INFORMATION AND CONTROL 15, 297–334 (1969)

A Mathematical Model of Transformational Grammars*

SEYMOUR GINSBURG

System Development Corporation and the University of Southern California, Los Angeles, California 90007

AND

BARBARA PARTEE

University of California at Los Angeles and System Development Corporation, Los Angeles, California 90024

A mathematical model of transformational grammars is presented which incorporates most current versions. Among other things, the model has a formal definition of transformations and a general scheme for ordering them. Numerous examples are given to illustrate the theory.

INTRODUCTION

One of the primary goals of linguistics is to find a suitable model of natural (human) language. Among the models proposed, one of them, the transformational grammar (abbreviated T-grammar) seems very promising and, in the past decade, has received considerable attention from linguists. At present, there are numerous alternative forms of the model under serious consideration. In fact, there is a noticeable lack of unanimity among linguists with respect to a number of aspects of the theory. As yet, no mathematical model has been given which encompasses most of these different versions of a T-grammar.[1] The purpose of

* Research sponsored in part by the Air Force Cambridge Research Laboratories, Office of Aerospace Research, USAF, under Contract F1962867C0008, and by the Air Force Office of Scientific Research, Office of Aerospace Research, USAF, under AFOSR Grant No. AF-AFOSR-1203-67.

[1] A formalization of one particular version has been made in a forthcoming paper (Peters and Ritchie). Some extensions of this version and some consequences have been explored in Kimball (1967). A computer model has recently been proposed by Friedman (1968).

this paper is to propose one such mathematical model. In particular, a model is presented which appears to incorporate most current versions of transformational theory.[2] As would be expected from linguistic experience, the model is general enough to yield all recursively enumerable sets as its generated languages. It appears likely that special natural restrictions on the model will yield many new types of formal languages of interest to the mathematician, linguist, and formal language theorist. In particular, because of the richness of the model, it seems reasonable to hope that appropriate restrictions will provide "grammars" for a number of families of formal languages introduced in the past few years by families of acceptors.

The paper is divided into four sections. In Section 1 we present a linguistic example to motivate the need for, and some underlying concepts of, T-grammars. In Section 2 we abstract some of the ideas of Section 1 and present a mathematical model of T-grammars. Section 3 contains some examples illustrating the ordering of T-rules and an example of a T-grammar in its entirety. Section 4 contains a number of possible restrictions on the general model. Taken in various combinations, these restrictions provide submodels which correspond to certain descriptions of T-grammars in the literature.

Stylistically, the paper has been written for the mathematical linguist, the mathematician, and the formal language theorist. In this way we hope to (1) make some of the issues more precise to the linguist and (2) make accessible to the mathematician and formal language theorist a number of definitions and concepts deeply embedded in the linguistic literature.

INFORMAL LINGUISTIC NOTIONS

As mentioned in the introduction, the purpose of the present paper is to obtain a formal definition of T-grammars. In this section we present an example to motivate the need for, and informally determine some features of, a T-grammar. The linguist will find nothing new here and we suggest he skip to Section 2.

Consider the following five English sentences:

(1′) The dog chased the cat
(2′) The cat ate the mouse
(3′) The mouse was eaten by the cat

[2] One important exception is the notion of "syntactic features" described in Chomsky (1965) and included in most subsequent T-grammars.

(4′) The cat that the dog chased ate the mouse
(5′) The mouse was eaten by the cat that was chased by the dog

Using certain morphophonemic rules (of no concern to us here), sentences (1′)–(5′) can be derived from the following forms:

(1) \$ the dog P_{ast} chase the cat \$[3]
(2) \$ the cat P_{ast} eat the mouse \$
(3) \$ the mouse P_{ast} be E_n eat by the cat \$
(4) \$ the cat that the dog P_{ast} chase P_{ast} eat the mouse \$
(5) \$ the mouse P_{ast} be E_n eat by the cat that P_{ast} be E_n chase by the dog \$

In the remainder of this section, we shall construct a sample T-grammar, which generates "sentences" (1)–(5) (and other sentences as well).

Sentences (1) and (2) are generated by the following CF rules[4]:

$$\begin{array}{ll}
S \to \$S'\$ & \left\{\begin{array}{l} V_b \to \text{chase} \\ V_b \to \text{eat} \end{array}\right. \\
S' \to N_pA_{ux}V_p & \\
V_p \to V_bN_p & \left\{\begin{array}{l} N_n \to \text{dog} \\ N_n \to \text{cat} \\ N_n \to \text{mouse} \end{array}\right. \\
N_p \to \text{the } N_n & \\
A_{ux} \to T_{ns} & \\
T_{ns} \to P_{ast} &
\end{array}$$

The nonterminals S', N_p, A_{ux}, V_p, V_b, N_n, and T_{ns} abbreviate "sentence," "noun phrase," "verb auxiliary," "verb phrase," "verb," "noun," and "tense," respectively. The S is the start symbol. The terminal symbols are P_{ast},[5] the, chase, eat, dog, cat, mouse, and the boundary marker \$. Clearly the rules generate (1) and (2) from S. If we were concerned only with sentences (1) and (2), then the above rules could be simplified. However, the rules as given provide linguistic structure needed to deal with sentences (3), (4), and (5).

The passive sentence (3) cannot be generated by these CF rules. Clearly we could add extra rules to generate (3). However, this would complicate the grammar considerably and would not indicate that (3)

[3] The symbol \$ is used as a sentence boundary marker.

[4] We assume that the reader has a working knowlecge of both context-sensitive (abbreviated CS) rules and context-free (abbreviated CF) rules.

[5] The symbols "P_{ast}," "the," "chase," etc., are to be regarded as single symbols.

is related to (2). For linguistic purposes it is thus necessary to introduce a different kind of rule. The type of rule suggested by Chomsky (1957) and currently receiving much attention is the "transformation rule" (abbreviated "T-rule"). Roughly described, a T-rule is a procedure which operates on "structured" sentences to form other structured sentences. This leads to the notion of a postulated set of "simple" structured sentences (in the current example, the set of sentences generated by the CF rules), and T-rules to generate more complicated structured sentences from them.

To see how a T-rule is used to generate (3), let us consider how passive sentences are formed. Loosely described, a passive sentence is formed from its corresponding active sentence (e.g., (3) from (2)) by interchanging subject and object, adding "by" before the original subject, and changing the verb to passive form. Linguistically speaking, a sentence of the form

(6) $\$N_p^{(1)}\ T_{ns}\ V_bN_p^{(2)}\$$

is made into a passive sentence by converting it to[6]

(7) $\$N_p^{(2)}\ \underbrace{T_{ns}\ \text{be}\ E_n}\ V_b\ \underbrace{\text{by}\ N_p^{(1)}}\$.$

In sentence (2), $N_p^{(1)}$ is "the cat," T_{ns} is "P_{ast}," V_b is "eat," and $N_p^{(2)}$ is "the mouse." Restructuring (2) as above gives (3).

The mapping that converts sentences of the form (6) into sentences of the form (7) is a T-rule, called T_{pas}, which can be symbolically written as follows:

T_{pas} : Domain statement: $\begin{array}{cccccccc} X^{(1)} & \$ & N_p^{(1)} & T_{ns} & V_b & N_p^{(2)} & \$ & X^{(2)} \\ \alpha_1 & \alpha_2 & \alpha_3 & \alpha_4 & \alpha_5 & \alpha_6 & \alpha_7 & \alpha_8 \end{array}$

Structural change statement: $\textcircled{1}-\textcircled{2}-\textcircled{6}-\textcircled{4}$ be $E_n-\textcircled{5}-$by$\textcircled{3}-\textcircled{7}-\textcircled{8}$

$N_p^{(1)}$ and $N_p^{(2)}$ are noun phrases[7] not necessarily identical to each other. $X^{(1)}$ and $X^{(2)}$ are variables representing arbitrary strings that are not necessarily identical to each other. In general, if a symbol occurs more than once in a domain statement, then each occurrence of the string it represents must be the same. In (6), each of the distinct symbols $X^{(1)}$ and $X^{(2)}$ is the empty string ϵ.[8] The domain statement specifies the form of the sentences to which the rule may be applied. The structural change

[6] E_n is a new terminal symbol which morphophonemically converts the verb following it to passive (i.e., past participle) form.

[7] Actually, $N_p^{(1)}$ and $N_p^{(2)}$ are not noun phrases; they are symbols representing trees that generate noun phrases. The meaning attached to these symbols is discussed rigorously in C of Section 2.

[8] An instance will shortly be given (Fig. 5), where neither $X^{(1)}$ nor $X^{(2)}$ is the empty string.

statement indicates the effect of the rule in a manner to be precisely explained in section two.

Since a T-rule applies to any sentence which can be parsed into the form given in the domain statement, in effect it applies to the tree associated with the CF derivation of the sentence.[9] The T-rule maps that

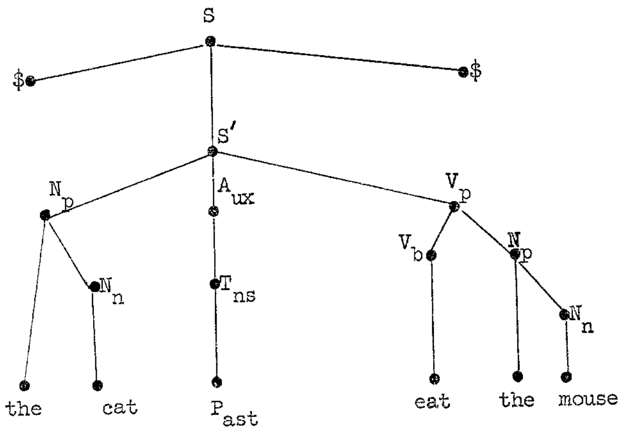

FIG. 1.[10]

tree into a new one in accordance with the specifications of the structural change statement in a manner to be described in Section 2. In our example, the tree in Fig. 1 is mapped into the tree in Fig. 2.

As noted above, the result of applying a T-rule to a tree structure is again a tree structure. This allows an iteration of the procedure, that is, allows T-rules to be applied in sequence (in a manner more fully described later). Thus, for example, the "question" T-rule (not given in the present paper) can be applied to passive sentences.

Consider sentence (4). Intuitively, sentence (4) is a combination of sentences (1) and (2). It turns out that we need another CF rule and another T-rule to generate (4). Let us add to our CF rules the CF rule

$$\mathrm{N_p} \rightarrow \text{the } \mathrm{N_n}\$\ S'\$.$$

The CF rules then generate

(4″) \$ the cat \$ the dog P_{ast} chase the cat \$ P_{ast} eat the mouse \$

[9] In Section 2 and thenceforth, we regard a T-rule as mapping a tree into a tree. In this section we occasionally speak loosely of a T-rule operating on a sentence.

[10] We write trees from top to bottom; i.e., the root is at the top.

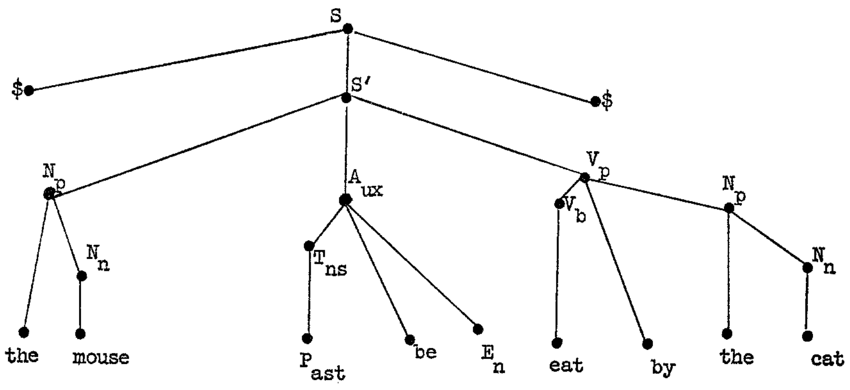
S
$
S′
$
N_p
A_ux
V_p
N_n
T_ns
V_b
N_p
N_n
the
mouse
P_ast
be
E_n
eat
by
the
cat

FIG. 2.

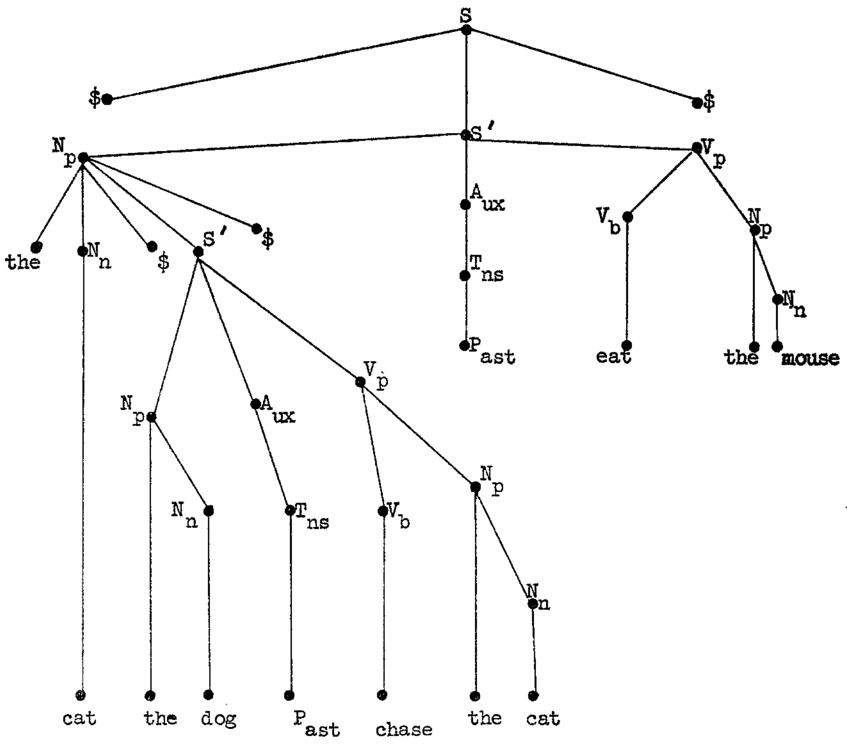
S
$
$
N_p
S′
V_p
the
N_n
$
S′
$
A_ux
T_ns
P_ast
V_b
eat
N_p
N_n
the
mouse
N_p
A_ux
V_p
N_n
T_ns
V_b
N_p
N_n
cat
the
dog
P_ast
chase
the
cat

FIG. 3.

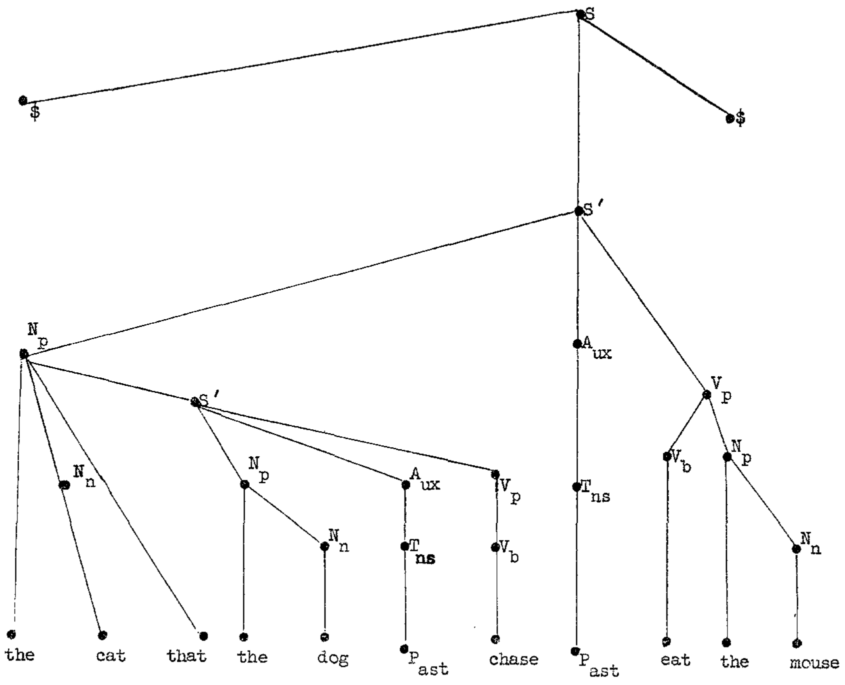

FIG. 4.

with the tree structure in Fig. 3. To derive (4), we need a T-rule which will convert the embedded sentence into a relative clause. The T-rule which accomplishes this is T_{rel}. In particular, T_{rel} checks to see that there is a noun in the embedded sentence which is identical to the noun immediately preceding the embedded sentence. If there is, then the noun in the embedded sentence is replaced by "that"[11], and the word "that" moved to the front of the embedded sentence. In a manner similar to T_{pas}, T_{rel} may be symbolically written as follows:

T_{rel} : Domain statement: $\underset{\alpha_1}{X^{(1)}}\ \underset{\alpha_2}{N_n^{(1)}}\ \underset{\alpha_3}{\$}\ \underset{\alpha_4}{X^{(2)}}\ \underset{\alpha_5}{\text{the}}\ \underset{\alpha_6}{N_n^{(1)}}\ \underset{\alpha_7}{X^{(3)}}\ \underset{\alpha_8}{\$}\ \underset{\alpha_9}{X^{(4)}}$.

Condition: N_p generates $\alpha_5\alpha_6$.

Structural change statement: ①–②–that–④–∅–∅–⑦–∅–⑨

The condition is a constraint on the domain of the T-rule. This par-

[11] The noun may also be replaced by "who" or "which," or it may even be deleted. For simplicity, only one case is considered here.

ticular condition means that in Fig. 3, N_p generates $\alpha_5\alpha_6$ with superscripts removed. The fact that the superscripts on the two occurrences of N_n are the same means that the subtrees they generate must be the same.

In the structural change statement, $\varnothing$ denotes the empty tree (generating the empty string ϵ). Its occurrence results in the deletion of the corresponding subtree designated by the domain statement. Thus, in this example, the second boundary marker \$ and the noun phrase "the cat" of the inner sentence are deleted (replaced by the empty tree), and "that" is substituted for the first boundary marker \$ of the inner sentence.

The terminal string associated with the tree in Fig. 3 generated by the CF rules satisfies the domain statement of T_{rel} in the following way:

\$ the	cat	\$	the dog P_{ast} chase	the	cat	ϵ	\$	P_{ast} eat the mouse \$
$X^{(1)}$	$N_n^{(1)}$	\$	$X^{(2)}$	the	$N_n^{(1)}$	$X^{(3)}$	\$	$X^{(4)}$
α_1	α_2	α_3	α_4	α_5	α_6	α_7	α_8	α_9

Therefore, T_{rel} can be applied to the tree in Fig. 3 to derive the tree in Fig. 4. The terminal string of the tree in Fig. 4 is the sentence (4).

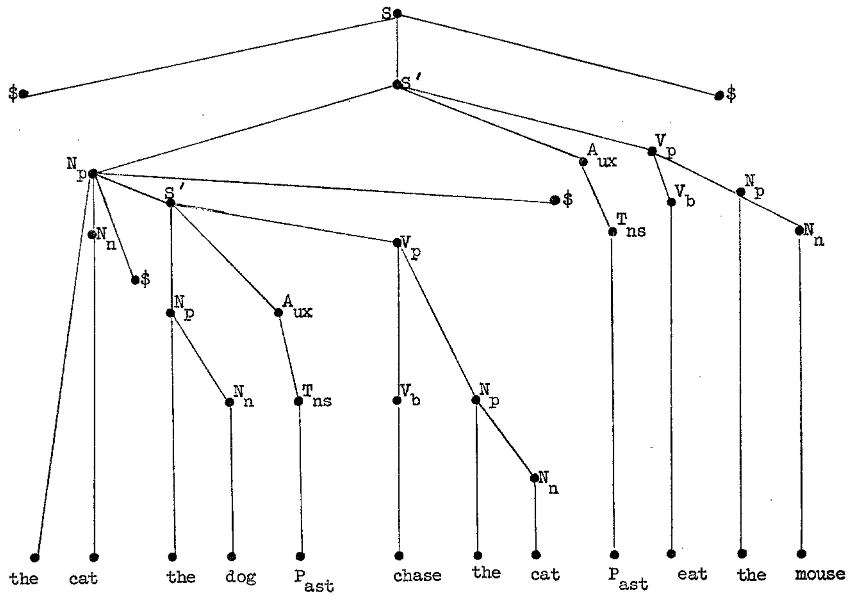

FIG. 5.

Finally, consider sentence (5). Sentence (5) requires no additional rules. Its derivation involves T_{pas}, T_{rel}, and T_{pas} as follows: The CF rules generate the tree in Fig. 5. The tree in Fig. 5 satisfies the domain statement of T_{pas} as follows:

$$\underbrace{\$\text{ the cat}}_{\alpha_1}\ \underbrace{\$}_{\alpha_2}\ \underbrace{\text{the dog}}_{\alpha_3}\ \underbrace{P_{ast}}_{\alpha_4}\ \underbrace{\text{chase}}_{\alpha_5}\ \underbrace{\text{the cat}}_{\alpha_6}\ \underbrace{\$}_{\alpha_7}\ \underbrace{P_{ast}\text{ eat the mouse }\$}_{\alpha_8}$$

The T_{pas} converts the tree in Fig. 5 into the tree in Fig. 6. The tree in Fig. 6 satisfies the domain statement of T_{rel} as follows:

$$\underbrace{\$\text{ the}}_{\alpha_1}\ \underbrace{\text{cat}}_{\alpha_2}\ \underbrace{\$}_{\alpha_3}\ \underbrace{\epsilon}_{\alpha_4}\ \underbrace{\text{the}}_{\alpha_5}\ \underbrace{\text{cat}}_{\alpha_6}\ \underbrace{P_{ast}\text{ be }E_n\text{ chase by the dog}}_{\alpha_7}\ \underbrace{\$}_{\alpha_8}$$

$$\underbrace{P_{ast}\text{ eat the mouse }\$}_{\alpha_9}$$

The T_{rel} converts the tree in Fig. 6 into the tree in Fig. 7. The tree in Fig. 7 satisfies the domain statement of T_{pas} as follows:

$$\underbrace{\epsilon}_{\alpha_1}\ \underbrace{\$}_{\alpha_2}\ \underbrace{\text{the cat that }P_{ast}\text{ be }E_n\text{ chase by the dog}}_{\alpha_3}\ \underbrace{P_{ast}}_{\alpha_4}\ \underbrace{\text{eat}}_{\alpha_5}\ \underbrace{\text{the mouse}}_{\alpha_6}$$

$$\underbrace{\$}_{\alpha_7}\ \underbrace{\epsilon}_{\alpha_8}$$

The T_{pas} converts the tree in Fig. 7 into the tree in Fig. 8. The terminal string of the tree in Fig. 8 is sentence (5), so that the derivation is now complete.

In the above example, we included a set of CF rules as part of the T-grammar. Since our main concern in the sequel is with T-rules and their usage, we are interested in CF or CS rules only insofar as they define a set of trees which serve as input to the T-rules. Now the linguistic literature generally includes ordered sets of CF or CS rules, ordered, frequently, in quite special ways (Chomsky, 1965). Therefore, we shall simplify our discussion by assuming that part of the T-grammar consists of a "base," i.e., set of finite trees. We shall not be concerned with the mechanism that generates this set of trees.

In summary, then, the above example illustrates the utility of a gram-

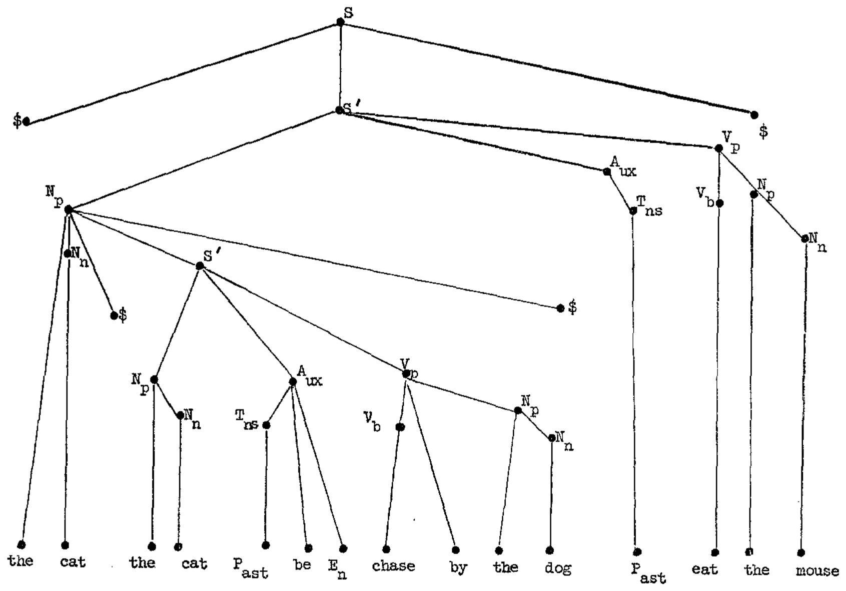
S
S′
$
$
Vp
Aux
Np
Vb
Np
Tns
Nn
Nn
S′
$
$
Np
Aux
Vp
Nn
Tns
Vb
Np
Nn
the
cat
the
cat
Past
be
En
chase
by
the
dog
Past
eat
the
mouse

FIG. 6.

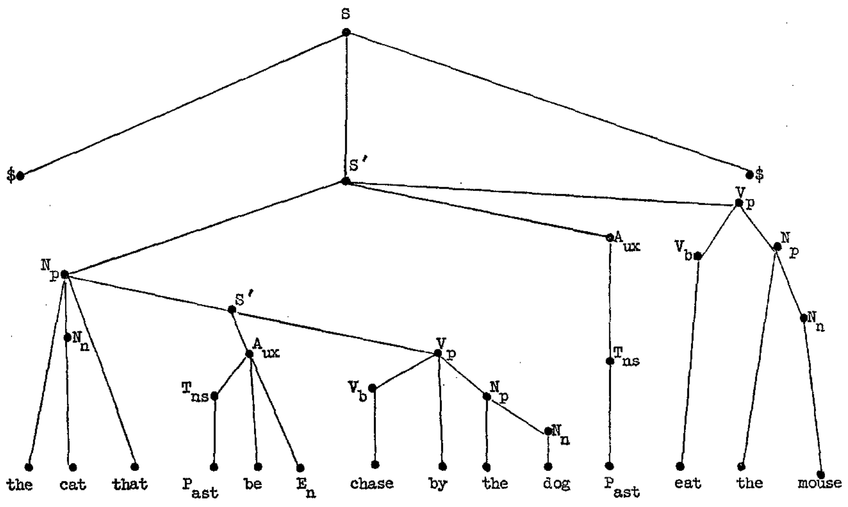
S
$
S′
$
Vp
Aux
Vb
Np
Np
S′
Nn
Nn
Aux
Vp
Tns
Tns
Vb
Np
Nn
the
cat
that
Past
be
En
chase
by
the
dog
Past
eat
the
mouse

FIG. 7.

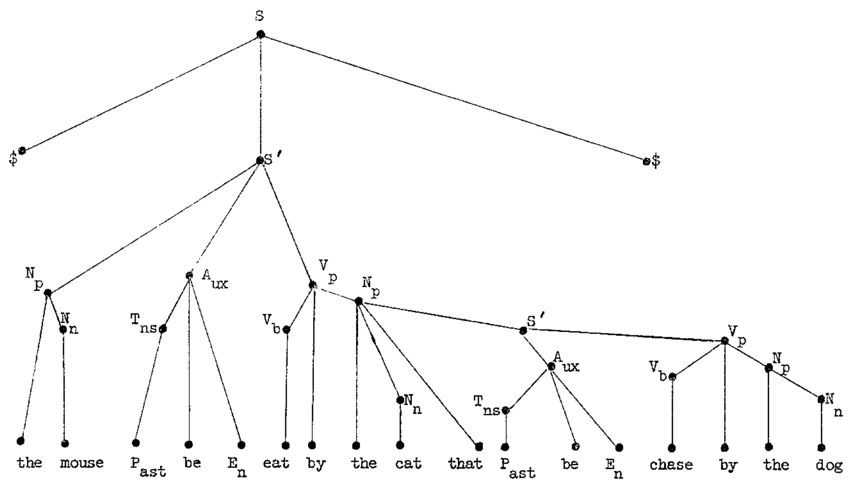

FIG. 8.

mar with the following features:

(1) The grammar contains a base consisting of trees with associated terminal strings.

(2) The grammar contains T-rules which map trees onto trees. In particular, they apply to trees of the base to yield structures to which further T-rules can apply, etc.

(3) Each T-rule is specified by a domain statement and a structural change statement.

In the next section, we shall formalize the definition of a T-grammar on the basis of the above three properties. We shall also incorporate some properties not exemplified in the sample grammar. For example, the ordering of the T-rules and the definition of the languages generated are concepts which require specification.

2. FORMALIZATION

In this section we present a mathematical model of T-grammars. In particular, we shall define T-rules, the operation of T-rules, the order in which the T-rules may be applied (i.e., the "traffic rules"), and the language generated by the grammar.

A. Base

As was mentioned in the preceding section, T-grammars are commonly said to have a set of CF or CS rules that generate trees to which T-rules apply. In actual practice, special orders are imposed on the use of these rules (Peters, 1966), so that their usage differs from that given in either CF or CS grammars. Since our interest in this paper is in the T-rules and not the schemes used for deriving an original set of trees, we shall ignore such schemes and assume the existence of a given set of trees, called a "base."[12]

In order to discuss "bases," we first consider "finite trees with labeled nodes," henceforth abbreviated "trees." We assume each non-empty tree has a "root."[13] Following the convention in the linguistic literature, we draw trees with the root at the top. We also need

Agreement. All trees in this paper are assumed to have the property that if a node has exactly one node immediately below it, then the two nodes have different labels. Each tree not of this form is to be automatically identified with the tree obtained by identifying such pairs of nodes in the obvious manner.

As an example, the tree in Fig. 1 is automatically considered to be the tree in Fig. 2.

The agreement is included because the T-rules as defined in our model cannot distinguish between such identified configurations. The condition is linguistically reasonable (and its equivalent has always been included in linguistic definitions of CF and CS grammars). It may be regarded as a special case of tree pruning. (See Ross (1965; 1967, Chapter 3) and Section 3, Example 6, below.)

We now briefly describe a base and the type of trees in it.

Definition. A *base* $\mathcal{B}$ is a triple (B, V, Σ), where V is a finite non-empty set, Σ (the set of *terminal* symbols) is a subset of V, and B is a set of non-empty trees, each of which (i) has its terminal nodes labeled with symbols of Σ (so that terminal nodes are labeled with terminals[14]) and (ii) has its nonterminal nodes labeled with symbols of $V - \Sigma$.

[12] A suggestion for specifying the set of base trees by a means other than CF or CS rules appears in McCawley (1968).

[13] We assume the reader is familiar with the concept of a finite tree with labeled nodes and a root.

[14] For some purposes it might be desirable to allow terminal nodes to be labeled with symbols of $\Sigma \cup \{\epsilon\}$, but for the purposes of this paper there is no need to do so.

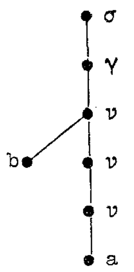

FIG. 1.

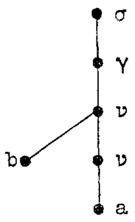

FIG. 2.

DEFINITION. The *associated word* or the *terminal string* of a tree τ satisfying (i) and (ii) above, whether or not in B, is the word formed by concatenating the labels of the terminal nodes of τ in the usual left-to-right order.

B. TRANSFORMATION RULES

As already noted, a T-rule is a rule which transforms trees into other trees. It has two parts: a "domain statement" and a "structural change statement." In this subsection we shall formally define T-rules, and in the next subsection we shall specify how they operate on trees.

Notation. Given $\mathcal{B} = (B, V, \Sigma)$, let Σ' denote a finite superset of Σ such that Σ' and $V - \Sigma$ are disjoint. Let X denote a new symbol (a "dummy variable") and let $V' = V \cup \Sigma' \cup \{X\}$. Let V'' denote a set consisting of V' plus a finite number of new symbols of the form $Y^{(i)}$, where Y is in V' and i is a nonnegative integer.

We now define the notion of a "domain statement."

DEFINITION. A *Boolean domain statement* is any expression formed as follows:

(1) Each string $\alpha_1 \cdots \alpha_k$, $k \geq 1$, every α_i in V'', is a Boolean domain statement.

(2) If D_1 and D_2 are Boolean domain statements, then $(D_1 \vee D_2)$, $(D_1 \wedge D_2)$, and $\sim D_1$ are Boolean domain statements.

We shall omit parentheses whenever no ambiguity results.

DEFINITION. A *domain statement* is any expression of the form D_I or $D_I \wedge D_O$, where D_I is a string $\alpha_1 \cdots \alpha_k$, $k \geq 1$, every α_i in V'', and D_O is a Boolean domain statement.

A typical domain statement is written as either D, $D = D_I$, or $D = D_I \wedge D_O$, with D_I and D_O as above.

We now define the notion of a "structural change statement."

DEFINITION. Let $D = D_I$ or $D = D_I \wedge D_O$ be a domain statement, with $D_I = \alpha_1 \cdots \alpha_n$. A *structural change statement* (*on* D) is a string $C = \beta_1 \cdots \beta_n$ such that for each i, $\textcircled{i}$ is a new symbol, $\beta_i = \textcircled{i}$ if i is in H, and β_i is in $((\Sigma' \cup \{\varnothing, \textcircled{1}, \cdots, \textcircled{n}\}) - H)^{+}$[15] if i is not in H, where $H = \{j/\alpha_j \text{ in } \{X, X^{(1)}, X^{(2)}, \cdots\}\}$.

It will be subsequently seen that the above conditions on H guarantee that a T-rule cannot alter or move any parts of the tree that are designated by one of the variables X, $X^{(1)}$, etc. The manner in which the structual change statement produces new constituent structure from old is specified in the next subsection.

Note that whereas $D = D_I$ or $D = D_I \wedge D_O$, C is defined only with respect to D_I. Intuitively described, the role of D_O is only to add further conditions, beyond those specified in D_I, which a tree must satisfy in order for the T-rule to be applicable to it. (See Section C below.) The role of D_I is both to impose conditions which must be satisfied by a tree and to define the structure on which the changes specified by C will operate.

We are now able to formally define a T-rule.

DEFINITION. A *transformation rule* T is an ordered pair (D, C), where D is a domain statement and C is a structural change statement on D.

We illustrate the concept of a T-rule by examining $\mathrm{T}_{\mathrm{pas}}$ and $\mathrm{T}_{\mathrm{rel}}$.

EXAMPLE. Consider $\mathrm{T}_{\mathrm{pas}}$. The domain statement is D_I, where $D_I = \alpha_1 \cdots \alpha_8 = X^{(1)}\$\mathrm{N}_{\mathrm{p}}^{(1)}\mathrm{T}_{\mathrm{ns}}\mathrm{V}_{\mathrm{b}}\mathrm{N}_{\mathrm{p}}^{(2)}\$X^{(2)}$. The structural-change statement is $\beta_1 \cdots \beta_8 = \textcircled{1}\text{–}\textcircled{2}\text{–}\textcircled{6}\text{–}\textcircled{4}\text{ be }\mathrm{E}_{\mathrm{n}}\text{–}\textcircled{5}\text{–by }\textcircled{3}\text{–}\textcircled{7}\text{–}\textcircled{8}$.[16]

EXAMPLE. Consider $\mathrm{T}_{\mathrm{rel}}$. A first thought is that the domain statement is D_I, where $D_I = \alpha_1 \cdots \alpha_9 = X^{(1)}N_n^{(1)}\$X^{(2)} \text{ the } N_n^{(1)}X^{(3)}\$X^{(4)}$. However, this does not provide for the restriction that $\alpha_5\alpha_6$ must be generated by N_{p}. The solution is to have the condition that $\alpha_5\alpha_6$ is generated by N_{p} expressed in a Boolean domain statement. In particular, the domain

[15] For each set E, E^+ is the set of all non-empty strings of elements from E and $E^* = E^+ \cup \{\epsilon\}$.

[16] The dash symbol,–, is used to separate the β_i. Thus, for example, $\beta_3 = \textcircled{6}$ and $\beta_4 = \textcircled{4}$ be E_{n}.

statement is $D_I \wedge D_O$, where

$$D_I = \alpha_1 \cdots \alpha_9 = X^{(1)}\mathrm{N}_{\mathrm{n}}^{(1)}\$X^{(2)} \text{ the } \mathrm{N}_{\mathrm{n}}^{(1)}X^{(3)}\$X^{(4)},$$

and $D_O = \alpha_{10} \cdots \alpha_{17} = X^{(1)}\mathrm{N}_{\mathrm{n}}^{(1)}\$X^{(2)}\mathrm{N}_{\mathrm{p}}X^{(3)}\$X^{(4)}$.

We shall see in C below how the new domain statement requires $\alpha_5\alpha_6$ to be generated by N_{p}. The structural change statement is $\beta_1 \cdots \beta_9 =$ ①–②–that–④–∅–∅–⑦–∅–⑨.

C. Use of T-Rules

We now turn to the problem of how to apply a T-rule. Hereafter, each tree mentioned is assumed to (i) have its terminal nodes labeled with symbols of Σ', and (ii) have its nonterminal nodes labeled with symbols of $V - \Sigma'$. We shall see that a T-rule is applied to certain trees to yield other trees. In particular, a T-rule may be applied to a given tree τ if τ "satisfies" the domain statement of the T-rule in a manner defined below.

Definition. A *graph-assignment* f is a function over V'' defined as follows:

(i) For $\xi = \gamma$ or $\xi = \gamma^{(i)}$, γ in $V' - \{X\}$, $f(\xi)$ is a tree with root labeled γ.

(ii) For each ξ in $\{X, X^{(1)}, X^{(2)}, \cdots\}$, $f(\xi)$ is an arbitrary concatenation of (possibly null) trees.[17]

Definition. (1) A tree τ *satisfies a string* $\alpha_1 \cdots \alpha_k$ *for a graph-assignment* f if $f(\alpha_1) \cdots f(\alpha_k)$ is a cofinal residual proper subgraph of τ[18]

(2) A tree τ *satisfies a conjunction* $D_1 \wedge \cdots \wedge D_p \wedge \sim D_{p+1} \wedge \cdots \sim D_{p+m}$ ($p, m \geq 0$, each D_i a string), if there exists a graph assignment f such that τ satisfies each D_i, $1 \leq i \leq p$, for f, and there exists no graph assignment f_1 such that $(\alpha)\tau$ satisfies each D_i, $1 \leq i \leq p$, for f_1 and $(\beta)\tau$ satisfies at least one D_{p+j}, $1 \leq j \leq m$, for f_1.

(3) A tree τ *satisfies a disjunction* $\mathfrak{D}_1 \vee \cdots \vee \mathfrak{D}_q$, $q \geq 1$, each $\mathfrak{D}_i$ a conjunction of form (2) above, if τ satisfies at least one $\mathfrak{D}_i$, $1 \leq i \leq q$.

For a given tree τ and a given string D, there may be zero, one, or more than one graph assignment for which τ satisfies D.

[17] Let $\eta_1, \cdots, \eta_k$ be subgraphs of a tree. Then $\eta_1 \cdots \eta_k$, the *concatenation* of the $f(\alpha_i)$, is the graph obtained by placing the η_i next to each other in the order given, taking the nodes of the η_i as pairwise disjoint.

[18] A subgraph τ_1 of a tree τ_2 is *cofinal residual*, if (a) for each node ν_2 in τ_2 there is a node ν_1 in τ_1 which is either ν_2 itself or below ν_2, and (b) if ν_1 is a node of τ_1 and ν_2 is a node of τ_2 below ν_1, then ν_2 is in τ_1.
A subgraph of τ_2 is *proper* if it is not τ_2 itself.

The adjective "proper" in condition (1) rules out the root of any of the trees $f(\alpha_i)$ being the root of the given tree τ. The necessity for having the root of no $f(\alpha_i)$ be the root of the tree τ arises in the description of the effect of a structural change, which could otherwise replace a tree with a sequence of trees having no common root.

We mention without proof that any two disjunctions of form (3) above, which are logically equivalent, are either both satisfied or both not satisfied for a given tree. Because of this, we shall say that a tree τ satisfies a domain statement if τ satisfies some equivalent domain statement of form (3) above.

EXAMPLE. Let $D = J^{(1)}XUJ^{(1)}$. Let f_1 be the following function on V'': $f_1(J^{(1)})$ is the tree in Fig. 3, $f_1(U)$ is the tree in Fig. 4, $f_1(X)$ is the structure in Fig. 5, and f_1 is irrelevant elsewhere. Let τ_1, τ_2, and τ_3 be the trees in Figs. 6, 7, and 8, respectively.

Then τ_1 satisfies D for f_1, τ_2 does not satisfy D for f_1 (but does for some other graph assignment f), and τ_3 does not satisfy D for any graph assignment f.

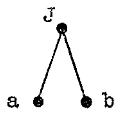

FIG. 3.

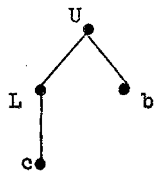

FIG. 4.

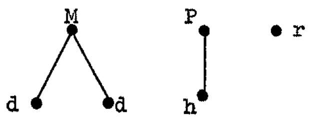

FIG. 5.

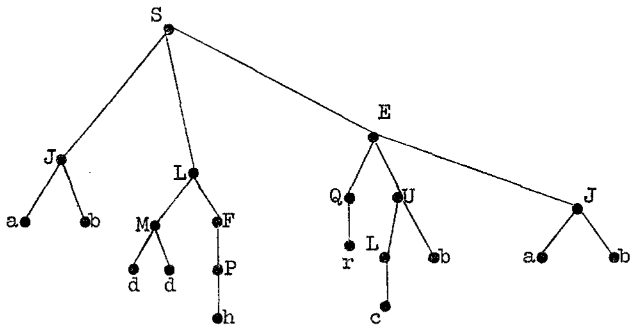

FIG. 6.

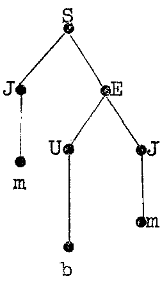

FIG. 7.

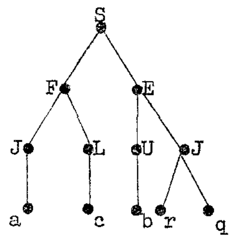

FIG. 8.

EXAMPLE. Let τ_1, τ_2, and τ_3 be as above. Let $D = D_I \wedge D_O = J^{(1)}XJ^{(2)} \wedge ((J^{(1)}XJ^{(1)} \wedge J^{(1)}E \vee (JLE \wedge {\sim}JMFE))$. Then τ_2 and τ_3 satisfy D but τ_1 does not.

We are now ready to define how a T-rule changes a tree τ into a tree τ', written symbolically $\tau \Rightarrow_T \tau'$ or $\tau \Rightarrow \tau'$.

Notation. Given a T-rule, $T = (D, C)$, where $D = D_I$ or $D = D_I \wedge D_O$, $D_I = \alpha_1 \cdots \alpha_n$, and $C = \beta_1 \cdots \beta_n$, and given a graph assignment f for which τ satisfies D, let $g_{T,f}$, abbreviated g when T and f are understood, be the function on $(\{\varnothing, \textcircled{1}, \cdots, \textcircled{n}\} \cup \Sigma')^+$ defined as follows:

(i) $g(\xi) = f(\alpha_k)$, for $\xi = \textcircled{k}$, $\textcircled{k}$ in $\{\textcircled{1}, \cdots, \textcircled{n}\}$;
(ii) $g(\xi)$ is the one-node tree with label ξ, for ξ in Σ';
(iii) $g(\varnothing)$ is the empty tree; and
(iv) $g(\xi_1 \cdots \xi_r) = g(\xi_1) \cdots g(\xi_r)$, for all $\xi_1, \cdots, \xi_r$ in

$$\{\varnothing, \textcircled{1}, \cdots, \textcircled{n}\} \cup \Sigma'.$$

Notation. Let $T = (D, C)$ be a T-rule and τ a tree which satisfies D for assignment f. Let $D = D_I \wedge D_O$ or $D = D_I$, with $D_I = \alpha_1 \cdots \alpha_n$. Then $f(\alpha_1) \cdots f(\alpha_n)$ is a cofinal residual subgraph of τ. Let τ_0 be the structure formed from τ by replacing, for each i such that $\beta_i \neq \textcircled{i}$, $f(\alpha_i)$ with $g(\beta_i)$ in the obvious manner. Write $\tau \Rightarrow_T \tau'$ or $\tau \Rightarrow \tau'$, where τ' is the largest tree (possibly the empty tree) contained in τ_0.[19]

Clearly there always exists a unique such tree τ' (for a given f).

To illustrate the relation of τ' to τ_0, if τ_0 is the structure in Fig. 9 then τ' is the tree in Fig. 10. Thus, τ' need not coincide with τ_0.

[19] Recall that as used here, a tree must have each of its terminal nodes labeled with terminals. The structure τ_0 need not have its terminal nodes labeled with terminals. (See Fig. 9.)

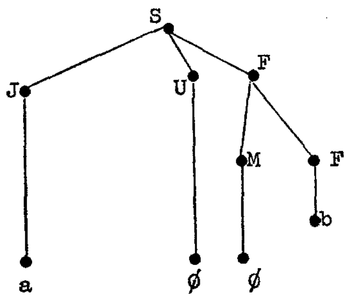

FIG. 9.

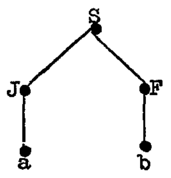

FIG. 10.

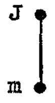

FIG. 11.

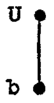

FIG. 12.

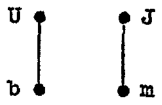

FIG. 13.

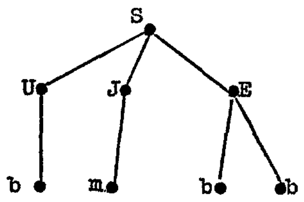

FIG. 14.

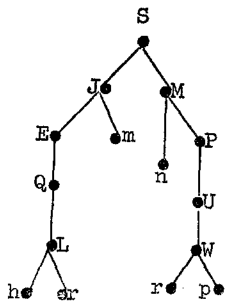

FIG. 15.

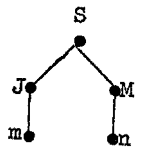

FIG. 16.

EXAMPLES. (a) T_{pas} changes the tree in Fig. 7 of section one into the tree in Fig. 8 of Section 1.

(b) Let $T = (D, C)$, where $D = D_I$, $D_I = \alpha_1 \cdots \alpha_4 = J^{(1)}XUJ^{(1)}$ and $C = \beta_1 \cdots \beta_4 = \textcircled{3}\textcircled{1}\text{-}\textcircled{2}\text{-}\varnothing\text{-}bb$. Let f be defined (in part) by $f(J^{(1)})$ being the tree in Fig. 11, $f(X) = \varnothing$, and $f(U)$ being the tree in Fig. 12. The tree τ_2 in Fig. 7 above satisfies D for f. For the function $g_{T,f} = g$, $g(\beta_1) = g(\textcircled{3}\textcircled{1})$ is the structure in Fig. 13, $g(\beta_2)$ is $\varnothing$, $g(\beta_3)$ is $\varnothing$, and

$g(\beta_4) = g(bb)$ is the sequence of trees $b^{\bullet}\,{}^{\bullet}b$. Then $\tau_2 \Rightarrow_T \tau'$, where τ' is the tree in Fig. 14.

(c) Let $T = (D, C)$, where $D = D_I = \alpha_1\alpha_2\alpha_3 = LXU$, and $C = \beta_1\beta_2\beta_3 = \varnothing\text{-}②\text{-}\varnothing$. Then $\tau \Rightarrow_T \tau'$, where τ is in Fig. 15 and τ' is in Fig. 16.

D. Transformational Grammar

We are now ready to formally define the concept of a T-grammar. In addition to the underlying base and the T-rules, the grammar will also have (i) a set Σ_A' of distinguished symbols over which the language generated by the grammar is defined, and (ii) a specification of the ordering of the T-rules. We shall discuss this last component after presenting the definition of the grammar.

Definition. A *transformational grammar* is a 4-tuple $\mathcal{G} = (\mathcal{B}, \Delta, R, \Omega)$ satisfying the following conditions:

(1) $\mathcal{B} = (B, V, \Sigma)$ is a base of trees.

(2) $\Delta = (\Sigma', \Sigma_A', X, V'')$, where

(a) Σ' is a finite set of abstract symbols such that $\Sigma \subseteq \Sigma'$, and Σ' is disjoint from $V - \Sigma$.

(b) X is an abstract symbol not in $V \cup \Sigma'$.

(c) $\Sigma_A' \subseteq \Sigma'$.

(d) V'' is a set containing $V \cup \Sigma' \cup \{X\}$ plus a finite number of new symbols of the form $Y^{(i)}$, where Y is in $V \cup \Sigma' \cup \{X\}$, and i is a nonnegative integer.

(3) R is a finite set of T-rules (D, C) with respect to V'', Σ', and X.

(4) $\Omega = (K, \mathfrak{N}, \delta, s_0)$, where

(a) K is a finite set (of *states*) and s_0 (the *start* state) is in K.

(b) $\mathfrak{N} = \{N(s)/s \text{ in } K\}$, where each $N(s)$ is a partially ordered set over $R \cup \{\textsc{stop}\}$, STOP being a special symbol occurring only as a maximal element.[20]

(c) δ is a mapping from a subset of $K \times R$ into K (*next state* function).

The components $\mathcal{B}$, Δ, and R have already been discussed. (As noted above, Σ_A' is the set of symbols over which the language generated by the grammar is defined.) The component Ω pertains to the order of application of the T-rules. In general, it is not true that a T-rule may be

[20] Given a partially ordered set $(Y, \leq)$, x is *maximal* (minimal) if, for y in Y, $y \leq x$ $(y \geq x)$ implies $y = x$. An element x is *isolated* if there is no $y \neq x$ such that either $x \leq y$ or $y \leq x$.

applied any time its domain statement is satisfied. In fact, all T-grammars in the literature which we have seen have restrictions on the order of application of the T-rules. We now show how Ω effects restrictions on the use of the rules, and we shall see that the restrictions in the literature can be obtained by appropriate Ω.

Informally, K is a set of states which record the past history of the use of the rules. Call a T-rule, T, "applicable" to a given tree τ at state s if (i) τ satisfies the domain statement of T, (ii) T occurs in $N(s)$, and ii) T is not preceded in $N(s)$ by a T-rule whose domain statement τ satisfies. A "derivation" consists of either

(α) a sequence of T-rules $T_1, \cdots, T_k$, states $s_1, \cdots, s_{k+1}$, and trees $\tau_1, \cdots, \tau_{k+1}$ such that for each i, $1 \leq i \leq k$, T_i is applicable to τ_i at s_i, yielding τ_{i+1}, and $s_{i+1} = \delta(s_i, T_i)$; or

(β) a sequence of T-rules $T_1, \cdots, T_{k-1}$, states $s_1, \cdots, s_k$, and trees $\tau_1, \cdots, \tau_k$ such that for each i, $1 \leq i < k$, T_i is applicable to τ_i at s_i, yielding τ_{i+1}, $s_{i+1} = \delta(s_i, T_i)$, and STOP is in $N(s_k)$ and is not preceded by any T-rule applicable to τ_k at s_k.

We now make the preceding motions more precise.

Notation. Let $\mathcal{G} = (\mathcal{B}, \Delta, R, \Omega)$ be a T-grammar and $\vdash$ the relation between pairs (τ, s), where τ is a tree and s is in K, defined as follows: Write $(\tau, s) \vdash (\tau', s')$ if there exists some T in R such that

(i) $\tau \Rightarrow_T \tau'$,

(ii) T occurs in $N(s)$,

(iii) there is no T' in $N(s)$ and τ'', with $T' < T$, such that $\tau \Rightarrow_{T'} \tau''$,

and

(iv) $s' = \delta(s, T)$.

Let $\vdash^*$ be the transitive, reflexive closure of $\vdash$, i.e., $(\tau, s) \vdash^* (\tau', s')$ if there exists $n \geq 1$ and $\tau_1 = \tau$, $s_1 = s, \cdots, \tau_n = \tau'$, $s_n = s'$ such that $(\tau_i, s_i) \vdash (\tau_{i+1}, s_{i+1})$ for each i, $i < n$. Write $(\tau, s)| \vdash$ STOP, if STOP is in $N(s)$ and there is no T in $N(s)$ and τ', with $T <$ STOP, such that $\tau \Rightarrow_T \tau'$.

Using the above notation, we now define the "language generated by a T-grammar."

DEFINITION. A word w is said to be *generated* by a T-grammar $\mathcal{G} = (\mathcal{B}, \Delta, R, \Omega)$, where $\mathcal{B} = (B, V, \Sigma)$, $\Delta = (\Sigma', \Sigma_A', X, V'')$, and $\Omega = (K, \mathfrak{N}, \delta, s_0)$ if

(i) w is in $(\Sigma_A')^*$, and

(ii) there exist τ, τ', and s' such that τ is in B, $(\tau, s_0) \vdash^* (\tau', s')| \vdash$ STOP, and w is the terminal string of τ'.
The *language* generated by $\mathcal{G}$, denoted by $L(\mathcal{G})$, is the set of all words generated by $\mathcal{G}$.

The purpose of STOP and of the distinction between acceptable terminals (Σ_A') and unacceptable terminals $(\Sigma' - \Sigma_A')$ is to eliminate from the language of a T-grammar, by methods akin to the methods in the linguistics literature, certain words generated by the rules of the grammar. The STOP symbol is primarily used to allow only certain words, derived at intermediate stages of words in the language, to also be in the language. The unacceptable terminals (cf. the use of # in [Chomsky (1965)]) are used primarily in those derivations which never lead to a word in the language. These uses are not mutually exclusive, however; i.e., each of these mechanisms may occasionally be used to perform the function of the other. In Section 4, we discuss some methods for reducing this "filter power."

In passing, we mention that the model as defined, with CF base, generates all recursively enumerable sets. The analogous theorem, for a different model, is proved in Kimball (1967).

3. ORDERING AND GRAMMAR EXAMPLES

We now present some examples to illustrate the above notions. The first five give some indication of the variety and generality of the use of partially ordered sets to specify the order of rule applications. The sixth illustrates the operation of "tree-pruning." The last one is a simple example of a T-grammar.

(1) The set of T-rules in virtually every T-grammar extant is linearly ordered, in part. The rules in a linearly ordered set of rules are divided into two types, "obligatory" and "optional." An obligatory rule is one which must be applied if its domain statement is satisfied at the appropriate point in the derivation; an optional rule is one which need not be. (See, for example, Chomsky, 1957.) If $R = \{T_1, \cdots, T_n\}$ is linearly ordered, in the order given, with $T_{i_1}, \cdots, T_{i_k}$ optional, then the ordering system Ω can be constructed as follows:

(i) $K = \{s_0, \cdots, s_n\}$.

(ii) $\delta(s_i, T_j) = s_j$, for all i, j, $0 \leq i \leq n$, $1 \leq j \leq n$.

(iii) In $N(s_0)$, the obligatory rules are linearly ordered, with the last of them immediately preceding STOP. Each optional rule T_{i_j} is represented in $N(s_0)$ as a maximal node, which is directly preceded by the last (in R) preceding obligatory rule, if there is one, and isolated if there is no preceding obligatory rule.

(iv) For all i, $1 \leq i \leq n$, $N(s_i)$ is formed from $N(s_{i-1})$ by removing T_i.

To illustrate, let $R = \{T_1, T_2, T_3, T_4, T_5\}$ be linearly ordered, in the order given, with T_1, T_2, and T_5 optional. Then Ω is as follows: $K = \{s_0, \cdots, s_5\}$ and $\delta(s_i, T_j) = s_j$, for all i, j, $0 \leq i \leq 5$ and $1 \leq j \leq 5$, and $N(s_0), \cdots, N(s_5)$ are in Fig. 1–Fig. 6, respectively.

(2) Recent versions of T-grammars, starting with Chomsky (1965), have assumed a cyclic order of application of T-rules. In this example, we ignore the linguistic problems involved in setting up the domain statement and simply show how a cyclic ordering is represented by our ordering scheme. If $R = \{T_1, \cdots, T_n\}$ is cyclically ordered (i.e., the rules are applied in the order given, but starting over with T_1 after T_n), with each rule obligatory, and the derivation halts whenever there is no rule that can be applied, then Ω is as follows:

(i) $K = \{s_0, \cdots, s_n\}$.

(ii) $\delta(s_i, T_j) = s_j$, for all i, j, $0 \leq i \leq n$ and $1 \leq j \leq n$.

(iii) For all i, $1 \leq i \leq n$, $N(s_i)$ is in Fig. 7, and $N(s_0) = N(s_n)$.

To illustrate, let $R = \{T_1, T_2, T_3\}$. This cyclic ordering, with the rules obligatory, is represented as follows:

(i) $K = \{s_0, s_1, s_2, s_3\}$.

(ii) $\delta(s_i, T_j) = s_j$, for all i, j, $0 \leq i \leq 3$, $1 \leq j \leq 3$.

(iii) $N(s_0) = N(s_3)$ is in Fig. 8. The $N(s_1)$ is in Fig. 9. The $N(s_2)$ is in Fig. 10.

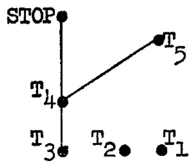

FIG. 1.

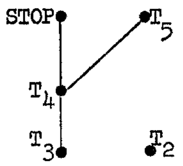

FIG. 2.

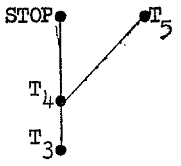

FIG. 3.

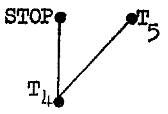

FIG. 4.

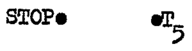

FIG. 5.

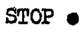

FIG. 6.

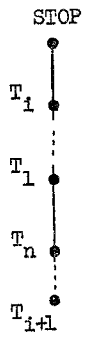

FIG. 7.

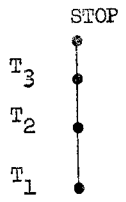

FIG. 8.

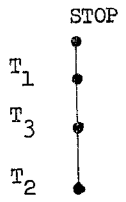

FIG. 9.

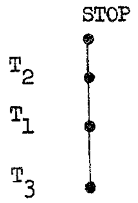

FIG. 10.

(3) In some grammars (e.g., Mitre Corp., 1964) the rules are cyclically ordered, except that certain rules are optionally and others obligatorily reapplied as often as possible before going to the next rule. (Such provisions are implicit in many grammars which appear to be cyclically ordered.) Here K, δ, and $N(s_0)$ are as in Example 2. If T_i is obligatorily reapplied as often as possible before going to the next (in R) rule, then $N(s_i)$ is in Fig. 11. If T_i is optionally reapplied indefinitely before going to the next rule, then $N(s_i)$ is as in Fig. 12.

To illustrate, let $R = \{T_1, T_2, T_3, T_4\}$ be cyclically ordered, with all rules obligatory, with T_2 obligatorily reapplied as many times as possible before going to T_3, and with T_3 optionally reapplied indefinitely before going to T_4. Then $K = \{s_0, s_1, s_2, s_3, s_4\}$, $\delta(s_i, T_j) = s_j$ for all i, j $0 \le i \le 4$ and $1 \le j \le 4$, $N(s_0) = N(s_4)$ is in Fig. 13, $N(s_1) = N(s_2)$ is in Fig. 14, and $N(s_3)$ is in Fig. 15.

(4) Recently the notion of an "anywhere" rule has been proposed (Ross, 1967). Roughly described, an "anywhere" rule is one which must (or may) be applied whenever its domain statement is satisfied. We consider only the case where $R = \{T_1, \cdots, T_n, T_A\}$, with T_1 to T_n obliga-

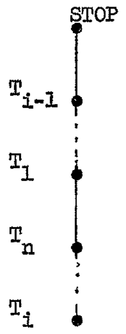

FIG. 11.

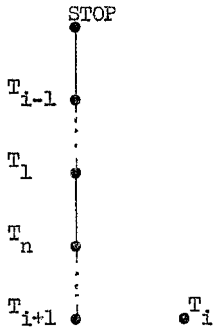

FIG. 12.

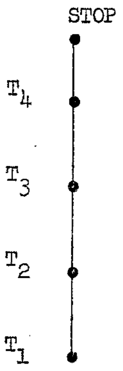

FIG. 13.

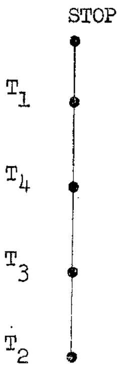

FIG. 14.

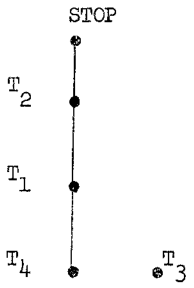

FIG. 15.

tory and cyclically ordered, and T_A an obligatory anywhere rule. (We assume that T_A is a reapplicable rule as in the preceding example; variations such as nonreapplicable anywhere rules, more than one anywhere rule, and optional anywhere rules, could all be represented by fairly obvious modifications.) Then $K = \{s_0, \cdots, s_n\}$. The next state function plays a more significant role in this case; $\delta(s_i, T_j) = s_j$ for i, j with $0 \leq i \leq n$, $1 \leq j \leq n$, and $\delta(s_i, T_A) = s_i$ for all i, $0 \leq i \leq n$. The sets $N(s_i)$ are as in the cyclic case except that T_A is added to each set as a zero element.[21]

For example, let $R = \{T_1, T_2, T_3, T_A\}$, with T_1 to T_3 obligatory and cyclic, and T_A an obligatory anywhere rule. Then $K = \{s_0, s_1, s_2, s_3\}$, $\delta(s_i, T_j) = s_i$ for all i and all T_j except T_A, and $\delta(s_i, T_A) = s_i$ for all

[21] Given a partially ordered set $(Y, \leq)$, x is a *zero* element if $x \leq y$ for all y.

i, $N(s_0) = N(s_3)$ is in Fig. 16, $N(s_1)$ is in Fig. 17, and $N(s_2)$ is in Fig. 18.

(5) The STOP symbol can be used in several ways to filter words from the language. The previous examples have illustrated its use in preventing words which occur in intermediate stages of certain derivations from being in the language. As another instance of its use, suppose a certain partially ordered set $N(s)$ contains no occurrence of STOP. Then any tree τ occurring at state s to which no rule in $N(s)$ can be applied is eliminated, since $(\tau, s)| \vdash$ STOP is false and there is no τ', s' such that $(\tau, s) \vdash (\tau', s')$. (For more drastic filtering, $N(s)$ may be empty for some states s.)

To illustrate the implications of the absence of STOP, let $R = \{T_1, T_2, T_3\}$ be linearly ordered, let $K = \{s_0, s_1, s_2, s_3\}$, let $\delta(s_i, T_j) = s_j$ for all i, j, $0 \leq i \leq 3$, $1 \leq j \leq 3$, and let $N(s_0)$ be as in Fig. 19, $N(s_1)$ as in Fig. 20, $N(s_2)$ as in Fig. 21, and $N(s_3)$ as in Fig. 22. Then any tree to which T_3 fails to apply as the last rule in the derivation is eliminated.

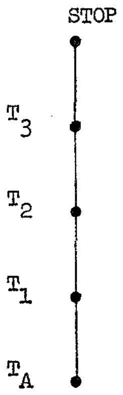

FIG. 16.

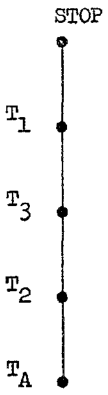

FIG. 17.

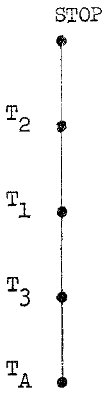

FIG. 18.

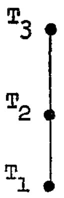

FIG. 19.

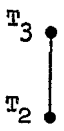

FIG. 20.

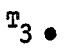

FIG. 21.

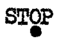

FIG. 22.

(6) We now present a simple example of "tree-pruning" (Ross, 1967; 1967, Chp. 3), and [R2, Chapter 3].

Suppose that the trees in Figs. 23 and 24 arise as subtrees in derivations. Furthermore, suppose we want to (a) "prune" the S-node in Fig. 23, i.e., to delete the S-node without deleting the subtree under it, thereby obtaining the tree in Fig. 24, and (b) leave unchanged the tree in Fig. 25. This cannot be accomplished by one application of a T-rule for the following two reasons:

(i) The domain statement $D = D_I \wedge D_O = \alpha_1\alpha_2\alpha_3 \wedge \alpha_4\alpha_5\alpha_6 = X^{(1)}\mathrm{N_p}X^{(2)} \wedge X^{(1)}\mathrm{S}X^{(2)}$ is satisfied by the trees in both Fig. 23 and Fig. 25.

(ii) Even if the trees in Fig. 23 and Fig. 25 could be distinguished by some domain statement, there is no way to substitute the $\mathrm{N_p}$-subtree for the S-subtree (for both subtrees cannot be included in one D_I, and only subtrees mentioned in D_I are affected by the structural change.)

The problem can be resolved by using the following sequence of T-rules, which are obligatory and linearly ordered with respect to each other. (We ignore here the question of how they might relate to the other rules of a grammar):

$T_1 = (D, C)$, where $D = D_I \wedge D_O = \alpha_1\alpha_2\alpha_3 \wedge \alpha_4\alpha_5\alpha_6 = X^{(1)}SX^{(2)} \wedge X^{(1)}N_pX^{(2)}$, $C = ①\text{-}②m\text{-}③$, and m is a new element of Σ'.

$T_2 = (D, C)$, where $D = D_I \wedge D_O = \alpha_1\alpha_2\alpha_3\alpha_4 \wedge \alpha_5\alpha_6\alpha_7 = X^{(1)}N_pmX^{(2)} \wedge X^{(1)}SX^{(2)}$, and $C = ①\text{-}\varnothing\text{-}②\text{-}④$.

$T_3 = (D, C)$, where $D = D_I = \alpha_1\alpha_2\alpha_3 = X^{(1)}mX^{(2)}$ and $C = ①\text{-}\varnothing\text{-}③$.

Then T_1 inserts the marker m adjacent to S in both Fig. 23 and Fig. 25, obtaining the trees in Fig. 26 and Fig. 27, respectively. T_2 applies only to the tree of Fig. 26, changing it to the tree in Fig. 24. T_3 removes the marker m from the tree of Fig. 27, restoring it to its original form (Fig. 25).

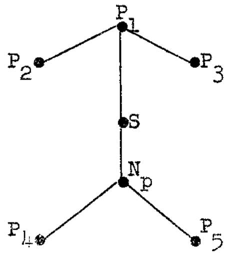

FIG. 23.

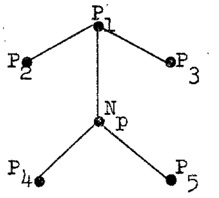

FIG. 24.

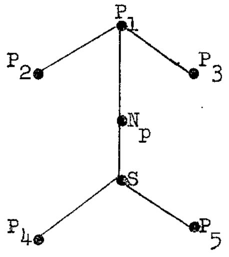

FIG. 25.

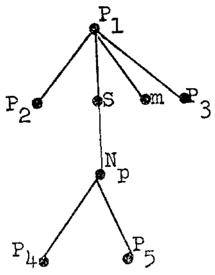

FIG. 26.

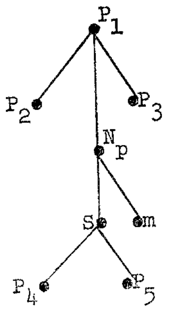

FIG. 27.

The above rules illustrate a type of procedure which can be used whenever two nodes, one below the other with no "branching" between, are both to be involved in a structural change. However, this technique does not allow one of two identically labeled nodes, one directly above the other, to be pruned. It was to avoid this situation that the agreement in Section 2A was made.

(7) For our last example, we present a T-grammar in its entirety. The grammar is $\mathcal{G} = (\mathcal{B}, \Delta, R, \Omega)$, where $\mathcal{B}$, Δ, R, and Ω are given below.

(a) $\mathcal{B} = (B, V, \Sigma)$, where

(i) $\Sigma = \{\$, \mathrm{P_{ast}}, \text{the, eat, chase, dog, cat, mouse, recently, yesterday}\}$.

(ii) $V = \Sigma \cup \{\mathrm{S}, \mathrm{S'}, \mathrm{N_p}, \mathrm{A_{ux}}, \mathrm{V_p}, \mathrm{A_{dv}}, \mathrm{V_b}, \mathrm{N_n}, \mathrm{T_{ns}}\}$.

(iii) B is the set of trees generated by the following set of CF rules, where S is the start symbol:

$$\mathrm{S} \to \$S'\$$$

$$\begin{cases} \mathrm{S'} \to N_pA_{ux}V_p \\ \mathrm{S'} \to N_pA_{ux}V_pA_{dv} \end{cases}$$

$$\mathrm{V_p} \to V_bN_p$$

$$\begin{cases} \mathrm{N_p} \to \text{the } N_n\$S'\$ \\ \mathrm{N_p} \to \text{the } N_n \end{cases}$$

$$\mathrm{A_{ux}} \to \mathrm{T_{ns}}$$

$$\mathrm{T_{ns}} \to \mathrm{P_{ast}}$$

$$\begin{cases} \mathrm{V_b} \to \text{chase} \\ \mathrm{V_b} \to \text{eat} \end{cases}$$

$$\begin{cases} \mathrm{N_n} \to \text{dog} \\ \mathrm{N_n} \to \text{cat} \\ \mathrm{N_n} \to \text{mouse} \end{cases}$$

$$\begin{cases} \mathrm{A_{dv}} \to \text{recently} \\ \mathrm{A_{dv}} \to \text{yesterday} \end{cases}$$

(b) $\Delta = (\Sigma', \Sigma_A', X, V'')$, where

(i) $\Sigma' = \Sigma \cup \{\mathrm{be}, \mathrm{E_n}, \mathrm{by}, \mathrm{that}, \mathrm{itself}, \#\}$

(ii) $\Sigma_A' = \Sigma' - \{\#\}$

(iii) $V'' = V \cup \Sigma' \cup \{X\} \cup \{X^{(1)}, \cdots, X^{(10)}, \mathrm{N_p^{(1)}}, \mathrm{N_p^{(2)}}\}$

(c) $R = \{\mathrm{T_{pas}}, \mathrm{T_{reflex}}, \mathrm{T_{adv}}, \mathrm{T_{rel}}, \mathrm{T_{fin}}, \mathrm{T_{bound}}\}$, where

(i) T$_{\text{pas}}$ is the rule (D, C), with $D = D_I \wedge D_O$, where

$D_I = \alpha_1 \cdots \alpha_9 = X^{(1)}\$\mathrm{N_p^{(1)}}T_{\mathrm{ns}}\mathrm{V_b}\mathrm{N_p^{(2)}}X^{(7)}\$X^{(2)}$,

$D_O = \alpha_{10} \cdots \alpha_{14} \wedge \sim\alpha_{15} \cdots \alpha_{21} \wedge \sim\alpha_{22} \cdots \alpha_{30} \wedge \sim\alpha_{31} \cdots \alpha_{39} = X^{(1)}\$S'\$X^{(2)} \wedge \sim X^{(1)}\$X^{(3)}\$X^{(4)}\$X^{(2)} \wedge \sim X^{(5)}\$S'\$X^{(6)}\$S'\$X^{(2)} \wedge \sim X^{(1)}\$\mathrm{N_p^{(1)}}\mathrm{T_{ns}}\mathrm{V_b}\mathrm{N_p^{(1)}}X^{(7)}\$X^{(2)}$, and $C = \beta_1 \cdots \beta_9 =$ ①–②–⑥–④ be E$_{\text{n}}$–⑤–by ③–⑦–⑧–⑨

All but the last conjunct of D_O are conditions designed to insure that the T-rule operate within the leftmost "lowest S' flanked by \$ markers" in the tree (see Footnote 23). Since these conditions will also be imposed on T$_{\text{reflex}}$, T$_{\text{adv}}$, and T$_{\text{rel}}$, we shall abbreviate them as "LLS" (leftmost lowest S'). Thus, LLS is D_O above without $\sim X^{(1)}\$\mathrm{N_p^{(1)}}\mathrm{T_{ns}}\mathrm{V_b}\mathrm{N_p^{(1)}}X^{(7)}\$X^{(2)}$. Since LLS requires 21 symbols to state, we shall assign it to $\alpha_{i+1} \cdots \alpha_{i+21}$, where $D_I = \alpha_1 \cdots \alpha_i$.

The last conjunct of D_O, in combination with D_I, asserts that the two noun phrases involved in the rule must be different.

(ii) R$_{\text{reflex}}$ is the rule (D, C), with $D = D_I \wedge D_O$, where

$D_I = \alpha_1 \cdots \alpha_9 = X^{(1)}\$\mathrm{N_p^{(1)}}\mathrm{A_{ux}}\mathrm{V_b}\mathrm{N_p^{(1)}}X^{(7)}\$X^{(2)}$,

$D_O = \alpha_{10} \cdots \alpha_{14} \wedge \sim\alpha_{15} \cdots \alpha_{21} \wedge \sim\alpha_{22} \cdots \alpha_{30} = \mathrm{LLS}$,

and $C = \beta_1 \cdots \beta_9 =$ ①–②–③–④–⑤–itself–⑦–⑧–⑨

(iii) T$_{\text{adv}}$ is the rule (D, C), with $D = D_I \wedge D_O$, where

$D_I = \alpha_1 \cdots \alpha_7 = X^{(1)}\$\,\mathrm{N_p}X^{(7)}\mathrm{A_{dv}}\$X^{(2)}$,

$D_O = \alpha_8 \cdots \alpha_{12} \wedge \sim \alpha_{13} \cdots \alpha_{19} \vee \sim\alpha_{20} \cdots \alpha_{28} = \mathrm{LLS}$,

and $C = \beta_1 \cdots \beta_7 =$ ①–②–⑤–③–④–∅–⑥–⑦

(iv) T$_{\text{rel}}$ is the rule (D, C), with $D = D_I \wedge D_O$, where

$D_I = \alpha_1 \cdots \alpha_{13} = X^{(1)}\$X^{(7)}\mathrm{N_n}\#X^{(8)}\ \text{the}\ \mathrm{N_n}X^{(9)}\#X^{(10)}\$X^{(2)}$,

$D_O = \alpha_{14} \cdots \alpha_{18} \wedge \sim\alpha_{19} \cdots \alpha_{25} \wedge \sim\alpha_{26} \cdots \alpha_{34} \wedge \alpha_{35} \cdots \alpha_{46}$
$= \mathrm{LLS} \wedge X^{(1)}\$X^{(7)}\mathrm{N_n}\#X^{(8)}\mathrm{N_p}X^{(9)}\#X^{(10)}\$X^{(2)}$,

and $C = \beta_1 \cdots \beta_{13} =$ ①–②–③–④–that–⑥–∅–∅–⑨–∅–⑪–⑫–⑬

(v) T$_{\text{bound}}$ is the rule (D, C), with $D = D_I \wedge D_O$, where

$D_I = \alpha_1 \cdots \alpha_5 = X^{(1)}\$X^{(3)}\$X^{(2)}$,

$D_O = \alpha_6 \cdots \alpha_{10} \wedge \sim\alpha_{11} \cdots \alpha_{17} \wedge \sim\alpha_{18} \cdots \alpha_{26} = \mathrm{LLS}$,

and $C = \beta_1 \cdots \beta_5 =$ ①–#–③–#–⑤

(vi) T$_{\text{fin}}$ is the rule (D, C), with $D = D_I$, where

$D_I = \alpha_1\alpha_2\alpha_3 = \#X\#$

and $C = \beta_1\beta_2\beta_3 = \varnothing\text{-}②\text{-}\varnothing$

(d) $\Omega = (K, \mathfrak{N}, \delta, s_{\text{bound}})$, where $K = \{s_{\text{rel}}, s_{\text{pas}}, s_{\text{reflex}}, s_{\text{adv}}, s_{\text{bound}}, s_{\text{fin}}\}$, $\mathfrak{N} = \{N(s_x)/s_x \text{ in } K\}$, with $\text{N}(s_{\text{rel}})$ as in Fig. 28, $\text{N}(s_{\text{pas}})$ in Fig. 29, $\text{N}(s_{\text{reflex}})$ in Fig. 30, $\text{N}(s_{\text{adv}})$ in Fig. 31, $\text{N}(s_{\text{bound}})$ in Fig. 32, $\text{N}(s_{\text{fin}})$ in Fig. 33, and δ is defined by $\delta(s_x, T_y) = s_y$ for all x and y.

Note that this T-grammar includes all the rules of the example in Section 1 (with the T-rules expressed formally here) as well as several others. In addition, this T-grammar includes adverbs (i.e., "recently", and "yesterday"). The CF-rules introduce adverbs at the end of a sentence, as in "the cat ate the mouse yesterday." The T-rule T_{adv} moves adverbs to the beginning of the sentence, as in "yesterday the cat ate the mouse." The inclusion of T_{reflex} permits such sentences as "the mouse chased itself." T_{bound} converts the original boundary marker \$ to # (at the end of each cycle). The provision that each rule apply within the leftmost lowest S', with \$ markers, combines (using the order of the rules) with the conversion of \$ to #. The effect is that the subtrees headed by S′ in the tree are operated on one at a time, from bottom to top (and left to right). At the end of the last cycle, T_{fin} deletes the outermost occurrence of #, which is the "unacceptable terminal" (i.e., $\Sigma' - \Sigma_A' = \{\#\}$).

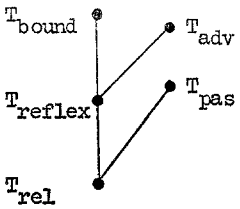

FIG. 28.

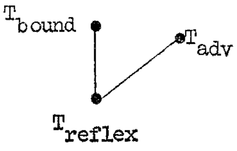

FIG. 29.

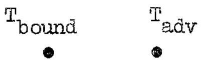

FIG. 30.

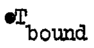

FIG. 31.

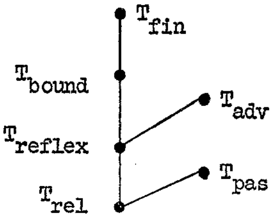

FIG. 32.

FIG. 33.

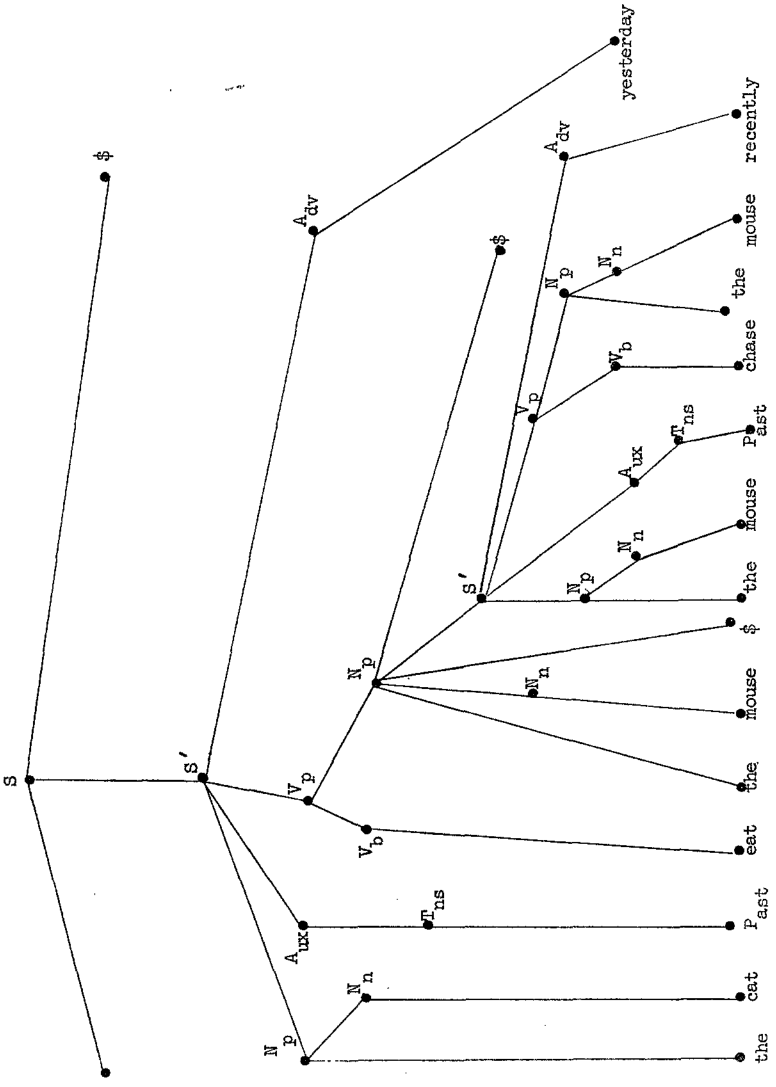
S
$
S′
A dv
yesterday
N p
A ux
T ns
V p
V b
N n
S′
A dv
recently
the
cat
Past
eat
the
mouse
$
the
mouse
Past
chase
the
mouse

FIG. 34.

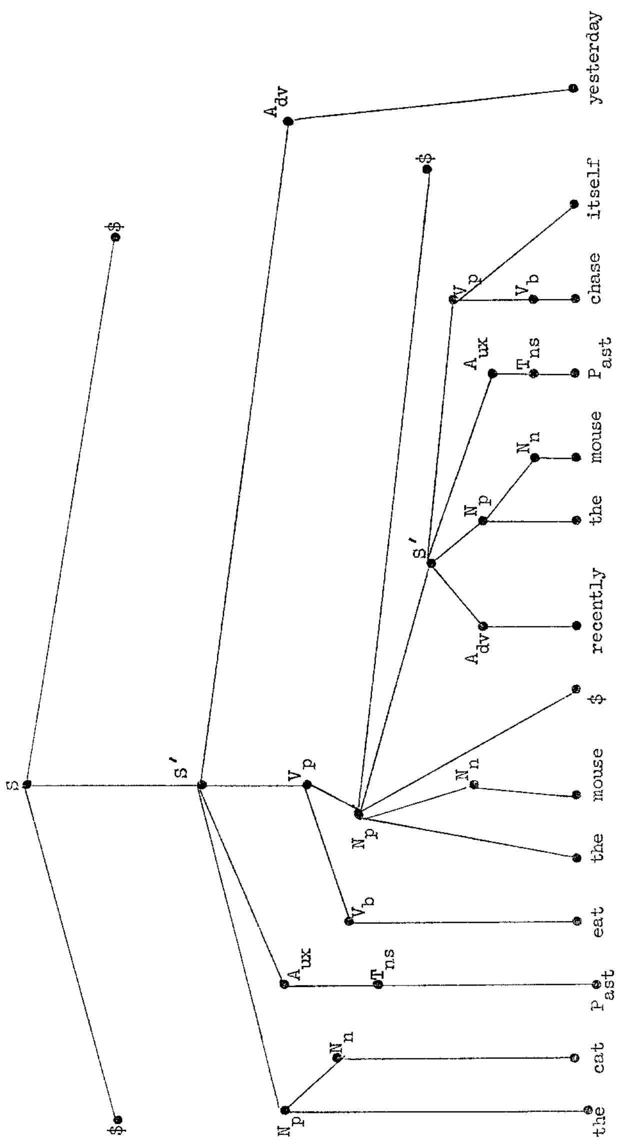
S
$
$
S′
Np
Nn
Aux
Tns
Vp
Vb
Np
Nn
Adv
$
S′
Adv
Np
Nn
Aux
Tns
Vp
Vb
the
cat
Past
eat
the
mouse
$
recently
the
mouse
Past
chase
itself
yesterday

FIG. 35.

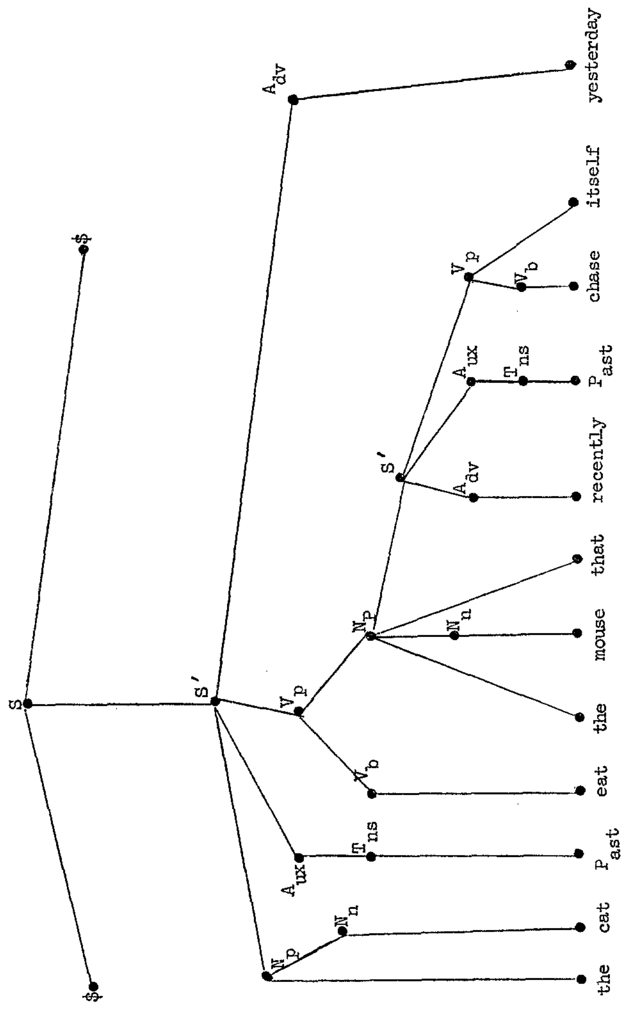
S
$
$
S'
A_{dv}
yesterday
N_p
N_n
A_{ux}
T_{ns}
V_p
V_b
N_p
N_n
S'
A_{dv}
A_{ux}
T_{ns}
V_p
V_b
the
cat
P_{ast}
eat
the
mouse
that
recently
P_{ast}
chase
itself

FIG. 36.

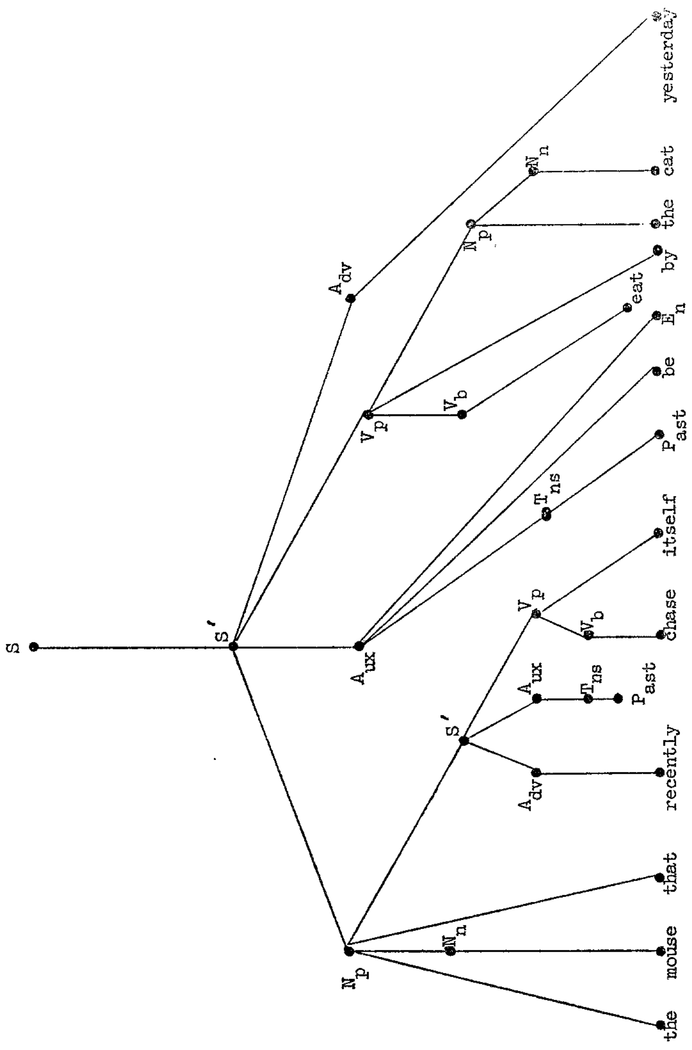
S
S'
N_p
N_n
the
mouse
that
S'
A_{dv}
recently
A_{ux}
T_{ns}
P_{ast}
V_p
V_b
chase
itself
A_{ux}
T_{ns}
P_{ast}
be
E_n
V_p
V_b
eat
by
N_p
the
N_n
cat
A_{dv}
yesterday

FIG. 37.

The ordering of the rules consists essentially of a cycle on the linear sequence T_{rel}, T_{pas}, T_{reflex}, T_{adv}, and T_{bound}, plus a T-rule T_{fin}, which is the last to be applied. T_{pas} and T_{adv} are optional rules. T_{rel} is reapplicable (obligatorily).

We illustrate the operation of the T-grammar with the derivation of the sentence "the mouse that recently chased itself was eaten by the cat yesterday."

(i) The CF rules generate the tree τ_0 given in Fig. 34.

(ii) $(\tau_0, s_{bound}) \vdash_{T_{reflex}} (\tau_1, s_{reflex})$, where τ_1 is the tree obtained by replacing the rightmost N_p in τ_0 with the symbol "itself."

(iii) $(\tau_1, s_{reflex}) \vdash_{T_{adv}} (\tau_2, s_{adv})$, where τ_2 is in Fig. 35.

(iv) $(\tau_2, s_{adv}) \vdash_{T_{bound}} (\tau_3, s_{bound})$, where τ_3 is the tree obtained by replacing the two inner occurrences of \$ with #.

(v) $(\tau_3, s_{bound}) \vdash_{T_{rel}} (\tau_4, s_{rel})$, where τ_4 is in Fig. 36.

(vi) $(\tau_4, s_{rel}) \vdash_{T_{pas}} (\tau_5, s_{pas}) \vdash_{T_{bound}} (\tau_6, s_{bound}) \vdash_{T_{fin}} (\tau_7, s_{fin}) \mid \vdash$ STOP, where τ_7 is in Fig. 37.

Since all the terminals of τ_7 are in Σ_A', the resulting word is in the language generated by the T-grammar. As in the example of Section 1, we assume that morphophonemic rules not shown here convert the terminal string "the mouse that recently P_{ast} chase itself P_{ast} be E_n eat by the cat yesterday" into "the mouse that recently chased itself was eaten by the cat yesterday."

4. RESTRICTIONS

The general model presented in Section 2 incorporates most of the important features of T-grammars described in the literature.[22] However, linguists are trying to determine the narrowest possible class of grammars adequate for describing natural languages. In this section, we present a number of restrictions on the general model. These restrictions, taken in various combinations, provide submodels which correspond to conceptions of T-grammars already in the literature.

A T-grammar has been defined as a quadruple $\mathcal{G} = (\mathcal{B}, \Delta, R, \Omega)$, where $\mathcal{B}$ is a base, Δ specifies the vocabulary and its subparts, R is a set of T-rules, and Ω is a specification of the order in which the T-rules can be applied. We therefore first list the various restrictions on Δ, then those on R, and finally those on Ω.

[22] See Footnote 2.

A. Restrictions on Δ

Definition. $\mathcal{G}$ is *terminal-preserving* if $\Sigma' = \Sigma$, and in each rule (D, C) in R, each β_i is in $\{\varnothing, ①, \cdots, ⓝ\}^+$.

In other words, a terminal-preserving grammar is one in which no T-rule can introduce a new terminal. This restriction is an extreme version of the notion that the trees of the base should contain all the semantically relevant material (Katz and Postal, 1964).

Definition. $\mathcal{G}$ is *terminal-blocking* if $\Sigma_A' \neq \Sigma'$, i.e., if some of the terminals are unacceptable. $\mathcal{G}$ is *a-blocking* if $\Sigma_A' = \Sigma' - \{a\}$.

In Chomsky (1965), p. 138, the "boundary symbol" # appears to be the only unacceptable terminal.

B. Restrictions on R

Definition. $\mathcal{G}$ has *weakly recoverable deletions* (Katz and Postal, 1964, p. 80) if, for each T-rule (D, C) and each i, β_i in C, at least one of the following holds:

(a) for some β_j in $C, \beta_j = u ⓘ v, u$ and v in $(\Sigma' \cup \{\varnothing, ①, \cdots, ⓝ\}$.

(b) $\alpha_i = \alpha_k$ for some $k \neq i$, and for some β_j in C, $\beta_j = u ⓚ v$, u and v in $(\Sigma' \cup \{\varnothing, ①, \cdots, ⓝ\})^+$.

(c) α_i is in Σ'.

$\mathcal{G}$ having weakly recoverable deletions requires each α_i whose index does not appear in the structural change statement to be either a terminal symbol (condition (c)), or identical to some α_k whose index appears in the structural change statement (condition (b)). Its effect is that given a tree and the T-rule applied to produce it, the preceding tree is determined to within a finite set of trees.

Definition. $\mathcal{G}$ has *strongly recoverable deletions* if for each T-rule and each β_i in C, either condition (a) or condition (b) in the previous definition holds.

This property, in slightly different form, is suggested in Chomsky (1965; p. 145, p. 222, fn 1; p. 225, fn 13). Chomsky suggests that in the general theory there be a requirement that deletion be effected only by identical structures, and that such a requirement would eliminate the need for quantifiers in domain statements. (In our model, quantifiers are implicit in the use of identical superscripts such as $N_P^{(2)}$, $N_P^{(2)}$ etc.)

C. Restrictions on Ω

Definition. $\mathcal{G}$ is *naturally ordered* if $R = \{T_1, \cdots, T_n\}$, $K = \{s_0, \cdots, s_n\}$, and $\delta(s_i, T_j) = s_j$ for all $i, j, 0 \leq i \leq n, 1 \leq j \leq n$.

Although ordering of T-rules is often not explicitly described, most of the literature implicitly conforms to the requirement of natural ordering. An exception is the "anywhere rule," given in Example 4 of Section 3. Note that $\mathcal{G}$ being naturally ordered imposes no restriction on the $N(s_i)$.

Among the natural orderings, we now describe two important subcases, the "linearly ordered" and the "cyclically ordered."

DEFINITION. $\mathcal{G}$ is *linearly ordered* if $R = \{T_1, \cdots, T_n\}$ is naturally ordered and for each i, $1 \leq i \leq n$,

(i) T_i is a minimal element of $N(s_{i-1})$,

(ii) $N(s_i)$ is formed by removing T_i from $N(s_{i-1})$, and

(iii) STOP is a maximal element of $\mathrm{N}(s_0)$ (and thus of each $\mathrm{N}(s_i)$).

Note that optional and obligatory rules, as described in Example 1 of section three, may appear in a linearly ordered T-grammar.

DEFINITION. $\mathcal{G}$ *is cyclically ordered* if, for $R = \{T_1, \cdots, T_n\}$, $\mathcal{G}$ has the following properties:

(i) $\mathcal{G}$ is naturally ordered.

(ii) $N(s_0) = N(s_n)$ and satisfies the following: the obligatory rules $T_{i_1}, \cdots, T_{i_k}$ of R are simply ordered, i.e., $T_{i_j} < T_{i_{j+1}}$ for each $j < k$, and each precedes STOP. For each optional rule T_i, $T_{i_j} < T_i$ if $i_j < i$.

(iii) For all i, $1 \leq i \leq n$, $N(s_{i-1})$ contains T_i as a minimal element.

(iv) Suppose T_i is an isolated node in $N(s_{i-1})$. Then T_i and STOP are incomparable in $N(s_i)$. Furthermore, for each j, if $T_j <$ STOP in $N(s_{i-1})$ then $T_j < T_i$ and $T_j <$ STOP in $N(s_i)$.

(v) Suppose T_i is not an isolated node in $N(s_{i-1})$. Then $T_i <$ STOP in $N(s_i)$. Furthermore, for each j, if $T_j <$ STOP in $N(s_{i-1})$ then $T_j < T_i$ in $N(s_i)$.

See Chomsky (1965) p. 134–138 for a linguistic discussion and a somewhat different formulation.[23]

DEFINITION. $\mathcal{G}$ is *unordered* if $R = \{T_1, \cdots, T_n\}$, $K = \{s_0\}$, $\delta(s_0, T_j) = s_0$, for all j, and in $N(s_0)$, for each i either T_i is an isolated node or T_i precedes STOP. (That is, optional rules are isolated, while obligatory rules precede STOP.)

[23] A typical rule of a cyclic grammar in the literature might have a domain statement which appears as # JL #. Since our model does not incorporate the notion of a "lowest sentence," such a domain statement would be restated in our system as $D_I \wedge D_O$, where $D_I = X^{(1)}\#JL\#X^{(2)}$, and D_O is similar to the LLS statement of Example 6 of Section 3.

To our knowledge an unordered T-grammar has not actually appeared in the literature, but its possibility has been suggested in informal discussions, frequently under the name "intrinsically ordered."

The final two definitions concern the use of ordering to accomplish blocking as illustrated in Example 5 of Section 3.

DEFINITION. $\mathcal{G}$ is *order-blocking* if for at least one s in K, $N(s)$ does not contain STOP as an isolated node.

Thus, a T-grammar is order-blocking if it does not permit free choice of STOP at every stage in every derivation. Such a grammar uses the ordering to prevent certain words which arise in intermediate stages from being in the language. This application of ordering is implicit in most generative grammars.

DEFINITION. $\mathcal{G}$ is *nonblocking* if $\mathcal{G}$ is neither terminal-blocking nor order-blocking.

Thus $\mathcal{G}$ is nonblocking if every word which arises in a derivation is a word in the language. This notion is a slight extension of the notion of "nonblocking" as used in Klima.

ACKNOWLEDGMENT

We are indebted to Dr. Joyce Friedman for a critical reading of the manuscript.

RECEIVED: July 15, 1968

REFERENCES

CHOMSKY, N. (1957), "Syntactic Structures." Mouton, The Hague.

CHOMSKY, N. 1965), "Aspects of the Theory of Syntax." MIT Press, Cambridge, Mass.

FRIEDMAN, J. (1968), A computer system for transformational grammar. Report CS-84 of the Computational Linguistics Project, Computer Science Dept., Stanford Univ.

KATZ, J. J., AND POSTAL, P. M. (1964), "An Integrated Theory of Linguistic Descriptions." MIT Press, Cambridge, Mass.

KIMBALL, J. P. (1967), Predicates definable over transformational derivations by intersection with regular languages. *Inform. Control* **11,** 177–195.

KLIMA, E. S. Current developments in generative grammars. MIT report, (to appear in *Kybernetica I*, Prague).

McCAWLEY, J. D. (1968), Concerning the base component of a transformational grammar. *Found. Language* **4,** 243–269.

The Mitre Corp. (1964), English preprocessor manual. Report SR-132, Bedford, Mass.

PETERS, S. (1966), A note on ordered phrase structure grammars. Report No.

NSF-17, The Computation Laboratory of Harvard University, pp. (VIII–1)-(VIII–19), Cambridge, Mass.

PETERS, S. AND RITCHIE, R. On the generative capacity of transformational grammars. (to appear in *Information and Control*).

Ross, J. R. (1965), A proposed rule of tree-pruning. Linguistic Society of America, winter meeting.

Ross, J. R. (1967), Gapping and the Order of Constituents." (to appear in the Proceedings of the Tenth International Congress of Linguists).

Ross, J. R. (1967), Constraints on variables in syntax. MIT Ph.D. dissertation.