ESD-TR-66-306

MTP-11

RECENT DEVELOPMENTS IN THE MITRE SYNTACTIC ANALYSIS PROCEDURE

SEPTEMBER 1966

D. E. Walker et al.

Prepared for

DEPUTY FOR ENGINEERING AND TECHNOLOGY
DIRECTORATE OF COMPUTERS
ELECTRONIC SYSTEMS DIVISION
AIR FORCE SYSTEMS COMMAND
UNITED STATES AIR FORCE
L. G. Hanscom Field, Bedford, Massachusetts

Distribution of this document is unlimited.

Project 7020
Prepared by
THE MITRE CORPORATION
Bedford, Massachusetts
Contract AF19(628)-5165

AD0642330

ESD-TR-66-306

MTP-11

RECENT DEVELOPMENTS IN THE MITRE SYNTACTIC ANALYSIS PROCEDURE

SEPTEMBER 1966

D. E. Walker et al.

Prepared for

DEPUTY FOR ENGINEERING AND TECHNOLOGY
DIRECTORATE OF COMPUTERS
ELECTRONIC SYSTEMS DIVISION
AIR FORCE SYSTEMS COMMAND
UNITED STATES AIR FORCE
L. G. Hanscom Field, Bedford, Massachusetts

Project 7020

Prepared by

THE MITRE CORPORATION
Bedford, Massachusetts

Contract AF19(628)-5165

ABSTRACT

The MITRE syntactic analysis procedure for transformational grammars has been used to process sentences on-line in a display-oriented mode as well as off-line. This report describes additions to the program structure and the grammar made since the last report and presents the results of experiments with the procedure.

REVIEW AND APPROVAL

This technical report has been reviewed and is approved.

[illegible]

Lt. J. B. FRASER
Directorate of Computers

TABLE OF CONTENTS

Introduction

The MITRE Syntactic Analysis Procedure was developed in the context of work on computer-based information systems. Descriptions of the MITRE transformational grammar, of the steps in the analysis procedure and of the initial computer programs for implementing the procedure were presented at the 1965 Fall Joint Computer Conference (Zwicky, Friedman, Hall, and Walker, 1965). The present paper describes further developments in the work program: additions to the grammar, changes in the procedure, and new computer programs that now provide a completely mechanical analysis of sentences -- on-line as well as off-line.

Overview of the Grammar and the Analysis Procedure

The model of language underlying the analysis procedure is that of transformational grammar as developed in particular by Chomsky. A transformational grammar distinguishes the surface structure from the base (deep or underlying) structure for a sentence. The surface structure represents the sentence in the form to be phonologically interpreted. The base structure for a sentence represents the categorial and relational information required for semantic interpretation.

The MITRE grammar contains two components: an ordered set of context-sensitive phrase structure rules and an ordered set of transformational rules. Lexical items occur as terminal symbols in certain of the phrase structure rules. The phrase structure rules generate a set of base trees. The transformational rules convert these base trees into surface trees. The procedures developed for analysis reverse this procedure in five processing steps: lexical lookup, surface grammar parsing, transformation reversal, context-sensitive phrase structure check, and synthesis. These steps, summarized below, were described in detail in the Fall Joint Computer Conference paper.

In lexical lookup the input sentence is mapped into a set of pre-trees, which are strings of trees containing both lexical and grammatical items. The pre-trees are obtained by the substitution of lexical entries for each word. A word may have several lexical entries; the number of pre-trees associated with a sentence is the product of the numbers of lexical entries for the words in the sentence.

The surface grammar is a context-free phrase structure grammar containing every expansion which can occur in a surface tree. Unavoidably, the surface grammar generates some trees which are not correct surface trees, i.e., not generable by the MITRE grammar, even though the corresponding terminal string may be a generable sentence with some other structure. In the second step of the analysis procedure, the surface grammar is used in a context-free parsing algorithm to construct from each pre-tree a set of presumable surface trees associated with the input sentence.

The next step reverses the effect of all transformational rules that might have applied to yield a given presumable surface tree. The "undoing" of the forward rules is achieved by rules that are very much like the forward rules in their form and interpretation. The reversal rules are applied in an order which is essentially the opposite of the order in which the corresponding forward rules are applied. The effect of transformation reversal is to map each presumable surface tree into a presumable base tree.

In the next step each presumable base tree is checked against the phrase structure component of the (forward) grammar. As a result of the check, presumable base trees that are not, in fact, base trees are discarded.

It is possible in the transformation reversal step to map a presumable surface tree into a base tree of the grammar which is not the base tree underlying that surface tree. In the synthesis step, the full set of (forward) transformations is applied to each base tree that survives the checking step. If the resulting surface tree is identical to the presumable surface tree being analyzed, then its base tree is a base tree of the input sentence.

The initial computer programs, written in FORTRAN (Friedman, 1965) and in TREET, a LISP-like list-processing language (Haines, 1965), were concerned exclusively with the last three steps in the analysis procedure. A simple form of lexical look-up has since been programmed; more sophisticated techniques are under study. An efficient algorithm has been programmed for step 2, the context-free parsing. Modifications in step 3, transformation reversal, have resulted in a more effective elimination of spurious surface trees and seem to make the synthesis step unnecessary. These changes, the additions to the grammar, and other current activities will be described in the following sections.

Surface Grammar Parsing

The surface grammar allows for the assignment to a given sentence of a number of surface trees. Since these surface trees share many common subtrees, the parsing algorithm must be efficient in discovering and representing their common parts. These requirements are met by the algorithm used in the analysis procedure, which is a modification of one originally proposed by Martin Kay (1964); the Kay algorithm is described in the English Preprocessor Manual (1964, 1965).

The procedure is a bottom-to-top algorithm (cf. Griffiths and Petrick, 1965); it finds all the trees which dominate any substring of the terminal string of a pre-tree. Any tree immediately dominated by more than one node will be found only once, and each of the nodes which dominate it will point to it. The algorithm operates simultaneously on all the pre-trees provided by the lexical lookup step. However, by way of illustration a parsing based on only the correct pre-tree will be considered first. The correct pre-tree for the sentence 'Can the airplane fly?' is

(1) $$\# \quad \text{PRES} \quad \text{SG} \quad \overset{\text{M}}{\overset{|}{\text{CAN}}} \quad \text{THE} \quad \overset{\text{NCT}}{\overset{|}{\text{AIRPLANE}}} \quad \text{SG} \quad \overset{\text{VINT}}{\overset{|}{\text{FLY}}} \quad \#$$

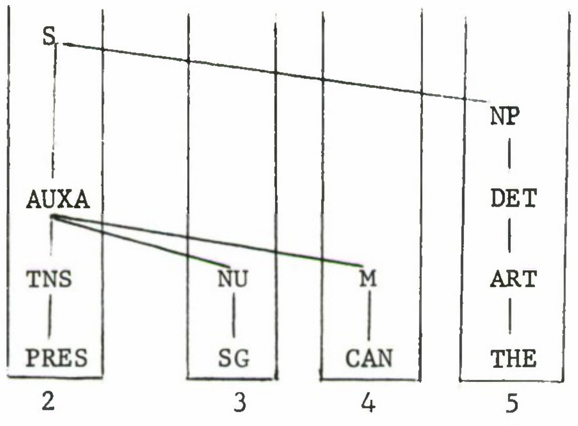
This pre-tree string is broken up into segments, and each segment is put into a separate bin:

(2) $$\underset{1}{\boxed{\#}} \; \underset{2}{\boxed{\text{PRES}}} \; \underset{3}{\boxed{\text{SG}}} \; \underset{4}{\boxed{\overset{\text{M}}{\overset{|}{\text{CAN}}}}} \; \underset{5}{\boxed{\text{THE}}} \; \underset{6}{\boxed{\overset{\text{NCT}}{\overset{|}{\text{AIRPLANE}}}}} \; \underset{7}{\boxed{\text{SG}}} \; \underset{8}{\boxed{\overset{\text{VINT}}{\overset{|}{\text{FLY}}}}} \; \underset{9}{\boxed{\#}}$$

An item in a bin is either a terminal node (e.g., SG in bin 3) or a non-terminal node with pointers (e.g., M in bin 4). If the node is non-terminal its left-most daughter (also an item) is in the same bin. Thus the leftmost terminal symbol of the tree dominated by a non-terminal node is in the same bin with that node. The pointers (illustrated in (2) as lines) from a node point to its daughters. The bin coming immediately after the bin containing the rightmost terminal symbol of a tree is the <u>next bin</u> for that tree. If a node has more than one daughter, then the (i + 1)st daughter must be in the next bin for the tree dominated by the ith daughter.

The notion of next bin is illustrated in the following segment from a parsing for the example sentence:

(3)



The first daughter of S is AUXA. The rightmost terminal symbol of the tree dominated by AUXA is CAN in bin 4. Therefore the <u>next bin</u> for that tree is bin 5, and the second daughter of S, NP, must be in bin 5.

A complete path exists between two nodes A and B if either B is in A's next bin or if there exists a node C such that a complete path exists between A and C and B is in C's next bin. Each node on a complete path is called a <u>path node</u>. There may be several complete paths between two nodes, and these paths may share path nodes.

The parsing algorithm is as follows:

1. Set the bin index i to the number of the last bin.

2. Call the first node in bin_i "A_1".

3. For each rule in the grammar of the form

$$A \rightarrow A_1\ A_2\ \dots\ A_k$$

find all complete paths between A_1 and A_k which have A_1, A_2, ... A_k, as path nodes. (In the case where $k = 1$ there is only one such path.) For each such path add to bin_i the symbol A with a pointer to each path node in that path.

4. If there still remains in bin_i a symbol which has not been processed by Step 3, then call this symbol "A_1" and go to Step 3.

5. If $i = 1$, the algorithm is terminated. Otherwise, decrease i by 1 and go to Step 2.

The algorithm will find all the trees that terminate in substrings of the terminal string. Therefore, if the sentence has a parsing, there should appear in column 1 the sentence node symbol (SS in the MITRE grammar) corresponding to the top node of a tree whose rightmost terminal symbol is the last terminal symbol of the string.

An excerpt from the JUNIOR surface grammar* sufficient for parsing (2) is presented in Table 1. For more efficient use in the algorithm the grammar is arranged so that rules with the same first symbol after the arrow are grouped together. Some rules have been omitted to make the diagrams clearer, but the parsing will produce as many trees as with the complete JUNIOR grammar.

Figure 1 shows the results of applying the parsing algorithm to (2). Notice the two S nodes in bin 2; they dominate different trees because the rule S → AUXA NP VP allows for two different complete paths since there are two NP's in bin 5. The tree dominated by the

*See the English Preprocessor Manual (1964, 1965), S010.

SS	→	# S #
DET	→	ART
S	→	AUXA NP AUX VP
S	→	AUXA NP VP
AUX	→	AUXA
NP	→	DET N NU
NP	→	DET
AUX	→	ING
NP	→	N NU
N	→	NCM
NCM	→	NCT
S	→	NP NP AUX VP
S	→	NP NP AUX
S	→	NP AUX VP
PRED	→	NP
NU	→	PL
VP	→	PRED
TNS	→	PRES
TNS	→	PST
NU	→	SG
ART	→	THE
AUXA	→	TNS NU BE
AUXA	→	TNS NU M
AUXA	→	TNS NU
VP	→	V NP
VP	→	V
V	→	VTR
V	→	VINT

Table 1. Surface grammar rules

18-18,963

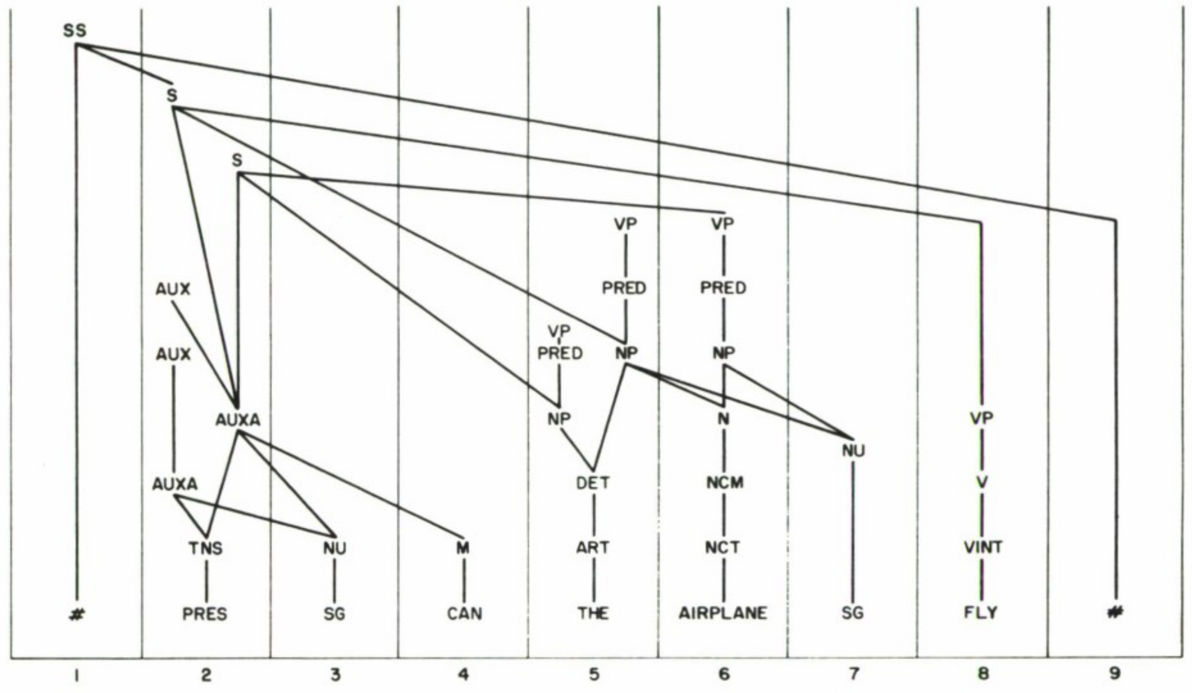


Figure 1. Result of the application of the parsing algorithm to the correct pre-tree for the sentence 'Can the airplane fly?'

lower S node (Figure 2) has bin 8 as its next bin; it is not dominated by the initial symbol SS and hence is not a surface tree.

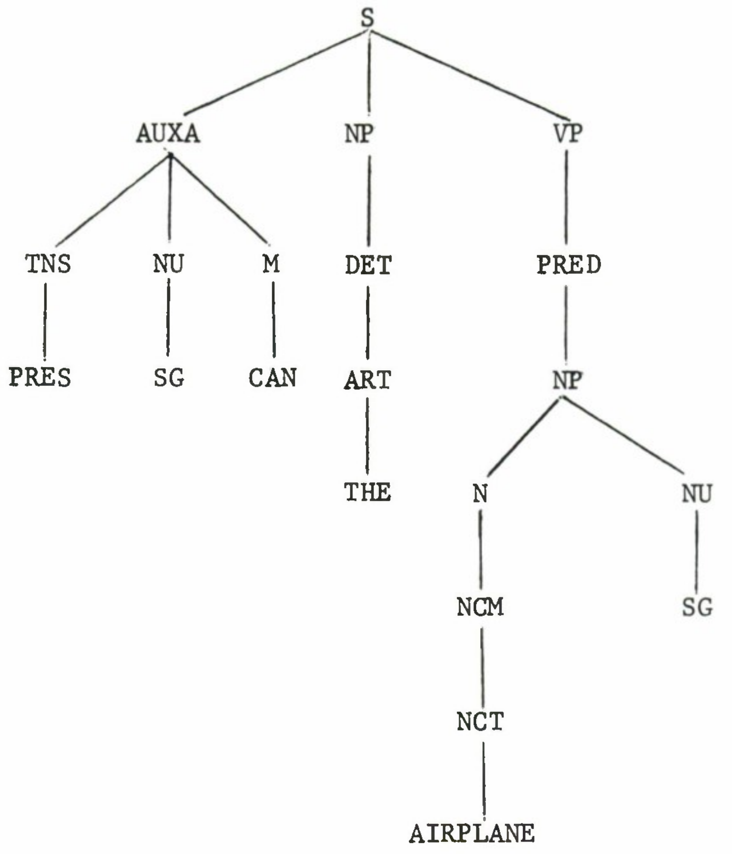


Figure 2. Tree dominated by the lower S node in Figure 1, bin 2

The tree with the higher S node is dominated by SS in bin 1 and has bin 9 as its next bin. Since the SS dominates the entire string, Figure 3 is the correct surface tree.

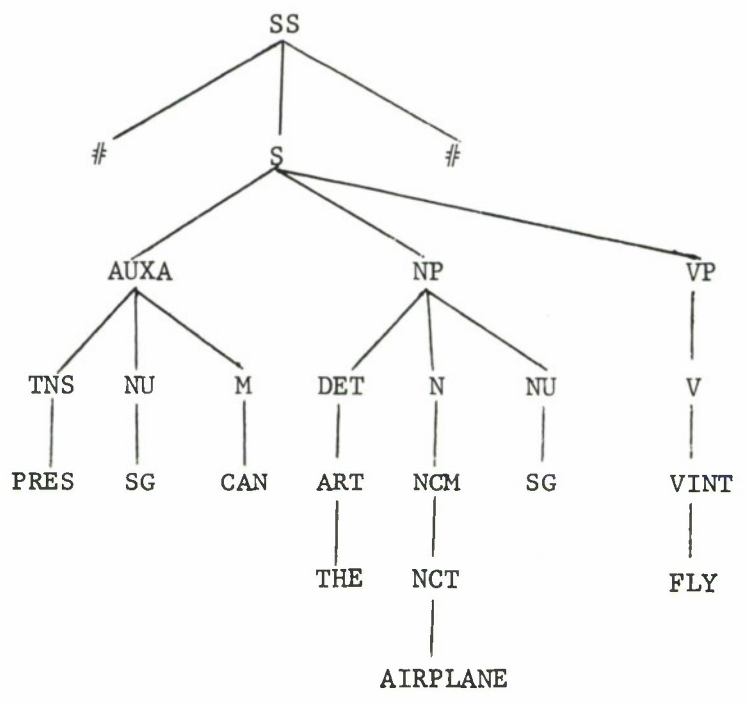


Figure 3. Correct surface tree

Notice that the tree in Figure 1 whose top node is AUXA in bin 2 is part of the two trees whose top nodes in bin 2 are S and that it is also part of a tree in bin 2 whose top node is AUX. This capability for compact representation of shared structure is an important aspect of the efficiency of the parsing procedure.

Having described the general form of the parsing algorithm for a simple case, it is now possible to consider the algorithm as programmed, using a more complex example. The same sentence 'Can the airplane fly?' will be parsed, but all the pre-trees derived from the lexical lookup will be included. Two changes in the algorithm were made in the course of programming. The changes relate to the way of handling "pointers" and "next bins" and to ways of coalescing items within a bin. These two changes will be described in sequence.

The fifteen pre-trees of the example sentence can be expressed compactly as in Figure 4; symbols within a pair of braces are alternatives.

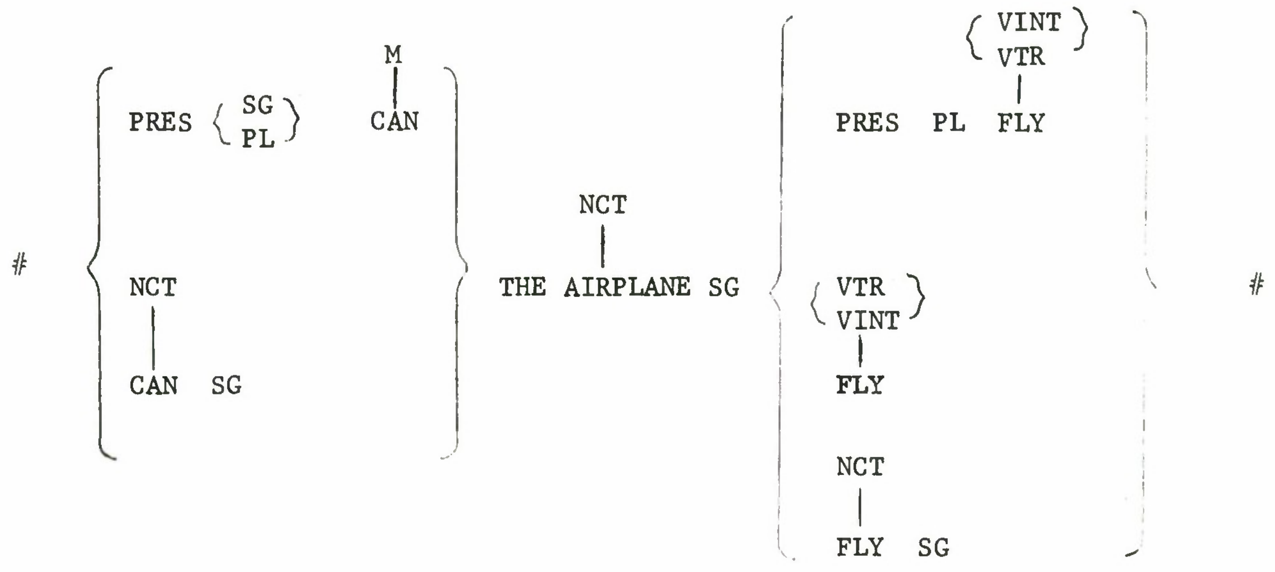


Figure 4. Pre-trees for the sentence 'Can the airplane fly?'

These fifteen pre-trees also may be represented by putting segments into thirteen different bins with a pointer from each terminal item indicating which bin is the next bin. In general, terminal nodes in the first column after a left brace will appear as items in the same bin, but each item will point to a different bin as the next bin for that item. Where there is a choice of non-terminal nodes which can dominate the same terminal node, each of the non-terminals points to the same terminal as its daughter. These characteristics are illustrated in Figure 5 which shows the set of bins corresponding to Figure 4. Note that there are six different paths through the set of bins and that no tree which has

IB-19,209

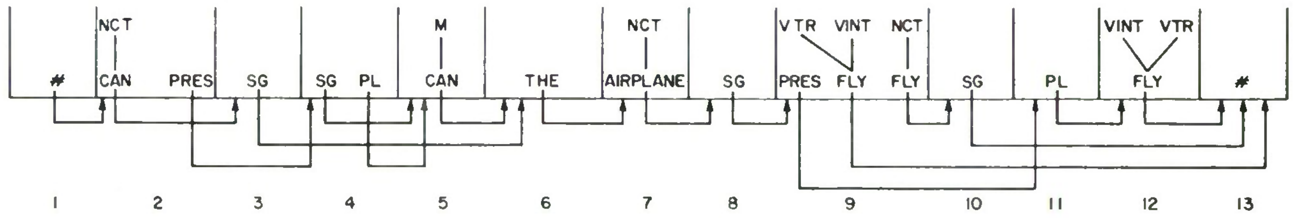
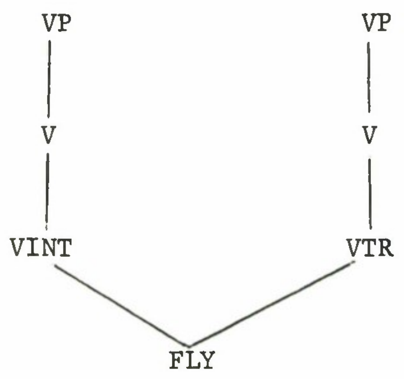


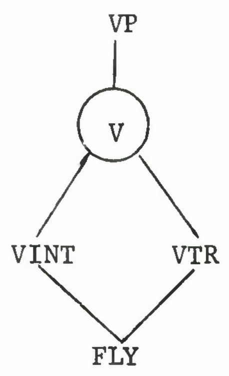
Figure 5. Segmentation of the complete set of pre-trees

a terminal in bin 3 may have a terminal in bins 4 or 5. The items that appear in a bin, then, are as in the earlier example except that the bin which is pointed to by the rightmost terminal symbol of a tree dominated by the node of an item is the "next bin" for that item. To make the program operate faster, each item has a pointer to its next bin even though for nonterminal nodes these pointers are redundant. To simplify the representation the redundant pointers are not included in Figure 5.

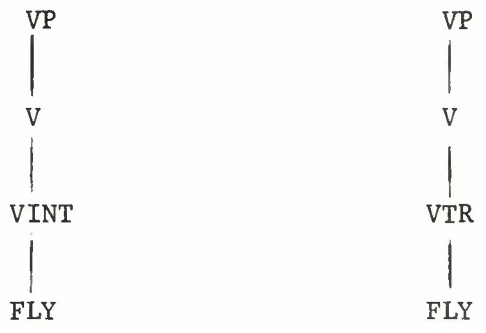
The second change in the algorithm as programmed requires that in a given bin, if two or more trees have the same top nodes and the same next bin, they must be expressed as a single item with a different set of daughter pointers for each tree. For example, in bin 12 of Figure 5 the items VINT and VTR have the same next bin and are both parsed as V in accordance with the surface grammar. Since V is parsed as VP, the unmodified parsing algorithm would result in the following representation of the items in bin 12:



By combining the two trees whose top nodes are V, a more compact representation is obtained:



Nodes, like V, that have more than one set of daughters are circled; the pointers to each set of daughters originate in a single place on the circle. This representation is equivalent to the two trees:



The more compact version reduces by two the number of items in the bin and, consequently, reduces the number of complete paths to be considered in further parsing. The savings on space afforded by allowing an item to have more than one set of daughters can become very large if this coalescing is possible in a number of bins. This change made a significant difference in the complexity of sentences that the program could handle with the core space available.

Figure 6 shows the complete set of bins which results from application of the modified algorithm to Figure 5. Each bin in Figure 6 contains a specification of all trees generated by the grammar which terminate in a substring

IB-18,065

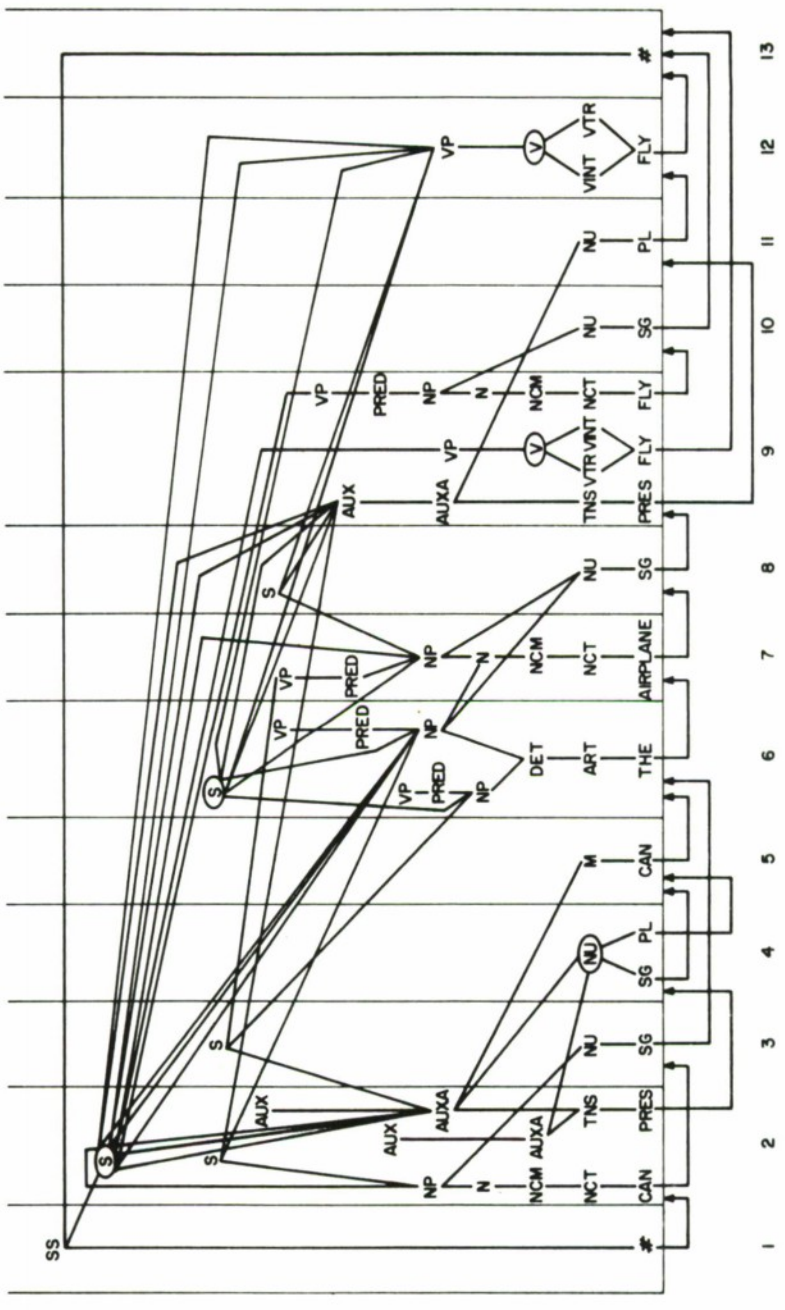


Figure 6. Parsing of the complete set of pre-trees

of any one of the pre-trees which has its leftmost symbol in that bin. The item SS in column 1 is the top node of twelve different surface trees with the terminal string of each tree being the terminal string of one of the pre-trees. Not every pre-tree need result in a surface tree, and several surface trees may result from the same pre-tree. Eliminating all items not dominated by that SS gives a structure of the form shown in Figure 7. The structure of Figure 7 unravelled into the set of 12 trees is shown in Figure 8.

The algorithm described above has been programmed for the IBM 7030 computer in the TREET list processing language (Haines, 1965). The program operates very rapidly, but uses a large amount of space. Both effects are due to the fact that it must remember all possible trees covering any string of terminal items belonging to any of the pre-trees. Since they are remembered they don't have to be recomputed, but remembering them is expensive. The sentence 'The general that Johnson met in Washington had traveled eight-thousand miles' has 144 presumable surface trees. The program found them all in about 10 seconds. The sentence 'The airplane that was seen by the general that met Joyce landed at MITRE for three hours' has 572 presumable surface trees, which were found by the program in about 38 seconds. Each of the 572 trees in this last sentence has about 100 nodes, which means that the total number of nodes in all the trees is about 57,200. To fit these trees into core, they must be stored in a fashion which allows maximum overlap. The algorithm used only about 3,150 nodes to represent all 572 trees.

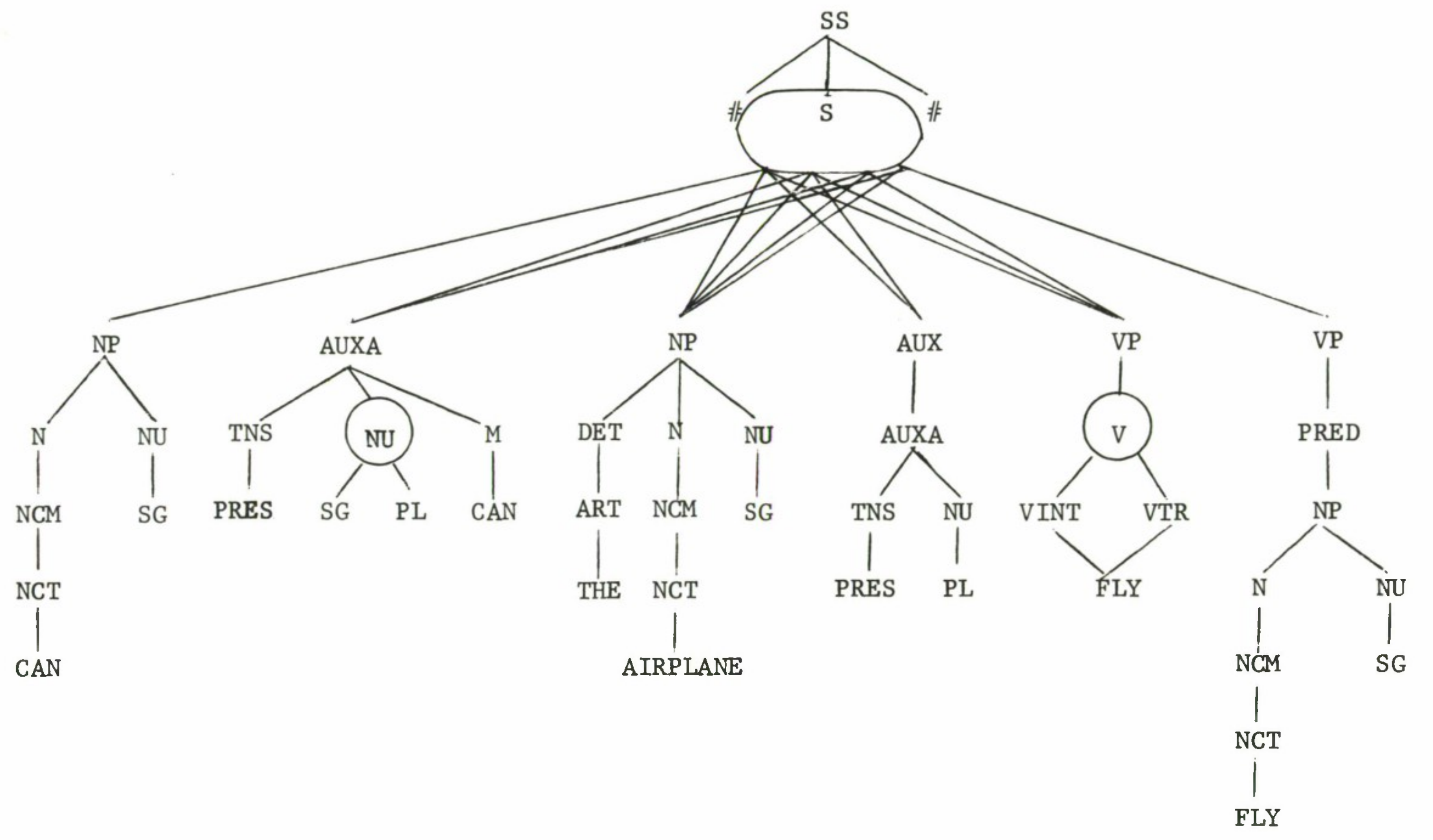
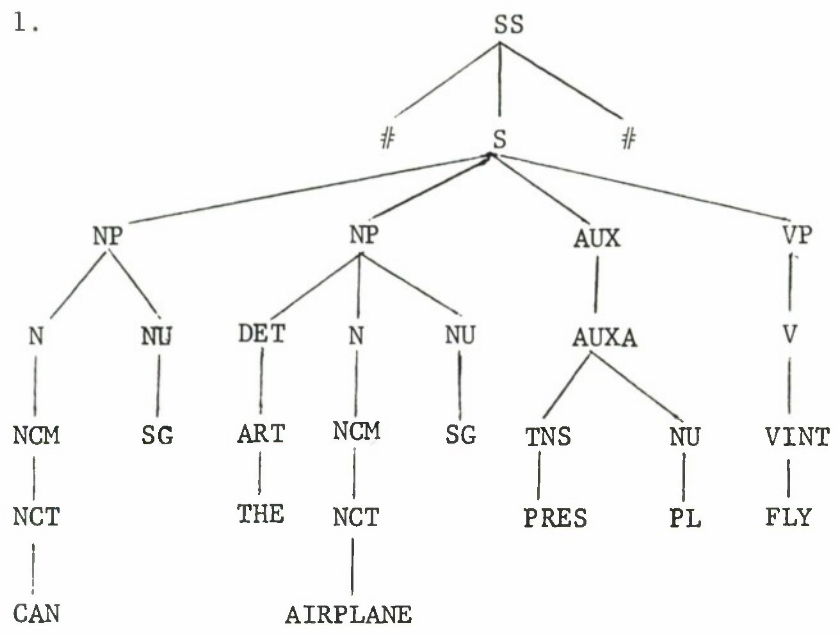


Figure 7. Structure produced by surface grammar parsing

Tree 1.



Tree 2.

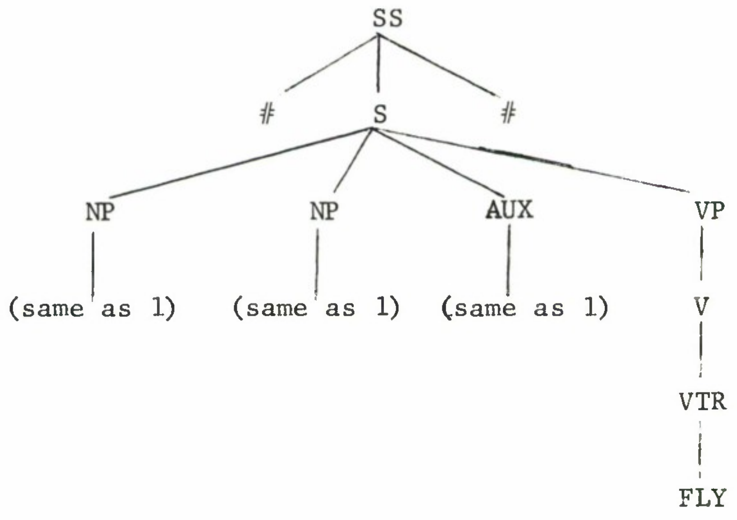
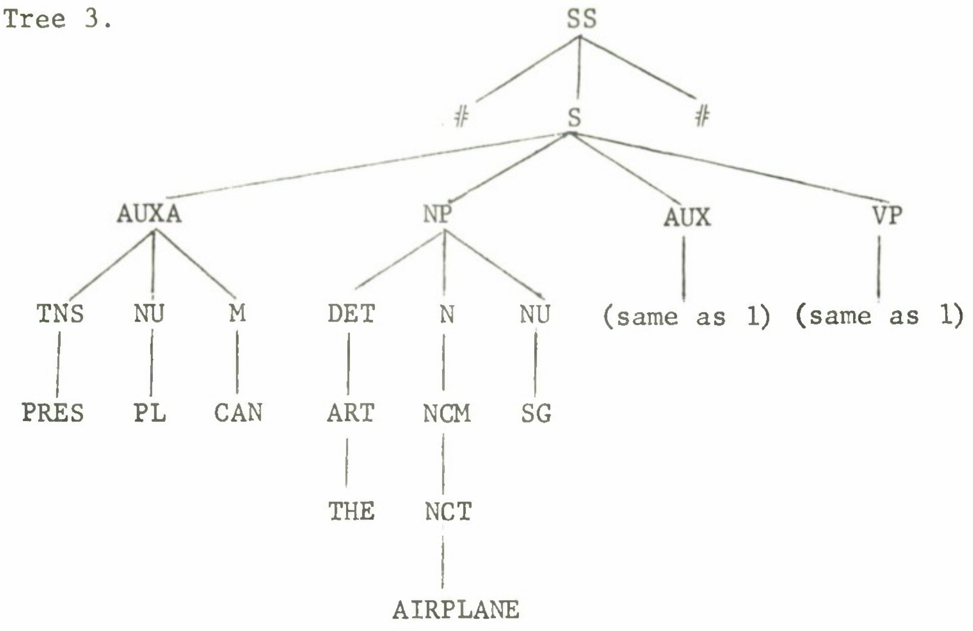
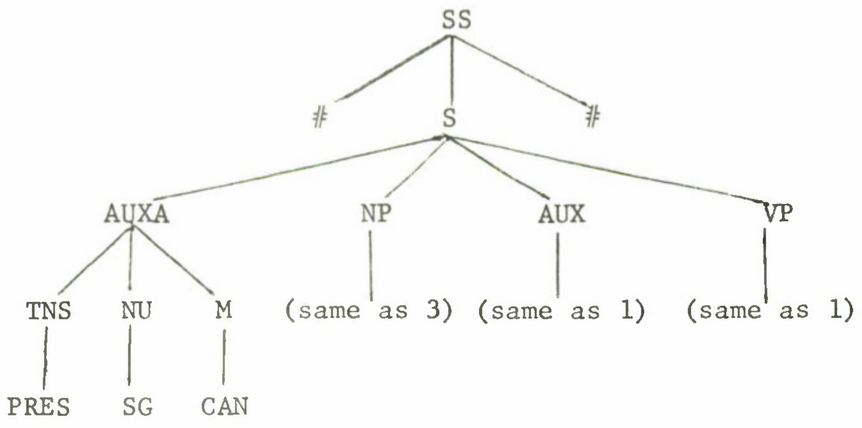


Figure 8. Presumable surface trees from surface grammar parsing

Tree 3.



Tree 4.



Tree 5.

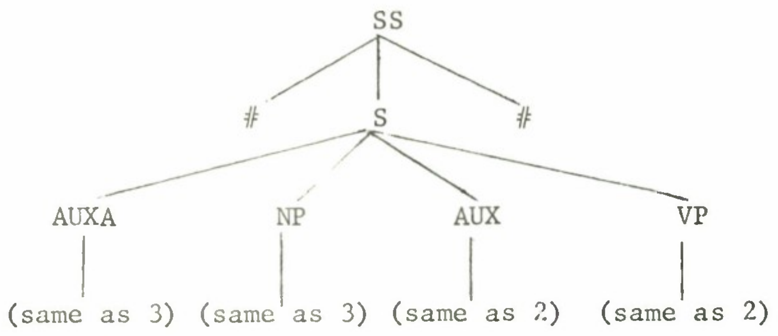
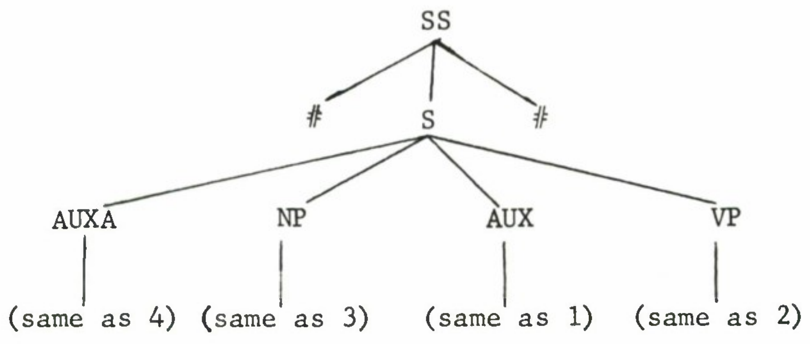
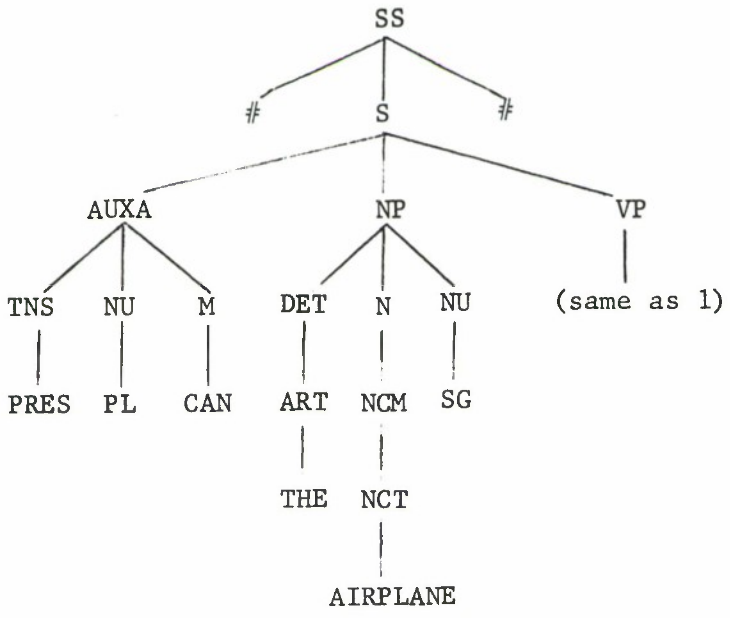


Figure 8. (continued)

Tree 6.



Tree 7.



Tree 8.

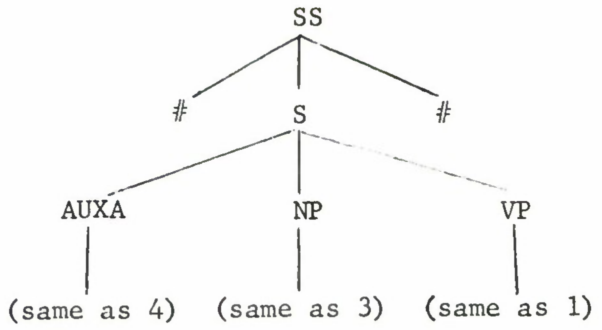
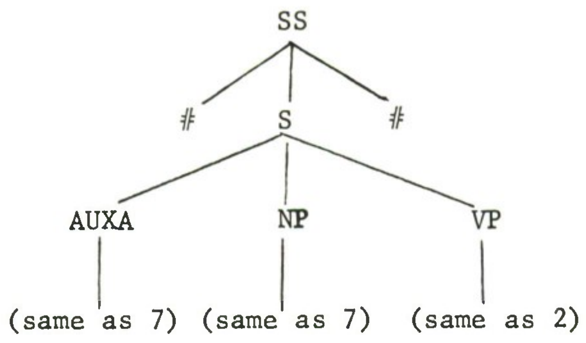
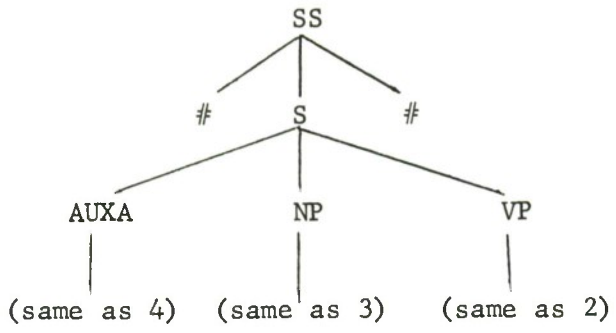


Figure 8. (continued)

Tree 9.



Tree 10.



Tree 11.

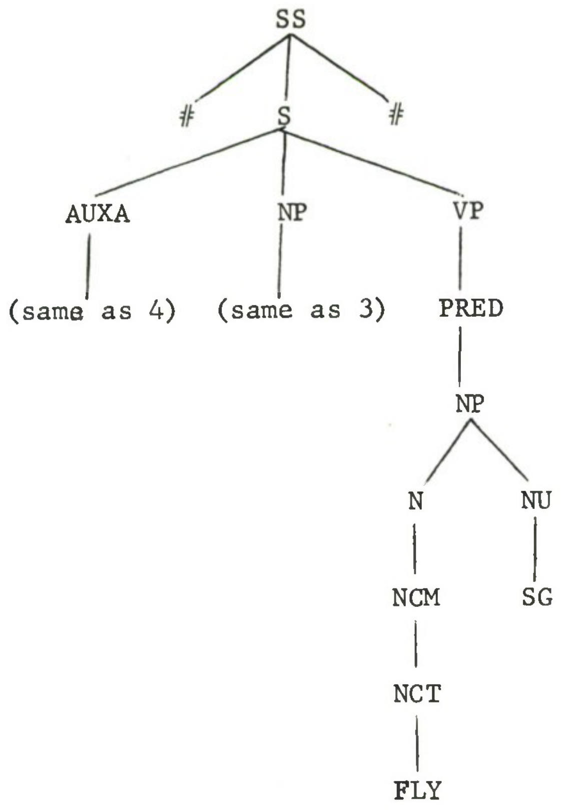


Figure 8. (continued)

Tree 12.

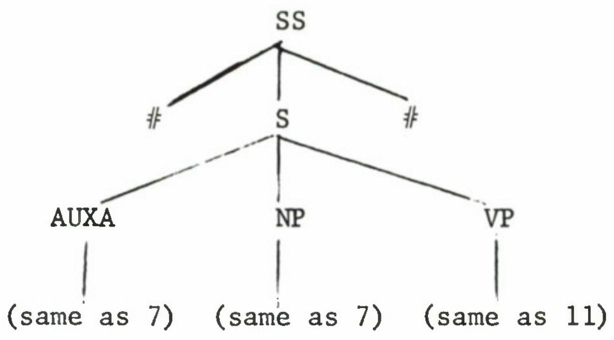


Figure 8. (concluded)

Transformation Reversal

The programming of the surface grammar parsing algorithm made it possible to experiment with the total analysis procedure. Since the last three phases of the analysis procedure -- transformation reversal, context-sensitive phrase structure checking, and synthesis -- involve processing all of the surface trees produced by the surface parsing, the number of those trees becomes a critical factor in the overall time for analysis. The figures cited at the end of the previous section indicate the magnitude of the problem. Two strategies are being developed to reduce the time required: (1) The use of special rules in transformation reversal that reject incorrect surface trees and (2) the application of reversal rules to structures like Figure 7 (to be called _supertrees_) that represent compactly a number of trees.

Rejection rules. Most of the presumable surface trees resulting from surface parsing are not actual surface trees for any sentence generated by the MITRE grammar. In fact, none of the spurious surface trees can be an actual surface tree for any sentence generated by the grammar. A context-sensitive surface grammar parsing would reduce the number of spurious surface trees produced. Preliminary experiments with a context-sensitive surface grammar used as a check immediately after the context-free parsing resulted in the elimination of most of the spurious surface trees. Further explorations with techniques involving context-sensitive rules will be carried out in the future. It is possible, however, to accomplish a similar effect during the reversal phase by adding rules that reject trees with incorrect structures. These rejection rules terminate the processing of a presumable surface tree when it meets a specified structural description. In the interest of efficiency it would be desirable to place these rules as early as possible in the reversal phase, and many can be applied at the beginning.

The operation of rejection rules can be illustrated by considering three of them in relation to the twelve trees of Figure 8. Trees satisfying the following structural descriptions or conditions are rejected:

1. # NP NP X

2. AUXA X AUXA Y

The sequence NP NP X is consistent with the surface grammar, since it is correct for certain embedded relative clauses, but it cannot occur immediately following a sentence boundary. AUXA X AUXA Y arises in questions (including the example sentence). It is not itself a valid sequence, but is always parsed as AUXA X AUX Y which can be valid.*
Rules 1 and 2 can both be applied at the beginning of the reversal phase.

3. NU(for subject NP) $\neq$ NU(for AUXA)

For questions, this condition can be assessed only after the reversal rule has moved the AUXA after the subject NP. Accordingly, this rejection rule can not be introduced until that point.

With respect to the twelve trees in Figure 8, trees 1 and 2 will be rejected by the first rule, 3 through 6 by the second rule, and 7, 9, and 12 by the third rule. Tree 8 will be transformed into the presumable base tree shown in Figure 9, which is the correct base tree for the sentence (the synthesis step will transform it back into tree 8).

---

*AUXA X AUX Y is the structure of questions like 'Is John running?', where the AUX dominates the ING of <u>running</u>.

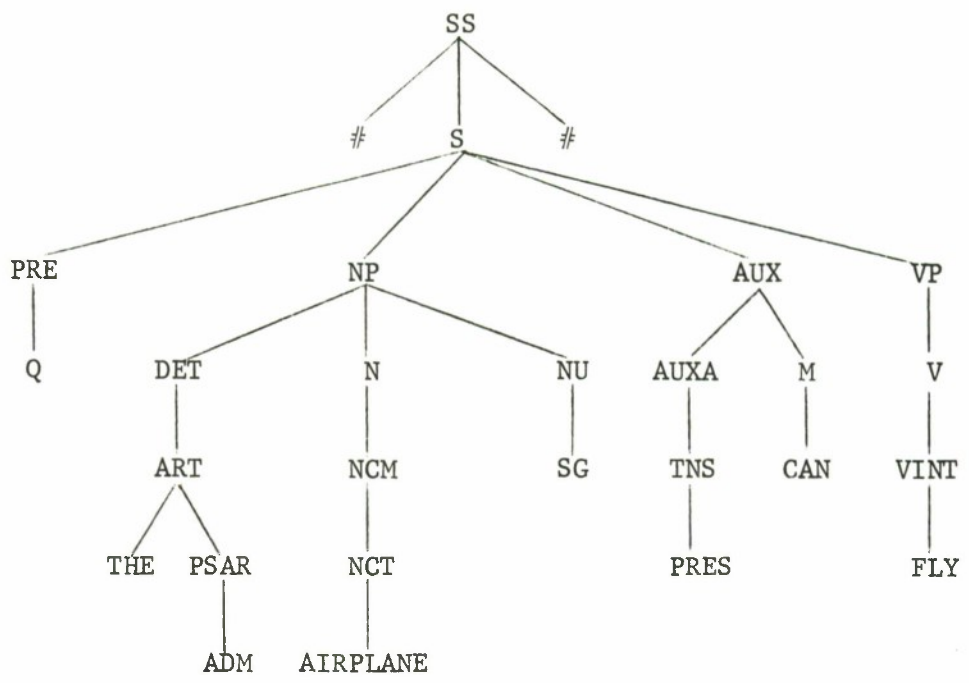


Figure 9. Base tree for the sentence.

Tree 10 will be transformed into a base tree differing from Figure 9 only in parsing FLY as VTR instead of VINT. This base tree will fail the context-sensitive phrase structure check because there is no object NP as context for the expansion of V into VTR. Tree 11 will be transformed into a base tree differing from Figure 9 only in the subtree dominated by VP. Since the VP in tree 11 expands into PRED rather than BE PRED, the tree will fail the phrase-structure check. Rules to reject trees 10 and 11 could be written, but both would have to follow the passive reversal rule, which is the last to apply. The relative economy of rejection rules in cases like these will have to be determined empirically.

The use of rejection rules has an additional consequence beyond expediting the reversal step. It now seems possible to eliminate the synthesis step entirely. Synthesis was required because in some cases a surface tree that is not correct for any sentence generable by the grammar can be mapped by reversal rules into a base tree which is correct for some sentence generable by the grammar. In every instance of this kind discovered so far, rejection rules have eliminated the incorrect surface tree. Further research will be required to determine whether rejection is possible in every such case. If it is possible, then the synthesis step is unnecessary.

The effect of rejection rules on the efficiency of the analysis procedure can be indicated by some comparisons. For the sentence 'The general that Johnson met in Washington had travelled eight-thousand miles,' 144 presumable surface trees were found. 108 were rejected by the initial rejection rules before any regular reversal rules were applied; 24 more were rejected after about half the reversal rules were applied; 8 more were rejected just prior to the context-sensitive phrase structure check; 3 were rejected during that check; 1 tree (corresponding to the correct base tree) passed that check and the synthesis check. The time required for the total processing was about 6 1/2 minutes in contrast to a time of about 36 minutes without the use of rejection rules.

Supertrees. It is obvious, even with the reduction in time afforded by rejection rules, that some additional ways to improve speed and efficiency are necessary. One technique under investigation involves the application of the reversal and rejection rules to supertrees. Figure 7, the supertree for the example sentence, is derived directly from Figure 6 by deleting all the structures that are not parts of presumable surface trees for the sentence. Applying the first rejection rule to the supertree would remove the first set of daughters attached to the S node -- all the trees with the analysis # NP NP X. The second rejection rule would remove the second set of daughters attached to the S node -- all the trees with the analysis AUXA X AUXA Y. Thus, the application of two rejection rules to the supertree would eliminate six presumable surface trees and result in the supertree shown in Figure 10. During the application of the reversal rules, the rejection rule involving number agreement would eliminate half the remaining trees. The supertree of Figure 11 represents the three remaining trees, to be checked against the context-sensitive phrase structure rules.

An even more compact form of supertree is possible. Those illustrated branch only to alternate sets of daughters; it is also possible to group together alternate sets of sisters. The procedures for constructing and operating on supertrees are still being developed.

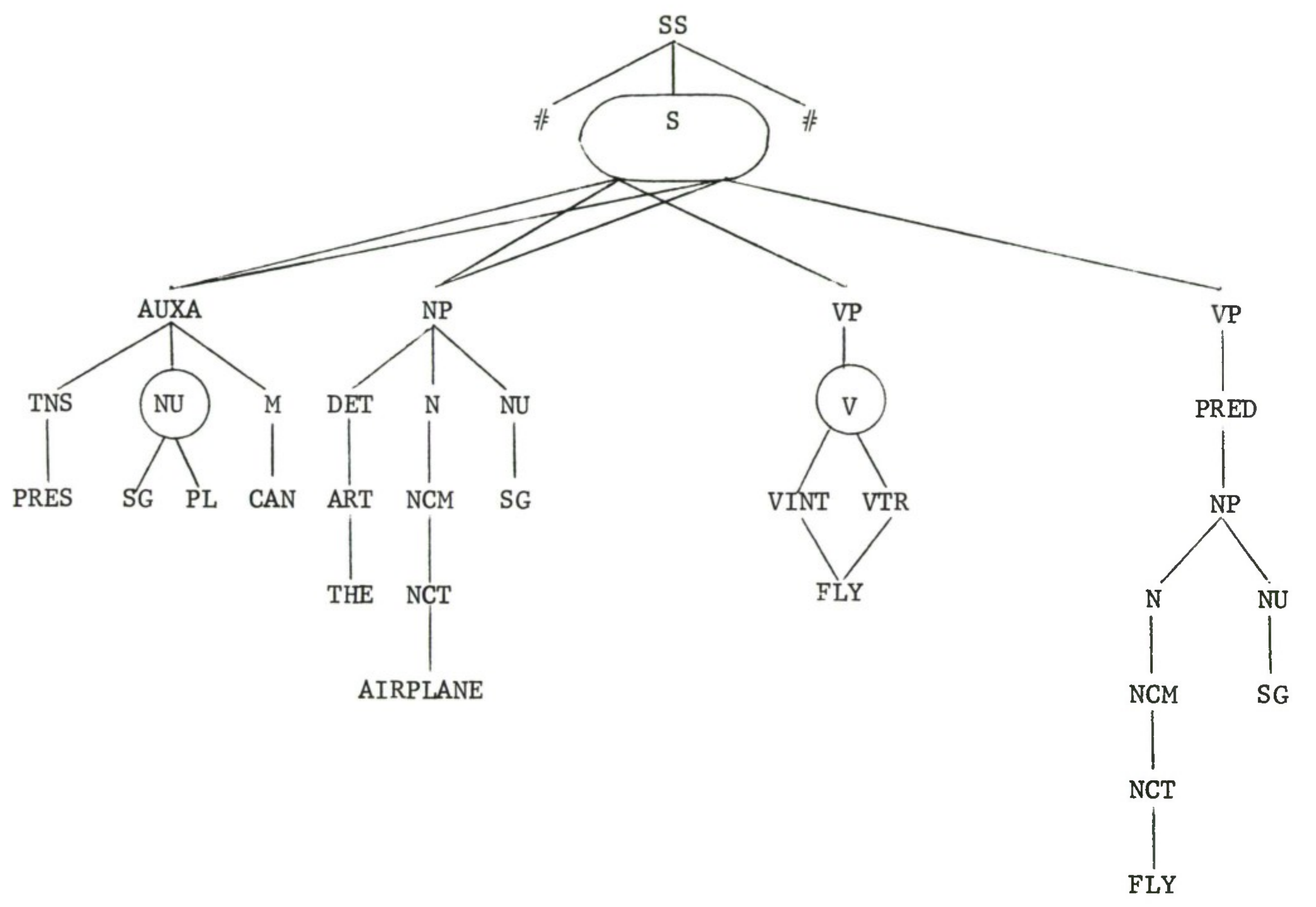


Figure 10. Supertree after initial rejection rules

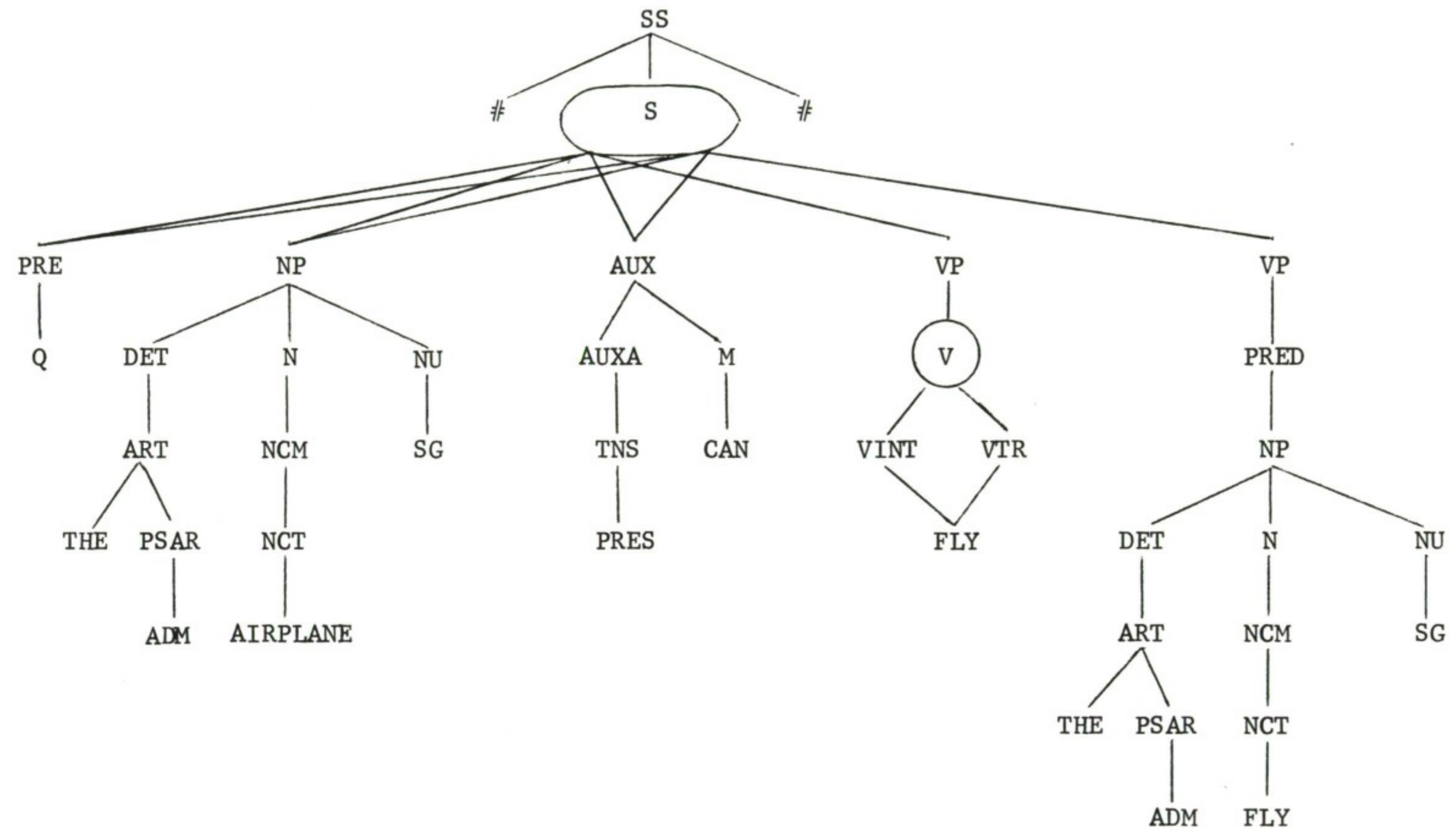


Figure 11. Supertree just before context-sensitive phrase structure check

Changes in The MITRE Grammar

The core of the MITRE syntactic analysis procedure is the MITRE grammar, a transformational generative grammar of a reasonably large subset of English. Improvements to the grammar continue to be made: extending the number of sentences it generates, reducing the number of non-sentences it generates, improving its internal consistency, and keeping it in conformity with substantial developments in linguistic theory. Several versions of the MITRE grammar have been issued. The most important change which has been made since the most recent of these versions appeared (Chapin and Geis, 1966) is a new treatment of coordinate conjunction. Other work in progress deals with lexical analysis and the incorporation into the grammar of syntactic fectures.

Coordinate conjunction. The earliest versions of the MITRE grammar contained a set of transformations which attempted to describe coordinate conjunction. The attempt was unsuccessful for a variety of reasons, chief among which was a self-imposed constraint on the form of grammars prohibiting the expression of a condition on a rule that two segments of its structural description be constituents of like type although not dominating identical subtrees. This constraint seems appropriate elsewhere in the grammar, but in coordination it led to the necessity of stating a separate transformation for every possibility of conjunction: a rule for subject NP's, another for object NP's, still another for verbs, and so on. The unwieldy set of transformations which resulted was reminiscent of efforts to generate transformationally derived phrase-makers using only context-sensitive rules: even if an adequate description of the facts could have been achieved, which in fact seemed not to be the case, generalizations were clearly being missed. Therefore, in the summer of 1965, Sanford Schane*undertook an exploration of the possibilities and implications of permitting the expression of such a condition.

The work on coordination is described in detail elsewhere (Schane, 1966); the following remarks summarize the results and describe some of the consequences. The proposed procedure generates conjoint sentences in the phrase structure grammar by a rule schema, as has generally been done in generative accounts, and then after certain transformations have applied (in particular, the passive transformation, in order to generate sentences like 'The Dodgers beat the Giants and were beaten by the Cardinals') searches the trees dominated by the conjoint S's. Schane's <u>principle for conjunction</u> is that any two (or more) trees may be compounded if they are identical except in the subtrees dominated by one grammatical node which dominates some lexical category (noun, verb, or adjective). The resulting compound is identical to the compounded elements down to and including the node dominating the subtree in which they differed. This node dominates the two (or more) differing subtrees, each dominated by the node which originally dominated it, separated by the conjunction(s) which originally separated the conjoint S's. For example, the (simplified) base tree of Figure 12, if left uncompounded, would produce the sentence 'John ran and Mary ran.' Suppose, however, it is decided to compound the tree (compounding is always optional). The two conjoint S's are identical except that one has <u>John</u> where the other has <u>Mary</u>. Dominating both <u>John</u> and <u>Mary</u> is N. N is a lexical category, but does not dominate any lexical category and is thus excluded from conjunction by the principle for

*Schane is now Assistant Professor of Linguistics at the University of California at San Diego.

conjunction. Dominating each of these N's is an NP, which satisfies the principle for conjunction. The two sentences may be compounded, with the subject NP as the compound constituent. The resulting compound will be 'John and Mary ran', with the structure shown in Figure 13.

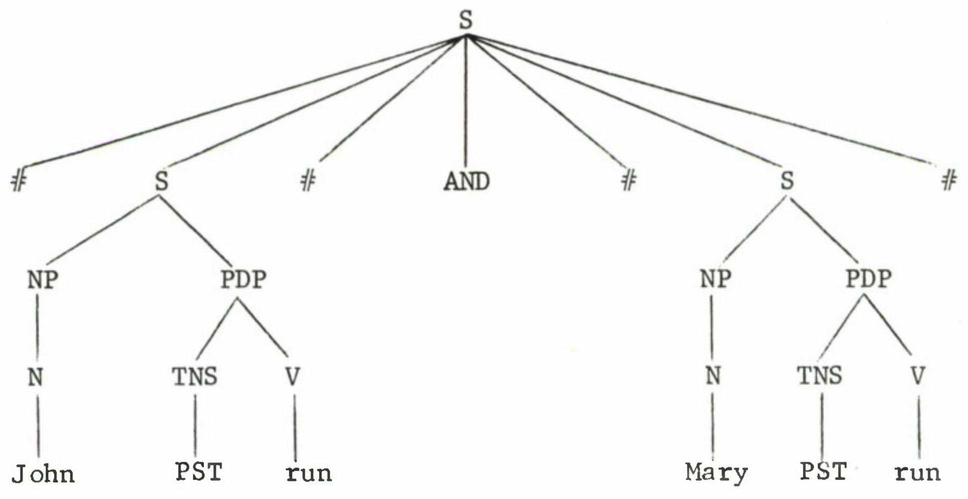


Figure 12. Base tree with conjoint S's

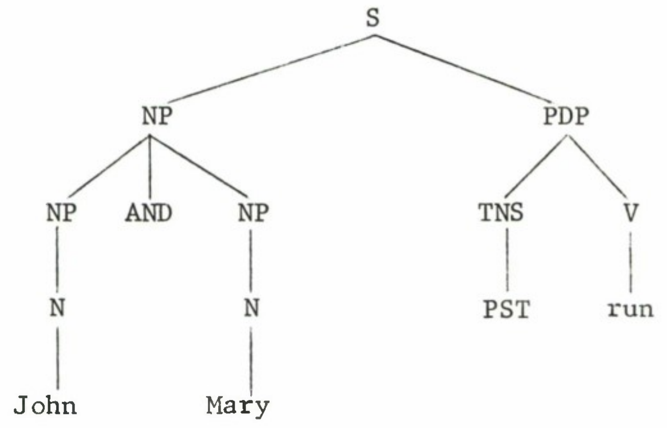


Figure 13. Structure of compound sentence

Perhaps Schane's most original and significant contribution was the hypothesis that only categories above the level of lexical category may be conjoined. This implies that apparent cases of conjunction of lexical items, as in 'The man and woman left', are actually derived by identity deletion of some element or elements of a fuller form, such as 'The man and the woman left.' Certainly these two sentences have the same meaning, but they could of course be derived by two separate compoundings, one of N, the other of NP. However, if compounding is only of the fuller form, certain otherwise strange facts can be readily accounted for. A prime case in point is the conjunction of complicated auxiliaries in English. Observe that the following sentences are all paraphrases:

> The plane could have been landing at eight o'clock and could have been leaving at nine o'clock.
>
> The plane could have been landing at eight o'clock and have been leaving at nine o'clock.
>
> The plane could have been landing at eight o'clock and been leaving at nine o'clock.
>
> The plane could have been landing at eight o'clock and leaving at nine o'clock.

One might say that these four sentences are derived from the same base by four different conjunction rules. In this case all generality is lost from the description of conjunction, since what are being conjoined are not unitary constituents, as in the case of "the man and woman", but rather sequences of constituents of the auxiliary, a different sequence for each of the four paraphrases. But if one says that in fact only one operation of conjunction has taken place, i.e., predicate phrase conjunction, and the shorter sentences are all merely stylistic variants, formed by optional deletion of repeated elements, exactly the right generalization is retained.

This principle for conjunction would also help toward a universal explanation of the conjunction process. In English 'The men and the women' and 'The men and women' are both legitimate compound phrases, but in French the analogous phrase 'Les hommes et les femmes' is possible, while *'Les hommes et femmes' is not. Could there be a special restriction on French, not applicable to English, that common nouns may not be conjoined (observe that proper nouns may be: "Jean et Paul sont ici")? Or is it not more likely that the conjunction process is the same in both languages, French simply lacking a low-level rule permitting deletion of repeated articles in conjoined noun phrases?

The overall MITRE syntactic analysis procedure requires a reversal rule corresponding to each transformation. Reversal of the identity deletion rules to restore the fullest compound form is straightforward. But, just as the rule for conjunction is not an ordinary transformation, differing both in the condition on its applicability and in its creation of an extra node, the reversal of conjunction is a special process which must be applied when the proper analysis for conjunction reversal is met. This process pulls out the compound constituent from the sentence, replacing it with a special marker. It then replicates the sentence with the special marker, so that there are as many instances of the sentence as conjuncts in the compound constituent, and transfers the conjunctions (<u>and</u> or <u>or</u>) from the compound constituent to the positions dividing the S's. The conjuncts are then substituted one by one for the special markers, and the process is complete.

Incorporation of the general treatment of conjunction into the MITRE grammar has motivated some fundamental changes in other parts of the grammar, which are of some linguistic interest. The most important of these has been the expansion of the initial symbol S. Previous versions of the grammar gave a tripartite division of S into noun phrase (NP), auxiliary (AUX) and verb phrase (VP), with an optional initial pre-sentence component (PRE). This division accorded with most extant analyses of English. In the conjunction scheme outlined above, however, it turned out that this analysis was untenable. In verb phrase conjunction in declarative sentences, part or all of the auxiliary is attached to each conjunct verb. Thus we have 'John sang and danced', rather than *'John sang and dance', 'John is singing and dancing' rather than *'John is singing and dance', and so on. This could be accounted for in two ways: by distributing the AUX among the verbs after VP conjunction, or by subsuming the AUX as a constituent of the VP. If the former course is chosen, the generality that only single nodes are conjoined is lost, since one must account for sentences like 'John can sing and will dance' by conjoining the constituent sequence AUX VP. Therefore, the latter decision is motivated, and some evidence provided for the decision of a vexing question of English syntax.

Lexical analysis. The theoretical framework of a more sophisticated lexical analysis procedure is under development. The MITRE lexicon so far has listed every variant form -- plurals, past tenses, etc.--of every word, the lexical lookup procedure being simply a scan of this list. A morphological decomposition procedure is being considered in which certain characteristic affixes and grammatical markers are removed from stems, which are then respelled according to some general rules of English orthography. The respelled stem is looked up in the lexicon. If a match is found, the syntactic characteristics of the stem are checked for appropriateness to the particular affix or marker removed. If this check passes, a lexical reading is attached to the original word which is determined by a combination of the properties of the stem and the affix or marker. If either check fails, the putative affix or marker is reattached to the stem and another pass is made through the lexicon.

Some concrete examples may make this procedure clearer. Consider the words _sincerity_, _charity_, _whiteness_, and _witness_. In _sincerity_ and _charity_, the suffix -_ity_ is discerned and "peeled off". A respelling rule turns the remaining stems into _sincere_ and _chare_. Then a search of the lexicon finds _sincere_ and attaches to it the notation _ADJ_ (among others). _Chare_, however, is not found. Therefore the respelling rule is undone and - _ity_ reattached, giving the original _charity_. This is sent through the lexicon again, where it is found and noted as N. _Sincere_ + -_ity_ is sent through a set of morpheme-combinatorial rules, one of which is of the form ADJ + -_ity_ $\rightarrow$ N. The appropriate notation is made and _sincerity_ is sent to the parser.

In _whiteness_ and _witness_ the suffix -_ness_ is peeled off. No respelling is necessary. _White_ and _wit_ both match lexical entries; _white_ is noted as ADJ and _wit_ as N. One of the morpheme-combinatorial rules is ADJ + -_ness_ $\rightarrow$ N. This rule applies to _white_; it does not apply, however, to _wit_. _Whiteness_ is therefore noted as N, but _-ness_ is reattached to _wit_ and another pass made through the lexicon to find _witness_.

The decomposition rules are ordered in such a way as to permit the removal of several nested affixes and/or markers from a stem. This ordering also serves to prevent a certain number of spurious analyses. For example, the rule removing _-ness_ is ordered before the rule removing _-less_; thus _witlessness_ and _witnessless_ are both correctly analyzed.

No machine implementation has yet been made of these lexical analysis procedures.

Syntactic features. In the MITRE grammar lexical items are treated as unanalyzable grammatical symbols. Grammatical distinctions like those between transitive and intransitive verbs, common and proper nouns, and count and mass nouns are treated as though they were lexical category differences. Thus, _send_ is identified as a member of the lexical category VTR, _arrive_ as a member of the category VINT, and _fly_ as a member of both VTR and VINT.

Recent developments in syntactic theory (Chomsky, 1965) indicate that a more effective way to deal with such distinctions is to treat lexical items as complex symbols composed of syntactic features. For example, verbs like _send_ that are necessarily transitive have the feature (+ ___ NP), indicating that they can occur in a base tree only if they are followed by an (object) NP. Verbs that are optionally transitive (_fly_) have the feature (+ ___ (NP)) where the parentheses around the NP indicate that the verb may be inserted into base trees with or without object NP's. Intransitive verbs (arrive) have the feature (+ ___), indicating that an NP may not follow the verb.

Besides features indicating the different structures into which a given lexical item may fit (called _strict subcategorization features_), the lexical item may be characterized by a set of _inherent features_. For example, common and proper nouns are differentiated by the features (+ common) and (- common).

Similarly, count nouns and mass nouns would have the features (+ count) and (- count).

There are important linguistic motivations for the use of features. There are also implications for syntactic analysis procedures. As a consequence of adopting this approach to the classification of verbs, for example, the number of surface parsings of a sentence containing a verb like fly can be reduced by as much as one-half, since differences between transitive and intransitive verbs are not represented by different surface trees. Moreover, all rejection rules dealing with the transitive-intransitive distinction can be eliminated.

In a similar manner, if differences in grammatical number were treated as a feature difference--say between +SG (= singular) and -SG (= plural)--the number of parsings of a sentence containing a verb whose morphological shape does not indicate number could be reduced by as much as one-half. In the current grammar number is treated as a grammatical category, and, thus, verb forms such as can in 'the general can' and 'the generals can' are structurally ambiguous. Differences in tense can contribute to structural ambiguity in a similar manner and also could be handled with a feature representation.

Features have not yet been incorporated into the analysis procedure. It is clear that their use will reduce the number of surface trees produced by the parsing procedure for many input sentences. It is likely that the reduction in structure should make the application of reversal rules to supertrees somewhat easier. However, there also may prove to be complications in programming the algorithms.

References

Chapin, P.G., and Geis, M. L. The MITRE Grammar (January 1966). MTR-121, MITRE Corporation, 1966.

Chomsky, N. _Aspects of the Theory of Syntax_. Cambridge, Massachusetts: M.I.T. Press, 1965.

English Preprocessor Manual. SR-132, MITRE Corporation, 1964, revised 1965.

Friedman, J. SYNN, An Experimental Analysis Program for Transformational Grammars. WP-229, MITRE Corporation, 1965.

Griffiths, T.V., and Petrick, S. R. On the Relative Efficiencies of Context-Free Grammar Recognizers. _Comm. ACM_, 1965, 8,289-300.

Haines, E. C. The TREET List Processing Language. SR-133, MITRE Corporation, 1965

Kay, M. A General Procedure for Rewriting Strings. Paper presented at the Annual Meeting of The Association for Machine Translation and Computational Linguistics, Bloomington, Indiana, 1964.

Schane, S. A. A Schema for Sentence Coordination. MTP-10, MITRE Corporation, 1966

Zwicky, A. M., Friedman, J., Hall, B. C., and Walker, D. E. The MITRE Syntactic Analysis Procedure for Transformational Grammars. _AFIPS Conference Proceedings: Fall Joint Computer Conference_, 1965, 27, 317-326.

Unclassified

Security Classification

14. KEY WORDS	LINK A		LINK B		LINK C	
	ROLE	WT	ROLE	WT	ROLE	WT
LANGUAGE, LINGUISTICS						
Transformational Grammar						
COMPUTERS						
Programming						
Sentence Analysis Procedures						
Time-Sharing						
On-line Processing						
Displays						

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)* The MITRE Corporation Bedford, Mass.	2a. REPORT SECURITY CLASSIFICATION Unclassified
	2b. GROUP

3. REPORT TITLE
RECENT DEVELOPMENTS IN THE MITRE SYNTACTIC ANALYSIS PROCEDURE

4. DESCRIPTIVE NOTES *(Type of report and inclusive dates)*
N/A

5. AUTHOR(S) *(Last name, first name, initial)*
Walker, Donald E. et al.

6. REPORT DATE September 1966	7a. TOTAL NO. OF PAGES 43	7b. NO. OF REFS 9

8a. CONTRACT OR GRANT NO. AF19(628)-5135	9a. ORIGINATOR'S REPORT NUMBER(S) ESD-TR-66-306
b. PROJECT NO. 7020	
c.	9b. OTHER REPORT NO(S) *(Any other numbers that may be assigned this report)* MTP-11
d.	

10. AVAILABILITY/LIMITATION NOTICES
Distribution of this report is unlimited

11. SUPPLEMENTARY NOTES	12. SPONSORING MILITARY ACTIVITY Deputy for Engineering & Technology, Directorate of Computers; Electronic Systems Division, L. G. Hanscom Field, Bedford, Mass.

13. ABSTRACT

The MITRE syntactic analysis procedure for transformational grammars has been used to process sentences on-line in a display-oriented mode as well as off-line. This report describes additions to the program structure and the grammar made since the last report and presents the results of experiments with the procedure.

DD FORM 1 JAN 64 1473