R E P O R T R E S U M E S

ED 010 874 AL 000 037

SPECIFICATION AND UTILIZATION OF A TRANSFORMATIONAL GRAMMAR. FINAL REPORT.

BY- ROSENBAUM, PETER S. BLAIR, FRED
INTERNATIONAL BUSINESS MACHINES CORP.

REPORT NUMBER AFCRL-66-762 PUB DATE OCT 66
EDRS PRICE MF-$0.18 HC-$2.40 60P.

DESCRIPTORS- *COMPUTATIONAL LINGUISTICS, *ENGLISH, *TRANSFORMATION THEORY (LANGUAGE), INFORMATION PROCESSING, SYNTAX, GRAMMAR, DEEP STRUCTURE, SURFACE STRUCTURE, ENGLISH GRAMMAR I, YORKTOWN HEIGHTS, SENTENCE SYNTHESIZING PROGRAM (SSP), LISP

RESEARCH IN THREE AREAS OF COMPUTATIONAL LINGUISTICS IS DESCRIBED--(1) THE DESIGN AND DEVELOPMENT OF A TRANSFORMATIONAL GRAMMAR FOR A SUBSET TO GRAMMATICAL SENTENCES IN ENGLISH, (2) THE IMPLEMENTATION OF THIS GRAMMAR IN TERMS OF A SENTENCE SYNTHESIZING PROGRAM WRITTEN IN LISP 1.5, AND (3) THE USE OF SENTENCE SYNTHESIZING PROGRAMS FOR TRANSFORMATIONAL GRAMMARS GENERALLY. THE PRIMARY OBJECTIVES HAVE BEEN TO SPECIFY A DESCRIPTIVELY ADEQUATE TRANSFORMATIONAL GRAMMAR FOR ENGLISH, INCORPORATING RECENT THEORETICAL DISCOVERIES, AND TO DEVELOP COMPUTATIONAL PROCEDURES FOR TESTING THE DESCRIPTIVE ADEQUACY OF TRANSFORMATIONAL GRAMMARS OF ADVANCED DESIGN. AN ANALYSIS IS GIVEN OF THE RESULTS OBTAINED, AND METHODOLOGICAL LIMITATIONS ARE DISCUSSED. THE SENTENCE SYNTHESIZING PROGRAM HAS BEEN FOUND VALUABLE IN REVEALING ERRORS IN THE TRANSFORMATIONAL RULES. (KL) 22

AFCRL-66-762

SPECIFICATION AND UTILIZATION OF A TRANSFORMATIONAL GRAMMAR

Peter S. Rosenbaum
and Fred Blair

INTERNATIONAL BUSINESS MACHINES CORPORATION
Thomas J. Watson Research Center
Post Office Box 218
Yorktown Heights, New York 10598

Contract No. AF 19(628)-5127

Project 4641

Task 464102

FINAL REPORT October 1966

Period Covered: July 1965 through September 1966

Prepared
for

AIR FORCE CAMBRIDGE RESEARCH LABORATORIES
OFFICE OF AEROSPACE RESEARCH
UNITED STATES AIR FORCE
BEDFORD, MASSACHUSETTS 01730

AL 000 037

AFCRL-66-762

SPECIFICATION AND UTILIZATION OF A TRANSFORMATIONAL GRAMMAR

Peter S. Rosenbaum
and Fred Blair

INTERNATIONAL BUSINESS MACHINES CORPORATION
Thomas J. Watson Research Center
Post Office Box 218
Yorktown Heights, New York 10598

Contract No. AF 19(628)-5127

Project 4641

Task 464102

FINAL REPORT October 1966

Period Covered: July 1965 through September 1966

Prepared
for

AIR FORCE CAMBRIDGE RESEARCH LABORATORIES
OFFICE OF AEROSPACE RESEARCH
UNITED STATES AIR FORCE
BEDFORD, MASSACHUSETTS 01730

Abstract

This report summarizes research carried out at the Thomas J. Watson Research Center in three areas of computational linguistics. These are 1) the design and development of a transformational grammar for a subset of grammatical sentences in English, 2) the implementation of this grammar in terms of a sentence synthesizing program written in LISP 1.5, and 3) the use of sentence synthesizing programs for transformational grammars generally.

The transformational grammar described provides a semantically interpretable deep structure and a phonologically interpretable surface structure for a set of English sentences. Surface structures are derived from deep structures by transformational rules. The sentence types characterized by the grammar include noun phrase complementation, verb phrase complementation, relative clauses, two types of question sentences, indirect object and prepositional phrase constructions, passives, aspectual constructions, and certain types of negation phenomena.

The sentence synthesizing program provides a facility for generating deep structures and surface structures. Further, it makes use of prototype fast-fail procedures which, in many cases, obviate either completely or partially the so-called proper analysis test for the applicability of transformations.

Observations on the use of the sentence synthesizing program include 1) an analysis of the results obtained in using the program as a device for evaluating the descriptive adequacy of the grammar and 2) a discussion of the methodological limitations imposed upon the use of the program by factors inherent in the linguistic subject matter.

LIST OF CONTRIBUTORS

F. Blair

D. Lieberman

D. Lochak

P. Postal

P. Rosenbaum

1.0 INTRODUCTION

This paper is a report on basic research activities directed toward the specification, development, and computer utilization of grammatical descriptions consistent with the advances in linguistic theory usually subsumed under the rubric of transformational linguistics. Our primary concerns have been 1) the specification of a descriptively adequate transformational grammar for English, and 2) the development of prototype computational procedures for testing the descriptive adequacy of transformational grammars of advanced design. In addition to a discussion of these topics, this report gives brief attention to our experiences in using the prototype sentence synthesizing program (SSP) and to program design considerations arising from our experiences.

2.0 THE IBM ENGLISH GRAMMAR I[1]

2.1 General Properties

In its linguistic essentials, the IBM English Grammar I (Grammar I) conforms to the most important of the recent theoretical discoveries in transformational linguistics.[2] In particular, sentences in English characterized by Grammar I are assigned two levels of representation; they are assigned a deep structure and a surface structure. Deep structures, generated by context-free rewriting rules, determine the semantic interpretation of sentences. Deep structures are mapped into surface structures by transformational rules. Thus, all surface structures derived from a common deep structure through the application of transformational rules are

synonymous necessarily. Deep structures, surface structures, and intermediate transformationally derived structures are formally represented by labelled bracketings known as P-markers.[3]

Deep structures are composed of categorial symbols such as S, NP, VP, (sentence, noun phrase, verb phrase) and lexical items consisting of a phonological distinctive feature matrix (abbreviated in Grammar I) and a syntactic feature vector specifying various inherent, distributional, and rule determining properties of particular lexical entries. (A sample deep structure is provided in Appendix I.) Categorial symbols are introduced by context-free rewriting rules (Cf. Appendix II for the rewriting rules of Grammar I) and lexical items are introduced into P-markers subject to various semi-transformational distributional constraints. (Cf. Appendix III for a sample lexicon.)

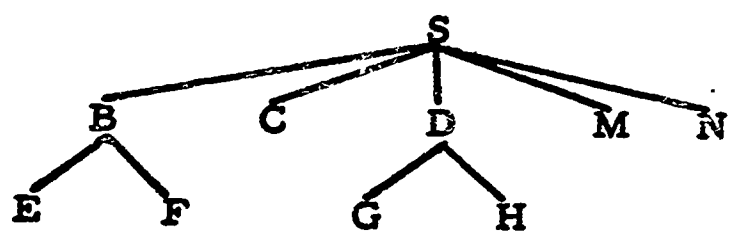
The domain of a particular transformational rule is provided in terms of conditions on P-markers. Any P-marker or set of P-markers meeting the conditions imposed by a particular rule falls under the domain of that rule. By way of clarification, consider the following hypothetical transformational rule.

(1)
B	+ C	D	X
1		2	3====>
1		∅	3

The numbered sequence of elements comprising the first two lines of the above rule (referred to as a structure index) defines the set of P-markers which may undergo the

transformational alteration stated in the third line of the rule. The structure index can be interpreted as asserting that any terminal string (last line of a P-marker) falls under the domain of the transformation (1) just in case it can be completely segmented into three consecutive substrings such that the first is a (member of the constituent or category sequence) B + C, the second is a D, and the third is anything at all. The diagram (2) contains a P-marker which falls under the domain of the transformation (1).

(2)



The terminal string of (2), i.e., E - F - C - G - H - M - N, can be segmented in such a way that the conditions stipulated by the structure index are met. Transformational rules stated in this fashion have the power of variable reference since each structure index characterizes a variety of P-markers. For example, the transformation (1) would be defined on the P-marker (2) regardless of the constituency dominated by B. If the phrase structure component from which this P-marker was constructed contains the rule B----> E + L + S, then an infinite number of P-markers are provided (since S, the sentence node, is recursive), all of which will fall under the domain of the transformation (1). Conditions imposed by transformational rules on P-markers include conditions on syntactic feature composition as well as on phrase structure. For instance, for the transformation (3), a P-marker falls under

its domain just in case it contains some constituent L which dominates a syntactic feature vector containing the feature ⟨+human⟩.

(3)

X	$[\langle +\text{human} \rangle]_L$	Y
1	2	3====>
1	∅	3

Transformational rules often involve two special types of restrictions. The first of these is dominance where some subtree in a P-marker must either have a certain analysis or must dominate some particular subtree. The second type of restriction is identity where a subtree must be identical or not identical to some other subtree.

A transformational rule specifies a finite sequence of formal operations called elementary transformations. The elementary transformations employed in Grammar I are as follows:

1. Substitution, where one subtree is substituted for another subtree.

2. Deletion, where a subtree is deleted.

3. Sister Adjunction, where a subtree is introduced under the immediate domination of some constituent which immediately dominates at least one other constituent.

4. Daughter Adjunction,[4] where some subtree is introduced under the immediate domination of some constituent which does not dominate any other constituent.

Transformational rules are ordered and are either cyclic[5] or post-cyclic. Cyclic rules apply to a lowest S in a deep structure where a lowest S is defined as any sequence of

terminal symbols 1) bounded left and right by sentence boundaries (#), 2) analyzable as an S, and 3) not containing any sentence boundaries except for those mentioned above. The final cyclic transformation deletes the sentence boundaries and, if the lowest S condition is still met, that is, if the sentence to which the cyclic rules were applying is an embedded sentence, the set of cyclic transformations reapplies. Otherwise, the set of post-cyclic transformations applies to the highest S, namely, one not dominated by S. The full set of cyclic and post-cyclic transformations of Grammar I is given in Appendix IV.

2.2 IBM English I

IBM English I (English I) is the subset of English sentences generated by Grammar I. Although the physical dimensions of Grammar I's rewriting rules are small (Cf. Appendix II), the system is relatively powerful. This power stems from the recursiveness of the initial symbol, S, under the domination of the verb phrase (VP), the noun phrase (NP), and the determiner (DET). The expansions of VP are of particular interest.

(4) VP----> V S

(5) VP----> V NP S

These expansions provide a deep structure characterization for the syntactic phenomenon of intransitive and transitive verb phrase complementation respectively. Transformed, these expansions give rise to the surface structures of sentences like (6) and (7).

(6) John condescended to play ball.

(7) John tempted Bill to play ball

Of equal importance are the two types of noun phrase complementation which arise through the expansions of NP given in (8) and (9).

(8) NP----> DET N S

(9) NP----> N S

These expansions yield a wide range of sentences including the following:[6]

(10) a. the fact that John came late worries me

b. it appears that John is honest

c. John appears to be honest

d. we stopped worrying

e. I dislike being here

f. she believes it to be true that life is good

The phenomenon of noun phrase complementation is extremely productive since NP's appear in diverse positions in deep structure; noun phrase complementation arises in all distributions.

In addition to the complex sentence formations described above, Grammar I characterizes such simple sentence phenomena as question formation (both the so-called "yes-no" questions and the "wh" questions), aspect, passive formation, certain types of negation phenomena, and, at the transformational level,[7] certain indirect object and prepositional phrase constructions. A set of sentences contained in English I which illustrates the major generative properties of Grammar I is

provided in Appendix V.

2.3 Deficiencies of Grammar I

Formal grammars invariably suffer from at least three types of deficiency: incompleteness, incorrectness, and theoretical slack. The general nature of these deficiencies is illustrated below with respect to Grammar I.

2.3.1 Incompleteness

It is a truism that no grammar constructed in the foreseeable future will generate all and only the sentences of an arbitrary natural language. There are a number of reasons for this, but acknowledgement simply of the immensity of natural language suffices. Since there is no reason a priori to assume that the completeness of a grammar is either necessary or sufficient for the computational utilization of formal grammars of portions of natural languages, it is important to recognize the bases of grammatical incompleteness. In so doing, it becomes clear that certain types of incompleteness are more susceptible to remedy than are others.

First, a grammar may intentionally omit treatment of a particular topic. For example, Grammar I does not deal with any form of conjunction. The reason for this is that theoretical support for the description of conjunction has been lacking. Only recently have theoretical developments indicated that a descriptive study of this phenomenon might become profitable.[8] It is currently expected that Grammar III will contain the results of such research.

Second, a grammar may be incomplete because of

simple oversight. To draw again on Grammar I, the formulation of the passive transformation in this grammar does not allow for the generation of passive sentences containing aspectual morphemes.[9] That is, Grammar I generates "John is teased by silly girls", but not "John is being teased by silly girls." Needless to say, such oversights are easily remedied.

2.3.2 Incorrectness

A grammar makes claims about the structure and derivation of well-formed sentences. Often, such claims turn out to be incorrect. Incorrectness may stem from a number of causes.

First, a grammar may be incorrect because it is incomplete. For example, in generating the "non-exceptional" sentence "John didn't want to behave himself", Grammar I provides a mechanism which incorrectly predicts the grammaticality of "*John said to behave himself." The problem here is that Grammar I lacks a mechanism for dealing with exceptions. The grammar is not basically wrong; it is simply incomplete, and this incompleteness leads to incorrectness.

Second, and far more embarrassing, a grammar may be linguistically incorrect thereby providing an incorrect analysis for generated sentences. For example, by allowing both progressive and regressive deletion in relative clause formation, Grammar I makes the incorrect claim that sentences (11) and (12) are synonymous.

(11) I discovered the mountain which John is admiring

(12) I discovered which mountain John is admiring

2.3.3 Theoretical Slack

Theoretical slack means simply that the linguistic theory in terms of which a grammar is constructed is insufficiently specific to allow a choice among competing descriptions of the same phenomenon. Grammar I, for example, is based upon a version of linguistic theory which allows various properties of nouns, verbs, and adjectives to be described in the deep structure either in terms of constituent structure or in terms of syntactic features subcategorizing nouns, verbs, and adjectives. To take a specific case, number on nouns may be viewed either as a constituent under the domination of NP, as specified in the rewriting rules of (13), or as the syntactic feature ⟨singular⟩ positively or negatively specified (where ⟨+singular⟩ indicates a singular noun and ⟨-singular⟩ indicates a plural noun).

(13) NP ----> N NU (S)

$$NU \text{ ----> } \left\{\begin{matrix}\text{Singular}\\ \text{Plural}\end{matrix}\right\}$$

Grammar I adopts the feature analysis of number, but this analysis is not theoretically determined. From the point of view of the linguistic theory, this analysis is arbitrarily selected.[10]

2.4 Directions in Ongoing Grammatical Research

An attempt is currently being made to remedy many of the deficiencies in Grammar I. Grammar II will correct the oversights of Grammar I and will, furthermore, treat such syntactic phenomena as pronominalization, reflexivization, genitivization, time and place adverbials, and verb-

preposition restrictions. In addition, corrections in the analyses are being made, e.g., for progressive and regressive deletion in relative clause formation, and a facility for the treatment of exceptions is being added.[11]

Finally, and by far the most important, Grammar II will be consistent with a revised linguistic theory which calls for the introduction of grammatical items such as articles, affixes, prepositions, aspectual morphemes, complementizers, and the like on the basis of the generated syntactic feature composition of the formatives N (noun) and VB (verbs and adjectives). The introduction of such items involves the process of transformational segmentalization.[12] It is generally significant that such a version of linguistic theory provides a near approximation to universal deep structure. (It seems clear, in any case, that potential computer applications involving the analysis of English sentences will be greatly facilitated by the removal of material entirely idiosyncratic to English from the deep structure.) The phrase structure rewriting rules for Grammar II are given in Appendix VI.

3.0 THE IMPLEMENTATION OF A SENTENCE SYNTHESIZER

3.1 General Description

A computer program for synthesizing sentences on the basis of Grammar I was implemented as several functions in LISP 1.5. The top level function, Deriv[], requires an expanded grammar[13] and a set of derivation control cards.

The latter consists of 1) a rewriting subrule[14] specification for the generation of a particular deep structure (syntactic feature vectors being also introduced by such rules in an ad hoc manner) and 2) a specification (possibly null) of the optional transformational rules whose applicability is to be tested. Deriv[], in generating deep structures, sequentially tests each rule in the expanded grammar for applicability. Establishing the applicability of a rule, Deriv[] determines whether a subrule of this main rule is specified on a derivation control card. Upon success, the subrule so specified is applied. For deep structure generation, rule application involves the replacement of the applicable left-hand symbol found in the current terminal string of symbols with the right-hand constituency of the specified subrule. At the same time, a P-marker is constructed which reflects the current state of the deep structure. Upon successful subrule application, a new derivation control card is read and the applicability of the same rule is retested. When a rule is not applicable, Deriv[] proceeds to test the applicability of the following rule. Deriv[] provides a printed record of the subrule applied, the current P-marker, and the current terminal string. Deriv[] applies in cycles to each unexpanded S node. When no more unexpanded S's remain, Deriv[] calls Derivtrans[], which tests the applicability of all obligatory transformations and the specified optional transformations, and converts the deep structure into a surface structure by applying transformations to current P-markers falling under their domain.

Derivtrans[] proceeds sequentially through the transformational rules of the expanded grammar testing each for

applicability. In the event that a proper analysis is obtained (i.e., where the pattern specified in the structural index of a transformation is found in the current P-marker), Derivtrans[] calls Dotran, which applies the transformation to the P-marker under either one of the following two conditions: First, the transformation is obligatory. Second, the transformation is an optional transformation specified on the current derivation control card. Derivtrans[] tests the set of cyclic transformations for each deepest S. When no more deepest S's remain, this function tests the applicability of the set of post-cyclic transformations.

3.2 The Pattern-Matching Function: Syntax and Semantics

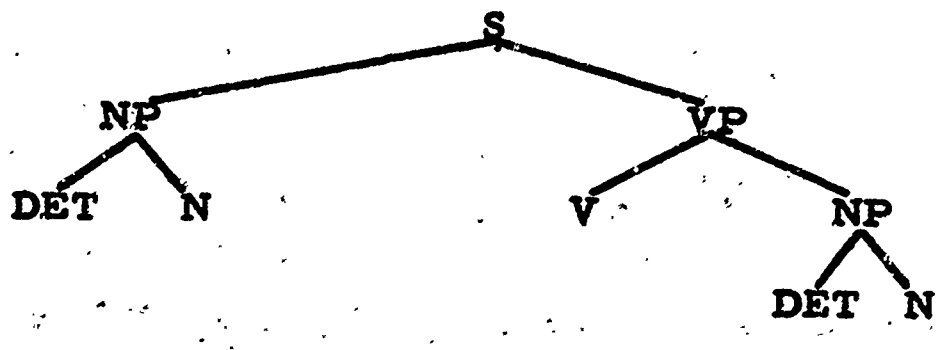
The heart of Derivtrans[] is the procedure for obtaining a proper analysis in a current P-marker. The function P-a allows the specification of the patterns in the structure indices of transformations and identifies these patterns in P-markers. This function has the following form:[15]

(14) P-a[⟨node:list⟩; ⟨pattern⟩; ⟨name#⟩; ⟨n:pairs⟩]

3.2.1 Syntax and the Modeling of Transformations

Subtrees in P-markers are represented as LISP s-expressions. Consider for example, the P-marker below.

(15)



This P-marker is represented as the following s-expression.

(16) (S(NP(DET)(N))(VP(V)(NP(DET)(N))))

The first argument to P-a is a list of sister nodes, ⟨node:list⟩ := (⟨node⟩*). A node is a constituent and all the constituents which it dominates in a P-marker, ⟨node⟩ := (⟨constituent⟩⟨node⟩*) and ⟨constituent⟩ := ⟨atom⟩. For instance, the node NP in the P-marker (15) is represented as follows:

(17) (NP(DET)(N))

A representative list of sister nodes supplying a first argument to P-a might be the following:

(18) ((NP(DET)(N))(VP(V)(NP(DET)(N))))

The second argument to P-a, ⟨pattern⟩, is a list consisting of an optional left-anchor[16] followed by an indefinite number of pattern elements to be matched, ⟨pattern⟩ := ({$0 | ∅} ⟨pattern:element⟩*).

The third argument to P-a, ⟨names⟩, is a list of names, ⟨names⟩ := (⟨name⟩*), where ⟨name⟩ := ⟨identifier⟩ | ⟨number⟩. For each member of the pattern list there is a corresponding name on the names list. Names are used to index matched nodes for subsequent reference. The convention in this system is to supply positive integers beginning with 1 as names.

The fourth argument to P-a, ⟨m:pairs⟩, is a list of names paired with the matched nodes indexed by these names, ⟨m:pairs⟩ := ((⟨name⟩{⟨node⟩ | ⟨node:list⟩ | ⟨m:pairs⟩})*).

Pattern elements are of especial interest because they characterize the devices available to the linguist in modeling structure indices.

(19) ⟨pattern:element⟩ := {()|NIL}|⟨literal⟩
|⟨alternation⟩|$|⟨special:form⟩|⟨p:form⟩

The pattern element {()|NIL} is the empty pattern element and is used to model optional non-empty patterns in a structure index. The pattern element ⟨literal⟩ specifies the content of a node to be matched. This element thus models the constituents in a structure index. Sets of alternative pattern elements are modeled as an alternation, ⟨alternation⟩ := (=OR ⟨pattern:element⟩*). The pattern element $ models the structural variables often found in transformational rules. The pattern element ⟨special:form⟩, where ⟨special:form⟩ := ⟨conjunction⟩|⟨matching:function⟩|⟨dominance:constraint⟩, is used by the linguist to model conjunctions within an alternation, to introduce special matching conditions such as identity and constraints on syntactic features, and to test constraints on dominance of elements within structure indices. Finally, ⟨p:form⟩, which was not employed in Grammar I as a pattern element, is of use in matching unusual tokens such as +, $, etc.

3.2.2 The Pattern-Matching Algorithm

P-a attempts to find a proper analysis for a pattern specification in various ways depending upon the character of the pattern elements in the pattern specification. Beginning with some candidate node, P-a tries to match the first

pattern element with that node. On success, P-a tries to match the next pattern element with the next contiguous node[17] in the P-marker. If all pattern elements are matched successfully, the function returns as its value a list of matched nodes and names (m:pairs) paired with the then current candidate node (which may be null). If a pattern element fails to match a candidate node, the left-hand daughter of the current candidate node becomes the candidate node and the function is reapplied. If, under these circumstances, no left-hand daughter exists, then the function has not found a proper analysis and returns NIL.

Various pattern elements affect this procedure in various ways. The null pattern element causes the current name to be paired with the empty fragment (empty list) and added to the mpairs list. The literal pattern element matches only nodes whose content is identical to itself. The alternation pattern element causes a match just in case one of the successively examined patterns causes a match when substituted for the alternation as the current pattern element.

The element $ matches a <u>fragment</u> (an indefinitely long list of nodes) in the following manner. If the current name does not appear on the mpairs list, the name is paired on the mpairs list with the empty fragment.

1. If a proper analysis can then be found for the rest of the current pattern (the pattern elements remaining to the right of the $ element) beginning with the current candidate node, this proper analysis is given as the resulting mpairs list (i.e., P-a [nodes; cdr[pattern]; cdr[names]; m:pairs].

2. Otherwise, the current candidate node is appended to the current $ fragment to yield mpairs' and the next contiguous node is taken as a current candidate'. If, on applying P-a recursively, a proper analysis is found for the current pattern (with the current $ as first element) beginning with the current candidate', then that proper analysis is given as the resulting mpairs list, (i.e., P-a [nxtcontiguous[node:list]; pattern; names; m:pairs']). In this way the $ is extended over next contiguous nodes.

3. On the failure of both 1 and 2, the left-most daughter of the current candidate node examined in 1 (not the current candidate') is taken as the new current candidate node and step 1 is reinitiated. If no such left-hand daughter exists, then the pattern match fails and the value of P-a is NIL.

3.2.3 Some Further Syntactic Considerations

Within P-a, pforms, where ⟨p:form⟩ := (QUOTE ⟨s-expression⟩) | ⟨back:reference⟩ | ⟨subscript:reference⟩ ! (* ⟨form⟩) | (*K ⟨arg⟩*), are used in a variety of ways. First, inasmuch as certain possible node contents are not allowable as literals, e.g., $, the pattern element pform provides a useful facility. A pform is first evaluated and the resulting value is taken as a literal. The value of the pform (QUOTE ⟨s-expression⟩) is simply the associated s-expression. If a node previously matched is required, e.g., as is often the case in special forms, either a back reference, where ⟨back:reference⟩ := ⟨name⟩, or a subscript reference, ⟨subscript:reference⟩ := (S/name*), must be employed. The pform (* ⟨form⟩) causes the form to be LISP evaluated,

e.g., (* DOLLAR) yields $. Finally, argument functions yield as values the values of the LISP function, ⟨function⟩, applied to the evaluated arguments. This device makes available the full power of the LISP language for the generation of arguments.

The pattern element ⟨special:form⟩ is either a conjunction, a matching function, or a dominance constraint. The value of a successful special form is an mpairs list rather than a node or list of nodes. Thus, mpairs lists may appear on mpairs lists and elements of such "contained" mpairs lists must be referred to by subscript references.

Conjunction, where ⟨conjunction⟩ := (=AND ⟨pattern: element⟩*), treats a pattern as a pattern element, as in an alternation where an alternate is a sequence of nodes.

Matching functions[18] provide an escape mechanism from the matching algorithm. This device allows the linguist to state a very wide range of special conditions on P-markers not otherwise specifiable. Consider the following matching function form:

$$(\text{FN } \langle\text{function}\rangle \; arg_1 \; arg_2 \ldots arg_n)$$

⟨function⟩ is a LISP defined function of n+1 arguments where the first argument, the current candidate node, is implicitly supplied by the pattern element interpreter. Matching functions obey the following conventions:

1. If the matching function fails, it returns the value NIL.
2. On success, the matching function returns a non-null value (usually an mpairs list).

P-a does not descend into syntactic feature vectors[19] (complex symbols). Where conditions on transformations involve complex symbols for Grammar I, they are implemented in terms of the following special matching functions.

1. (FN FEATURE arg_1 arg_2), where arg_1 evaluates to a terminal node (e.g., N, V) dominating a syntactic feature vector and where arg_2 evaluates to a list of features which must be contained in the feature vector. Consider, for example, the "WH pronoun" transformation.

(20) X DEF $\left[\begin{matrix}\langle +PRO \rangle \\ \langle +Sg \rangle\end{matrix}\right]_N$ WH + Y

1 2 3 4====>

1 2 Ø 4

The structure index for this transformation is modeled as follows:

(21) ($ DEF N WH (FN FEATURE 3 (QUOTE(/Sg /PRO))))

2. (FN ALPHA arg_1 arg_2) is identical to the above FN except that the coefficient of the syntactic feature(s) of arg_2 is ignored. This matching function is useful for modeling structure indices employing the variable coefficient, e.g., ⟨α C⟩, in the Complementizer Placement Transformation.

The syntax of a dominance constraint on P-markers is as follows:

(22) ⟨dominance:constraint⟩ := (*** ⟨arg⟩⟨pattern⟩)

The first argument specifies a previously matched node. The second argument is the pattern to be matched in the subtree dominated by the node specified by the first argument. The

value of the pattern element is either an mpairs list, on success, or NIL. The structure index

$$\begin{array}{cccc} A & B & [C & D]_E \\ 1 & 2 & 3 & 4 \end{array}$$

is modeled as follows:

(23) (A B E(*** 3(QUOTE ($0 C D))))

3.3 The Pattern Transformation

P-markers are transformed by the LISP function Dotran[u; tr; ct] where u is an mpairs list produced by the successful application of P-a, ct is the segment of the P-marker falling under the domain of the transformation identified by its highest node and tr is a list specifying the transformation.

⟨tr⟩ := (⟨replacement⟩*)

⟨replacement⟩ := (⟨arg⟩⟨arg⟩*)

The first argument of a replacement is either a node produced as the value of a back reference or subscript reference, or else is an argument function. In the first case, the remaining arguments specify the nodes which are to replace the node referenced by the first argument. In the second case, an argument function may be employed to modify ct in some other manner than by sister adjunction, substitution, or deletion.[20]

Consider how the relative placement transformation of Grammar I is modeled.

(24) # X ART S N Y #
1 2 3 4 5 6 7====>
1 2 3 ∅ 5+4 6 7

(25) ((// $ ART S N)(4 0)(5 5 4))

Observe that deletion involves replacement of a node by ∅.

4.0 THE LINGUISTIC BASES OF HIGH SPEED SENTENCE GENERATION

Perusal of the transformational rules in Grammar I will reveal that structural variables appear quite often as the first and last pattern elements of a structure index. These variables capture the linguistically significant generalization that a particular transformational process is completely independent of the constituent material falling under the domain of such variables in a particular P-marker. This generalization is not reflected in P-a, which must search through a P-marker for an occurrence of a pattern element even though much of the constituent structure which is traversed in this process may fall under the domain of the structural variable. Needless to say, such irrelevant searching is time-consuming.

To a certain degree, the situation is salvageable inasmuch as the imposition of an arbitrary depth-of-embedding constraint on the rewriting rules renders it possible in principle to state transformations in terms of literals. In other words, it becomes possible to specify all of the constituent structure material ordinarily subsumed under the variable. At the very least, such a procedure is lingustically distasteful. Worse, however, are the consequences if the grammar

under study is even minimally complex, e.g., handles complementation. Under such circumstances, a vast number of environments would have to be specified for the most trivial of rules. Consider, for example, the number of "left" environments Grammar I would be forced to provide, even if the depth of embedding were arbitrarily limited, for the trivial post-cyclic rule which assigns an affix to plural nouns.

There is little question that the routine for obtaining a proper analysis which we have devised is not optimal and that greater processing efficiency can be expected in future versions of SSP.[21] Our efforts to solve the problem of the structural variable have been based on the assumption that the fastest routine for obtaining a proper analysis is no routine at all. Less glibly, we have addressed two questions. If a particular P-marker does not fall under the domain of a transformation, is it possible 1) to prevent entry into the proper analysis routine in the first place, and 2) if not, to achieve a rapid termination of this routine? The answer to both of these questions is yes. We refer to techniques accomplishing these tasks as fast-fail procedures.

4.1 How Not to Proper Analyze

4.1.1 Node Listing

Node listing is a simple procedure for obviating a proper analysis which takes advantage of the fact that a necessary (though not sufficient) condition for obtaining a proper analysis is that the P-marker in question must contain every literal contained in the structure index. Before the application of P-a, a test is made on the P-marker to

determine whether it contains a single instance of literals supplied appropriately with each transformation. This procedure turned out to be of minimal value since most P-markers contain most constituents.

4.1.2 Sister Listing

Of far greater value is a similar procedure which tests the P-marker for the sisterhood of two or more constituents mentioned as literals in a structure index. For example, the Relative Placement Transformation mentions the sisters ART and S. Node listing would be of little value since virtually all P-markers contain ART and all P-markers do, in fact, contain S. The sister listing procedure checks the current P-marker to determine whether ART and S are contained somewhere as sisters. Such a test will be successful just in case the P-marker contains a relative clause construction. Otherwise, it will fail and Derivtrans[] will immediately proceed to the next transformation.

4.2 The Fast NIL for P-a

4.2.1 Terminalizing

Careful study of Grammar I's transformational rules shows that the literals mentioned in the structure indices of these rules are quite often terminal symbols, i.e., symbols which uniquely appear in the terminal strings of P-markers. This fact suggests the possibility of reducing considerably the work which must be performed by P-a in the event that entry into this routine is unavoidable. More specifically, observe that when the structure index of a transformation contains only terminal symbols, a search by P-a of the

non-terminal constituents of the current P-marker is unnecessary. Such redundant searching can be obviated by requiring P-a to apply to a temporary P-marker, P-marker', which contains the highest node, S, and the terminal nodes of the original P-marker, but none of its intermediate nodes. This "tree pruning" procedure, which turns out to be extremely valuable and will be even more so for Grammar II, is called terminalizing.

4.2.2 Node Weighting

When entry into P-a is unavoidable and when the structure index of the current transformation contains non-terminal as well as terminal symbols, terminalizing is impossible. The only fast-fail procedure currently operating in SSP for reducing the amount of work done by P-a in this instance is node weighting. This procedure takes advantage of the fact that the number of eligible proper analysis nodes in the current P-marker must always be equal to or greater than the number of literal pattern elements remaining to be matched. Under this procedure each pattern element is assigned a weight reflecting a minimum node requirement for the current P-marker. Similarly, nodes in the current P-marker are assigned weights in accordance with their position in the P-marker for possible proper analysis. At such time as the weight of the pattern element exceeds the weight of the node being examined, P-a terminates.

Grammar II, which has considerably different properties than Grammar I, presents several possibilities for effective fast-fail procedures. We plan to report on these at a

later date.

5.0 THE USE OF THE SENTENCE SYNTHESIZING PROGRAM

5.1 Research Goals and Their Consequences

Our conclusions concerning the use of SSP in the development of English Grammar I only have meaning in terms of our research goals. Our central research goal in computational linguistics is to install in an electronic computer a knowledge of natural language (English in the present case) which reflects the English speaker's ability to relate the form of a sentence to its meaning. Inasmuch as the immensity of English[22] renders impossible a full reconstruction of this knowledge in the form of a transformational grammar, our less ambitious goal is to construct a transformational grammar for a subset of English sentences which is both useful and learnable. The usefulness of such a subset is completely a function of its expressive power.[23] Clearly, the existence of such a subset is meaningless if this subset (which, in all likelihood, will be infinite) is not learnable.[24]

This goal establishes three requirements for computational linguistic research. The first is to pursue topics in linguistic theory since it is generally true that the more advanced the linguistic theory, the more general are the grammars whose form is a consequence of this theory. The second is 1) to develop precisely specified grammars which are descriptively correct with respect to the assignment of deep structures to surface structures and 2) to study computational

procedures for implementing these grammars. The third is to study the useability and learnability of the subsets of natural language generated by these grammars. Our views on the use of SSP are couched in terms of these considerations.

5.2 General Conclusions

5.2.1 Sentence Synthesis and Linguistic Theory

The relation obtaining between linguistic theory and a sentence synthesizing program is one of specification, in that the linguistic theory specifies the form of the grammar which is implemented by the program. This fact is perhaps disconcerting since, inasmuch as constancy over time has not yet become a property of transformational linguistic theory, it suggests the necessity of constant revision for the SSP. In a weaker moment, one may fancy an SSP which allows a linguist to make arbitrary changes in his theoretical formulation, but recognition of the utter nonexistence of discovery procedures for linguistic theories, i.e., the complete lack of any basis for projecting future developments, persuades us that such a device is an impossibility.

It is always possible to take the linguistically and, in the long run, computationally unfortunate option of theoretical compromise, thus constructing a theoretically antiquated grammar for the sake of computation. If, however, the goal is to develop a grammar that is theoretically sound, it is then our conclusion that the major responsibility for developing and maintaining an adequate SSP belongs to the linguist and not to the programmer. There is no computational procedure for resolving difficulties inherent in linguistic methodology.

5.2.2 Sentence Synthesis and the Construction of Grammars

Eschewing linguistic description for its own sake, there are two reasons for constructing transformational grammars. The first of these is to confirm or disconfirm theoretical hypotheses. Since generative rigor involving the construction of anything more than a grammatical sketch has never been a necessary condition for the wholehearted acceptance or rejection of such hypotheses, the usefulness of a sentence synthesizing program in the construction of such a grammar segment is extremely doubtful. On the other hand, a necessary condition for computer applications based upon transformational grammars is the generative correctness of large descriptive grammars. In this respect, a sentence synthesizing program which tests the rules of the grammar is a must, as anyone who has studied the mind-warping properties of complex transformational grammars will readily appreciate.

It is our observation that the uses of a sentence synthesizing program are most reasonably determined not so much by the grammar but by the applications requiring the grammar. Specifically, the facility which is critical to computer applications involving the speaker's knowledge of his language is the transformational reconstruction of the relation obtaining between the form of a sentence and its meaning, between surface structure and deep structure. This relation is precisely specified by the transformational rules of the grammar. The implication here is that there is simply no good reason to provide a computer implementation of those

facilities of a transformational grammar which do not have direct bearing on the relation between meaning and form. This assertion is reflected in our inability to find any use for that extensive component of SSP which allows the expansion of the rewriting rules of Grammar I and the automatic generation of deep structures. In the testing of transformational rules, a generative phrase structure component is just so much baggage, and we normally introduce deep structures for Derivtrans[] by hand. We are chagrined to have spent so much time developing a subroutine which turns out to have neither linguistic nor applicational significance (save perhaps for an audio-visual aid in Linguistics I). Much as a result of this unfortunate experience, we have not implemented a blocking facility for terminating a derivation where two constituents are required to be identical, but, in a particular P-marker, are not. An adequate transformational grammar must no doubt provide such a blocking facility, but the necessity for computer implementation of this facility is doubtful since, on the theoretical hand, the theoretical claim made by any such blocking device could be trivially evaluated manually and since, on the applicational hand, deep structures exemplifying such cases of non-identity would never arise. A blocking mechanism in the sentence synthesizing program itself would provide nothing more than a laboratory curiosity.

SSP is of great value in answering the following question: Given deep structure D, does the set of transformational rules generate surface structure S? Most dramatic are those cases where the transformations generate an incorrect sentence S'. An illustrative example concerns the transformation

which assigns the affix "s" to plural nouns in Grammar I.

X	$[\langle$-Sg$\rangle]_N$	Y
1	2	3====>
1	2 + s	3

This rule asserts: If a noun carries the syntactic feature ⟨-singular⟩ then add an "s" under the domination of this noun regardless of all other aspects of the environment. Since P-markers are reanalyzed after the application of a transformation[25] for the reapplicability of the same transformation, the above transformation will and did reapply indefinitely producing strings like "noun s s s s s ...". This consequence resulted from the fact that the morpheme "s" was subsumed under the variable Y on each reanalysis by P-a.

Such cases as the above are fairly uncommon. More often than not, the output of SSP, when it turns up an error, is completely undramatic since when something is wrong the transformation most commonly does not apply at all and the output is some intermediate structure. This circumstance was especially unnerving while the SSP was being debugged since it was not always easy to determine whether a rule failed to apply because it would not or because it could not.

Summarizing, SSP has turned up errors of the following sort:

1. Transformational rule applies incorrectly.
2. Transformational rule fails to apply.

Reasons:

a. Traffic information incorrect, e.g., obligatory rule marked optional.

b. Transformational rule stated incorrectly.

c. P-marker stated incorrectly.

Finally, mention should be made of the fact that SSP operates in what might be called an automatic mode in transformational generation whereby all obligatory transformations are tested against a P-marker without specification on a derivation control card. Only optional rules are so specified. It is often the case, however, that linguistic attention is focussed on particular transformations and the effects of others are beside the point. For such cases, which arise very often, we are developing a manual mode of operation for the sentence synthesizing program which will implement Grammar II. In this manual mode, the linguist will specify all transformations that he wishes to be tested against a particular P-marker. The manual mode provides the linguist with less information than the automatic mode, but such omitted information is often superfluous and can be profitably sacrificed for speed.

FOOTNOTES

[1]English Grammar I was formally presented in P. S. Rosenbaum and D. Lochak, "The IBM Core Grammar of English," Specification and Utilization of a Transformational Grammar, Scientific Report No. 1, (IBM Corporation, Yorktown Heights, N.Y., 1966).

[2]Cf. N. Chomsky, Aspects of the Theory of Syntax (MIT Press, Cambridge, Mass., 1965).

[3]Cf. J. J. Katz and P. M. Postal, An Integrated Theory of Linguistic Descriptions, (MIT Press, Cambridge, Mass., 1964).

[4]The need for daughter adjunction in Grammar I is an artifact. This elementary transformation does not appear in the transformational rules of Grammar II.

[5]Empirical justification for the cyclic principle is provided in P. S. Rosenbaum, "The Empirical Basis of the Cyclic Principle," (forthcoming).

[6]For a detailed discussion of noun phrase and verb phrase complementation, cf. P. S. Rosenbaum, The Grammar of English Predicate Complement Constructions, MIT Doctoral Dissertation, (Cambridge, Mass., 1965).

[7]The actual introduction of prepositions into P-markers is not treated in Grammar I but is in Grammar II.

[8]Cf. G. Lakoff and S. Peters, "Phrase Conjunction and Symmetric Predicates," (forthcoming).

[9]This embarrassing fact was kindly pointed out to us by C. Valenti, C.F.X. Cf. his unpublished manuscript, "Suggested Adjustments in the IBM Core Grammar," NDEA Summer English Institute, The Ohio State University, 1966.

[10]For a grammar in which the constituent analysis is adopted, cf. "English Preprocessor: English Grammar--Rules

and Examples," English Preprocessor Manual, MITRE Corp., (May, 1965).

[11] The problem of exceptions is given its most complete treatment in G. Lakoff, On the Nature of Syntactic Regularity, Mathematical Linguistics and Automatic Translation, Report No. NSF-16, (The Computation Laboratory, Harvard University, Cambridge, Mass., 1965).

[12] P. Postal, "On So-called 'Pronouns' in English," (to appear in F. Dineen, ed., Monograph Series on Languages and Linguistics Number 19, Georgetown University Institute of Languages and Linguistics, Washington, D. C.).

[13] F. Blair, "Programming of the Grammar Tester," Specification and Utilization of a Transformational Grammar, Scientific Report No. 1, (IBM Corporation, Yorktown Heights, N. Y., 1966).

[14] Rosenbaum and Lochak, op. cit., p. 1.

[15] The syntax notation employed in this report is a modified Backus-Naur form. The symbols employed are interpreted as follows: := (syntactic definition), ⟨ ⟩ (metalinguistic variable delimiters), | (alternation specifier), {} (metalinguistic grouping brackets), * (indefinite number), ∅ (null).

[16] Cf. D. Bobrow, "METEOR: A LISP Interpreter for String Transformations," THE PROGRAMMING LANGUAGE LISP: Its Operation and Applications, Informational International, Inc. (1964). In the present case $0 requires the first pattern element (if either a literal or a pform) to find a match among the left-hand descendants of car[nodes]. In the absence of a $0, an initial $ is supplied automatically.

[17] The next contiguous node, N, of some node, N', is either the right-hand sister of N' or, if N' has no right-hand sister, the next contiguous node of the parent of N'.

[18] Cf. Bobrow, op. cit., pp. 178-179.

[19] Rosenbaum and Lochak, op. cit., pp. 9-17.

[20] Ibid., pp. 22-27.

[21] Illustrative of possible improvements is S. Kuno's proposal in "Polish Notational Representation of Phrase Markers and Matching of Structural Index with Polish String," unpublished manuscript.

[22] For example, cf. G. A. Miller, E. Galanter, and K. H. Pribram, Plans and the Structure of Behavior, (Henry Holt, New York, 1960), pp. 145-148.

[23] In other words, simply stated, does the subset of English allow the user to say what he wants to say, albeit not making full use of full glory of the English language?

[24] P. S. Rosenbaum, A. Baldwin, J. Samsky, "On the Useability and Learnability of a Transformationally Generated Subset of English," (forthcoming).

[25] This requirement applies only to Grammar I. The theory underlying Grammar II requires transformations to apply to all legitimate proper analyses "simultaneously" as it were. After the application of a transformation, the transformed P-marker is not reanalyzed for the purpose of applying the same transformation on the same cycle.

[26] For a description of the design and implementation of a computational aid for compiling a transformational lexicon, cf. D. Lieberman and D. Lochak, "Computer Support for Lexicon Development and Use," Specification and Utilization of a Transformational Grammar, Scientific Report No. 1, (IBM Corporation, Yorktown Heights, N. Y., 1966).

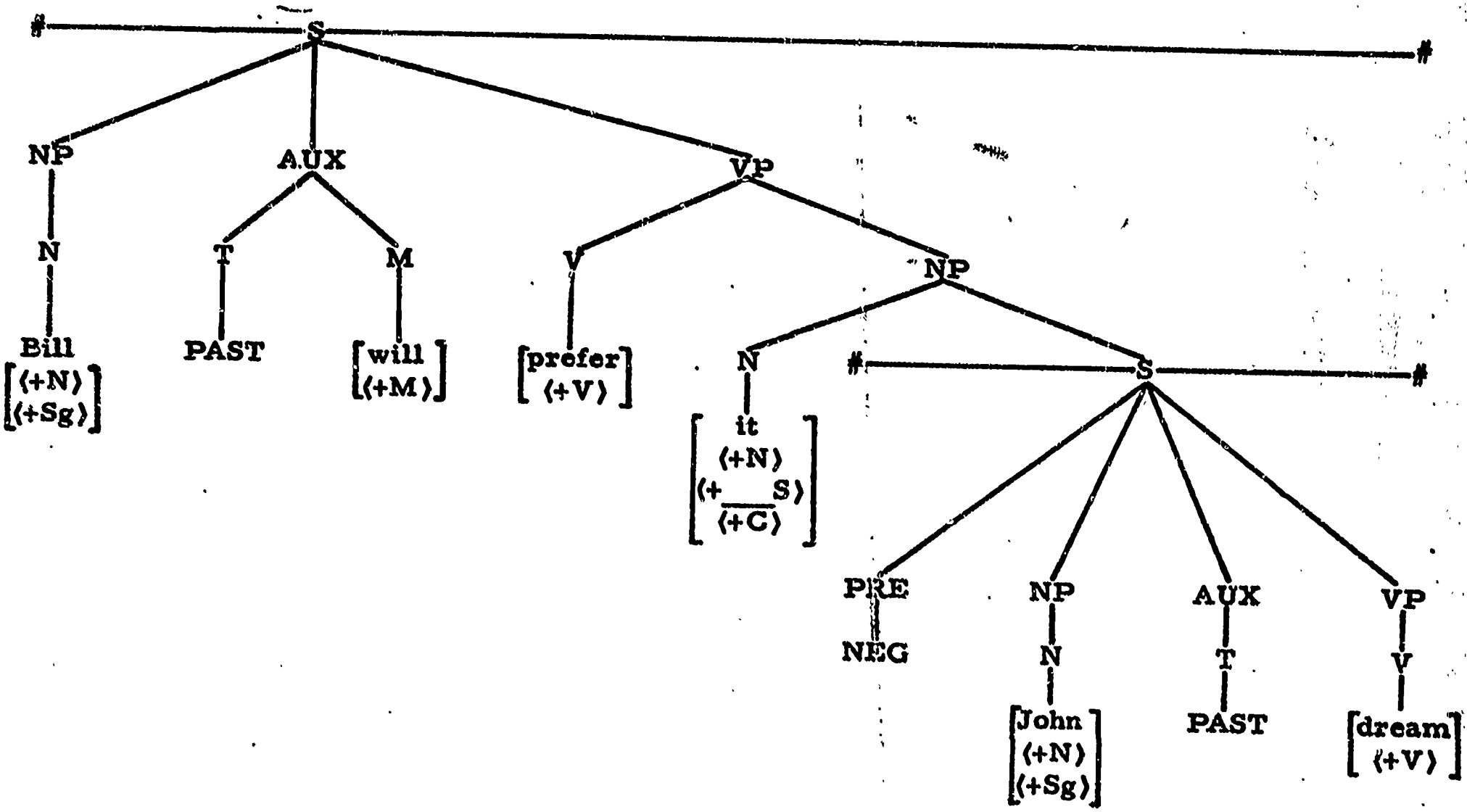
APPENDIX I

Sample Deep Structure

Actual string: Bill would prefer for John not to have dreamed

Derived string: Bill will ed prefer for John not to have dream en

Deep structure:



APPENDIX II

Phrase Structure Rewriting Rules for Grammar I

S ----> # (PRE) NP AUX VP #

PRE ----> (NEG) (Q)

AUX ----> T (M)

T ----> {PRES / PAST}

VP ----> (have en)(be ing) { V({NP / PP})({S / PP})(MAN) / be (ADJ) }

PP ----> PREP NP

MAN ----> PREP P

NP ----> (DET) N (S)

ART ----> (WH) {DEF / INDEF}

APPENDIX III

Sample Lexicon[26]

Syntactic Category

⟨+N⟩	boy	⟨+ADJ⟩	honest
⟨+V⟩	slay	⟨+M⟩	must

Strict Subcategorization

Verbs ⟨+V⟩		Nouns ⟨+N⟩	
⟨+__NP S⟩	tempt	⟨+DET__S⟩	fact
⟨+__NP⟩	disappoint	⟨+DET__⟩	teapot
⟨+__PP⟩	approve	⟨+__S⟩	it
⟨+__S⟩	condescend	⟨+__⟩	John
⟨+__⟩	elapse		

Inherent Subcategorization

Nouns ⟨+N⟩			
⟨+human⟩	boy	⟨+animate⟩	mongoose
⟨+abstract⟩	blame	⟨-animate⟩ ⟨-abstract⟩	table

Selectional Subcategorization

Verbs ⟨+V⟩	
⟨+__ ⟨+__S⟩⟩	suppose
⟨+ ⟨+__S⟩ S AUX__⟩	bother
⟨+__(DET) ⟨+human⟩ PREP ⟨+__S⟩⟩	remind

Adjectives ⟨+ADJ⟩	
⟨+ ⟨+__S⟩ S AUX__⟩	obvious

APPENDIX IV

Transformational Rules

I. CYCLIC RULES

1. CP 1 Complementizer Placement 1

#	X	$[(_{\alpha}C)]_N$	NP + {T, be, have, V}	Y	#	
1	2	3	4	5	6	6====>
1	2	3	$_{\alpha}C$ + 4	5	6	

---

2. CP 2 Complementizer Placement 2 OB

#	X	$[(+C)]_V$	(NP)	NP + {T, be, have, V}	Y	#	
1	2	3	4	5	6	7	7====>
1	2	3	4	+C + 5	6	7	

---

3. CP 3 Complementizer Placement 3 OB

#	X	N	$[NP + Y]_S$	Z	#	
1	2	3	4	5	6	7====>
1	2	3	that + 4	5	6	7

---

4. IE Identity Erasure OB

#	W	(NP)	X	$_{\alpha}C$	NP	Y	(NP)	Z	#	
1	2	3	4	5	6	7	8	9	10	10====>
1	2	3	4	5	Ø	7	8	9	10	

Condition: An NP_j is erased by an identical NP_i if and only if there is an S_n such that

(i) NP_j is dominated by S_n

(ii) NP_i neither dominates nor is dominated by S_n

(iii) for all NP_k neither dominating nor dominated by S_n, the distance between NP_j and NP_k is greater than the distance between NP_j and NP_i where distance between two nodes is defined in terms of the number of branches in the path connecting them.

---

5. IOI Indirect Object Inversion OP

#	X	V	$\left\{\begin{matrix}PP\\NP\end{matrix}\right\}$	to + NP	Y	#
1	2	3	4	5	6	7====>
1	2	3 + 5	4	Ø	6	7

---

6. TO To Deletion OB

#	X	V	to	NP	(PREP) + NP	Y	#
1	2	3	4	5	6	7	8====>
1	2	3	Ø	5	6	7	8

---

7. PASSIVE Passive OB

#	(PRE)	NP_1	AUX	V	(PREP)	NP_2	X	PREP	P	Y	#
1	2	3	4	5	6	7	8	9	10	11	12====>
1	2	7	4	be+en+5	6	Ø	8	9	3	11	12

Condition: $3 \neq 7$

---

8. EXTRA Extraposition OP

#	X	$[\langle + \_\_ S \rangle]_N$	S	Y	#
1	2	3	4	5	6====>
1	2	3	Ø	5	4 + 6

---

9. PROREP Pronoun Replacement OB

#	X	$[[\langle +\_\_S\rangle]_N]_{NP}$	(AUX(be en) V + (MAN))	αC	NP	Y	#
1	2	3	4	5	6	7	8====>
1	2	6	4	5	Ø	7	8

10. WHA WH-Attraction OB

#	U	ART	[NP	W	$\left\{\begin{matrix}\text{PREP} + [\text{WH X}]_{NP} \\ [\text{WH X}]_{NP}\end{matrix}\right\}$	Y]$_S$	Z	#
1	2	3	4	5	6	7	8	9====>
1	2	3	6 + 4	5	Ø	7	8	9

11. RELPLACE Relative Placement OB

#	X	ART	S	N	Y	#
1	2	3	4	5	6	7====>
1	2	3	Ø	5 + 4	6	7

12. AUXFILL Auxiliary Filler OB

#	X	T	$\left\{\begin{matrix}\text{be} \\ \text{have}\end{matrix}\right\}$	Y	#
1	2	3	4	5	6====>
1	2	3 + 4	Ø	5	6

13. AG Agreement OB

#	(PRE)	[(DET)[$\langle \alpha$Sg$\rangle$]$_N$X]$_{NP}$	$\left\{\begin{matrix}\text{PRES} \\ \text{PAST}\end{matrix}\right\}$	Y	#
1	2	3	4	5	6====>
1	2	3	4<αSg	5	6

Condition: $4 < \emptyset$

14. EVER Ever OP

#	X	INDEF	[$\langle$-human$\rangle$]$_N$	WH	$\left\{\begin{matrix}\text{INDEF} \\ \text{DEF}\end{matrix}\right\}$	N	Y	#
1	2	3	4	5	6	7	8	9====>
1	2	3	4	5	6+ever	7	8	9

15. REGDEL Regressive Deletion a. OP b. OB

#	X	$\left\{\begin{matrix}\text{ART}\\ \text{INDEF}\end{matrix}\right.$	$[\langle +\text{PRO}\rangle$ / N$]_N\}$	WH+	INDEF +	$\left\{\begin{matrix}\text{a. } \emptyset\\ \text{b. ever}\end{matrix}\right\}$	N	Y	#	
1	2	3	4		5		6	7	8	====>
1	2	$\emptyset$	$\emptyset$		5		6	7	8	

Condition: 4 = 6

---

16. DEFI Definitization OB

#	X	N	[(PREP) +	WH	INDEF	N	Y]$_S$	Z	#	
1	2	3		4	5	6	7	8	9	====>
1	2	3		4	DEF	6	7	8	9	

---

17. WHAG WH-Agreement OB

#	X	WH	$\left\{\begin{matrix}\text{INDEF}\\ \text{DEF}\end{matrix}\right\}$	(ever)	[$\langle$ human)]$_N$	Y	#	
1	2	3	4	5	6	7	8	====>
1	2	3	4 < human	5	6	7	8	

Condition: 4 < $\emptyset$

---

18. PROGDEL Progressive Deletion OB

#	X	N	(PREP) +	WH +	DEF	N	Y	#	
1	2	3		4		5	6	7	====>
1	2	3		4		$\emptyset$	6	7	

Condition: 3 = 5

---

19. RELDEL Relative Deletion OP

#	X	N	[WH Y]$_{NP}$ +	be +	PRES	ADJ	Z	#	
1	2	3		4		5	6	7	====>
1	2	3		$\emptyset$		5	6	7	

---

20. ADJPLACE Adjective Placement OB

#	X	N	ADJ	Y	#	
1	2	3	4	5	6	====>
1	2	4 + 3	Ø	5	6	

21. CDUP Complementizer Duplication OB

#	X	αC	NP	Y	#	
1	2	3	4	5	6	====>
1	2	3	4 + 3	5	6	

22. CNEG C Negative Placement OP

#	X	αC	$(\{^{be}_{have}\})$ + T	NEG	Y	#	
1	2	3	4	5	6	7	====>
1	2	5 + 3	4	Ø	6	7	

23. CTENSE C Tense OB

#	X	αC	$(\{^{be}_{(+V)}\})$ + PAST	Y	#	
1	2	3	4	5	6	====>
1	2	3 + have + en	4	5	6	

24. TS Tense Suppression OB

#	X	αC	$\{^{be}_{(have + (en))((+V))}\}$	T	Y	#	
1	2	3	4	5	6	7	====>
1	2	3	4	Ø	6	7	

25. CD Complementizer Deletion OP

#	X	V	$[(+\_\_S)]_N$	αC	NP	Y	#	
1	2	3	4	5	6	7	8	====>
1	2	3	4	Ø	6	7	8	

26. TAG Tag Question OP

#	NEG	Q	NP	AUX	VP	Y		#
1	2	3	4	5	6	7		8====>
1	2	Ø	4	5	6	7	3+4+5+8	

27. NEGPLACE Negative Placement OB

#	NEG	(Q)	NP	AUX	X	#
1	2	3	4	5	6	7====>
1	Ø	3	4	5 + 2	6	7

28. NEGTAG Tag Negative Placement OP

#	X	AUX	NEG	VP	(S)	Q	NP	AUX	#
1	2	3	4	5	6	7	8	9	10====>
1	2	3	Ø	5	6	7	8	9 + 4	10

29. NEGAUX Negative Auxiliary OP

#	X	T + ($\left\{\begin{matrix}\text{be}\\\text{have}\\\text{M}\end{matrix}\right\}$)	NEG	Y	#
1	2	3	4	5	6====>
1	2	3 + 4	Ø	5	6

30. QUES Question OB

#	Q	X	$\left\{\begin{matrix}\text{PREP} + [\text{WH} + \text{Y}]_{\text{NP}}\\[\text{WH} + \text{Y}]_{\text{NP}}\end{matrix}\right\}$	Z	#
1	2	3	4	5	6====>
1	4+2	3	Ø	5	6

31. YESNO Yes-No Question OB

#	X	Q	NP	AUX	Y	#
1	2	3	4	5	6	7====>
1	2	5 + 3	4	Ø	6	7

32. AF Affix OB

#	X	$\left\{\begin{matrix}T\\-C\\ing\\en\end{matrix}\right\}$	$\left\{\begin{matrix}NP\\\langle+V\rangle\\\langle+M\rangle\\be\\have\end{matrix}\right\}$	Y	#
1	2	3	4	5	6====>
1	2	∅	4 + 3	5	6

Condition: 5 = a terminal string σ_1, σ_2, ..., σ_n, such that $\sigma_1 \neq$ -C, ing, en, or T, and 2 = a terminal string ϕ_1, ϕ_2, ..., ϕ_n, such that $\phi_n \neq$ ⟨+V⟩ or ⟨+M⟩.

---

33. PREPDEL Preposition Deletion OB

#	X	PREP	$\left[\text{not } \begin{matrix}\langle+\_\_S\rangle\\\langle-C\rangle\end{matrix}\right]_N$	Y	#
1	2	3	4	5	6====>
1	2	∅	4	5	6

---

34. PD Pronoun Deletion OB

#	X	$[\langle+\_\_S\rangle]_N$	(to + NP)	S	Y	#
1	2	3	4	5	6	7====>
1	2	∅	4	5	6	7

---

35. AGDEL Agent Deletion OP

#	X	$[\text{PREP} + \text{INDEF} + [\langle+\text{PRO}\rangle]_N]_{\text{MAN}}$	Y	#
1	2	3	4	5====>
1	2	∅	4	5

---

36. THAT that Deletion OP

#	X	$\left\{\begin{matrix}V\\ADJ\end{matrix}\right\}$	(NP)	$[\text{that} \quad Y]_S$	Z	#
1	2	3	4	5	6	7	8====>
1	2	3	4	∅	6	7	8

37. VPCOMP VP Complement Placement OP

#	X	MAN	+C + {AUX / have} + VP	Y	#
1	2	3	4	5	6====>
1	2	4 + 3	Ø	5	6

38. BEDEL be Deletion OB

#	X	M	be	+C	Y	#
1	2	3	4	5	6	7====>
1	2	3	Ø	5	6	7

39. MCDEL Modal Complementizer Deletion OB

#	X	M	+C	Y	#
1	2	3	4	5	6====>
1	2	3	Ø	5	6

40. QDEL Q Deletion OB

#	X	Q	Y	#
1	2	3	4	5====>
1	2	Ø	4	5

41. ERASE Boundary Erasure OB

#	X	#
1	2	3====>
Ø	2	Ø

II. POST-CYCLIC RULES

42. PAST Past OB

X	{(+V) / (+M)}	PAST	Y
1	2	3	4====>
1	2	ed	4

43. MTDEL Modal Term Deletion OB

X	⟨+M⟩	PRES	Y
1	2	3	4====>
1	2	∅	4

---

44. PLUDEL Plural Deletion OB

X	⟨+V⟩	$[-Sg]_{PRES}$	Y
1	2	3	4====>
1	2	∅	4

---

45. NUM Number OB

X	$[\langle -Sg \rangle]_{N}$	Y
1	2	3====>
1	2 + s	3

---

46. NUAG Number Agreement OB

X	⟨+V⟩	+Sg	Y
1	2	3	4====>
1	2	s	4

---

47. CONTR Contraction OB

X	$[Y \quad NEG]_{AUX}$	Z
1	2 3	4====>
1	2 n't	4

---

48. NEGSPELL Negative Spelling OB

X	NEG	Y
1	2	3====>
1	not	3

49.	DO 1	Do 1		OB
X	$[[\omega Sg]_{PAST}$	$Y]_{AUX}$	Z	
1	2	3	4====>	
1	did	3	4	

50.	DO 2	Do 2		OB
X	$[[+Sg]_{PRES}$	$Y]_{AUX}$	Z	
1	2	3	4====>	
1	does	3	4	

51.	DO 3	Do 3		OB
X	$[[-Sg]_{PRES}$	$Y]_{AUX}$	Z	
1	2	3	4====>	
1	do	3	4	

52.	BE 1	Be 1		OB
X	be	$[+Sg]_{PRES}$	Y	
1	2	3	4====>	
1	Ø	is	4	

53.	BE 2	Be 2		OB
X	be	$[-Sg]_{PRES}$	Y	
1	2	3	4===>	
1	Ø	are	4	

54.	BE 3	Be 3		OB
X	be	$[+Sg]_{PAST}$	Z	
1	2	3	4====>	
1	Ø	was	4	

55. BE 4 Be 4 OB

X	be	$[-Sg]_{PAST}$	Y
1	2	3	4====>
1	∅	were	4

---

56. HAVE 1 Have 1 OB

X	have	$[+Sg]_{PRES}$	Y
1	2	3	4====>
1	∅	has	4

---

57. HAVE 2 Have 2 OB

X	have	$[-Sg]_{PRES}$	Y
1	2	3	4====>
1	∅	have	4

---

58. HAVE 3 Have 3 OB

X	have	PAST	Y
1	2	3	4====>
1	∅	had	4

---

59. WHPD 1 WH Pronoun Deletion 1 OP

X	DEF	$\begin{bmatrix}\langle +PRO\rangle \\ \langle +Sg\rangle\end{bmatrix}_N$	WH + Y
1	2	3	4====>
1	2	∅	4

---

60. WHPD 2 WH Pronoun Deletion 2 OB

X	WH + INDEF + (ever)	$\begin{bmatrix}\langle +PRO\rangle \\ \langle +Sg\rangle\end{bmatrix}_N$	Y
1	2	3	4====>
1	2	∅	4

---

61. WHDEL WH-Deletion OP

X	N	$[WH + Y]_{NP}$	NP	Z
1	2	3	4	5====>
1	2	∅	4	5

---

62. DEFTHAT Definite that OB

X	DEF	WH + Y
1	2	3====>
1	that	3

---

63. WH 1 WH 1 OB

X	$[WH\ [\langle +human \rangle]\ \begin{Bmatrix} DEF \\ INDEF \end{Bmatrix}]_{NP}$	Y
1	2	3 4====>
1	∅	who 4

---

64. WH 2 WH 2 OB

X	WH + DEF	Y
1	2	3====>
1	which	3

---

65. WH 3 WH 3 OB

X	WH + INDEF	Y
1	2	3====>
1	what	3

---

66. PLADEL Plural Article Deletion OB

X	INDEF	$[\langle -Sg \rangle]_{N}$	Y
1	2	3	4====>
1	∅	3	4

---

67. C 1 C 1 OB

X	+C	NP	Y
1	2	3	4====>
1	for	3	4

---

68. C 2 C 2 OB

X	+C	Y
1	2	3====>
1	to	3

---

69. C 3 C 3 OB

X	NP	-C	Y
1	2	3	4====>
1	2	's	4

---

70. C 4 C 4 OB

X	-C	X
1	2	3====>
1	ing	3

---

71. BY By OB

X	[PREP NP]$_{MAN}$	Y
1	2	3	4====>
1	by	3	4

---

72. INDEF Indefinite OB

X	INDEF	Y
1	2	3====>
1	a	3

---

73. DEF Definite OB

X	DEF	Y
1	2	3====>
1	the	3

APPENDIX V

Sentence Types Contained in English I

1. the boy likes the girl
2. the boys like the girl
3. the boy liked the girl
4. the boy does not like the girl
5. the boy will like the girl
6. the boy would like the girl
7. the boy will not like the girl
8. the boy is admiring the girl
9. the boy isn't admiring the girl
10. the boy has been admiring the girl
11. the boy will have been admiring the girl
12. does the boy like the girl?
13. doesn't the boy like the girl?
14. John likes the girl doesn't he?
15. John does not like Mary does he?
16. is John admiring Mary?
17. the books were purchased by John
18. must Mary be tormented by John?
19. John gave the book to Mary
20. John offered Mary the book
21. the book was offered to Mary by John
22. Mary was offered the book by John
23. who sleeps?
24. what boy sleeps?
25. which things slip?
26. what slips?

27. what book has John not taken?
28. about what did John speak?
29. the boy who must leave will leave
30. the book of which John speaks is awful
31. the book John speaks of is awful
32. John touched that which annoys Bill
33. Bill can visualize what will fall
34. whatever falls will bounce
35. a tall boy arrived
36. which tall boy did John see?
37. John would like for Mary to leave
38. John wants Mary to leave
39. John wants Mary to be loved by Bill
40. John prefers for Bill not to leave
41. Bill would prefer for John not to have dreamed
42. for John not to drown would be preferred
43. it is required for John to stand
44. Bond was believed to be dead by Goldfinger
45. John loves to run
46. John likes to be taken
47. John thinks Bill to be silly
48. John decided for Bill to represent Harry
49. John decided on Bill to represent Harry
50. John appears to have fallen
51. it embarrasses Bill to trip
52. John may resemble Bill
53. John dislikes Bill's annoying Mary
54. John dislikes Bill annoying Mary

55. John dislikes annoying Mary
56. John decided on going
57. John thinks that Bill will go
58. John thinks Bill smokes
59. that Bill smokes was mentioned by John
60. Bill mentioned to Mary that John smokes
61. it was mentioned by John that Bill smokes
62. Bill tells Mary John smokes
63. Bill reminded Mary to go
64. John tempted Mary to go
65. John condescended to go
66. John stops wondering

APPENDIX VI

Phrase Structure Rewriting Rules for Grammar II

S ----> # T NP VP #

VP ----> VB (NP) $\left(\begin{Bmatrix} NP \\ S \end{Bmatrix}\right)$

NP ----> $\begin{Bmatrix} NP & S \\ N & (S) \end{Bmatrix}$

UNCLASSIFIED

Security Classification

DOCUMENT CONTROL DATA - R&D

(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)

1. ORIGINATING ACTIVITY *(Corporate author)*
International Business Machines Corp.
T. J. Watson Research Center
P. O. Box 218, Yorktown Heights, N. Y. 10598

2a. REPORT SECURITY CLASSIFICATION
UNCLASSIFIED

2b. GROUP

3. REPORT TITLE
"SPECIFICATION AND UTILIZATION OF A TRANSFORMATIONAL GRAMMAR"

4. DESCRIPTIVE NOTES *(Type of report and inclusive dates)*
Final Scientific Report July 1965-September 1966

Approved: 23 Nov. 1966

5. AUTHOR(S) *(Last name, first name, initial)*
Rosenbaum, Peter S.
Blair, Fred

6. REPORT DATE
October 1966

7a. TOTAL NO. OF PAGES
53

7b. NO. OF REFS
16

8a. CONTRACT OR GRANT NO.
AF 19(628)-5127

b. PROJECT AND TASK NO.
4641,02

c. DOD ELEMENT
62405304

d. DOD SUBELEMENT
674641

9a. ORIGINATOR'S REPORT NUMBER(S)

9b. OTHER REPORT NO(S) *(Any other numbers that may be assigned this report)*
AFCRL-66-762

10. AVAILABILITY/LIMITATION NOTICES
Distribution of this Document is Unlimited.

11. SUPPLEMENTARY NOTES

12. SPONSORING MILITARY ACTIVITY
Hq. AFCRL, OAR (CRB)
United States Air Force
L. G. Hanscom Field, Bedford, Mass.

13. ABSTRACT

This report summarizes research carried out at the T. J. Watson Research Center in three areas of computational linguistics. These are 1) the design and development of a transformational grammar for a subset of grammatical sentences in English, 2) the implementation of this grammar in terms of a sentence synthesizing program written in LISP 1.5, and 3) the use of sentence synthesizing programs for transformational grammars generally.

DD FORM 1 JAN 64 1473

UNCLASSIFIED

Security Classification

UNCLASSIFIED

Security Classification

14. KEY WORDS	LINK A		LINK B		LINK C	
	ROLE	WT	ROLE	WT	ROLE	WT
Transformational Grammar-English Grammar I						
Phrase Structure Rewriting Rule						
Transformational Rule						
Deep Structure						
Surface Structure						
P-marker						
Sentence Synthesizing Program-SSP						
Fast-Fail Procedure						
Structure Index						
Elementary Transformation						
Proper Analysis						
Lexicon						

INSTRUCTIONS

1. **ORIGINATING ACTIVITY:** Enter the name and address of the contractor, subcontractor, grantee, Department of Defense activity or other organization *(corporate author)* issuing the report.

2a. **REPORT SECURITY CLASSIFICATION:** Enter the overall security classification of the report. Indicate whether "Restricted Data" is included. Marking is to be in accordance with appropriate security regulations.

2b. **GROUP:** Automatic downgrading is specified in DoD Directive 5200.10 and Armed Forces Industrial Manual. Enter the group number. Also, when applicable, show that optional markings have been used for Group 3 and Group 4 as authorized.

3. **REPORT TITLE:** Enter the complete report title in all capital letters. Titles in all cases should be unclassified. If a meaningful title cannot be selected without classification, show title classification in all capitals in parenthesis immediately following the title.

4. **DESCRIPTIVE NOTES:** If appropriate, enter the type of report, e.g., interim, progress, summary, annual, or final. Give the inclusive dates when a specific reporting period is covered.

5. **AUTHOR(S):** Enter the name(s) of author(s) as shown on or in the report. Enter last name, first name, middle initial. If military, show rank and branch of service. The name of the principal author is an absolute minimum requirement.

6. **REPORT DATE:** Enter the date of the report as day, month, year, or month, year. If more than one date appears on the report, use date of publication.

7a. **TOTAL NUMBER OF PAGES:** The total page count should follow normal pagination procedures, i.e., enter the number of pages containing information.

7b. **NUMBER OF REFERENCES:** Enter the total number of references cited in the report.

8a. **CONTRACT OR GRANT NUMBER:** If appropriate, enter the applicable number of the contract or grant under which the report was written.

8b, 8c, & 8d. **PROJECT NUMBER.** Enter the appropriate military department identification, such as project number, subproject number, system numbers, task number, etc.

9a. **ORIGINATOR'S REPORT NUMBER(S):** Enter the official report number by which the document will be identified and controlled by the originating activity. This number must be unique to this report.

9b. **OTHER REPORT NUMBER(S):** If the report has been assigned any other report numbers *(either by the originator or by the sponsor)*, also enter this number(s).

10. **AVAILABILITY/LIMITATION NOTICES:** Enter any limitations on further dissemination of the report, other than those imposed by security classification, using standard statements such as:

(1) "Qualified requesters may obtain copies of this report from DDC."

(2) "Foreign announcement and dissemination of this report by DDC is not authorized."

(3) "U. S. Government agencies may obtain copies of this report directly from DDC. Other qualified DDC users shall request through

_____."

(4) "U. S. military agencies may obtain copies of this report directly from DDC. Other qualified users shall request through

_____."

(5) "All distribution of this report is controlled. Qualified DDC users shall request through

_____."

If the report has been furnished to the Office of Technical Services, Department of Commerce, for sale to the public, indicate this fact and enter the price, if known.

11. **SUPPLEMENTARY NOTES:** Use for additional explanatory notes.

12. **SPONSORING MILITARY ACTIVITY:** Enter the name of the departmental project office or laboratory sponsoring *(paying for)* the research and development. Include address.

13. **ABSTRACT:** Enter an abstract giving a brief and factual summary of the document indicative of the report, even though it may also appear elsewhere in the body of the technical report. If additional space is required, a continuation sheet shall be attached.

It is highly desirable that the abstract of classified reports be unclassified. Each paragraph of the abstract shall end with an indication of the military security classification of the information in the paragraph, represented as *(TS), (S), (C), or (U).*

There is no limitation on the length of the abstract. However, the suggested length is from 150 to 225 words.

14. **KEY WORDS:** Key words are technically meaningful terms or short phrases that characterize a report and may be used as index entries for cataloging the report. Key words must be selected so that no security classification is required. Identifiers, such as equipment model designation, trade name, military project code name, geographic location, may be used as key words but will be followed by an indication of technical context. The assignment of links, rules, and weights is optional.

UNCLASSIFIED

Security Classification