1		2	3	4	5 $\Rightarrow$
0		2	0	0	0

where 4 = c

Condition: 2 is not an M, no substring of Y is an S.

$\mathcal{T}_{9l1}$ $[_S\Delta\text{-}\overbrace{\#\text{-}\ldots\text{-}\#}^{q}\text{-}U_1]_S\,[_S\Delta X\text{-}[_M\Delta]_M\text{-}\Delta\text{-}\ldots\text{-}\Delta\text{-}\underbrace{\left\{\begin{matrix}\Delta\text{-}\ldots\text{-}\Delta\text{-}\overbrace{\Delta\text{-}\ldots\text{-}\Delta}^{q}\\ \#\text{-}\ldots\text{-}\#\text{-}\#\text{-}\ldots\text{-}\#\end{matrix}\right\}}_{r}\text{-}\#\text{-}\ldots\text{-}\#\text{-}\left\{\begin{matrix}a_1\\ \vdots\\ a_n\\ b\end{matrix}\right\}\text{-}[_D Y]_D\text{-}Z]_S\#$

1 2 ...q+1 q+2 q+3 q+4 ... r+3 r+4 ... q+r+3 q+r+5
q+r+4 q+r+6 ⇒

1 2 ...q+1 q+2 q+3 q+4 ... q+3.q+3 2 ... q+1 q+r+4
q+r+5 q+r+6

where $U_1 = \#^{r-q+1}a_1\ldots a_n b$ and q+r+4 = c

Condition: $M \not\subset X$, no substring of X or Z is an S.

$\mathcal{T}_{9l2}$ $[_S\Delta\#^{r+1}a_1\ldots a_n b]_S[_S\Delta X M^{r-1} - [_M Y]_M - \left\{\begin{matrix}a_1\\ \vdots\\ a_n\\ b\end{matrix}\right\} - MZ]_S\#$

1 2 3 4 ⇒
1 3 2 4

where 3 = c

Condition: $M \not\subset X$, $M \not\subset Z$, no substring of X or Z is an S.

$\mathcal{T}_{9l3}$ $[_S\Delta\#^{r+1}a_1\ldots a_n b]_S[_S\Delta X M^{r-2} - [_M Y]_M - \left\{\begin{matrix}a_1\\ \vdots\\ a_n\\ b\end{matrix}\right\} - M^2Z]_S\#$

1 2 3 4 ⇒
1 3 2 4

where 3 = c Condition: $M \not\subset X$, $M \not\subset Z$, no substring of X or Z is an S.

etc.

$$\mathcal{T}_{9\ell r+1}[_S\Delta\#^{r+1}a_1\ldots a_n b]_S[_S\Delta X - [_M Y]_M - \begin{Bmatrix} a_1 \\ \vdots \\ a_n \\ b \end{Bmatrix} - M^r Z]_S - \#$$

$$\begin{array}{ccccc} 1 & 2 & 3 & 4 & 5\Rightarrow \\ 1 & 3 & 2 & 4 & 0 \end{array}$$

where 3 = c

Condition: $M \notin X$, $M \notin Z$, no substring of X or Z is an S.

$$\mathcal{T}_{9\ell r+2}[_S\Delta\#^{r+1}a_1\ldots a_n b]_S[_S - \Delta - X\Delta^{r-p+1}\#^p - \begin{Bmatrix} a_1 \\ \vdots \\ a_n \\ b \end{Bmatrix}]_S - \#$$

$$\begin{array}{ccccc} 1 & 2 & 3 & 4 & 5\Rightarrow \\ 0 & 2 & 0 & 0 & 0 \end{array}$$

where 4 = c

Condition: $M \notin X$, no substring of X is an S.

$$\mathcal{T}_{10}\ [_S\Delta\#^r - \# - a_1\ldots a_n b]_S - [_S X - \Delta^{r+j+1}\#^j - Y]_S$$

$$\begin{array}{cccccc} 1 & 2 & 3 & 4 & 5 & 6\Rightarrow \\ 0 & 2 & 0 & 4 & 0 & 6 \end{array}$$

where $1 \leq j \leq r$

Condition: $M \notin X$, $M \notin Y$, no substring of X or Y is an S.

$$\mathcal{T}_{11}\ \#[_S X - \Delta - \begin{Bmatrix} b \\ e \end{Bmatrix} - \begin{Bmatrix} e \\ a_1 \\ \vdots \\ a_n \end{Bmatrix} - Y]_S - \#$$

$$\begin{array}{cccccc} 1 & 2 & 3 & 4 & 5 & 6\Rightarrow \\ 1 & 0 & 0 & 4+2 & 5 & 0 \end{array}$$

Condition: $5 \neq e$, no substring of X or Y is an S.

$$\mathcal{T}_{12}\ \#[_S - X - \Delta - \begin{Bmatrix} b \\ e \end{Bmatrix} - \begin{Bmatrix} e \\ a_1 \\ \vdots \\ a_n \end{Bmatrix}]_S - \#$$

$$\begin{array}{cccccc} 1 & 2 & 3 & 4 & 5 & 6\Rightarrow \\ 0 & 2 & 0 & 0 & 5 & 0 \end{array}$$

$\mathcal{T}_{12}$ (continued)

Condition: no substring of X is an S.

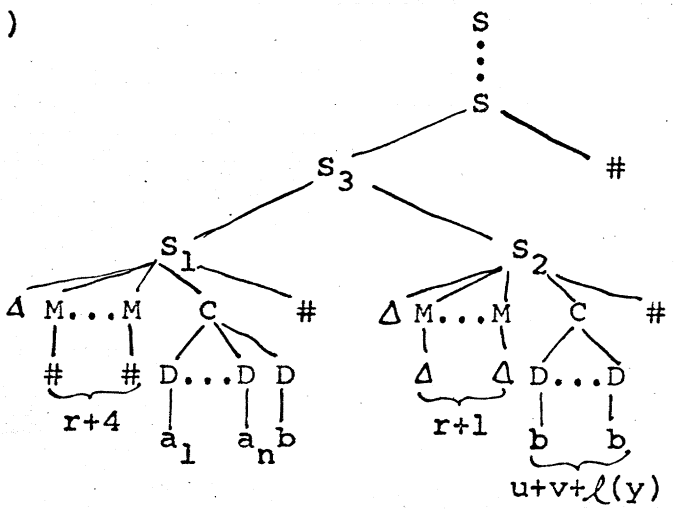
Now let $\mathcal{G}$ be the grammar with the terminal alphabet $\{a_1,\ldots a_n\}$ consisting of $(\mathcal{P},\mathcal{T})$. We must show that $L(\mathcal{G}) = L(G)$.

($\supseteq$) First, we will show the $L(G)\subseteq L(\mathcal{G})$, verifying as we go that all the transformations of $\mathcal{T}$ obey the structure-preserving constraint. So let $z\in L(G)$. Then there is some transformational derivation of z with respect to G such that the line in the derivation after T_1 through T_4 and only those transformations have applied is (6).

$$(6)\ [_S[_S\ldots[_S[_S\#^r a_1\ldots a_n b\#^4 b]_S b^u\#yb^v]_S\#]_S\ldots\#]_S$$

Then the following transformational derivation with respect to $\mathcal{G}$ from the phrase marker (7) generated by $\mathcal{P}$ also yields z.

(7)



where total number of S's = $2\ell(y)+u(r+2)+v+w+3$

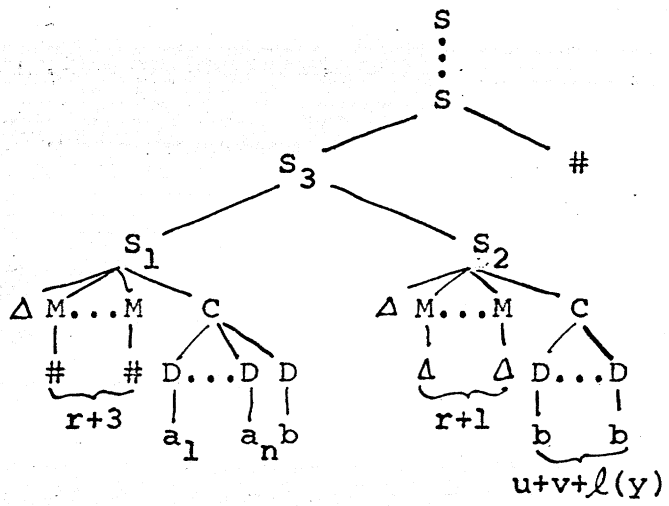
First cycle: Vacuous, since no transformations apply to S_1.

Second cycle: Vacuous, since no transformations apply to S_2.

Third cycle: Normally, $\mathcal{T}_1$ would apply to S_3, substituting b for any a_i already on the copy of the tape contained in S_2. In this case, since the tape contains only b's, $\mathcal{T}_1$ is not applicable. Notice, however, that $\mathcal{T}_1$ consists of one structure-preserving elementary deletion (of 6) and a structure-preserving elementary substitution of 2 (which is a D) for 4 (which is a D), so $\mathcal{T}_1$ obeys the structure-preserving constraint.

However, $\mathcal{T}_2$ applies, filling in the last b on the tape in S_2 and changing (7) to the derived structure (8).

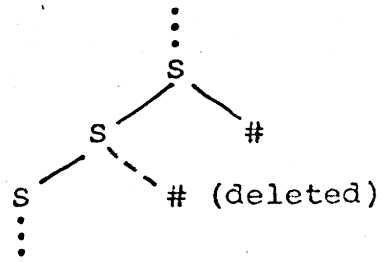
(8)



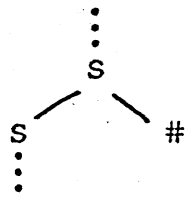
$\mathcal{T}_2$ consists of structure-preserving deletions of 2, 5, and 9 and a structure-preserving substitution of 4 for 7, so it obeys the structure-preserving constraint.

Notice also that at every cycle after this one, the final # is deleted, leaving a structure like (9) which reduces by the A-over-A principle[6] to (10).

(9)

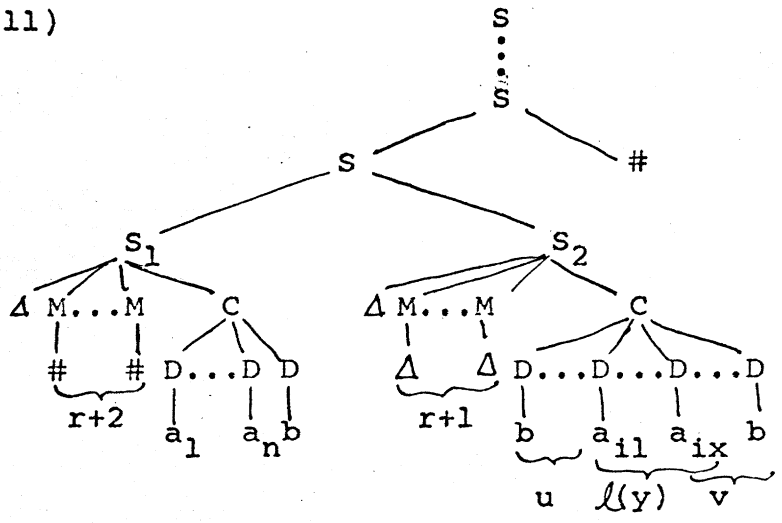


(10)



Next $\ell(y)$ cycles: $\mathcal{T}_3$ applies, filling in one letter of the input string each time. On all but the last of these cycles, we choose the first analysis of constituents 1 and 2; as soon as we use the second analysis, $\mathcal{T}_3$ cannot apply any more and we must go on. Suppose the input string is $a_{i1}\ldots a_{ix}$. Then at the end of the $\ell(y)$+3rd cycle, the derived structure looks like (11).

(11)



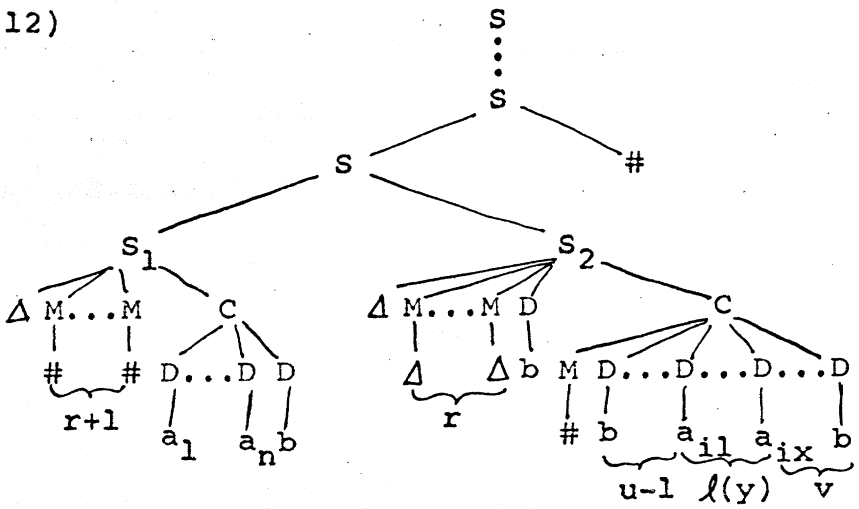
$\mathcal{T}_3$ obeys the structure-preserving constraint, since it

consists of structure-preserving deletions of 2 and 9 and a structure-preserving substitution of 4 for 6.

$\ell(y)$+4th cycle: Either $\mathcal{T}_4$ or $\mathcal{T}_5$ applies, in general. Both put the simulated machine in its initial state and start the process of putting it into reading position. $\mathcal{T}_5$ handles the case where there are no blanks on the tape to the left of the leftmost a_i (in which case the head is already in reading position); $\mathcal{T}_4$ handles the case where there are intervening blanks. Both $\mathcal{T}_4$ and $\mathcal{T}_5$ contain structure-preserving deletions of 2 and structure-preserving substitutions of 2 for 6; $\mathcal{T}_4$ also has a minor movement of 6 (which is an M and therefore a function category node) over 7 (a D, which is a lexical category node). Moreover, 6 and 7 are mutually in construction since S_2, the node immediately dominating M, dominates D. So $\mathcal{T}_4$ and $\mathcal{T}_5$ obey the structure-preserving constraint.

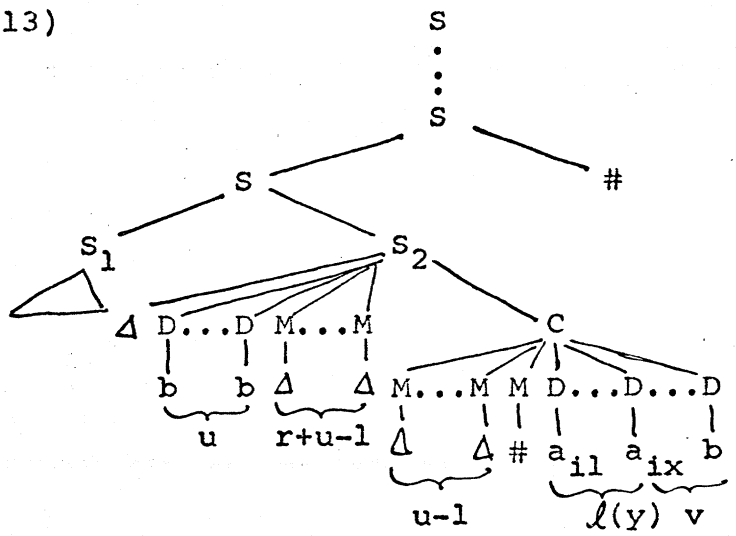
After $\mathcal{T}_4$ applies to phrase marker (11) but before the end of the cycle, the derived structure is (12).

(12)



Now, and on the next u(r+1)-1 cycles, $\mathcal{T}_6$ applies, bringing the reading head into position adjacent to the input. $\mathcal{T}_6$ consists of a structure-preserving deletion of 7 and a minor movement of 4 over 5. At the end of the $\mathcal{L}(y)$+u(r+1)+3rd cycle, the derived phrase marker is (13), assuming r+1> u (there can be at most u constituents to the left of a_{il} under C -- cf. footnote 4). This assumption causes no loss of generality.

(13)



Suppose Z computes an output, given input y, in exactly w steps. Then on the next w cycles, either $\mathcal{T}_{7j}$ applies or $\mathcal{T}_{8kl}$ through $\mathcal{T}_{8kr+1}$ apply or $\mathcal{T}_{9\ell 1}$ through $\mathcal{T}_{9\ell r+1}$ apply. $\mathcal{T}_{7j}$ simulates a print instruction by overprinting the old symbol with the new and changing the state of the machine. To accomplish this it makes r+1 structure-preserving elementary substitutions and one structure-preserving deletion. $\mathcal{T}_{8kl}$ through $\mathcal{T}_{8kr+1}$ simulate a move left instruction by moving the reading head left one symbol and

changing the state of the machine. In detail, $\mathcal{T}_{8k1}$ changes the state with r structure-preserving elementary subsitutions and moves the leftmost M over the D immediately to its left. $\mathcal{T}_{8k2}$ through $\mathcal{T}_{8kr+1}$ each make one minor movement of an M over that same D, until at the end of the cycle the whole reading head has moved over the D. Each of these transformations consists of one minor movement, and $\mathcal{T}_{8kr+1}$ has a structure-preserving elementary deletion as well. $\mathcal{T}_{8kr+2}$ covers the case when an instruction says to move left but there is no tape to the left of the reading head (this situation cannot arise with the phrase marker we have been examining, since we have provided u b's to the left of the input). In this case, $\mathcal{T}_{8kr+2}$ deletes everything but the Δ in S_2, so the phrase marker no longer meets the structural descriptions of any transformations. All further cycles are vacuous, so the resulting string contains a symbol not in the terminal vocabulary of the transformational language, and the phrase marker is filtered out. All the deletions of $\mathcal{T}_{8kr+2}$ are structure-preserving, of course. The move right transformations $\mathcal{T}_{9\ell 1}$ through $\mathcal{T}_{9\ell r+1}$ and $\mathcal{T}_{9\ell r+2}$ work analogously. All these transformations, as we have seen, obey the structure-preserving constraint.

Let us work through a specific example. Suppose Z has two states, q_1 and q_2, and the following instructions:

(14) $I_1 = (q_1, a_1, L, q_1)$

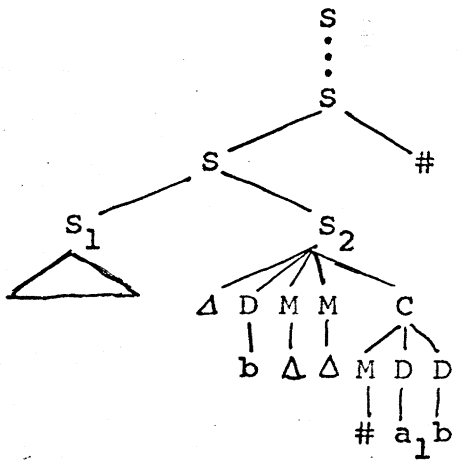
$I_2 = (q_1, b, a_1, q_2)$

$I_3 = (q_2, a_1, R, q_2)$

$$I_4 = (q_2, b, a_2, q_1)$$

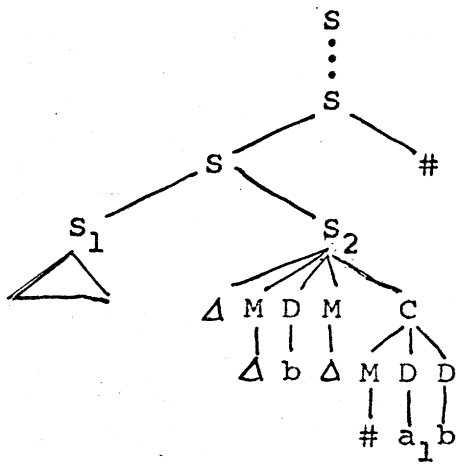
We will compute the output of Z, given the input a_1. (We can check by hand simulation that u = 1 and v = 1.) So the transformations apply first to a phrase marker (15).

(15)



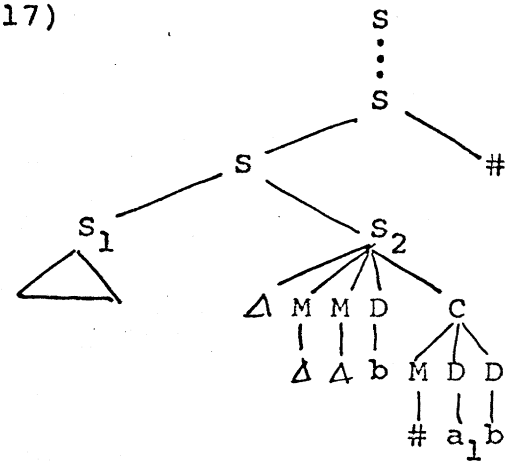
First, $\mathcal{T}_{811}$ applies:

(16)



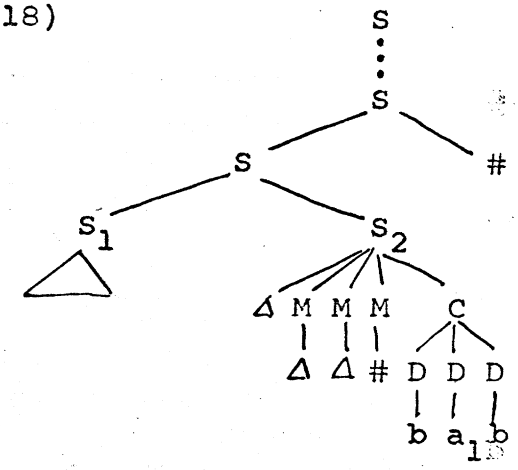
Then $\mathcal{T}_{812}$ applies:

(17)



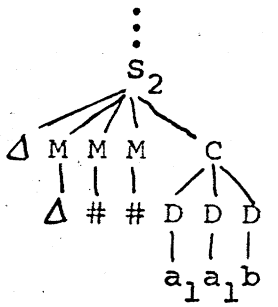
Finally, $\mathcal{T}_{813}$ applies.

(18)



On the next cycle, $\mathcal{T}_{72}$ applies:

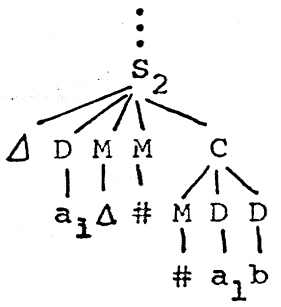
(19)



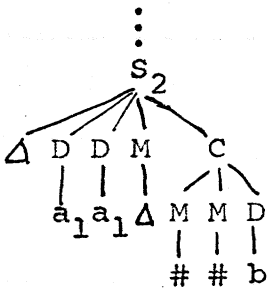
On the next two cycles, $\mathcal{T}_{931}$, $\mathcal{T}_{932}$, and $\mathcal{T}_{933}$ apply, leaving (20) at the end of the first and (21) at the end of

the second cycle.

(20)

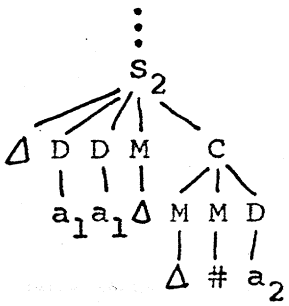


(21)



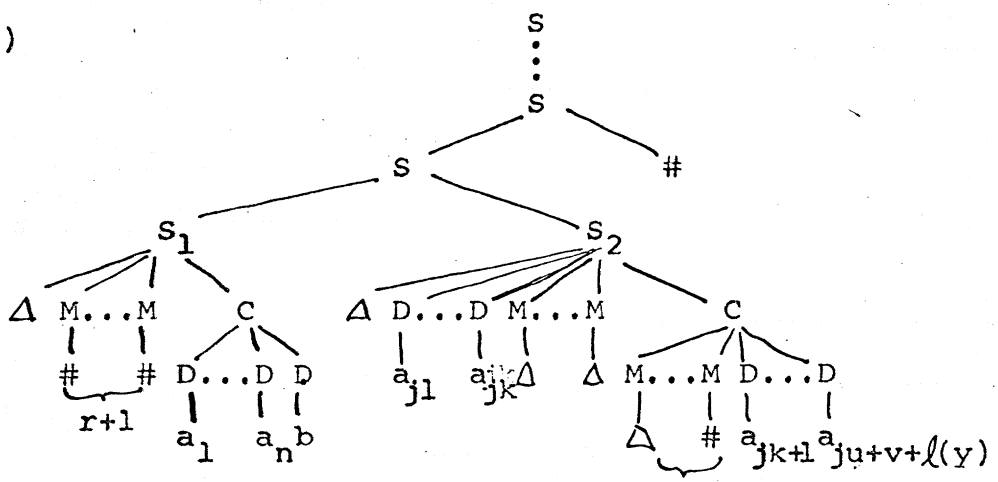
On the next cycle, $\mathcal{T}_{74}$ applies and the machine halts, giving us the derived phrase marker (22) and eventually the output string $a_1a_1a_2$.

(22)



Let us return to phrase marker (13). At the end of the $\ell(y)+u(r+1)+w+3$rd cycle, after all the computations of Z on y have been simulated, we are left with a phrase marker like (23), assuming that the output of Z is $a_{j1}\ldots a_{ju+v+\ell(y)}$ (where b can be in this string as well as the a_i's).

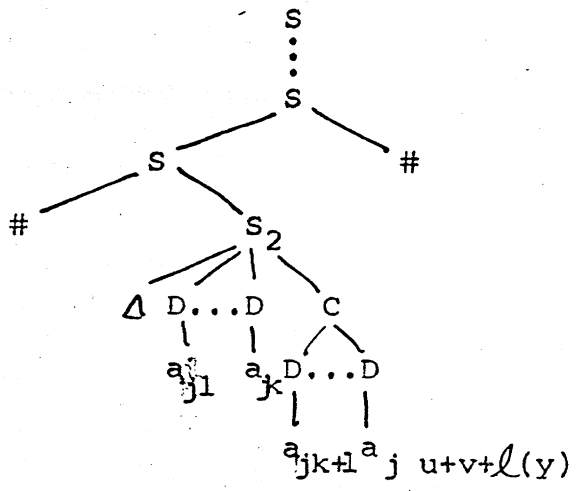
(23)



f, where q_f is the state Z halted in.

$\ell(y)+u(r+1)+w+4$th cycle: $\mathcal{T}_{10}$ applies, deleting everything but one # in S_1 and deleting the reading head (i.e. all the M nodes) in S_2. All the deletions are structure-preserving. This gives us phrase marker (24).

(24)

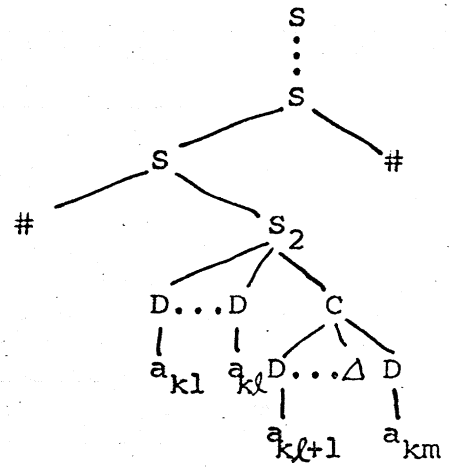


Then $\mathcal{T}_{11}$ applies for the first time, passing the Δ of S_2 over the D immediately to its right by a minor movement, and deleting the D if it dominates B.

Next $u+v+\ell(y)-2$ cycles: $\mathcal{T}_{11}$ applies, passing Δ over the D immediately to its right as above. At the end of the

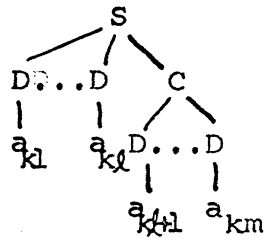
$2\ell(y)+u(r+2)+v+w+2$nd cycle, we have (25), assuming that $a_{k1}\ldots a_{km}$ is the string $a_{j1}\ldots a_{u+v+\ell(y)}$ with internal blanks deleted.

(25)



$2\ell(y)+u(r+2)+v+w+3$rd cycle: $\mathcal{T}_{12}$ applies, deleting everything but the string of a's in S_2, leaving (26):

(26)



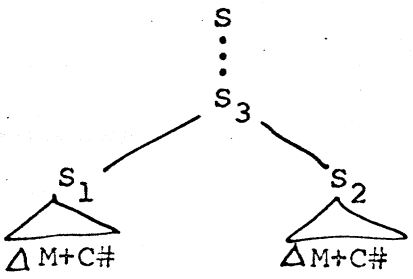
But the debracketization of (26) is just $a_{k1}\ldots a_{km} = z$, the string computed by Z given input y. So $z \in L(\mathcal{G})$, and this completes the proof that $L(G) \subseteq L(\mathcal{G})$.

($\subseteq$) For the other half of the proof of the theorem, that $L(\mathcal{G}) \subseteq L(G)$, we will show that phrase markers other than (7) either lead to a sentence of L(G) or cause the derivation to block.

If S_3, the S immediately dominating the lowest S in the

phrase marker, is expanded except as in (27),

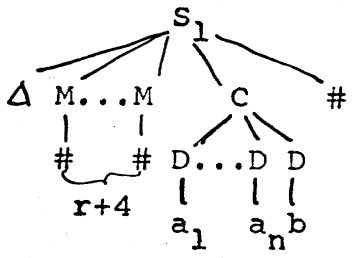
(27)



e.g. as S# or as ΔM+C#, the phrase marker does not meet the structural description of any of the transformations. Hence all the cycles of the derivation are vacuous and the debracketization of the last line contains several Δ's (and #'s). Since these symbols are not members of the terminal vocabulary of the transformational language, their presence in a string filters out the original phrase marker as a possible deep structure for the language. So the lowest expanded S must be expanded as in (27).

If S_1 looks like anything but (28),

(28)



The phrase marker does not meet the structural description of $\mathcal{T}_2$, and its final # is not deleted. Hence no later transformations, and in particular $\mathcal{T}_{10}$, can apply. So the Δ in S_1 is never deleted and the debracketization of the final phrase marker will contain this Δ. So any phrase marker

whose leftmost inner S is unlike (28) will be filtered out as a possible deep structure. In fact, we can assume from now on that the leftmost inner S of any phrase marker we discuss is expanded as in (28).

If the M+ in S_2 does not contain exactly r+1 M's dominating Δ, the structural description of $\mathcal{T}_4$ (and therefore of all later transformations) is not met. All further cycles are vacuous, and the Δ in S_2 filters out the phrase marker as a possible deep structure. So M+ must contain r+1 M's dominating Δ.

Since $\mathcal{T}_1$ and $\mathcal{T}_2$ fill up the copy of the tape in S_2 with b's, the initial content of the D nodes in S_2 is irrelevant. Let us consider, however, the number of D's in C; suppose there are exactly t of them.

If $t < \ell(y)$, only a truncated portion of y can fit on the tape. But in applying $\mathcal{T}_{7j}$ through $\mathcal{T}_{9\ell r+2}$ the derivation will simulate the action of Z on this truncated input as if it were an input itself. So if there are enough b's to the right and left of the truncated input for Z to complete its calculation, we will get a sentence in L(G) (although probably not the sentence Z calculates given input y). If not, either $\mathcal{T}_{8kr+2}$ or $\mathcal{T}_{9\ell r+2}$ will apply, depending on whether we run off the left or right end of the tape. In either case, no further transformations can apply and the final string will be Δ, which is not in $L(\mathcal{G})$.

If $\ell(y) \leq t < \ell(y)+u+v$ and $\mathcal{T}_3$ writes the whole input string y on the tape, then either $\mathcal{T}_{8kr+2}$ or $\mathcal{T}_{9\ell r+2}$ will apply,

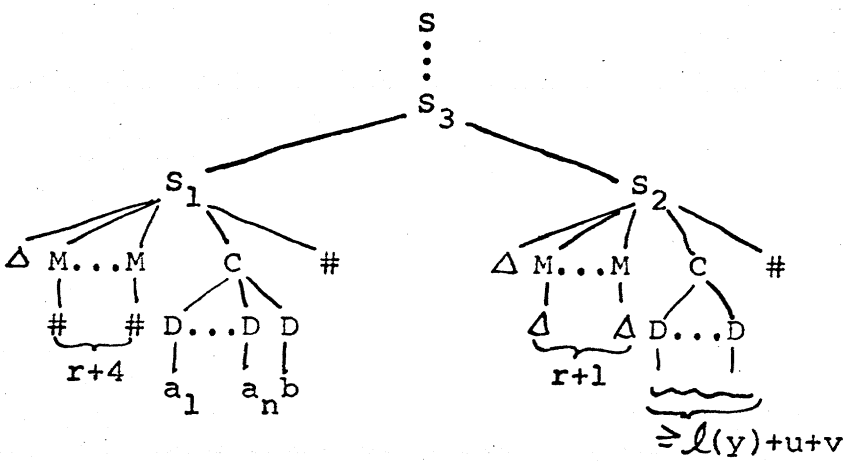
with the same results as above. If $\ell(y) \in t < \ell(y)+u+v$ and $\mathcal{T}_3$ does not write the whole input string on the tape, the same reasoning holds as for the $t < \ell(y)$ case above.

If $t \geq \ell(y)+u+v$ and $\mathcal{T}_3$ writes the whole input string y on the tape, the derivation will go through as in the first half of the proof because there will be enough room on the tape for Z to perform its computation, and $\mathcal{T}_{11}$ and $\mathcal{T}_{12}$ wipe out any extra blanks. If $t \geq \ell(y)+u+v$ and $\mathcal{T}_3$ does not write the whole input string y on the tape, the same reasoning holds as for the $t < \ell(y)$ case above.

So if we get any string in $L(\mathcal{G})$ at all, this string is also in L(G).

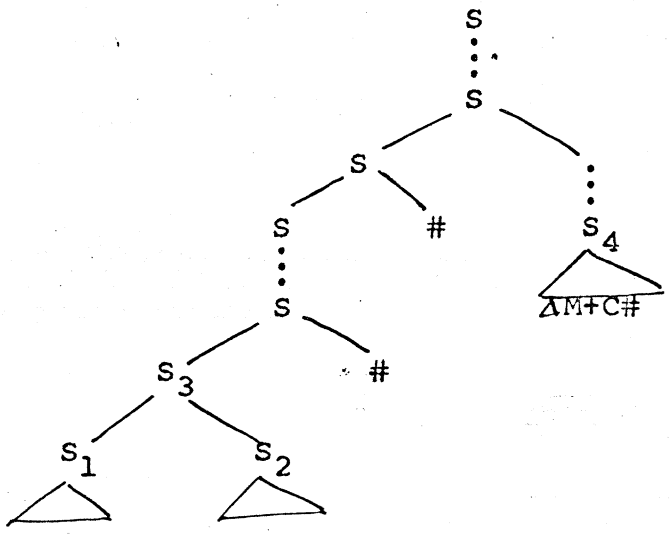
So far the only phrase marker that can possibly yield a sentence of $L(\mathcal{G})$ looks like (29):

(29)



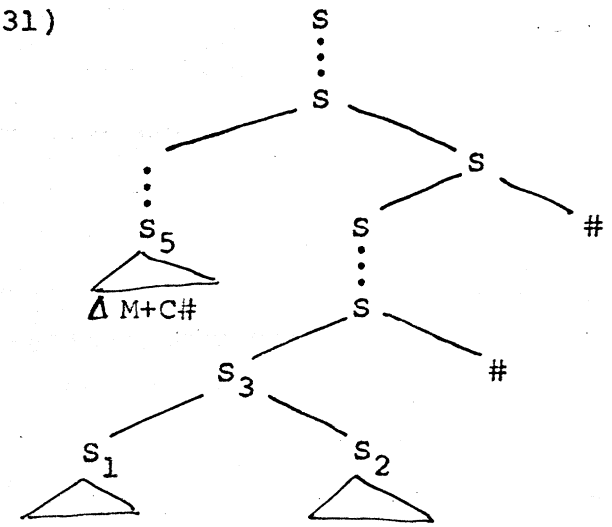
Now suppose we have extra expanded S's above S_3, as in (30) or (31). Let S_4 be the highest and rightmost such S and let S_5 be the highest and leftmost such S.

(30)



If nothing goes wrong before we get there, let us consider the first cycle which includes both S_3 and S_4. This cycle and all further cycles are vacuous, since (30) does not meet the structural description of any of the transformations. So the Δ in S_4 is never deleted and (30) is filtered out as a possible deep structure for $L(\mathcal{G})$.

(31)



Again, if nothing goes wrong before we get there, let us

consider the first cycle which includes both S_5 and S_3. All further cycles are vacuous, since (31) with its three embedded S's does not meet the structural description of any transformation. So the Δ in S_5 is never deleted and (31), like (30), is filtered out.

We have been assuming so far that the phrase marker we are looking at has enough embedded S's to give us as many cycles as we need. We must now verify this assumption. Moreover, since we are working inductively and have already used this assumption above, we must be careful not to support it with statements we have used it to prove.

Given the rules of $\mathcal{P}$, we know that there is at least one cycle in every phrase marker.

If there are not enough cycles for $\mathcal{T}_1$ and $\mathcal{T}_2$ to fill the whole tape with blanks, the Δ in S_1 is sufficient to filter out the phrase marker. Similarly, there must be enough cycles for $\mathcal{T}_3$ to fill in the input string, for $\mathcal{T}_4$ through $\mathcal{T}_6$ to put the machine into its initial state and in reading position, for $\mathcal{T}_{7j}$ through $\mathcal{T}_{9\ell r+2}$ to apply until the machine halts given this input, and for $\mathcal{T}_{10}$ and $\mathcal{T}_{11}$ to apply. Finally, $\mathcal{T}_{12}$ must apply or else Δ will be left in S_2.

Now suppose the phrase marker has more cycles than are necessary to complete a derivation. Notice that after $\mathcal{T}_{12}$ has applied once, no other transformations can apply since the derived phrase marker does not meet the structural description of any transformation. So if the phrase marker contains extra #'s above where we want to stop, further

cycles will be vacuous and the final string will contain at least one #, which is sufficient to filter out the original phrase marker. If there are only extra expanded S's above this point, again all further cycles are vacuous since $\mathcal{T}_{12}$ has already applied. So the obligatory Δ in any such S will not be deleted and will filter out the phrase marker.

Finally, notice that if machine Z does not halt on a particular input, the corresponding phrase marker we would need to start with for $\mathcal{T}_1$ through $\mathcal{T}_{12}$ to apply has an infinite number of cycles and hence is not generated by $\mathcal{P}$. In short, we never get an output where Z does not halt.

This completes the proof that $L(\mathcal{G}) \subseteq L(G)$ and the proof of the theorem.

FOOTNOTES

1. This is the key fact needed to show that every r.e. language can be generated by a context-sensitive-based transformational grammar obeying the structure-preserving constraint.

2. This is a change from the definition in Kravif (1970), where the formalization of Emonds' constraint was worked out in detail. There, a structure-preserving deletion was a substitution of Δ for non-Δ material. In that formulation, however, any deletion leaves behind a Δ which later filters out the phrase marker as a possible deep structure.

3. This is an unfortunate choice of terminology, since the root of a tree is usually considered to be its highest node--actually its least node according to the partial ordering on the tree.

4. We could equally well define only elementary minor movement left and right adjunctions and define a minor movement rule of B over C as a left or right adjunction of B to C and a structure-preserving elementary deletion of the original B. This formulation changes some of the trees in section 3 but otherwise has no effect on the results of this paper.

5. It is not clear whether Emonds intends the phrase nodes to be exactly the non-preterminal nodes and the lexical and function category nodes to be exactly the preterminals. We interpret his remark (1970;4) that the proposed restrictions do not "exclude lexical (or other) nodes from dominating phrase nodes" as implying that he does not so intend, and have in our phrase structure grammar a lexical category node C that is not a preterminal. We can regularize C's status by replacing the rule $C \rightarrow D+$ by $C \rightarrow CD$, but only at the expense of making C a phrase node. Choosing the second alternative does not affect the results of this paper.

6. The A-over-A principle reduces trees. In phrase markers this reduction is automatic, since a phrase marker is a <u>set</u> of structural analyses. Two structural analyses S yield only one S in the phrase marker.

BIBLIOGRAPHY

Bowers, J. (1970), Rough draft of doctoral dissertation, unpublished, MIT.

Bresnan, J. (1970), "On Complementizers: Toward a Syntactic Theory of Complement Types", Foundations of Language 6:3.

Emonds, J. (1969), "Constraints on Transformations", Indiana University Linguistics Circle multilith, Bloomington, Indiana.

Emonds, J. (1970), Root and Structure-Preserving Transformations, unpublished doctoral dissertation, MIT.

Kimball, J. (1967), "Predicates Definable over Transformational Derivations by Intersection with Regular Languages", Information and Control 11.

Kravif, D. (1970), "On the Generative Capacity of Constrained Transformational Grammars", unpublished, MIT.

Kravif, D. (1971), "Weak Generative Capacity and Emonds' Constraint", Linguistic Inquiry 2:1.

Peters, S. (1970), "Why There are Many 'Universal' Bases", Papers in Linguistics 2:1.

Peters, S. and R.W.Ritchie (1969a), "On the Generative Power of Transformational Grammars", Technical Report CSci 69-2-3, University of Washington, Seattle, Washington.

Peters, S. and R.W.Ritchie (1969b), "On Restricting the Base Component of Transformational Grammars", unpublished, University of Texas, Austin, Texas.