CS - 103

LEXICAL INSERTION IN TRANSFORMATIONAL GRAMMAR

BY

JOYCE FRIEDMAN and THOMAS H. BREDT

This research was supported in part by the United States Air Force Electronic Systems Division, under Contract F196828-C-0035.

STANFORD UNIVERSITY COMPUTER SCIENCE DEPARTMENT
COMPUTATIONAL LINGUISTICS PROJECT

JUNE 1968

LEXICAL INSERTION IN TRANSFORMATIONAL GRAMMAR

by

Joyce Friedman and Thomas H. Bredt

This research was supported in part by the United States Air Force Electronic Systems Division, under Contract F196828-C-0035.

cs - 103

AF - 25

June 1968

LEXICAL INSERTION IN TRANSFORMATIONAL GRAMMAR

by

Joyce Friedman and Thomas H. Bredt

Abstract

In this paper, we describe the lexical insertion process for generative transformational grammars. We also give detailed descriptions of many of the concepts in transformational theory. These include the notions of complex symbol, syntactic feature (particularly contextual feature), redundancy rule, tests for pairs of complex symbols, and change operations that may be applied to complex symbols. Because of our general interpretation of redundancy rules, we define a new complex symbol test known as compatibility, This test replaces the old notion of nondistinctness. The form of a lexicon suitable for use with a generative grammar is specified.

In lexical insertion, vocabulary words and associated complex symbols are selected from a lexicon and inserted at lexical category nodes in the tree. Complex symbols are lists of syntactic features. The compatibility of a pair of complex symbols and the analysis procedure

used for contextual features are basic in determining suitable items for insertion. Contextual features (subcategorization and selectional) have much in common with the structural description for a transformation and we use the same analysis procedure for both. A problem encountered in the insertion of a complex symbol that contains selectional features is side effects, We define the notion of side effects and describe how these effects are to be treated.

The development of the structure of the lexicon and the lexical insertion algorithm has been aided by a system of computer programs that enable the linguist to study transformational grammar. In the course of this development, a computer program to perform lexical insertion was written. Results obtained using this program with fragments of transformational grammar are presented. The paper concludes with suggestions for extensions of this work and a discussion of interpretations of transformational theory that do not fit immediately into our framework.

Table of Contents

Table of Contents (continued)

1. Introduction

The form and role of the lexicon in transformational grammar have been considered by Chomsky in Aspects of the Theory of Syntax [3]. The notions of syntactic feature and complex symbol introduced there have substantially altered the role of the lexicon in the theory., In earlier discussions, vocabulary words were selected by terminal rules in the phrase structure. Context-sensitive phrase structure rules and subcategorization in the phrase structure were the only devices available to constrain the choice of vocabulary words. Chomsky now gives two alternative ways of introducing vocabulary words into the tree, In the first, the rewriting rules of the phrase structure are modified to introduce complex symbols. Vocabulary words from a lexicon are suitable for insertion in the tree if their complex symbol is nondistinct from the complex symbol in the tree. In the other alternative, the phrase structure is a simple context-free grammar. In this case, a vocabulary word may be inserted if its complex symbol is nondistinct from a complex symbol appearing in the tree (there are conventions to insure that the lexical category node in the tree and the category feature of the complex symbol are the same) and if the tree structure is analyzable in such a way as to satisfy the contextual features appearing in the complex symbol in the lexicon. In this paper we will describe an interpretation of this second alternative for lexical insertion,

To provide perspective for the discussions to follow, we digress to discuss the environment in which this research has been conducted. For the past two years we have been developing computer aids for the

study of transformational grammar. We have developed a comprehensive system that is capable of modeling all phases of the generation of a sentence according to a grammar [6, 10]. These phases include the phrase structure phase, the transformational phase, and what we will call here the lexical insertion phase, A valuable byproduct of our work has been the deveiopment of a formal description of a transformational grammar [11]. This description serves a dual purpose,, It serves as a definitive statement of what we mean when we refer to transformational grammar and also as a definition of the input formats in which a grammar is given for testing by our system of programs, In Appendix A of this paper we reproduce the formal syntax that defines the form of a transformational grammar as we will use it. This syntax is given in a modified Backus Naur Form described in [11]. We will make frequent reference to this syntax in our discussions. For the purposes of exposition we adopt an interpretation of transformational theory that may be sketched as follows. The generation of a sentence begins with the generation of a base tree by the phrase structure phase. The terminal string contains symbols for lexical categories, During the lexical insertion process, which occurs next, suitable vocabulary words and their associated complex symbols are-selected from the lexicon and inserted in the tree. Transformations are then applied and a surface tree is obtained., This interpretation is not universally accepted by all linguists. However,, other proposals such as the alternation of lexical insertion and transformations (Klima [16]) or the possibility of doing some lexical insertion after the transformations have applied are consistent with the basic lexical insertion process that we will describe,

The primary purpose of this paper is to specify the lexical insertion process. However, we also include a description and definition of many of the concepts in transformational grammar that relate to the lexicon and the insertion process. The notions of complex symbol, syntactic feature, contextual feature (which includes subcategorization and selectional features), and redundancy rule are treated. We define the tests used for complex symbols and also the change operations allowed for complex symbols. Our interpretation of redundancy rule makes necessary a new test between complex symbols called compatibility. This test includes the old test for nondistinctness. Contextual features are viewed in much the same way as the structural description for a transformation and we use the same analysis procedure for both. These features determine the tree environment that is suitable for the insertion of a vocabulary word. Our notion of contextual feature is a generalization of the concept introduced by Chomsky [3]. We include in this type both subcategorization and selectional features. A lexicon will be defined that includes the provision for the definition of labels for contextual features as well as the definition of the other syntactic features, redundancy rules, and the lexical entries themselves. In the lexical insertion process, the main items of interest are the selection of a lexical item for insertion in the tree and the treatment of the side effects from the insertion of a lexical item. Side effects must be eonsidered whenever a complex symbol containing a contextual feature with an embedded complex symbol is inserted.

Our development of an algorithm to describe the lexical insertion process has been aided by the system of programs mentioned earlier. We

have tested the algorithm by generating sentences using grammars from the literature [3, 18, 21, 22]. The results of these experiments are reported in [7, 8, 9]. In each case, the grammar was written according to the syntax of Appendix A and the sentences were generated by the computer. These tests have made apparent many otherwise subtle points in the grammars. Results from the computer runs will be used to illustrate the operation of the algorithm. For each grammar, a small lexicon of approximately 100 vocabulary words was defined. We felt it more instructive to understand the process through the use of a small lexicon than to attack the data-processing problems incurred in dealing with large lexicons. The program that executes the lexical insertion algorithm was written in the FORTRAN programming language [15] and is discussed by Bredt [1].

In the next section, we discuss the basic concepts that are essential in the understanding of the remainder of the paper. These concepts include complex symbol, feature, and feature specification. We also discuss complex symbol operations, the use of redundancy rules, and the compatibility test for complex symbols.= We conclude the section with a discussion of contextual features, In the following section, the form and content of the lexicon are given. Next we give a description of the lexical insertion algorithm and an extended example of its use. The paper concludes with suggestions for extensions of this work and with a discussion of alternative formulations of transformational theory that might benefit from study with the methods used in this research.

2. Basic Concepts

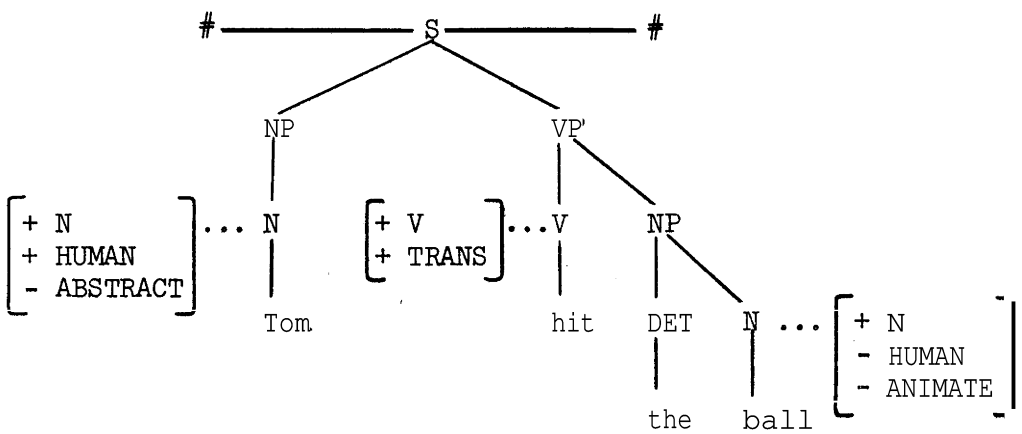
Complex symbols, features, and feature specifications

A complex symbol is a list of feature specifications, interpreted as a conjunction, and enclosed in vertical bars "|". A feature specification consists of a value and a feature. We allow the values + - and * . The indefinite value * in a feature specification in a complex symbol indicates that the complex symbol is marked for that feature, but that the value is unspecified. The-value * is not allowed in a complex symbol in a tree; when a complex symbol is inserted in a tree, each value * is changed to either + or - .

Four types of features are provided: category features, inherent features, rule features, and contextual features. A category feature denotes a lexical category such as noun (N) or verb (V). A positively specified category feature in a complex symbol indicates the type of category node at which the complex symbol may be inserted. We require that a complex symbol contain at most one positive feature specification for a category feature; complex symbols in the entries of the lexicon must contain precisely one. If a complex symbol is positively specified for one category feature, it is -implicitly specified negatively for all other category features.

Inherent features denote unanalyzable qualities such as HUMAN or ABSTRACT . Rule features are transformation names used as features. For the purposes of lexical insertion, they in no way differ from inherent features. A contextual feature is used to describe a particular environment (or context) for lexical insertion. We defer the discussion of these features until later in this section.

Complex symbols are introduced into a tree by the lexical insertion process and may be moved or altered by transformations. In lexical insertion, complex symbols are attached only to lexical category nodes. When a vocabulary word has been selected for insertion, the associated complex symbol is merged with any complex symbol that may already be on the lexical category node; and the vocabulary word, without its complex symbol, is attached as the daughter of the lexical category node. Some grammars, e.g. [21], have assumed that the complex symbol is associated with the vocabulary word directly, Minor modifications are required in such grammars to adapt them to our interpretation., The example below shows the position of complex symbols in a tree,



Complex symbol comparisons

In [6], basic comparisons and changes involving complex symbols are defined. Each of these operates on pairs of feature specifications for a feature. The result depends on the combination of the values + - and * and also abs (absent). A comparison is conveniently represented as a matrix in which the entry indicates the result of the

comparison. Complex symbol changes are likewise specified by a matrix, but in this case the entry is the final value for the feature after the change is made.

For lexical insertion, the basic comparison will be compatibility. To define compatibility we must first define nondistinctness.

Definition: Two complex symbols are nondistinct if all their feature specifications are nondistinct; that is if no feature has the value + in one and the value - in the other.[1]

Using T to represent true, F for false, and abs for absent, the nondistinctness comparison for two feature specifications A and B is given by the following matrix.

NONDISTINCTNESS

A \ B	+	-	*	abs
+	T	F	T	T
-	F	T	T	T
*	T	T	T	T
abs	T	T	T	T

We defer the discussion of the compatibility test until later in this section after more concepts have been introduced.

1. Category features are a special case. Two complex symbols are distinct if they are positively specified for different category features. We could have avoided this convention by using redundancy rules to indicate that if a complex symbol is positively specified for one category, it is negatively specified for all others, e.g., |+ N| => |- V - COP - DET| .

The comparison operation, known as inclusion, is defined by the matrix below. The matrix represents the comparision A included in B . Notice that neither + nor - is included in * . This will be important in the discussion of redundancy rules.

INCLUSION

(of A in B)

A \ B	+	-	*	abs
+	T	F	F	F
-	F	T	F	F
*	T	T	T	F
abs	T	T	T	T

Complex symbol changes

In applying redundancy rules and in combining a complex symbol with another complex symbol already in the tree, the operation merge is used. In merging A into B, each feature specification of A is merged into the corresponding feature specification of B, according to the following matrix..

MERGE

(A into B)

A \ B	+	-	*	abs
+	+	+	+	+
-				-
*	+	-	*	*
abs	+	-	*	abs

This test is useful in making structural changes in transformations and in expanding complex symbols by redundancy rules, Notice that if the complex symbols A and B are distinct, the values of features in B will be altered.

Redundancy rules

The purpose of redundancy rules is to add feature specifications to a complex symbol by a general rule whenever this is possible. These rules are useful in avoiding the specification of redundant features when defining lexical entries, A redundancy rule has the form A => B where A and B are complex symbols, Any complex symbol that includes the complex symbol on the left-hand side of the redundancy rule implicitly includes the complex symbol on the right. Notice that we allow complex symbols to appear on the left of redundancy rules, not just single feature specifications. This means that we can have rules of the form:

$$|+A \quad - B| \Rightarrow |+ \ C|$$

This capability is useful in the following situation. First, a redundancy rule may be made to apply only in the context of a specific category symbol. If we wrote the rule |+ HUMAN| => |+ ANIMATE|, and then moved the feature specification + HUMAN from a noun to a related WH, we would implicitly mark the WH also as + ANIMATE . If we wish to avoid this and mark the WH only with + HUMAN, the redundancy rule can be written |+ N + HUMAN| => |+ ANIMATE| .

Chomsky [3], uses rules very similar in form to redundancy rules in his first alternative for introducing complex symbols into trees. These rules, such as |+ ANIMATE| → |± HUMAN| are not actually redundancy rules in the sense used here; rather, they are generative rules. The interpretation of the rule just given would be that any complex symbol containing the feature specification + ANIMATE must also be specified for the feature HUMAN . Rules of this form, though superficially like the redundancy rule |+ ANIMATE| => |* HUMAN!, are not used in our interpretation, since this would mean that for any particular animate vocabulary word the choice between + HUMAN and - HUMAN was arbitrary. For vocabulary words this is not the case. Therefore, we do not allow redundancy rules in which the value * is used in the complex symbol on the right.

A redundancy rule with the indefinite value * on the left is admissible. The rule

$$|* \text{ HUMAN}| \Rightarrow |+ \text{ ANIMATE}|$$

is equivalent to the two rules

$$|+ \text{ HUMAN}| \Rightarrow |+ \text{ ANIMATE}|$$

$$|- \text{ HUMAN}| \Rightarrow |+ \text{ ANIMATE}|$$

and may be regarded as an abbreviation for that pair of rules.

We now observe why the result of the inclusion comparison for |- A| included in *|A or |+ A| included in |* A| must be false.

If we had the redundancy rule |- A(=> |- B| then the complex symbol |* A| would be expanded to |* A - B| which is equivalent to |+ A - B| or |-A - B| . Actually, the result of the application of the redundancy rule should be equivalent to either |+ A] or |- A - B| .

The compatibility test

Allowing redundancy rules to have complex symbols on the left-hand side introduces a complication into the testing of pairs of complex symbols, The basic test for lexical insertion can no longer be nondistinctness, but must be compatibility, defined as follows:

> Definition: Two complex symbols are compatible if (a) they are nondistinct, and (b) the application of the redundancy rules to the result of merging them does not change the value of any feature.

To see that this criterion must replace nondistinctness as a test, notice that the complex symbols |+ A| and |- B - C| are nondistinct, but in the presence of the redundancy rule |+ A - B| => |+ C| they are not compatible.

The actual execution of the compatibility test is complicated by the possible presence of the value * in the complex symbol obtained when the pair of complex symbols is merged prior to expansion by the redundancy rules. The appearance of the value * denotes an abbreviation for two complex symbols, one positively specified for the feature and one negatively specified. In general, if the value * appears n times, the complex symbol is an abbreviation for 2^n complex symbols. To

illustrate, if we are to test the complex symbols |+ A + D| and |+A *B * C| for compatibility, we find first that they are non-distinct. The result of merging them is |+ A * B * C + D| which is an abbreviation for the four complex symbols

|+ A +B + C + D|

|+A +B - C + D|

|+A - B + C + D|

|+A - B - C + D| .

Now the redundancy rules may be such that all, some, or none of these possibilities are good. For example, if the redundancy rule |+ D| => |- C| exists, then the second and fourth complex symbols above are possible choices. For this reason the result of the compatibility test must be not only true or false to indicate compatibility or incompatibility, but it must also give the resultant expanded complex symbol. When * values are present, the possible complex symbols are tested in a random manner such that each possibility has equal likelihood of being tested until a compatible result is found. This complex symbol is returned as part of the test. If all complex symbols fail when expanded by the redundancy rules, the test returns the value false to indicate incompatibility. Notice that the complex symbol that is the result of the compatibility test does not contain the value * . The value * is changed to either + or - before a complex symbol is inserted in the tree.

Contextual features,

Contextual features are used in the lexicon to describe tree environments where a vocabulary word may appear. If a vocabulary word is positively specified for a contextual feature, the word may appear only in that context; if negatively specified, the word may not appear in that context. A contextual feature with the value * has no interpretation. Strict local subcategorization [3, p.99] is an available option, but is not a requirement in the definition of contextual features. A contextual feature may be given either by a contextual feature description or by a contextual feature label. The label is simply an abbreviated notation for the complete feature description and is defined in the prelexicon portion of the lexicon.

The form of a contextual feature description is defined by syntax rule 4.09 of Appendix A given below.

contextual feature description ::= (structure opt[, WHERE restriction])

This format is very similar to the definition of a structural description for a transformation (rule 2.01). The structure portion must begin with a node name. The same analysis procedure is used for structural descriptions and for contextual features.

Syntax rule 2.02 of Appendix A gives the full syntax for a structural analysis. A structural analysis may be embedded in the structure portion of a contextual feature. The analysis process is discussed by Friedman and Martner [12], and by Friedman [6]. We discuss some of the more important aspects as they relate to contextual features. The

underline __ corresponds to the category node for which lexical insertion is to be performed. The symbols ⟨ and ⟩ are used to indicate dominance in the tree. If a slash / precedes the ⟨ ... ⟩, the dominance may be nonimmediate. A skip symbol $ may be used to avoid the specification of irrelevant structure. The negation of a structural analysis is indicated by preceding it by the symbol ¬.

In the examples to follow, the trees can be considered to have been generated according to the following phrase structure rules.

$$S \rightarrow \# \text{ NP VP } \#$$

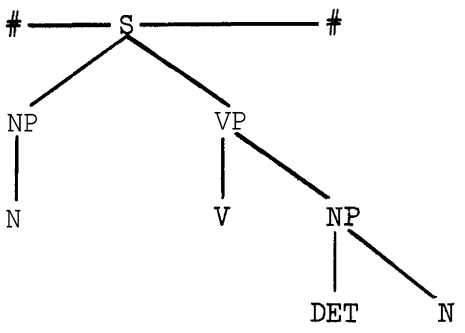
$$VP \rightarrow V \text{ (NP)}$$

$$NP \rightarrow \text{(DET) N}$$

The contextual feature label TRANS defined by

$$\text{TRANS} = \langle \text{ VP } \langle \text{ \_\_ NP } \rangle \rangle$$

appearing in a complex symbol for a verb could be used to indicate a transitive verb (+ TRANS) or an intransitive verb (- TRANS). An examination of the tree-below for the contextual feature TRANS shows that the structure is present. Thus a verb with the feature specification + TRANS would be suitable for insertion, and one with the feature specification - TRANS would not. If the feature specification is omitted, the verb will be suitable for insertion.

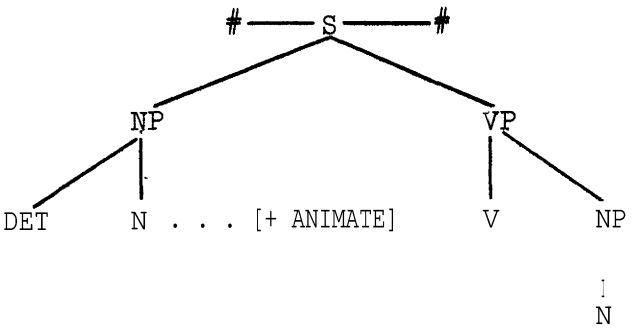


The contextual feature TRANS is an example of a strict local subcategorization feature as discussed by Chomsky [3].

The contextual feature label ANIMSUB defined by

$$\text{ANIMSUB} = \langle\ S\ \langle\ \#\ NP\ \langle\ \%\ N|+\ \text{ANIMATE}|\rangle\ VP\ \langle\ \_\_\ \%\ \rangle\ \#\ \rangle\ \rangle$$

may be used to specify that a verb must take an animate subject. A verb positively specified for this contextual feature can appear in the tree below.



Contextual features that include complex symbols correspond to Chomsky's notion of a selectional feature [3]. Contextual features appearing in complex symbols that are embedded in contextual features are treated in the same manner as inherent features.

We have extended the notion of a contextual feature by optionally allowing it to contain a restriction We have included this capability by analogy with our definition of structural descriptions for transformations, Integer labels may be assigned to structures or choices in a contextual feature and referred to in the restriction to impose further requirements on the environment., Such restrictions appear to be useful, e.g., the verb "condescend" requires that the subject noun phrase of the matrix sentence and that of the embedded sentence be the same. The use of restrictions makes it possible to-give alternative formulations for contextual feature descriptions. For example, the contextual feature label ANIMSUB defined earlier could also be given as:

ANIMSUB = ⟨ S ⟨ # NP ⟨ % 1N ⟩ VP ⟨ __ % ⟩ # ⟩ , WHERE 1 COMP |+ ANIMATE|)

COMP indicates that |+ ANIMATE| is compatible with the tree node corresponding to the node labeled 1 in the feature description.

When a vocabulary word has been found suitable for insertion in the tree, that is, its complex symbol is compatible and the tree meets the contextual feature specifications, the major function of the contextual feature has been served.' In many grammars, contextual features play no further part in sentence generation and could be removed from the complex symbol before it becomes part of the tree, However, a contextual feature describes the tree structure at the time of lexical insertion. Furthermore, this structure may be modified by the application of transformations. Therefore, by carrying the contextual features into the tree, it is possible to use them as "memory" devices. After lexical insertion, transformations may reference these features to determine the state of the

tree at the time of lexical insertion even though the tree may have been altered by the application of earlier transformations.

In the remainder of the section, we point out some differences between contextual features and structural descriptions for transformations. An important distinction is the underline symbol __ . This symbol appears in a contextual feature to denote the location of the lexical category node in the tree. It never appears in the structural description for a transformation except possibly in a contextual feature in an embedded complex symbol used to test early tree structure as just discussed, With this exception, the underline must appear only once in the structure portion of a contextual feature. A contextual feature specifies that the tree be examined around a particular node located by the underline symbol. A transformation structural description specifies environment that does not have this constraint. Since a contextual feature indicates that an "inside-out? analysis is to be performed around the underline __, we restrict the general form of a transformation structural description by requiring that a contextual feature be a structure (rule 2.04, Appendix A) rather than a structural analysis (rule 2.02, Appendix A) . The result is that a contextual feature must be dominated by a single element, which must be a node,

3. Lexicon

We consider only syntactic aspects of the structure of the lexicon, and do not consider definition and representation of phonological and semantic properties. Aspects of phonology are treated by Halle [4]. The semantic content of the lexicon remains an open question,

The lexicon has two parts, the prelexicon and the lexical entries. The prelexicon contains feature definitions and redundancy rules. The lexical entries are comprised of the vocabulary words and their complex symbols. The format for the lexicon is defined by syntax rules 7.01 through 7.13 of Appendix A.

Prelexicon

An ordered list of category features is given as part of the feature definitions. This list must contain all categories for which lexical insertion is to be performed. The order in the list specifies the category order for insertion in the tree. For example, the category list

CATEGORY V N COP DET .

would specify that verbs (V) are to be inserted first, followed by nouns (N), then copulas (COP), and then determiners (DET) .

Inherent features may optionally be defined as part of the feature definitions. This serves as a record of the inherent features that are used. Features not defined as category features or inherent features or contextual features are assumed to be inherent.

The definition of labels for contextual feature descriptions is a useful option. The labels appear in complex symbols and avoid the necessity of writing the complete contextual feature description. For example, a label denoting commonness for nouns could be defined by:

$$\text{COMMON} = \langle\ \text{NP}\ \langle\ \text{DET}\ \_\_\ \rangle\ \rangle$$

This label would then appear in complex symbols, as in:

$$\text{boy}|+\ \text{N}\ +\ \text{COMMON}|$$

If contextual feature labels are defined for all contextual features, updating the lexicon to reflect modifications in the phrase structure is not difficult because only the descriptions appearing in the label definitions need be altered. The redundancy rules that apply to the complex symbols that appear in the grammar are also defined in the pre-lexicon.

Lexical entries

Lexical entries comprise the major portion of the lexicon. A lexical entry is a set of vocabulary words, and an associated set of complex symbols. Each complex symbol represents a different sense in which one of the vocabulary words may be used. Each vocabulary word in the list may be paired with any one of the complex symbols in the complex symbol list. Thus, the complex symbol set is interpreted as a disjunction of complex symbols. Because a complex symbol is a list of

feature specifications that is interpreted as a conjunction, we have, in effect, a normal form in which any logical combination of complex symbols and feature specifications may be represented. We refer to the pair consisting of a word and a complex symbol as a "lexical item", It is lexical items that are selected from lexical entries in the lexicon for insertion in the tree. Thus, the lexical entry George Bill |+ N - COMMON - COUNT + HUMAN| represents two lexical items. This form for a lexical entry makes possible a compact representation, and can 'be used to indicate that certain vocabulary words have the same syntactic properties., However, if desired, each lexical item may be written as a distinct lexical entry.

The lexicon shown in Figure 1 was constructed from examples given by Chomsky [3]. The phrase structure rules are included because the form of the contextual feature depends on the phrase structure. The category order for lexical insertion will be verbs (V), then nouns (N), and finally, determiners (DET) . Four features are defined as inherent: ABSTRACT, ANIMATE, COUNT, and HUMAN. The contextual feature labels are TRANS (to distinguish transitive and intransitive verbs), COMMON (for nouns), ANIMSUB (for verbs taking + ANIMATE subjects), NANIMSUB (for verbs taking - ANIMATE subjects), NABSTOBJ (for verbs taking - ABSTRACT objects), ANIMOBJ (for verbs taking + ANIMATE objects), and NANIMOBJ (for verbs with - ANIMATE objects). Four redundancy rules are given, followed by the lexical entries, This lexicon is not intended to be complete; it is given only to illustrate the formats.

To simplify the selection of items from the lexicon, each lexical entry is linked to an appropriate category list. That is, all the nouns

```
            "  FRAGMENT FROM ASPECTS  "
PHRASESTRUCTURE
      S = # NP VP # .
      VP = V (NP) .
      NP = (DET) N .
$ENDPSG
LEXICON
      CATEGORY  V  N  DET  .
      INHERENT  ABSTRACT  ANIMATE  COUNT  HUMAN  .
      CONTEXTUAL
            TRANS = <VP<_NP>>,
            COMMON = <NP<DET_>>,
            ANIMSUB =  <S<#NP<%N|+ANIMATE|>VP<_%>#>>,
            NANIMSUB = <S<#NP<%N|-ANIMATE|>VP<_%>#>>,
            ANIMOBJ =  <VP<_NP<%N|+ANIMATE|>>>,
            NANIMOBJ = <VP<_NP<%N|-ANIMATE|>>>,
            NABSTOBJ = <VP<_NP<%N|-ABSTRACT|>>> .
      RULES
            |+COUNT| => |+COMMON|,
            |+ABSTRACT| => |+COMMON -ANIMATE|,
            |+HUMAN| => |+ANIMATE|,
            |+ANIMATE| => |-ABSTRACT| .
      ENTRIES
            SINCERITY VIRTUE  |+N -COUNT +ABSTRACT|
            BOY  |+N +COUNT +HUMAN|
            GEORGE BILL  |+N -COMMON -COUNT +HUMAN|
            THE  |+DET|
            EAT  |+V +TRANS +ANIMSUB +NABSTOBJ|
                 |+V -TRANS +ANIMSUB|
            GROW |+V +TRANS +ANIMSUB +ANIMOBJ|
                 |+V -TRANS +ANIMSUB|
            FRIGHTEN |+V +TRANS +ANIMOBJ|
            ELAPSE OCCUR |+V -TRANS +NANIMSUB|
            ADMIRE |+V +TRANS +ANIMSUB|
            READ |+V +TRANS +ANIMSUB +NANIMOBJ +NABSTOBJ|
            BUTTER |+N -COUNT -ABSTRACT -ANIMATE|
            BOOK |+N -ANIMATE +COUNT|
            BEE |+N +COUNT +ANIMATE -HUMAN|
            WEAR |+V +NANIMOBJ|
                 |+V -TRANS +ANIMSUB|
            OWN |+V +TRANS +ANIMSUB +NABSTOBJ|
            KNOW |+V +TRANS +ANIMSUB|
            EGYPT |+N -COMMON -COUNT -ANIMATE|
            DOG |+N +COUNT -HUMAN +ANIMATE|
            CARROT |+N +COUNT +ANIMATE -HUMAN|
            RUN |+V +ANIMSUB| .
$ENDLEX
```

Figure 1

are linked together, all the verbs, and all the determiners. This is the only form of hierarchical structure provided, and has been done for convenience rather than linguistic necessity.

There remain unanswered questions regarding the exact nature of the lexicon. More hierarchical structure may be useful, although it is probably not necessary. We have preferred to keep the lexicon in a simple and flexible format so that it will be easy to understand and to modify.

4. Lexical Insertion

The algorithm for lexical insertion will be described in an informal manner. This algorithm has been implemented in a working computer program [1]. The examples used to illustrate the operation of the algorithm were generated by this program working with other routines in the system described in [6].

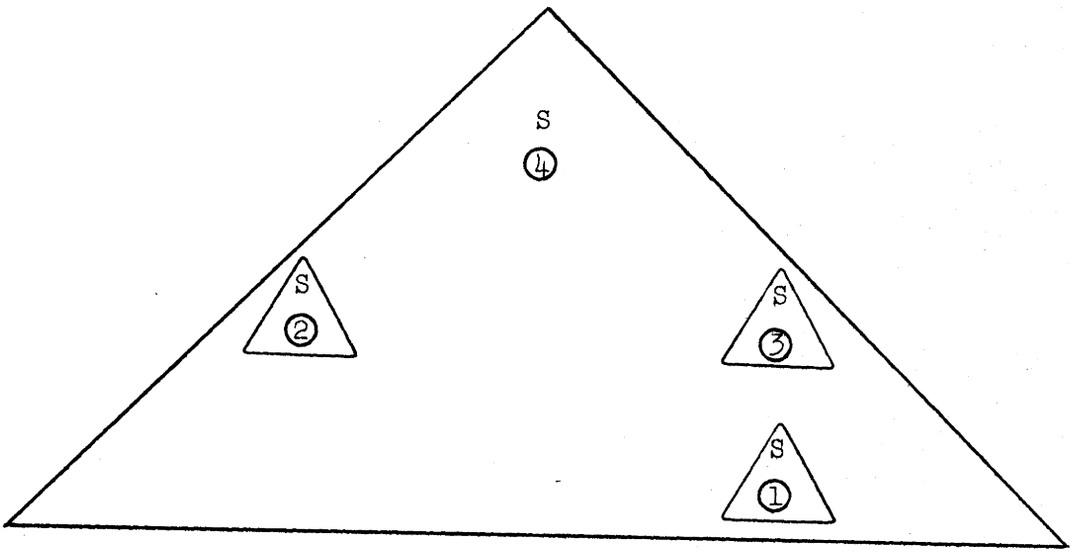
Lexical insertion, as it will be described here, occurs after the generation of a preterminal string by the phrase structure component. Lexical items are selected from the lexicon and inserted in the tree, Following lexical insertion,, transformations are applied, The directed random generation of base trees in our system (excluding lexical insertion) is described by Friedman [5], and application of transformations is discussed by Friedman and Pollack [13].

Chomsky [3] proposed two styles of lexical insertion. In the first, complex symbols, including inherent and contextual features, were introduced by rewriting rules. A lexical item was suitable for insertion if its complex symbol was nondistinct from a complex symbol in the tree. The alternative style did not introduce complex symbols by rewriting rules. Rather, a lexical item was suitable if its complex symbol was nondistinct from a tree complex symbol and if each contextual feature specification in the complex symbol was present in the tree. Chomsky remarks [3, p. 122] that the contextual features may be thought of as the Boolean structure index (structural description) of a transformation, and that the insertion of a lexical item can be viewed as a substitution transformation. It is not clear from his discussion whether he intends lexical insertion to be merely thought of as a transformation,

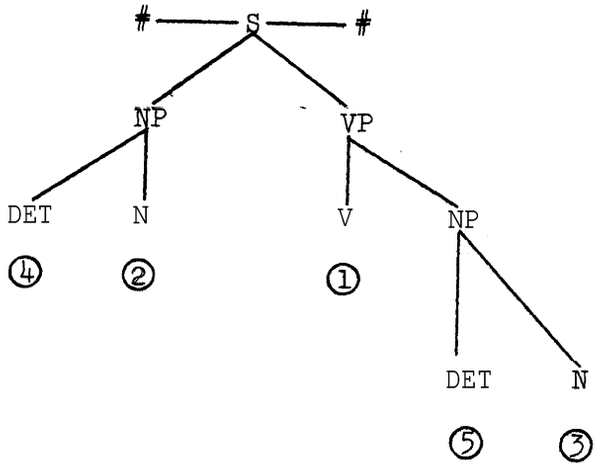
or actually implemented as a transformation. In our interpretation, lexical insertion is done independent of the transformation phase, However, there is much in common between the two, The separation was made for several reasons. First, it was desired to study the insertion process to better understand its basic nature. Second, there are differences in the analysis performed for a contextual feature and the analysis performed for a transformation. These differences were discussed in section 2, and are considered further by Friedman and Martner [12]. Third, a complex symbol may contain more than one contextual feature, each of which is a type of structural description. Thus, to specify a conjunction of contextual features as a structural description for a transformation, we would have to either combine all contextual features into a single inclusive one, or allow conjunctions of structural descriptions to appear in transformations,, Also, there is no convenient way to specify the selection of a vocabulary word as part of the structural change of a transformation, If the vocabulary words are included in the transformation, the concept of a lexicon as a distinct entity is lost.

The a-lgorithm

If a tree has embedded sentence subtrees, they are considered in lowest to highest, right to left order, In the example below, subtrees would be considered in the numbered order for the insertion of lexical items,



Within a sentence subtree, lexical items are inserted for each lexical category node appearing in the subtree. The category nodes are considered by type in the order specified in the lexicon In each category, the order is left to right in the tree. Thus, if the lexicon defined the category order as V N DET, nodes would be considered in the numbered order shown in the tree below.



The order in which categories are considered can effect the efficiency of the insertion algorithm. That is, more lexical entries may have to be tested before a suitable complex symbol is located. Since vocabulary words are selected in a random manner, the same terminal strings will not necessarily be obtained when different category orders are used,

Selection of a lexical item

When a particular category node in the tree has been found, the lexicon must be searched to find a vocabulary word and complex symbol that are suitable for insertion. Since the lexical entries are linked by category, it is possible to search the lexicon considering only entries in the proper category, We desire to select a lexical item from the set of acceptable items that could be inserted. It would be inefficient to form a list of all acceptable items and then choose from this list. Therefore, a different method of selection has been devised.

This method tests random entries in the following manner. An entry is selected at random[2] from the list of entries of the appropriate category, Each-entry may contain many complex symbols so each one is compared with the complex symboi associated with the category node. The comparison used is compatibility. An

2. The phrase "selected at random" as used in this paper has the following interpretation. Given a collection of n objects (lexical entries, complex symbols, vocabulary words, etc.), number them from 1 to n . Compute a random number using a uniform probability density function over the interval (1,n) . This number identifies the object that has been selected at random. A uniform probability density function is such that each object in the collection has equal chance of being selected,

analysis of the tree is performed for each contextual feature specification to determine if the tree has the desired structure. In the analysis, the underline _ in the contextual feature must correspond to the category node. If the contextual feature has embedded complex symbols, they are compared for compatibility with the corresponding complex symbols in the tree. Since the analysis of a contextual feature can be complicated, it is performed only once for any contextual feature. The value of the feature is saved in case the contextual feature appears in another complex symbol that is tested for insertion at the same category node.

When acceptable complex symbols are found in an entry, one is selected at random. A vocabulary word is then selected at random from the list of vocabulary words for the entry. This vocabulary word and the complex symbol that is the result of the compatibility test form the lexical item that will be inserted in the tree. If the lexical entry does not contain an acceptable complex symbol, the number of entries to 'be tested is reduced by one and the entry examined is marked so that it will not be retested. An increment is computed that gives another random entry. The complex symbols for this entry are tested and the process continues. The process terminates when an acceptable lexical item is selected, or when no more entries remain to be tested. In the latter case, there is no acceptable item for insertion at the node and the insertion algorithm continues with the next node to be considered.

This method of selection weights lexical entries equally. Since an entry may have more than one complex symbol, complex symbols do not have exactly equal probabilities of being selected, If this is an important consideration, the lexicon should be defined so that each entry consists of a single complex symbol with its associated vocabulary words, If the lexical items are to receive equal probability of selection, the lexicon should be defined so that each entry is a single vocabulary word and a single complex symbol.

Insertion of a lexical item

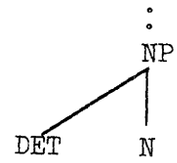
The vocabulary word selected is inserted as a daughter of the category node. The complex symbol obtained as a result of the compatibility test is attached to the category node. Suppose the lexical item selected was

dog|+ N + COUNT - HUMAN|

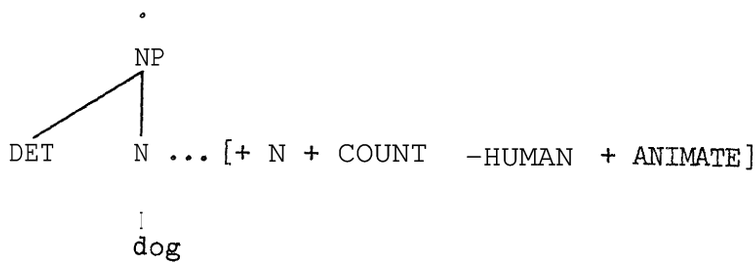
and the redundancy rule

|- HUMAN| => |+ ANIMATE|

had been defined. This complex symbol is suitable for insertion in the partial tree below.

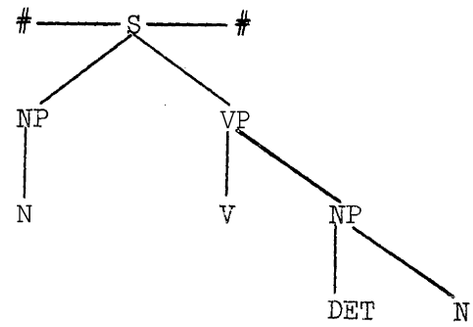


After insertion, we have

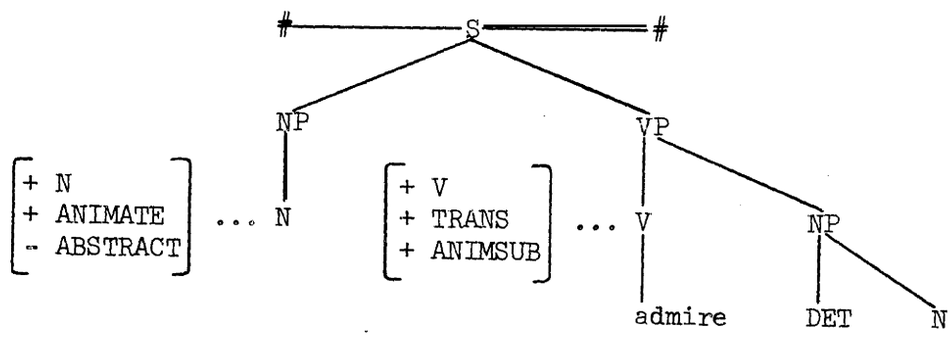


Side effects

The process of inserting a lexical item is now complete, except for the treatment of side effects. Side effects must be considered when a complex symbol containing a contextual feature with an embedded complex symbol is inserted (e.g., a selectional feature). Such a feature is defined by the label ANIMSUB defined in the lexicon in Figure 1. This feature appears in the complex symbol for the verb "admire? Given the base tree shown below, this verb would be suitable for insertion since the subject noun is unspecified for the feature ANIMATE and thus compatible (nondistinct), as desired.



Once this verb is inserted, the subject noun must be positively specified for the feature ANIMATE . This will insure that an appropriate noun is selected. The complex symbol obtained from the compatibility test performed during the analysis of the tree for the contextual feature is attached to the category node for the subject noun.



This discussion of side effects is not complete. The generality of the definition of the structure (rule 2.04, Appendix A) of a contextual feature description allows complex environments to be described, If negations (¬) appear in the feature description, appropriate action is often difficult to determine. Side effects may also be introduced by the restriction (rule 3.01, Appendix A)

in the feature description Again, the form of a restriction makes a thorough treatment difficult,[3] The principle issue is clear however: the insertion of a lexical item may produce effects on other nodes in the tree that must be accounted for if a grammatical sentence is to be obtained,

After the lexical item is inserted, the algorithm continues with the next appropriate category node. The process terminates when all category nodes suitable for lexical items have been considered,

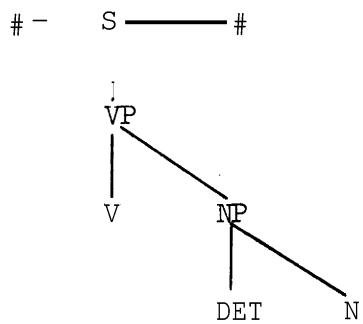
Negatively specified contextual features

Before giving examples of the operation of the insertion algorithm, we digress to discuss negatively specified contextual features, A negatively specified contextual feature implies that the environment described by the feature must not be present The interpretation is usually clear when the contextual features do not contain complex symbols. But if we attempt to describe a verb that must take an inanimate subject by the feature specification - ANIMSUB, where ANIMSUB is a contextual feature label as defined in Figure 1, we encounter difficulties. First, the tree just given, prior to verb insertion, satisfies this feature. Thus, the feature specification - ANIMSUB in a complex symbol would cause the lexical item to be rejected, This is clearly not what is desired, Note, however, that if a new contextual feature label NANIMSUB is defined as in Figure 1, and if the complex symbol contains the feature specification

3. In the current implementation of the algorithm, side effects are not treatedifthey are introduced by a restriction.

+ NANIMSUB, then the complex symbol would be acceptable,, at least for this feature. Further,, the subject noun would be correctly specified as - ANIMATE by side effects.

One might propose that the lexical insertion algorithm perform a function similar to side effects when a negatively specified contextual feature with an embedded complex symbol is encountered, This function would attempt to modify the tree so that the contextual feature would fail. This could easily be done in the previous example by specifying the subject noun as - ANIMATE . The tree would fail for the feature ANIMSUB and a complex symbol marked - ANIMSUB would be acceptable. Such a proposal encounters difficulties, For example, the tree shown below would also fail for the feature ANIMSUB, since an NP dominating an N is not present.



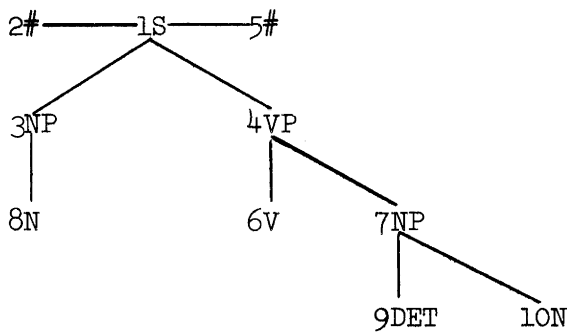
Therefore, a verb with the feature specification - ANIMSUB would be acceptable for insertion in this tree, as well. This may not be desirable and is not what was intended by the definition of the feature. We conclude that contextual features containing complex symbols should be positively specified,

Similar difficulties may be encountered with negatively specified subcategorization features. The possibilities for satisfying the negative feature specifications must be carefully considered. In this case, the problem is less complex since compatibility and side effects are not issues,

If contextual features such as ANIMSUB are negatively specified, they are treated in the following manner No attempt is made to cause the feature to fail. Thus as in the example, if verbs are inserted first, no verb with the feature specification - ANIMSUB would be acceptable for insertion in a tree such as shown on page 30. Note, however, that by inserting nouns before verbs in this tree, the feature specification - ANIMSUB would be treated as intended. Therefore, category order in which items are inserted can be important. Side effects for negatively specified contextual features are not considered,

An example

Using the phrase structure and lexicon of Figure 1, we monitor the operation of the insertion algorithm on the base tree below. This base tree, while simple, will illustrate the issues.

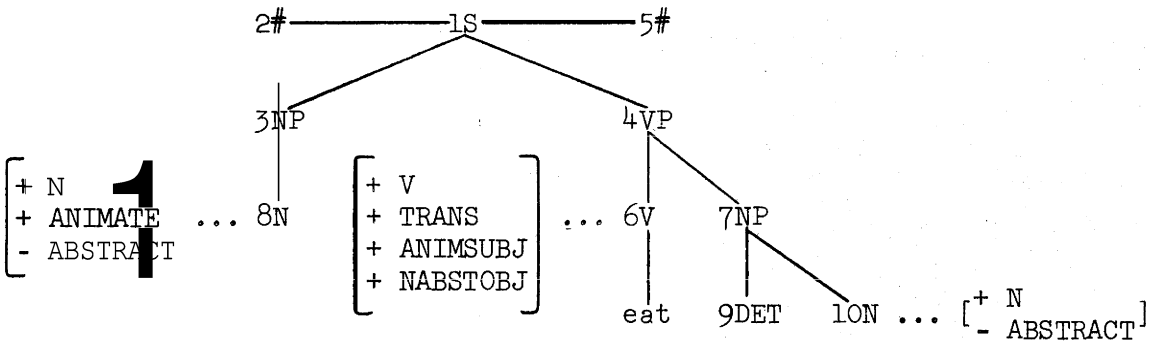


Each node is numbered to simplify reference. Verbs are inserted first. There are ten verb entries in the lexicon. These entries are (complex symbols omitted):

V1 eat
v2 grow
V3 frighten
v4 elapse occur
V5 admire
V6 read
V7 wear
V8 own
V9 know
V10 run

Node 6 is the only verb node in the tree., Entry V4 in the verb list is randomly selected to be tested for insertion, The complex symbol |+ V - TRANS + NANIMSUB| is compatible with the complex symbol for node 6, so the tree is analyzed for the contextual feature with the label TRANS . This analysis determines that the tree does have the structure for a transitive verb and this complex symbol is rejected. The result of the analysis for TRANS is saved. Since there are no more complex symbols to be tested for entry V4, this entry is marked as unacceptable for insertion at this node. A new random entry, entry V1, is selected in the manner described earlier, This entry has the complex symbols |+ V + TRANS + A-NIMSUB + NABSTOBJ| and |+ V - TRANS + ANIMSUB| . The first complex symbol is compatible, so its contextual features are analyzed, The contextual feature TRANS is already known to be present, so no analysis is necessary for this feature. The tree is analyzed for the features defined by ANIMSUB and NABSTOBJ, and found to be acceptable, The second complex symbol is then tested and rejected because of the feature specification - TRANS . This entry has only a single vocabulary word, so this word and the complex symbol are

inserted. Next, side effects are considered, The feature specification + ANIMSUB requires that node 8 be specified as + ANIMATE ; and the feature specification, + NABSTOBJ, requires that node 10 be specified - ABSTRACT . After insertion of the verb, the tree appears as shown below.



There are no more verbs in the tree, so node 8 is next. There are nine noun entries in the lexicon, Omitting complex symbols, they are:

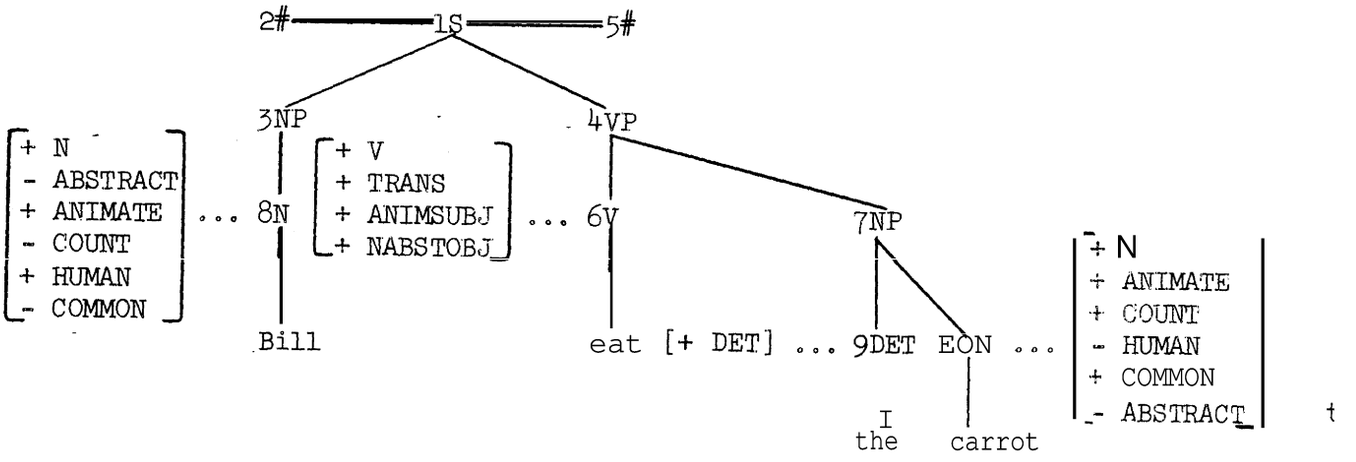
N1 sincerity virtue
N2 boy
N3 George Bill
N4 butter
N5 book
N6 bee
N7 Egypt
N8 dog
N9 carrot

Entry N2 is randomly selected. The complex symbols are compatible, but the tree does not have the contextual feature specification +⟨ NP ⟨ DET __) ⟩ and, therefore, this entry is rejected. Entry N9 is randomly selected from the eight remaining entries, and rejected for

the same reason (the analysis 'is not performed a second time)" Entry N3 is selected next. This complex symbol is compatible, and has the proper contextual feature specification, There are two vocabulary words for this entry, so "Bill" is selected at random, and the word and complex symbol are inserted. This complex symbol has no contextual features with embedded complex symbols, so there are no side effects,

Node 10 is next. Entry N7 is selected and found to be compatible, but an analysis of the tree for the feature specification - COMMON fails,, Notice that the analysis must be performed again, since the __ in the contextual feature corresponds to node 10 instead of node 8, as on the previous analysis, Entry N9 is selected and found to be acceptable. The vocabulary word, "carrot", and the complex symbol are inserted, There are no side effects

This completes processing of the nouns. The determiner, "the", is selected for insertion at node 9, and lexical insertion is complete for this tree. The base tree after lexical insertion is shown below.



Lexical insertion with directed random sentence generation

Friedman [13] describes techniques for the random generation of base trees according to a phrase structure grammar. These techniques allow the generation to be "directed", so that a certain subclass of possible trees can be obtained, These techniques are implemented in a computer program, and this program can 'be used to produce base trees for input to the program that performs lexical insertion. Some of the methods for constraining the form of the base tree may have an effect on lexical insertion and must be discussed briefly,

The form desired for the base tree is indicated by a "skeleton", This skeleton is actually a portion of the base tree with extra "restrictions". The complete base tree is built around the skeleton 'by random selection of appropriate phrase structure rules, The skeleton may contain complex symbols, These complex symbols are considered in lexical insertion There may be restrictions demanding equality of terminal nodes. The skeleton may also contain particular vocabulary words, which are to appear in the base tree. In the last two cases, the lexical insertion process is modified slightly. First, the lexical insertion algorithm is executed treating only these effects If vocabulary words are given in the input base tree, their corresponding complex symbols must be located and inserted in the tree. The side effects of these complex symbols must also be treated. If there are equality restrictions in the skeleton, lexical items are selected and inserted to fulfill these restrictions, After these operations have been performed, the algorithm is executed again, to complete the insertion process Without these considerations, side effects might result in a distinct feature

specification for the complex symbol for a vocabulary word present in the skeleton.

5. Suggestions for Further Study

The nature of vocabulary words

A topic of current research is the form of words that appear in the lexicon. Two positions are advocated. The first maintains that the lexicon should contain word stems and affixes. Words such as "readable" and "self-reliant" are derived by the application of transformations, This view is known as the "transformationalist" position since the major burden is on the transformation phrase as opposed to the phrase structure or lexicon. This view is supported by Chapin [2] and Lakoff [17]. The second position holds that the lexicon must contain separate entries for distinct words and that the phrase structure component must be enriched to introduce the appropriate categories and structure. This position is known as the "lexicalist" view and has been advocated by Chomsky [4]. A compromise is no doubt necessary* Chomsky, himself, argues [4] that derived nominals should have the form of base sentences while gerundive nominals should be formed by transformation.

Arguments on questions such as these are based on empirical evidence. The unfortunate difficulty is that without thorough evaluation of the hypotheses, the results are often hard to accept. There is the feeling (often justified) that there are unmentioned sentences or words that provide counter-examples. The system we have defined provides an excellent tool for this type of study,, Sentences can be quickly generated and examined, The randomness of the generation and selection of lexical items gives a degree of confidence in the results if they are positive, The availability of a lexicon gives a permanent and visible record of

the vocabulary considered, With computer aids, it should be possible to give a more quantitative flavor to future research,

Syntactic features

The study of syntactic. features has also been hampered by the lack of a clear exposition of the issues. Examples of complex symbols often do not give complete feature specifications, and this casts doubt on the validity of the assertions. The concept of a syntactic feature is in need of thorough study., Consistent sets of inherent features should be defined for limited sets of vocabulary words. The vocabulary should then be extended until it is representative of the language, To illustrate the difficulties, the lexicon of Figure 1 contains the entries:

bee |+ N + COUNT + ANIMATE - HUMAN|

carrot|+ N + COUNT + ANIMATE - HUMAN|

These vocabulary words do not represent the same type of entity, but there is no distinguishing them on the basis of their complex symbols, There is a question whether the-feature ANIMATE refers to animate in the sense of being alive or in the sense of being animal. Are the above entries really syntactically identical with the differences established on semantic grounds?

The definition of complex symbols for verbs is also subtle. How does one account for the different senses of "grow" as in

Bill grows carrots,

Bill grows tired,

One sense appears to take animate objects and the other inanimate objects.,

Recognition procedures

In this paper we have studied grammar as a generative device. With recognition procedures for transformational grammar, the base and surface structure of the sentence is constructed given the terminal string of the surface tree. Recognition procedures have been studied elsewhere [19, 23] but none of this work has dealt with syntactic features. Lexical lookup in recognition is the counterpart of lexical insertion in generation. Lexical lookup is difficult since there are often many complex symbols for a vocabulary word.

Idioms

Although idioms play an important role in language, grammars do not usually account for them. One possibility is that an idiom should be represented in the lexicon as a single entry, For example "quick as a wink" might be found as an adverb entry, This entry might appear in

He went quick as a wink.

The representation of the "vocabulary word" for the idiom could be either a tree structure representing the structure of the idiom or simply a single "word", e.g., quick-as-a-wink,

Alternative formulations for transformational theory

The work described has been based on one general interpretation of transformational grammar. There are other formulations, particularly of the lexicon, which could be studied and formalized using the same techniques. The syntax we used to define a complex symbol for example could be revised to allow other interpretations such as Boolean combinations of feature specifications as proposed by Lakoff [17]. We feel quite strongly that regardless of the final form of the grammar it should be evaluated for inconsistencies and as to its adequacy by an unbiased device such as the computer. We have defined a system that treats the components of transformational grammar. The programs of this system can be combined in different ways to yield different interpretations of the theory. We now mention some alternative interpretations related to the lexicon and lexical insertion that could be evaluated if interest is shown.

— The introduction of complex symbols by rewriting rules as proposed by Chomsky [3] and used in Lakoff [17] has already been discussed.

— Chapin [2] has defined the principle of "homogeneity of components". This principle requires that parts of one component of the grammar are not used in another. For example, transformations do not appear in the phrase structure, An investigation could be made of the consequences of violating this principle, Perhaps lexical insertion should be split, part taking place after the phrase structure phase and part after the transformations

have been applied. Klima [16] does suggest such an approach. He would insert the lexical items for the lowest embedded sentence subtree then immediately perform the transformations that are local to that subtree. Lexical insertion and transformations would continue to be performed, alternately, until a surface tree is obtained.

— Lakoff [17] introduces extra features such as "structure description features". He also uses dual sets of complex symbols in the tree. One set generated by the phrase structure and the other set copied from the lexicon. These additions are used to observe the types of ungrammatical sentences obtained when nondistinctness comparisons and contextual feature specifications are violated,

— Robinson [20] has studied the use of dependency grammars for generation. These grammars generate trees with vocabulary words assigned to nonterminal nodes in the tree.

COMPLETE SYNTAX FOR TRANFORMATIONAL GRAMMAR

0.01 TRANSFORMATIONAL GRAMMAR ::= PHRASE STRUCTURE LEXICON TRANSFORMATIONS SEND

1.01 SPECIFICATION ::= TREE optf , clist< WORD TREF >> .
1.02 TREE : := NODE opt; COMPLEX SYMBOL > optf < listf TREE >>>
1.03 NODE : := WORD or SENTENCE SYMBOL or BOUNDARY SYMBOL
1.04 SENTENCE SYMBOL : := S
1.05 BOUNDARY SYMBOL : := #

2.01 STRUCTURAL DESCRIPTION ::= STRUCTURAL ANALYSIS optf , WHERE RESTRICTION > .
2.02 STRUCTURAL ANALYSIS ::= listf TERM >
2.03 TERM ::= opt< INTEGER > STRUCTURE or op INTEGER > CHOICE or SKIP
2.04 STRUCTURE ::= ELEMENT optf COMPLEX SYMBOL > opt< opt< " > optf />< STRUCTURAL ANALYSIS >>
2.05 ELEMENT ::= NODE or * or _ .
2.06 CHOICE ::= (c l i s t f STRUCTURAL ANALYSIS >)
2.07 SKIP ::= %

3.01 RESTRICTION ::= booleancombinationf CONDITION >
3.02 CONDITION ::= UNARY CONDITION or BINARY CONDITION
3.03 UNARY CONDITION ::= UNARY RELATION INTEGER
3.04 BINARY CONDITION : := INTEGER BINARY TREE RELATION NODE DESIGNATOR or
INTEGER BINARY COMPLEX RELATION COMPLEX SYMBOL DESIGNATOR
3.05 NODE DESIGNATOR ::= INTEGER or NODE ,
3.06 COMPLEX SYMBOL DESIGNATOR ::= COMPLEX SYMBOL or INTEGER
3.07 UNARY RELATION : := TRP: or . NTRM or NUL or NNUL or DIF or NDIF
3.08 BINARY TREE RELATION ::= EQ or NEQ or DOM or NDOM or DOMS or NDOMS or DOMBY or NDOMBY
3.09 BINARY COMPLEX RELATION ::= INC1 or NINC1 or INC2 or NINC2 or CSEQ or NCSEQ or NDST
or NNDST or COMP or NCOMP

4.01 SYMBOL ::= | | B O ::= | listf FEATURE SPECIFICATION > |
4.02 SPECIFICATION : := VALUE FEATURE
4.03 ::= CATEGORY FEATURE or INHERENT FEATURE or CONTEXTUAL FEATURE or RULE FEATURE
4.04 CATEGORY FEATURE ::= CATEGORY
4.05 CATEGORY ::= WORD
4.06 INHERENT FEATURE ::= WORD
4.07 RULE FEATURE ::= TRANSFORMATION NAME
4.08 CONTEXTUAL FEATURE ::= CONTEXTUAL FEATURE LABEL or CONTEXTUAL FEATURE DESCRIPTION
4.09 CONTEXTUAL FEATURE DESCRIPTION ::= < STRUCTURE opt< , WHERE RESTRICTION > >
4.10 VALUE ::= + or - or *

5.01 STRUCTURAL CHANGE ::= clist< CHANGE INSTRUCTION >
5.02 CHANGE INSTRUCTION : := CHANGE or CONDITIONAL CHANGE
5.03 CONDITIONAL CHANGE ::= IF < RESTRICTION > THEN < STRUCTURAL CHANGE >
optf ELSE < STRUCTURAL CHANGE >>
5.04 CHANGE ::= UNARY OPERATOR NODE DESIGNATOR or
TREE DESIGNATOR BINARY TREE OPERATOR NODE DESIGNATOR or
COMPLEX SYMBOL DESIGNATOR BINARY COMPLEX OPERATOR NODE DESIGNATOR
or NODE SYMBOL DESIGNATOR TERNARY COMPLEX OPERATOR NODE DESIGNATOR NODE DESIGNATOR
5.05 NODE DESIGNATOR : := INTEGER WORD
5.06 COMPLEX SYMBOL DESIGNATOR ::= COMPLEX SYMBOL INTEGER
5.07 TREE DESIGNATOR ::= (TREE) or N O D E
5.08 BINARY TREE OPERATOR ::= ADLAD or ALADE or ADLADI or ALADEI or ADFID or AFIDE or
ADRIS or ARISE or ADRISI or ARISEI or ADLES or ALESE or ADLESI or ALESEI
or ADRIA or ARIAE or SUBST or SUBSE or SUBSTI or SUBSEI
5.09 BINARY COMPLEX OPERATOR : := ERASEF or MERGEF or SAVEF
5.10 UNARY OPERATOR ::= ERASE or ERASEI
5.11 TERNARY COMPLEX OPERATOR ::= MOVEF

```
6.01  PHRASE STRUCTURE : : = PHRASESTRUCTURE listf PHRASF STRUCTURF RULE }$END
6.02  PHRASE STRUCTURE RULE : := RULE LEFT = RULE RIGHT .
6.03  RULE LEFT : := NODE
6.04  RULE RIGHT ::= NUDE or list{ RULE RIGHT } or (list{ RULF RIGHT }) or ( clistf RULF RIGHT })

7.01  LEXICON ::= LEX I CON PRELEX I CON LEX I CAL ENTR I ES $END
7.02  PRELEXICON : : = FEATURE DEF I N IT I GNS opt{REDUNDANCY RULES }
7.03  FEATURE DEFINITIONS ::= CATEGORY DEFINITIONS opt{INHERENT DEFINITIONS } opt{ CONTEXTUAL DEFINITIONS }
7.04  CATEGORY DEFINITIONS ::= CATEGORY 1 ist{ CATEGORY FEATURE } .
7.05  INHERENT DEFINITIONS : : = INHERENTlist{INHERENT FEATURE } .
7.06  CONTEXTUAL DEFINITIONS : := CONTEXTUAL clist{ CONTEXTUAL DEFINIT I ON }
7.07  CONTEXTUAL DEF INITION ::= CONTEXTUAL FEATURE LABEL = CONTEXTUAL FEATURF DESCRI PTION
7.08  CONTEXTUAL FEATURE LABEL::= WORD...
7.09  REDUNDANCY RULES: : = RULES clist{ REDUNDANCY RUI E } .
7.10  REDUNDANCY RULE : : = COMPLEX SYMBOL => COMPLEX SYMBOL
7.11  LEXICAL ENTRIES ::= ENTRIES list< LEXICAL ENTRY } .
7.12  LEXICAL ENTRY ::= list< VOCABULARY WORD } list{ COMPLEX SYMBOL }
7.13  VOCABULARY WORD : := WORD

8.01  TRANSFORMATIONS ::= TRANSFORMAT IONS 1 ist{ TRANSFORMAT I ON } CP CONL PR POGRAM. $END
8.02  TRANSFORMATION ::= TRANS IDENTIFICATION SD STRUCTURAI DESCRIPTION opt{ SC STRUCTURAI CHANGE . }
8.03  I DENTIFICATION ::= opt{ J NTEGER } TRANSFORMATION NAME opt{list{PARAMETER }} opt{ KEYWORDS } .
8.04  PARAMETER : : = GROUP NUMBER or OPTIONALITY O R REPETITION
8.05  GROUP NUMBER : : = I or II or III or IV or V or VI or VI I
8.06  OPTIONALITY ::= OB or O P
8.07  REPETITION ::= AC or ACAC or AACC or AAC
8.08  KEYWORDS ::= ( list{ NODE } )

9.01  CONTROL PROGRAM : : = sclist{opt{ LABEL : }INSTRUCTION }
9.02  LABEL ::= WORD
9.03  INSTRUCTION ::= KPT INSTRUCTION or IN I NSTRUCTION or I F I NSTRUCT ION
                      or GO INSTRUCTI ON or TRACE I NSTRUCTION or STOP INSTRUCTION
                      or T INSTRUCTION or < sclist{ INSTRUCTION } >
9.04     NSTRUCT I ON : : = TRANSFORMATION NAME or GROUP NUMBER
9.05  RPT INSTRUCTION : := E P T opt{ INTEGER } < CONTROL PROGRAM >
9.06  I NINSTRUCTION : : = IN TRANSFORMATION NAME (INTEGER) DO < CONTROL PROGRAM >
9.07  IF INSTRUCTION ::= I F INSTRUCTION T H E N GO INSTRUCTION o p t f E L S E GO INSTRUCTION }
9.08  GOINSTRUCTION : := GO TO LABEL .
9.09  TRACE I NSTRUCT I ON : := TRACE T INSTRUCTION TRACE SPECI F I CATION or UNTRACE T INSTRUCTION or TREE
9.10  TRACE SPECIFICATION::= BEFORE TEST or AFTER FAILURE or AFTER SUCCESS or AFTER CHANGE
9.11  STOP INSTRUCTION : :- STOP
```

REFERENCES

[1] Bredt, T. H. A computer program for lexical insertion. AF-26, Computer Science Department, Stanford University (May, 1968).

[2] Chapin, P. On the Syntax of Word Derivation in English. M.I.T. Thesis (1967).

[3] Chomsky, N. Aspects of the Theory of Syntax. M.I.T. Press, Cambridge, Massachusetts (1965).

[4] Chomsky, N. Remarks on nominalization, To appear in Rosenbaum, P. S., and Jacobs, R. (Eds.) Readings in Transformational Grammar. Blaisdell Publishing Co.

[5] Friedman, J. Directed random generation of sentences. CS-80, AF-15, Computer Science Department, Stanford University (October, 1967).

[6] Friedman, J. A computer system for transformational grammar. CS-84, AF-21, Computer Science Department, Stanford University (January, 1968).

[7] Friedman, J. Computer experiments in transformational grammar, I: fragments from Aspects. AF-22, Computer Science Department, Stanford University (February, 1968).

[8] Friedman, J. Computer experiments in transformational grammar, II: Traugott's grammar of Alfredian prose. AF-23, Computer Science Department, Stanford University (February, 1968).

[9] Friedman, J., and Martner, T. S. Computer experiments in transformational grammar, III: the IBM core grammar. AF-27, Computer Science Department, Stanford University (May, 1968).

[10] Friedman, J. (Ed.) Manual for a system of computer programs for transformational grammar. Computer Science Department, Stanford University (forthcoming).

[11] Friedman, J., and Doran, R. W. A formal syntax for transformational grammar. CS-95, AF-24, Computer Science Department, Stanford University (March, 1968).

[12] Friedman, J., and Martner, T. S. Analysis in transformational grammar. Computer Science Department, Stanford University (forthcoming),

[13] Friedman, J., and Pollack, B.W. A control language for transformational grammar. Computer Science Department, Stanford University (forthcoming).

[14] Halle, M. On the bases of phonology. in Fodor, J.A. and Katz, J.J. (Eds.) The Structure of Language, Prentice-Hall (1964).

[15] IBM System/360 FORTRAN IV language. File No. S360-25, Form C28-6515-5 (1966).

[16] Klima, E. S. Current developments in generative grammar. Kybernetika 1(1965) 184-197.

[17] Lakoff, G. On the Nature of Syntactic Irregularity. NSF-16, The Computation Laboratory, Harvard University (1965).

[18] Partee, Barbara H. Computer tests of AFESP case grammar I. Working paper 23, English Syntax Project, U.C.L.A. (March, 1968).

[19] Petrick, S. R. A Recognition Procedure for Transformational Grammars, M.I.T. Thesis (1965).

[20] Robinson, Jane J. A dependency-based transformational grammar. IBM Research Report, RC-1889, Yorktown Heights, N.Y.(1967).

[21] Rosenbaum, R., and Lochak, D. The IBM core grammar of English. In Lieberman, D. (Ed.) Specification and utilization of a transformational grammar. AFCRL-66-270 (1966).

[22] Traugott, Elizabeth C. Deep and surface structure in Alfredian prose. mimeographed. PEGS (1967).

[23] Zwicky, A.M., Friedman, J., Hall, B.C., and Walker, D.E. The MITRE syntactic analysis procedure for transformational grammars. Fall Joint Computer Conference 27 (1965),317-326.