University of Pennsylvania
TRANSFORMATIONS AND DISCOURSE ANALYSIS PROJECTS

27. Procedure for Left-to-Right Recognition of Sentence Structure

Naomi Sager

I. Introduction[1]

1. The string structure of sentences

The program described in this paper takes a particular theory of language structure (substring theory[2]) and recognizes a sentence as an instance of that structure. This constitutes grammatical analysis of the sentence. Preliminary to the analysis, the sequence of words of a given sentence is represented by a sequence of sentence symbols indicating the grammatical class and subclass memberships of the sentence words, such as N for noun, V for verb. (The classification symbols are listed on p. 35, Appendix A1). Some words have more than one classification, e.g., increase (N/V), and certain sequences of words (called word complexes) have a single classification for the whole sequence, e.g., because of (P).[3]

The sequence of sentence symbols which represent the successive words of a sentence will be called a sentence formula, e.g., TNWV for The growth may begin. If the i^{th} word (or word complex) in the sentence has m classifications the formula will have in its i^{th} position a choice of m symbols, e.g., TNW N/V for The growth may increase. Such a formula will be treated as a family of m formulas which are identical except for having different choices in the i^{th} position: TNWN and TNWV. A formula or any other sequence of sentence symbols (one which represents a proper part of a sentence), which contains no multiple choice of sentence symbols at any position will be called definite.

The substring theory of language structure associates with certain definite sentence formulas one or more groupings of the sentence symbols, expressible by placing brackets with various identifications at certain points among the symbols. These points are the boundaries of groupings which will be identified below as occurrences of strings. Each way of placing the brackets within a formula according to some general rules will be called an <u>analysis</u> of that formula, and of the sentences which that formula represents. Formulas with such an analysis are recognized as being grammatical. In the example above, the representation TNW N/V gave rise to two definite formulas, of which TNWV was analyzable, hence grammatical, while TNWN was not.

In the substring theory, grammatical sentence formulas are described recursively as follows:

1. The theory lists axiomatically certain definite sequences of sentence symbols called (axiomatic) <u>strings</u>. There are one or more distinguished axiomatic strings called <u>center strings</u>, which are grammatical sentence formulas. (In the present program only the major center string of English, 'NV', is considered; the others are closely related to it.)

2. The theory lists rules of <u>replacement</u> of the following form: The result of replacing the occurrence of a certain symbol in a grammatical sentence formula by one of a specified list of axiomatic strings is a grammatical sentence formula. (It is understood that a replacing string is grouped with the rest of the sentence formula exactly as the replaced symbol was.)

3. The theory lists rules of <u>adjunction</u> of the following form: The result of adjoining one of a specified list of axiomatic strings to an occurrence of a certain symbol in a grammatical sentence

formula is a grammatical sentence formula. (Note: Adjunction could be defined as replacement where the replacing string contains the replaced symbol, e.g., $x/x^\frown y$.)

4. In addition to this recursive characterization, there are certain dependences, i.e., cases in which the occurrence of a particular symbol in a grammatical sentence formula restricts the occurrence of a specified symbol at a distance stateable in terms of combinations of string membership. (Example: subject-verb agreement in number).

5. English and many other languages have the property (given by Rules 1-3) that if a string is included in another string, it is wholly included, including its adjuncts (adjoined strings). E.g., in: We saw a book in the library entitled "Ghosts" on 42nd St., the intervention of entitled Ghosts as an adjunct of book between library and its adjunct on 42nd St. would be ungrammatical [4]

A distinction should be made between a string and the occurrence of a string in a sentence formula. In a sentence formula, certain sequences of symbol occurrences are not strings, even if the corresponding sequence of symbols is a string. E.g., in the sentence:

```
        (1)  Light from Mars was analyzed.
               N     P    N   V+    S
```

the sequence N V^+S (Mars was analyzed) would not be an occurrence of a string, although N V^+ S is an axiomatic string in a substring grammar of English.

The occurrence of a string in a sentence formula is determined by application of Rules 1-4 above.[5]

A grammatical sentence formula is then composed of string occurrences satisfying the above rules. Let the occurrence of a string in a sentence be enclosed in a pair of brackets.* Each bracketed string occurrence in a sentence formula has a depth of parenthesization associated with it which is given by the excess of left brackets (including its own left bracket) over right brackets to the left of the 1st symbol of the string, up to but not including the opening bracket of the center string. In the course of computation from left to right, a depth is assigned to each encountered string. In some cases, the depth is relative, and the actual value is determined later in the analysis.

2. Left-to-right recognition of string composition.

A substring grammar is conveniently arranged for left-to-right recognition in the following way.

1) We list the axiomatic strings, each identified by its string head (i.e., its leftmost symbol). If more than one string begins with the same string head, the heads of the different strings are differentiated by lower-left subscripts: e.g., $H\ N\ V^+$ is written ${}_1H\ N\ V^+$ with head ${}_1H$, while $H\ N\ V^-$ is written ${}_2H\ N\ V^-$ with head ${}_2H$. (See the strings in Appendix A-2, pp. 37-41).

2) For each symbol of the alphabet of sentence symbols, we list the strings which replace or adjoin that symbol in some sentence of the language, e.g., for N we list replacers of N, left adjuncts of N, right adjuncts of N; for V, we list left adjuncts of V, right adjuncts of V. The lists for some symbols may be empty. (See the lists in Appendix A-3, p. 42.)[6]

---

* The types of brackets used are described on p. 10.

3) We list dependence instructions which are associated with a particular symbol either in all its occurrences (i.e., as a sentence symbol) or as a member of a particular string; these instructions operate on some other symbol which does not precede the given symbol in the string in which it occurs.

On this basis, it is possible to construct for the rth member of every string, a <u>successor list</u> which consists of the r+1th member (if there is one) of the same string, and every string head which occurs immediately following the rth member in some sentence of the language.

E.g., the successor list of N of #N V$^+$ consists of:

1. V, the next member of the #N V$^+$ string
2. <u>lav</u>, the list of heads of left adjunct strings of V
3. <u>sa</u>, " " " " " sentence adjunct strings which may follow the subject of a sentence.
4. <u>ran</u>, " " " " " right adjunct strings of N

For example, in the following four sentences the successors of N (<u>light</u>) of #N V$^+$ are from 1 - 4 above (in the same order).

(1) <u>Light travels with velocity c.</u>
successor: V (<u>travels</u>)

(2) <u>Light always travels with velocity c.</u>
successor: D (<u>always</u>)

(3) <u>The light, moreover, is monochromatic.</u>
successor: & (<u>moreover</u>)

(4) <u>Light from Mars was analyzed.</u>
successor: P (<u>from</u>)

The successor list of the r-1th member of a string will be called the <u>position list</u> at the rth position in that string.

In analyzing a sentence formula from left to right, when we encounter a particular string head, we open a bracket for that string, thus assigning a particular depth to that string (see the end of II., p. 4). Until all the members of the string (and their adjuncts) have been recognized in the sentence formula we can say that we are in that string. It follows from Rules 1-3 in 1.1 that at every point in the analysis of a formula from left to right we are in at least one string; in particular, we are in all the strings for which opening brackets have been placed and closing brackets have not yet been placed, and we are at the position in each string (call this the active position) following that of the string member which was most recently recognized in the formula.

We now have a basis for analyzing each successive sentence symbol in terms of the analysis to the left of that symbol. Knowing the strings which are in progress at a given point in the sentence formula, and knowing where we are in each of these strings, we can state that the sentence symbol at the given point must be either the string member or a string head in the position list at the active position of one of the strings in progress.

So we set up a combined list for the n^{th} sentence symbol which consists of all the active position lists which apply at this point in the analysis of the sentence formula, and compare the n^{th} sentence symbol with this list. The symbol(s) in the combined list which have the same form (i.e., N, V, P, etc.) as the sentence symbol are said to match the sentence symbol. The symbols in the combined list are specified as to whether they are string members or heads of particular strings and also as to depth. For each match, then, we have an analysis of the sentence symbol, i.e., it is the head of such and such a string at a particular depth of parenthesization, or it is the next string member of a string at a particular depth, etc., which is the desired output.

3. Application to the sentences of English

This program analyses English sentences within the framework of a symbol-by-symbol, left-to-right process. A symbol-by-symbol recognition process can be applied to language only on the basis of a recursive theory of sentence composition. The use of a left-to-right process is invited by certain features of English (and of various other languages), chiefly:

1. In the list of strings, the number of strings which adjoin to the right is greater than the number of strings which adjoin to the left.

2. Most right adjunct strings have on their left, as a string head, an introducer which indicates any special dependences that the string may have; i.e., the string head is a convenient identifier of the string. Furthermore, most adjuncts can be identified by their leftmost element (string head) more conveniently than by their rightmost element, (e.g., NV⁻ strings have as their leftmost element, N or its left adjuncts or its replacers, but as their rightmost element, whatever ends the reduced object of the particular V).

The features of English which are not naturally described by a left-to-right process are restricted in such a way as to make it reasonably easy to fit them into such a process.

3. Left adjunct strings are short, in almost all cases consisting of one symbol.

4. Replacement strings occur only in the place of the symbol they replace.

5. Whereas all adjunctions and replacements of strings can be treated as restrictions on (or permissions of) symbol occurrence in the next or same position in the string, the other dependences (Rule 4 in I,1) operate

between symbols at a distance. These dependences are relations between two or more string members and their lists. We adjust the recognition of the dependences to a left-to-right process by identifying the leftmost participant in the relation, and if it is matched, sending an instruction to the later participants, which will be carried out if the later participant is matched. The addressing of the instruction is a relatively simple matter because the later participant is always a later member of the same string or a member of an adjunct of that string.

6. Consider a left adjunct or replacement string Y whose first symbol is identical with the symbol it adjoins or replaces, e.g., D as left adjunct of D, N G^+ as replacement of N. When we meet D in a position where D is in the list, we do not know if this is the D of the list or a left adjunct of that D, with the D of the list still to come. For such strings Y, we cannot specify the final depth of parenthesization until we reach, later to the right, the adjoined string member or the member which follows the replaced symbol. We assign to such strings Y, the least depth required by the computation up to that point and increase this depth if we find, later to the right, symbols which show that the string Y had been a left adjunct or replacement string. The contiguity of adjunction and the non-intercalation of strings (Rules 1-3 pp. 2-3) enables us to say that after we have matched the string member adjoined by Y or the string member which follows the member replaced by Y, we will no longer meet any grounds for increasing the depth of Y. Hence, from this point on in the computation the depth of Y is absolute.

This program analyzes not some simple part of the set of English sentences, but all sentences of English which are described by a substring grammar. If the substring grammar is more detailed, the program will be able to analyze additional sentences; and if changes or corrections are made in the grammar, the program will reflect them, as soon as the changes are put in the form of revised strings, lists, and dependence instructions.

The program is thus a direct method of sentence structure recognition for a substring grammar. Its basic method would apply for substring grammars of other languages, though additional operations may have to be developed for more complex left adjuncts or for other types of dependence than those met within English.

II. Main Procedure

1. Form of the output

The output displays a substring analysis of the sentence, e.g.

Sentence: What follows now is a history of the evolution of living creatures.

Output:* $\langle NV^{+}\ ;_{rn}[K_4V^{+};$ what follows $_{sa}(D;$ now$)]$ is $\{N;_{lan}(T;$ a$)$ history

$_{ran}(PN;$ of $_{lan}(T;$ the$)$ evolution $_{ran}(PN;$ of $_{lan}(G;$ living$)$ creatures$))\}\ \rangle$.

Four types of brackets are used:

$\langle\ \rangle$ center string

$_x(\ )$ adjunct string

$_x[\ ]$ replacement string

$\{\ \}$ object string

The subscript x specifies the type of string and the symbol replaced or adjoined (where applicable):

x	type of string
lan	left adjunct of N
lav	left adjunct of V
laa	left adjunct of A
ran	right adjunct of N
rav	right adjunct of V
raa	right adjunct of A
rn	replacement string of N
sa	sentence adjunct

These are also the names of the lists of string heads given in Appendix A3. (Lower case letters are used to distinguish names of lists from sentence symbols). Sentence adjuncts adjoin the string in the next inclusive bracket pair; adjuncts of the symbol X adjoin the word whose symbol is X, or the X -replacement string (whichever occurs) in the next inclusive bracket pair.

*In the output, center string $\#NV^{+}$ is written NV^{+}.

Inside any bracket pair,[7] the first element, up to semicolon, is the name of the string and consists of a sequence of symbols which are the class names of the members of the string. These symbols are in one to one correspondence with the words and replacement strings after the semicolon (disregarding included adjuncts) inside the bracket pair. There are several exceptions to the one to one correspondence, as follows:

1) In the case of the verbs BE, HAVE, DO, in such constructions as <u>are going</u>, <u>have gone</u>, <u>did go</u>, the verb following BE, HAVE, or DO is not separately bracketed, e.g., not <u>are {going}</u>.[10,11] Hence in these cases there is more than one word corresponding to the V in the string name. Since V always occurs as the rightmost string member, if the output is read from left to right, the word(s) corresponding to V can be identified.

2) Left adjuncts of N which begin with N are, for the moment, grouped into one N or NA bracket e.g., (N;vacuum tube) diodes.

3) The words (or replacement strings) which follow the conjunction C repeat a segment of the string and hence correspond to a repetiéion of a segment of the string name. (See III 5.)

Further examples of output are:

Our present physical knowledge leaves us even more uncertain about the equivalence or non-equivalence of positive and negative electricity.

Output: (NV^{+}; $_{lan}$(A;our) $_{lan}$(A;present) $_{lan}$(A;physical) knowledge leaves {NA;us $_{laa}$(D; $_{laa}$(D;even) more) uncertain $_{raa}$(PN;about $_{lan}$(T;the) equivalence or non-equivalence $_{ran}$(PN;of $_{lan}$(A;positive) and $_{lan}$(A;negative) electricity))})*

*Where the program produces more than one analysis of a given sentence only one of the possible outputs is shown in the example. E.g., another analysis of the last bracket is: ... $_{ran}$(PN;of $_{lan}$(A;positive and negative) ... . (5.1 OUTPUT 3a). Cf. IV.

More recent achievements in neurophysiology have been the development of hypotheses relating to the events occurring during the propagation of an impulse over the surface of the nerve cell and its branches, and to the events occurring at excitatory and inhibitory synapses.

Output: (NV$^+$;$_{lan}$(A;$_{laa}$(D;more)recent)achievements$_{ran}$(PN;in neurophysiology)
have been {N;$_{lan}$(T;the)development$_{ran}$(PN;of hypotheses$_{ran}$(G$^+$;
relating {P$_{to}$ N;to$_{lan}$(T;the)events$_{ran}$(G$^+$; occurring$_{rav}$(PN;during
$_{lan}$(T;the)propagation$_{ran}$(PN;of$_{lan}$(T;an)impulse)$_{ran}$(PN;over$_{lan}$(T;the)
surface$_{ran}$(PN;of$_{lan}$(T;the)$_{lan}$(N;nerve)cell and$_{lan}$(A;its)branches)))!}
and {P$_{to}$ N;to$_{lan}$(T;the)events$_{ran}$(G$^+$;occurring$_{rav}$(PN;at$_{lan}$(A;excitatory)
and$_{lan}$(A;inhibitory)synapses))}))})

2. <u>Position lists of a string</u>

2.1 With each string is associated a sequence of lists which has a format composed as follows:

1. The members of the string (obtained by looking up the string head in a table of strings) are placed in the top sections of successive positions, with one empty position provided to the right of the last member.
2. Lists of string heads (obtained by looking up string members in the appropriate string head tables) are added in sections at each position, in the following order:

Left adjunct heads	→	position		of the calling member*
Replacement string heads	→	"		" " " "
Sentence adjunct heads	→	"	to the right	" " " "
Right adjunct heads	→	"	" " "	" " " "

*E.g., for the left adjuncts of N, N is the calling member.

Each position but the last now contains an ordered list consisting of a string member and zero or more sections of string heads; the last position contains only a list of zero or more sections of string heads. For a particular string, the sequence of lists at positions, as defined by 1. and 2. above, are <u>the position lists of the string</u>.

E.g., the position lists of the string # N V^+ are:

#	N	V^+	
	lan	lav	sa
	rn	sa	rav
	sa	ran	

2.2 In assembling position lists of a string, if Y is in a list and Y has left adjuncts and/or replacement strings, we call the lists of left adjunct and/or replacement string heads <u>sublists of Y</u> and we add these string heads to the list containing Y, with (Y) added to each symbol in the sublist. The same holds for string heads in the sublist which have left adjuncts and/or replacement strings, so that we may get entries in a list of the form $Y_1\ (Y_2(Y_3\ldots))$ where Y_1 is in a sublist of Y_2, which is in a sublist of Y_3, etc. The OUTPUT and SIP procedures treat the parenthesized string heads to give e.g., the output $_{ran}(NV^-;\ _{lan}(\tilde{A};\ _{laa}(D;$ for the list entry $D(\tilde{A}(_4N))$. (See II 4.3, Step 3d; II 4.4, Step 3f, and the example, pp. 49-50.) To prevent endless sublisting, symbols are not allowed to appear on adjunct or replacement lists of themselves. In particular

a) N in <u>rn</u> cannot be in a sublist of another N in <u>rn</u>, e.g., not $_7N(_7N)$ [$_7N$ is the head of replacement string $_7NG^+$]

b) $\tilde{A}$ cannot be listed under $\tilde{A}$, i.e., not $\tilde{A}(\tilde{A})$

c) D " " " D, i.e., not D(D)

These cases, i.e., constructed like a), b), and c) above, are handled by special "push down" instructions. (See III 4.)

3. Strings in progress

The information required at any point in the analysis is stored in a stack of registers, called Strings in Progress, SIP(k), where each SIP contains the position lists of a string for which a bracket pair at depth k has been opened but not yet closed. SIP(o) is reserved for the center string; thereafter, when a string head in a position list in SIP(k) is matched, the position lists of the string whose head was matched are copied into SIP(k+1). If the string head is in a sublist, there are additional instructions (described in 4.3, below).

When the string member in a position list is matched, the list is marked with a star. We define the active list of SIP(k), M(k), to be the leftmost unstarred position list in SIP(k).

4. The computation of the nth sentence symbol

The computation of the nth sentence symbol (N>O) consists of procedures 4.1-4. First, certain of the active lists from the current SIP(k) are assembled into a combined list for the nth sentence symbol; (4.1, below). Then, the sentence symbol is compared with this list and some symbol in the list is matched (otherwise this particular analysis of the sentence fails) (4.2, below). Knowing the depth of the position list containing the matched symbol and whether the symbol was a string member or string head, the OUTPUT procedure closes and opens various brackets (4.3) and the SIP procedure erases or makes changes in existing SIPs and/or brings in the position lists for new SIPs (4.4). For n)= o, 4.3 - 4 are used with the stated initial conditions (4.5)

LIST:

4.1 ⟨Given that the n-1th sentence symbol ($n > o$) was analyzed and assigned the depth k, <u>the combined list for the nth sentence symbol</u> contains:

1. M(k), the current active list in SIP(k), and

2. if the string member section of M(k) is empty, then 1. and 2. with $k-1 \rightarrow k$.

3. C and CC (See III 5.)

MATCH:

4.2 ⟨The nth sentence symbol is compared with the combined list.

If no symbol in the list is matched, this particular analysis is dropped.

If more than one symbol in the list is matched, the following procedures apply separately for each match. (This case is discussed in Part IV).

For each match, the matched symbol is in some M(k). Let this $k = \underline{k}$.

4.3 <u>OUTPUT Procedure</u>: carried out after 4.4.

Brackets determined by the match of the nth sentence symbol are placed in the sentence to the left of the sentence word (or word complex) corresponding to the nth sentence symbol.

STEP 1. For $k > \underline{k}$: brackets are closed.

These strings are complete; no more adjuncts are possible since the sentence has returned to their including string.

STEP 2. If the matched symbol is a string member: no opening bracket is placed.

STEP 3. If the matched symbol is a string head:

a) the appropriate bracket is opened.

b) the "name" of the string (i.e., the symbols of the string obtained by looking up the string head in the table of strings), followed by a semicolon, is written to the right of the opening bracket.

c) the name of the list in which the matched string head appeared is specified on the outside lower left of the opening bracket.

d) in addition, if the matched string head is in a sublist, i.e., the symbol Y_1, in a list entry of the form $Y_1(Y_2(Y_3...))$:

Step 3 is repeated for each parenthesized symbol reading from left to right; for each parenthesized symbol the opening bracket (with name, etc.) is placed to the left of all other opening brackets due to the previous application of 2d for this match.

(The application of OUTPUT step 3d is shown in the computation of <u>fully</u>, Note 3 on p. 49 and note 1 on p. 50.)

4.4 <u>SIP Procedure</u>

STEP 1. For $k > \underline{k}$: SIP(k) are dropped.

STEP 2. If the matched symbol is a string member: M($\underline{k}$), the position list containing the matched string member, is marked with a star.

As far as the main procedure is concerned, this is equivalent to erasing the list. Starred lists are used in computing strings following C.

STEP 3. If the matched symbol is a string head:

a) a new SIP($k = \underline{k}+1$) is set up, containing the position lists of the string;

b) the zero position of the new SIP is starred;

c) the name of the list in which the matched string head appeared is noted in the zero position of the new SIP (See III 5.1 CONJ, LIST 2

In addition:

d) if the matched string head is the head of a replacement string, the list at the position of the replaced symbol, M($\underline{k}$), is starred;

This is similar to the action for a string member in STEP 2, since the occurrence of a replacement string satisfies the requirement for the string member it replaces.

e) If the matched string head is the head of an object string:

the list of object strings is blocked in M(<u>k</u>).*

Object strings are not repeatable.

f) If the matched string head is in a sublist, i.e., Y_1 in a list entry of the form $Y_1(Y_2(Y_3...))$:

STEP 3 (except b) is repeated for each parenthesized symbol, reading from left to right; at each application, the SIPs set up for parenthesized symbols to the left in the entry, are "pushed down," i.e., SIP(<u>k</u>+1) → SIP(<u>k</u>+2) → SIP(<u>k</u>+3), etc.

E.g., If the matched symbol is $D(\bar{A}({}_4N))$ in the active list of SIP(o), after applying 3f we have:

SIP(1)	N	V^-	
	lan	lav	sa
	rn	sa	rav
		ran	
SIP(2)	$\bar{A}({}_4N)$		
	$laa({}_4\bar{A}({}_4N)$		raa
	$D(\bar{A}({}_4N)$		
SIP(3)	$\overset{\#}{D}(\bar{A}({}_4N))$		△

4.5 <u>The computation at n = o</u>

At the beginning of the analysis (n = o) we set up the requirement for a center string and open the center string bracket-pair in the output. It is possible to do this with the usual OUTPUT and SIP Procedures (4.3 and 4.4) since #, the sentence initial mark, has been defined to be the head of the center string, $\#NV^+$. If we set <u>k</u> = -1, # will set up SIP (<u>k</u> + 1 = o). The conditions at n = o, therefore, are:

1) <u>k</u> = -1

2) # is a string head

3) SKIP 4.1 and 4.2 (for n = o).

*"Blocking" is defined on the top of page 19.

III. Dependence Instructions

In addition to the basic apparatus for inserting a string next to particular symbols of another string, languages commonly have dependences, i.e., the computation of a symbol at one point of a string is affected by the occurrence of a particular symbol at another point in a string (e.g., Life is ... and The life is ..., but only The book is ... and not Book is ...). Since these dependences occur (in English) only between a member of a string and other members or adjuncts of the same string, a dependence-instruction carried by a symbol in SIP(k) will make a suitable change in some position list of that same SIP(k) or a SIP set up by a string head in a position list in SIP(k).

The dependence instructions are executed after the SIP Procedure and before the OUTPUT Procedure. Their point of application is referred to M(k), the position list which contains the matched symbol. However, when an instruction is carried by a symbol X in a sublist of Y*, the point of application of the instruction is referred to the position list which contains Y as a string member in the SIP set up for Y, instead of to M(k).

Instructions may be carried by a symbol either in its occurrence as a sentence symbol or as a symbol in the list. When the sentence symbol and the symbol in the list which matches it both carry instructions, the instructions carried by the sentence symbol are executed first. Where

*X being in a sublist of Y indicates that X and Y open brackets at the same point in the sentence formula, and that the string headed by X is included in the string headed by Y. Step 3f of SIP Proc. (II, 4.4) will set up SIPs for X and Y at the appropriate depth, with the depth of X being one greater than the depth of Y.

several instructions are carried by a symbol, they are executed in the order of listing (from left to right).

An important operation in many instructions is blocking. When a symbol in a position list in a SIP is blocked, it is ignored in further computations involving this list in this SIP.

1. Partial ordering

Sections of a position list and subsections of a section are arranged in "blocking order," i.e., when a sentence symbol matches a symbol in a position list, all the subsections and sections below the matched symbol in the list are blocked.

The blocking relation of sections and subsections handles the partial ordering of adjuncts with respect to the relation: order of occurrence in the sentence formula, e.g., some fine lines but not fine some lines, people who travel sometimes find... but not people sometimes who travel find... The examples in Appendix B illustrate how sectional blocking works. (See p. 45, bot., N. 1; p. 46, top, N.1; p. 46, bot., N.1; p. 48, N.1.)

2. Verb Object[8]

Instructions:	OBJ +
	OBJ -
	OBJ P
	OBJ S
	OBJ B
	OBJ E
	TENS

Associated with each particular verb is a set of object strings and each time the verb occurs, it occurs with one or another of its associated object strings. One of the object strings may be a zero (e.g., for exists). When a verb in the sentence is matched, its object strings are placed in some register, henceforth called the <u>object register</u>. The first symbol of each string in the object register is an <u>object string head</u>. E.g., for the verb <u>influence</u>, the stored object strings are <u>N</u> and <u>N to V^+</u> and the list of the corresponding string heads is:

N'
lan
rn
N''
lan
rn

(See for example the contents of the object register on p. 49, and note 3 on p. 49).

Instruction OBJ+ brings the object strings for the particular verb into the computation. OBJ- causes an N to be omitted from certain object strings when they are included in particular strings, e.g., certain k and H strings. OBJP covers omission in PK strings and OBJS omission due to the passive. OBJB and OBJE adjust the output for <u>be</u>, <u>have</u>, <u>do</u> TENS covers tense agreement.

2.1 <u>Instruction OBJ+</u>

(OBJ+ is carried by certain V or G or S. In the lists (pp. 37-39) and examples (pp. 45-60), a V or G or S which carries OBJ+ is written V^+, G^+, S^+ resp.)

1. If the list of object strings heads, <u>os</u>, contains no ∅ (zero object), the list is placed in the string member section of the position list to the right of M(<u>k</u>).

   With the string member section not empty, the list for the next sentence symbol will be this position list. (II 4.1, p. 15)

2. If <u>os</u> contains ∅, the ∅ is removed and the list is placed just below the string member section of the position list to the right of M(<u>k</u>).

If the object string is zero, the string containing V^+ or G^+ or S^+ is completed by the V or G or S itself and the correct list for the next sentence symbol is obtained by considering that there is no string member in the position to the right of M(_k_), i.e., the assembly of the combined list will not stop at this position list, but continue with $k - 1 \rightarrow k$. (II 4.1) (A ∅ object string occurs in the computation of _started_ on pages 55-56. (See note 5 on p. 56.)

2.2 _Instruction OBJ-_

(OBJ- is carried by certain V or G or S. In the examples, $V^- = V$ carrying OBJ- ; $G^- = G$ carrying OBJ- ; $S^- = S$ carrying OBJ-[9])

1. For each N occurring in each object string, the object string is rewritten in the object register without that N,

   e.g., N to $V^+ \rightarrow$ to V^+

   $N \rightarrow \emptyset$

   N P N $\rightarrow$ N P and P N

2. If a V^+ or G^+ or S^+ occurs within an object string (in its original state, not the resultant of 1.), the string is rewritten with - in place of +,

   e.g., N to $V^+ \rightarrow$ N to V^- by 2

   N to $V^+ \rightarrow$ to V^+ by 1

3. If no N or V^+ occurs the string is dropped from the register.

4. Perform OBJ+ .

Instruction OBJ- therefore consists of a) modifying the object strings and b) performing instruction OBJ+. N-adjunct strings containing a verb are similar in most cases to the $\#NV^+$ center string except that one N is omitted -- either the subject, or an N of the object: _The monkey which saw us_, _The monkey which we saw_. Omission of the subject is treated by listing different strings, sometimes two for the same string head, e.g., for K_3(_which_): ${}_1K_3V^+$, ${}_2K_3NV^-$. Omission in the object string is handled by OBJ- . (See notes 4 and 5 in the computation of _started_ on p. 56.)

If the verb object string contains a verb, the N may be omitted from the object string of the contained verb (step 2). E.g., in The children whom I want to tell to go home, the N is omitted from the object string N to V^+ of tell, contained in the object string to V^+ of want.

2.3 Instruction OBJP

(OBJP is carried by V or G or S in strings which begin with PK; in the strings (p. 37-39) and example (p. 59), V^P or G^P or S^P stand for V or G or S with OBJP.)

1. For each object string which contains V^+ or G^+ or S^+, $V^+ \rightarrow V^P$, $G^+ \rightarrow G^P$, $S^+ \rightarrow S^P$ in the object register.

2. For each object string which contains a specified (sentence word) P, compare this P with the P (sentence word) which heads the string containing V^P. (The latter is obtained from the output or will be stored in the SIP when a $2\text{-}10^P$ is matched.)

   Match: Perform step 1 and 3 of OBJ-, with PN → N in steps 1. and 3. of OBJ- .

   No Match: go immediately to step 3 of OBJP

3. Perform OBJ+

E.g., in The people of whom I spoke, the P(sentence word), of, specified in the object string of N of the verb speak, matches the P(sentence word), of, which heads the PK_2NV^P string in the sentence formula. Hence, by step 2, using the modified OBJ-, object string of N → Ø and we accept spoke, instead of, e.g., spoke of him. (OBJP is used on p. 59, top, N. 3-4.)

2.4 Instruction OBJS

(OBJS is carried by sentence symbol S and is executed when S is matched.)

1. If S occurred in the list lan, erase the object register and skip 2-4.*

2. Object strings which do not begin with N or PN (except HNV^+) are dropped from the object register.

3. The 1st N is dropped from the remaining strings (except in HNV^+).

4. Then perform OBJ+, OBJ-, or OBJP whichever is the case.

*As a left adjunct of N (e.g., privileged), S is not treated as a verb.

E.g., in This was attributed to him, the object PN(to him) came from the object N^{P}to N of attribute. (See the computation of interrupted p. 60, N. 3.)

2.5 OBJb[10,11]

OBJB is carried by $V_{(b)}$, the verbs be and have and do. When the sentence symbol representation of one of these verbs is matched, OBJB is executed.)

1. Whatever instruction was attached to the matched V or G or S in the list is transferred to each $V^{E,+}$ or $G^{E,+}$ or $S^{E,+}$ in the object register in place of + (OBJ+).
2. Perform OBJ+ (for the matched V or G or S).

(OBJB is used in the computation of were and was in the example in Appendix B2. See p. 58, bot., N. 3-4 and p. 59, bot., N. 3-4)

2.6 Instruction OBJE[10,11,12]

(OBJE is carried by V or G or S or to when one of these symbols is the first symbol of an object string of $V_{(b)}$; OBJE is indicated by writing V^E or G^E or S^E or to^E.)

1. Step 3 of the OUTPUT and SIP procedures are blocked for this match.
2. M(k) is starred and the lists sa and rav are moved (added) to the position list to the right of M(k). If the matched symbol is to^E (of to^E $V^{E,\ldots}$), $V^{E,\ldots}$ is placed in the string member section of the position list to the right of M(k).

The operation of OBJE in conjunction with OBJB removes the brackets which would have made, e.g., G(building), the object of V(is), in He is building the apparatus. (OBJE is used in the computation of waiting and interrupted, p. 59, top, N. 1-2; p. 60, top, N. 1-2.)

2.7 Instruction TENS[11]

(TENS is carried by sentence symbol W and is executed when W is matched.)

W blocks Vs and Vp in M(k)

TENS insures that we will not have, e.g., he may goes or he will went. (See p. 48, N. 4.)

3. Noun Dependences[8]

Instructions: NMBR
CNTN
CPDN
CPDR
LFTE
PRON
PRDN
SUBJ
GOFN

Instruction NMBR treats subject-verb number agreement. CNTN applies the restriction that certain nouns (called count nouns) require a preceding article. CPDN and CPNR adjust the bracketing of compound nouns. LFTE handles the left elements which are non-repeatable. PRON and PRDN block (or release a block on) certain adjuncts following pronouns and certain nouns (called predicate nouns). SUBJ insures that certain replacement strings occur only in the subject position. GOFN(G of N) sets up the object string in replacement strings of the form: N's G of N or N's G of NPN, e.g., John's wiring of radios..., peoples' basing of conclusions on limited evidence... .

3.1 Instruction NMBR

1. If N or a symbol in rn is matched, and if the string member section of the position list to the right of M(k) (of M(k +1) if N was a string head) contains V_o or V_s, then

   a) N_1 or N_{sg} or a symbol in rn (not R_{pl} or T_2) blocks V_o

   b) N_{pl} or R_{pl} blocks V_s

2. The match of W releases the block on V_o (if there is one).

> Thus we can obtain <u>The man goes</u>, <u>The men go</u>. <u>The man will go</u>, but not <u>The man go</u> and not <u>The men goes</u>. (See p. 47, bot., N. 2; p. 54, top, N.3; p. 55, N. 4.)

3.2 Instruction CNTN

(CNTN is carried by sentence symbols T, B, and is executed when T or B is matched. String member N_1 is normally blocked.)

The match of T or B releases the block on string member N_1 in M(<u>k</u>).

> This permits <u>Life is ....</u>, <u>The life is ....</u>, <u>The book is ....</u>, but not <u>Book is ...</u> (CNTN is used in computing <u>the experiment</u>, pp. 53-4; see p. 53, N. 5.)

3.3 Instruction CPDN

(CPDN is carried by ${}_1N$ and ${}_2N$ in the list of left adjuncts of N.)

1. The OUTPUT procedure of the <u>next</u> match is blocked.

2. Instruction CPDR is attached to string member N in M(<u>k</u>).

3.4 Instruction CPDR

(CPDN associates CPDR with string member N)

The block on the OUTPUT procedure due to CPDN is released.

> As a result of 3.3 and 3.4, the left adjunct section of a compound noun is placed into a single parenthesis: <u>(book) burnings</u>, <u>(electron beam focussing) system</u>. Related instructions have to be stated for compound adjectives, e.g., <u>(worry)free</u>. (CPDN and CPDR are used in the <u>computation of radio astronomy stations</u>, p. 46, bot., N 5; p. 47, top, N. 2; p. 47, bot., N. 3.)

3.5 Instruction LFTE[13]

(LFTE is carried by sentence symbols T and B and T_q.)

The match of T or B blocks T and B in M(<u>k</u>)

" " " T_q " T_q " " " "

Thus we can have AAN(<u>bright industrious students</u>) but not TTN(<u>The the students</u>). (LFTE is used in computing <u>the</u>, p. 53, bot., N.3.

3.6 Instruction PRON

(PRON is carried by R in the list of replacement string heads of N.)

All the right adjuncts of N except those which begin with H, K, $_3N$, $_5N$ are blocked in the position list to the right of M(<u>k</u>).

Thus we can have <u>The man who thinks</u>, <u>He who thinks</u>, <u>The man on the street</u>, but not <u>He on the street</u>.[14] (PRON is used in the computation of <u>we</u> on the top of p.55; see note 6.)

3.7 Instruction PRDN

(PRDN is carried by sentence symbol N_a.) The right adjuncts of N which begin with $_1For$, $_1H$, or $_6N$ are normally blocked.)

PRDN releases the blocks on these string heads in M(<u>k</u>).

E.g., we can have <u>The reason (N_a) it is necessary to describe this function</u>, but not <u>The book it is necessary to describe this function</u>.

3.8 Instruction SUBJ[6]

(SUBJ is carried by $_1For$, $_1to$, $_1H$ in the list of replacement strings of N.

If the string member section in the position list to the right of M(<u>k</u>) (of M(<u>k</u>+1) if N is a string head) is not filled by V, then drop this particular analysis.

E.g., we can have <u>What we said is known</u>, <u>That we said it is known</u>, <u>It follows from what we said</u>, but not <u>It follows from that we said it</u>; i.e., The replacement string $_1HNV^+$ (<u>that we said it</u>) is acceptable only in the subject position, whereas K_4NV^- (<u>what we said</u>) has no such restriction.

3.9 Instruction GOFN

(GOFN is carried by G in the string headed by $_2$N's.)

1. All object strings except N and NPN are dropped from the object register.

2. The particular P, _of_, is inserted as the head of each remaining object string.

3. Perform instruction OBJ+.

GOFN sets up the computation of the string which represents e.g., _John's wiring of radios_, given the object strings of _wire_.

4. <u>Push-down Instructions</u>

Instructions: PDN+
PDND
PDNZ
PDNR
PDNG

PDN instructions cover the cases discussed in I, 3.6., p. 8. and II, 2.2, a, b, c, in which the first symbol of a left adjunct or replacement string is identical with the symbol it adjoins or replaces. PDN+ associates the appropriate PDN instruction with particular symbols. PDND and PDNZ cover the left adjunct cases, and PDNR and PDNG cover the replacement string cases.

4.1 Instruction PDN+

Certain sentence symbols carry instructions to associate PDN instructions with particular symbols: (See, for example, p. 50, N. 4.)

The match of	D	causes	PDND	to be	associated	with	D	in M($\underline{k}$)
" " "	Z	"	PDNZ	" "	"	"	Z	" "
" " "	N's	"	"	" "	"	"	N's	" "
" " "	A or S or G	"	"	" "	"	"	Z and N's	" "

There is an "A counter" and a "D counter" which are set to zero when M($\underline{k}$) is starred.

The match of A or S or G steps the A counter.

" " " D^{PDND} " " D " . (See, for example, p. 51, top, N.

4.2 PDND and PDNZ (PDND and PDNZ are identical except that PDND uses the D counter and PDNZ uses the A counter).

Let q = the contents of the appropriate counter and $\bar{q}$ = the current value of q. $\bar{q}$ + 1 different analyses are recorded for this match, one for each value of q from o through $\bar{q}$. (E.g., note two outputs for $\bar{q}$ = 1, p. 51, top.)

OUTPUT: as usual, except that in each reading, the opening bracket (with name, etc.) is placed to the left of a q pairs of brackets.[15]

SIP: as usual, but if the count of k is kept, the k of each bracket pair included in the bracket opened by PDND or PDNZ is raised by 1. (Here, the depth of the new SIP does not correspond to the depth of the string in the output; advantage is being taken of the fact that all the strings involved in PDN instructions are only one symbol long, so that there are no unfinished PDN strings to carry. If this were not the case, the new SIP would have to be inserted at the depth corresponding to the placing of the opening bracket.)

These instructions yield such outputs as ((completely)unashamedly)(angrily),(completely)((unashamedly)angrily);((dark)red)and(dark)(red);((definite)(young)man's)ideas,(definite)((young)man's)ideas;etc.
(PDND is used in computing <u>fully systematically planned measurements</u>; in see the example in Appendix B1; see p.51, top, N.1-4.)

4.3 Instruction PDNR

(PDNR is normally associated with ${}_7N$ and ${}_{13}N$ in the list of replacement strings of N.)

A symbol G^+(string head), with the instruction PDNG attached to it, is added at the bottom of the position list to the right of M(<u>k</u>).

4.4 Instruction PDNG

1. The opening bracket for G^{PDNG} is placed in the output to the left of the bracketed string to the left. The string name is NG^+.

2. Perform PDNR

PDNR and PDNG enable us to enter a replacement string which can contain a case of itself as its own beginning, i.e., where the X in XY is replaceable by XY, and to decide the depth of the first X only when we reach the second (or third, etc.) Y. Thus we obtain <u>((Children walking at night) constituting their main worry)amazed me</u>.

5. <u>Conjunctions</u>

Instructions:	CONJ
	CC Instructions
	NOTC
	END$\cancel{C}$

Another type of dependence is found in those substrings whose form depends on what precedes them in a given sentence. In the substrings considered up to now, we could say that each string head introduces one or more substrings of fixed form, e.g., HNV^-, HV^+. Most of the conjuntions (those in class $\cancel{C}$j) are string heads of this type. However, the string headed by C (<u>and</u>, <u>or</u>, <u>but</u>, <u>i.e</u>; <u>-er than</u>, <u>as...as</u>) has no fixed form; instead it repeats a preceding string[16] from(almost) any point up to the point at which C occurred, and then completes the preceding string. If the preceding string was finished at the point at which C was matched, this process can also apply to strings which include that string. E.g., <u>The experiment which we started yesterday and</u> can be followed by <u>again today</u>, <u>finished today</u>, <u>they described</u>, <u>which failed</u>, <u>the one which we expect to start</u>, all followed by the required V^+. When C occurs at the end of a string, the repetition may also consist of initial segments of a preceding string, e.g., <u>they will not take it but you may</u>.

Instruction CONJ (5.1) sets up for computation the string headed by C. The CC Instructions (5.2) treat the occurrence of C with a discontinuous prior part, e.g., <u>either ... or</u>. C and CC appear in every combined list (II 4.1),[17] except where Instruction NOTC causes C or particular members of C (such as CA (<u>and</u>), CB (<u>but</u>), CD (<u>or</u>)) to be blocked.

5.1 Instruction CONJ

The transfer to CONJ (in place of the usual SIP and OUTPUT procedures) takes place when C is matched, by having a jump instruction in place of a string as the entry for string-head C in the table of strings. Since C operates differently depending on whether it occurs at the end of a string or before the end, we define, for convenience, an END-STRING indicator for each SIP(k), which is + when the string member section of M(k) is empty and is − otherwise, i.e. "END-STRING(k) is +" means that all the members of the string in SIP(k) have been recognized in the sentence formula. Instruction CONJ is stated in two parts, corresponding to the two main types of strings headed by C: repetitions of a preceding string which reach up to C (Part I), and repetitions of initial segments of a preceding string which may not reach up to C (Part II). Pictorially, if XYZ is a string preceding C, we have:

Part I : C at interior or end of a string

```
X Y C | X Y Z          X Y Z C | X Y Z
      | Y Z                    | Y Z
                               | Z
```

Part II : C at end of a string

```
X Y Z C | X
        | X Y
        | X Z
```

Part I Let C be the symbol in the nth position of the sentence formula. When C is matched there is no output until after the match at n+1. The LIST, MATCH, OUTPUT, and SIP procedures described below are used instead of II 4.1-4 at n+1.

<u>LIST</u>: Starting with the value of k of the deepest SIP:

1. The list at n+1 contains all the starred position lists of SIP(k) with previous blocks released and with right adjuncts and sentence adjuncts blocked.[18] After all C except CS and CT, if SIP(k) contains a CC tag, the position lists to the left of the tagged list are omitted and LIST steps 2-3 are skipped.*

   The list at n+1 is composed of previously matched string members (with their left adjuncts and replacement strings). When the n+1th sentence symbol matches one of the symbols in the list, it thereby decides from which point in the preceding string the string headed by C begins its repetition.

2. If END-STRING(k) is +, then,

   a) instead of the string head in position zero of SIP(k), we place on the list at n+1 the list of string heads from which the string head in position zero had come; (if replacement-string heads, then

*The CC tag is described in 5.2.

also the replaced-string member).

When C occurs at the end of a string, more types of C-strings are permitted than otherwise. LIST 2a allows the string headed by C to be a different string in the same list as the preceding string, e.g., different right adjuncts of N, as in <u>the experiment which we started and for which you waited</u>.

b) LIST 1 and 2, above, are repeated with $\underline{k}-1 \rightarrow k$

This allows the string headed by C to begin the repetition from a point in any one of the strings preceding C (see footnote 16), e.g., <u>We considered the plan which the committee proposed and offered a solution</u>; or using 2a and 2b, <u>He is a person fighting for the rights which are legally his and determined to maintain them</u>.[19]

3. A list of string heads, $\bar{D}$, with depth equal to that of the deepest SIP is added to the list at n+1.

$\bar{D}$ lists adjuncts of a string which can occur directly after C following that string. (The repetition, in this case, begins at the end of the string and consists of a final adjunct.) E.g., <u>We left and fast</u>; <u>He solved the puzzle and while driving</u>.

4. CC is added to the list at n+1. CA, CB, CO add CT and CS also.

After any C we may have CC, e.g., <u>I like blue better than either red or green</u>; after CT or CS we do not have any C, e.g., not <u>than and</u>, <u>as or</u>; after CA, CB, CO, however, we may have CT or CS, e.g., <u>He plays Bach better than Brahms or than Prokofieff</u>.

MATCH: If the matched symbol at n+1 is in a position list from SIP(k), let this k = $\underline{k}$. Separate analyses are made for each match, as described in II 4.2.

OUTPUT: Closing brackets are placed immediately to the left of C and opening brackets immediately to the right of C.

1. For $k > \underline{k}$: brackets are closed

2. If the matched symbol was a string member in SIP($\underline{k}$) (i.e., was a string member copied from SIP($\underline{k}$) by LIST), no opening bracket is placed. If the matched symbol is the head of a left adjunct or replacement string of a string member from SIP($\underline{k}$), an opening bracket is placed according to II 4.3.

3. If the matched symbol was a string head in SIP($\underline{k}$) or is in the class $\bar{D}$:

   a) if END-STRING($\underline{k}$) is +, there are two outputs:

      1) the bracket for $\underline{k}$ is closed, then step 3 of II 4.3 is performed;

      2) no brackets are placed

      E.g., for <u>The experiment which we started and which they finished analyzing</u>, the two outputs are:

      1) $_{lan}$(T;the)experiment$_{ran}$(K_3NV$^-$; which we started) and (K_3NV$^-$; which they finished {G;analyzing}).

      2) $_{lan}$(T;the)experiment$_{ran}$(K_3NV$^-$;which we started and which they finished {G;analyzing})...

In the second analysis, analyzing is the object string of both started and finished. When and which occurred we did not know whether the following material would concern only the string headed by C or both it and the preceding string, i.e., whether or not to close the bracket.

b) if END-STRING($\underline{k}$) is- , no brackets are placed.

Here, since C occurs before the end of the string, we know that what follows must be included in the on-going string, e.g., There are some things which we and which they too cannot explain.

SIP:

1. For $k > \underline{k}$: SIP(k) are dropped

2. If the matched symbol was a string member in SIP($\underline{k}$),

   a) position lists are placed in SIP($\underline{k}$) immediately following the rightmost starred list, replacing any unstarred lists. The inserted lists are copies of the original (totally unblocked) position lists of the string in SIP($\underline{k}$), starting with the one containing the matched symbol; this list is starred.[20] If the matched symbol is the head of a left adjunct or replacement string of a string member from SIP($\underline{k}$), then in addition to the above a new SIP is set up according to II 4.4.

   b) (After CA only) If the matched string member from SIP($\underline{k}$) is N and if there is a blocked V_o in the string member section of the position list to the right of N in SIP($\underline{k}$), the block is released; step 1a of NMBR is blocked from further use in SIP($\underline{k}$).

When CA (<u>and</u>) occurs in the subject position, it is necessary to lift the (singular) agreement restriction, since a singular subject may have become plural, e.g., <u>The man and his dog seem ...</u> but <u>The man seems ...</u> .

3. If the matched symbol was a string head in SIP(<u>k</u>)

a) if END-STRING(<u>k</u>) is +, then

1) (corresponding to OUTPUT 3a1) SIP(<u>k</u>) is erased and a new SIP(<u>k</u>) is set up according to step 3 of II 4.4.

2) (corresponding to OUTPUT 3a2) do 2a,b above.

b) if END-STRING(<u>k</u>) is - , do 2 a,b above.

In the cases corresponding to OUTPUT 3a2, 3b, the string headed by C is treated as part of the preceding string and the position lists are placed in SIP(<u>k</u>), as they were for the match of a string member.

Part II (Carried out after MATCH of Part I, in addition to the rest of Part I; applies only if END-STRING(<u>k</u>) is +)

1. If the matched symbol is N in SIP(<u>k</u>) (or head of a left adjunct or replacement-string of N), and if the position list to the right of N in SIP(<u>k</u>) contains string-member V or G or S (i.e., N is in the subject position[6]), then three additional analyses, A1, A2, A3, are possible. For each analysis, 1 and 2 of OUTPUT and SIP (above) are performed, with the following additions to SIP 2a, respectively:

A1. Erase V (or G or S) and <u>lav</u> in the position list to the right of N, and erase all position lists further to the right.

A2. Same as for A1, but instead of erasing <u>lav</u>, add a top section to <u>lav</u> which consists of $V_{(b)}$. The members of $V_{(b)}$ are also to be in blocking order: <u>do</u>, <u>be</u>, <u>have</u>.

A3. Erase the position list to the right of N, and erase rav from the list to the right of that one. Add ran as the last section of the latter list.

> A1, A2, A3 give rise, respectively, to strings of the type X, XY, XZ (shown at the beginning of 5.1); e.g., A1: She likes tennis more than I, A2: She has been practicing but I haven't, A3. We finished our game and they theirs. In all three strings, N corresponds to the element X. In A2, W (may, can, will, etc.) and $V_{(b)}$ (do, be, have) correspond to the element Y.[21] The blocking order allows, as Y element in XY, sequences like will have been in they will be in Florence more than we will have been, but not such sequences as do be, be have, or have will.

> In A3 we are counting on the fact that the object strings of the verb in the preceding string have been prepared (OBJ instructions performed and resulting strings stored), and that the list of the heads of these strings, os, now appears in the position list following V in SIP(k) (III 2.1). Thus when V is omitted (corresponding to the omission of Y from X Y Z) it is still possible to obtain the correct object (as the Z element of the resulting X Z string).[22]

2. (After CT and CS only) Add to 1. (above):

A4. Erase all OBJ instructions carried by V(or G or S) and erase the list, os, from the position list to right of V.[23]

> In addition to A1-3, we may have: She is prettier than I thought (but not, e.g., She is prettier and I thought).

5.2 CC Instructions

The CC instructions apply to C-conjunctions which occur with a discontinuous/first part, i.e., with an extra sentence symbol which may appear at a preceding point in the sentence formula (e.g., either ... or). The extra element may (or may not) require the occurrence of its associated C, and in most cases it serves as a scope marker for the repetition initiated by C.

CC symbols:[24]

CCA	both	CA	and
CCO	either	CO	or
CCN	neither	CN	nor
CCS	as	CS	as
CCT	more	CT	than
	less		
	rather		
	----er		

5.21 Instruction CCα1 (α = A, O, N)

When CCα is matched, action is delayed until the next non-sublisted position list is starred.[25] Then:

a) This list is given a CC tag; the tag is specific to α.

As a scope marker, the CC tag marks the leftmost point beyond which the string headed by C may not repeat elements of the preceding string, e.g. He will both plan and carry out the test, but not He will both plan and he will carry out the test. The tag works in conjunction with 5.1 LIST 1: After the match of C, CONJ LIST 1 sets up a list of previously matched string members for re-match (repetition); this list may include string members only as far back as the CC tag.

b) Cα is written in the string member section of the last position list of the SIP which contains the tagged position list.[26]

A CC element may require the occurrence of its associated C before the closing bracket is placed on the string containing the CC tag. E.g. in the sentence The people either walking or riding the bus will arrive first, the CC tag occurs in the string begun after either; in The people who either walk or ride the bus will arrive first, the CC tag occurs in the on-going string; in either case, or is required before will arrive closes the bracket on the tagged string. In placing the required C in the string member section in the last position list of the tagged SIP, we use the fact that a closing bracket cannot be placed on a string until the last string member has occurred in the sentence formula. If the required C occurs before this position is reached, the C is erased (CCα2).

5.22 Instruction $C C\alpha 2$

When $C\alpha$ is matched, action is delayed until k is determined (5.1 MATCH). Then a $C\alpha$ is erased from the string member position of the last position list of SIP(k) (if there is a $C\alpha$ present).

Illustrate release requirement

5.23 Instruction$CC\beta 1$ ($\beta = S, T$)[27]

When $CC\beta$ is matched the block on $C\beta$ is released

> CT(than) and CS(as) are normally blocked. However, e.g. after more occurs, we may have than (but it is not required).

5.3 Other conjunction instructions[28]

5.31 Instruction NOTC

The match of T, or T_2 or # blocks C from the combined list for the next sentence symbol.

> E.g. not a and, the and or And at the beginning of a sentence.[29]

5.32 Instruction END$\emptyset$

When END-String (k) is +, the block on $\emptyset_o$ is released.

illustrate

IV. Alternative Analyses

An analysis of a formula branches into two or more analyses if the sentence symbol at some position matches two or more entries in the list at that position. This may happen either because the sentence symbol was a multiple classification and two or more of its choices were matchable in the list, or because a single sentence symbol matched several list entries which were identical in form although related to different strings (and hence carrying different lower left subscripts). An analysis of a formula is uncompleteable (and hence rejected) if the sentence symbol at some position does not match any symbol in the list at that position, or if the end of the formula is reached with some string in progress still requiring the occurrence of a string member in the formula.

Some multiple classifications are resolved immediately if only one of the choices at the multiple choice position has a match in the list at that position. E.g., T N/V W N/V for <u>The rate may increase</u> yields the definite formulas TNWN, TNWV, TVWN, and TVWV. But we can find no V in the list for the sentence symbol following T, so that the formulas TV .... are eliminated. (See p. 46, bot., N.3; p. 51, bot., N.3; p.60, top, N.5 for examples of resolution.) Otherwise the multiple classifications give rise to separate formulas, each with its distinct analysis from this point on, though some of the analyses may be rejected at a later position.

As an example of alternative analyses due to the match of several entries in the list by a single sentence symbol, consider the two sentences:

Figure sculptors like to cast.

Steel sculptors like to weld.

In the second sentence, two analyses are possible until we reach the period, at which position one of them is rejected. The second analysis would have been successful if the sentence had continued, for example:

Steel sculptors like to weld is
low in carbon content.

If more than one analysis remains unrejected after the period, we have more than one grammatical analysis of the formula and of the sentences it represents, indicating in most cases, more than one meaning of the same word sequence (ambiguity).

Various methods can be used to keep track of the several analyses (as long as they last). One method, of course, is simply to make copies at each branch point. Or for less copying, the diverging segments could be named, so that a particular analysis is given by a sequence of names of segments. Alternatively, at each sentence position, the output(s) (and the state(s) of the SIPs) for each analysis could be numbered so that each analysis would be identified as a sequence of nodes in the branching tree structure of all analyses.

Appendix A-1

SYMBOLS IN SENTENCE FORMULAS

A	adjective
A_v	adjective which can follow N e.g., present
$\bar{A}$	A, S, G, Z
B	pronoun which = TA: this, his
C	sentence conjunction which repeats part of the preceding string (III 5): CA and, CB but, CI i.e., CN nor, CO or, -CS as, -CT than
CC	discontinuous part of C-conjunction: CCA both, CCO either, CCN neither, CCS as, CCT more, less, rather, ---er
D	adverb
$\bar{D}$	D, ${}_1P$, ${}_{11}P$, ${}_1to$, $\not{c}_{2-3}$
D_2	adverb after is, e.g., here
G	verb with ing
H	that [string head]
K	wh-word: K_1 who, K_2 whom, K_3 which, K_4 what, K_5 whose, K_6 whoever, K_7 whomever, ... K_{10} whosever, K_{11} where, K_{12} whether
N	noun (see footnote 8)
N_1	count noun; N which requires an article in the singular: the book
N_{sg}	singular non count noun
N_{pl}	plural noun
N's	noun with 's
N_S	noun which can have HNV^* adjunct: reason
O_B	objects of be: A, G^+, D_2, N, PN, S^+

Appendix A-1 (continued)

P	preposition
Q	quantifier: many, few, one, two, ...
R	pronoun (=TN)
S	verb with -en, -ed (passive)
T	article: T_1: a; T_2: the; T_3: no, one; T_4: some, either
T_q	quantifier which can precede T: all(the), many (a)
T_D	adverb which can precede T, T_q: just, only, barely, scarcely.
V	verb (see footnote 8)
V_o	verb with no suffix
V_s	verb with -s
V_{pp}	verb with -ed or -en occurring after have
V_p	verb with -ed (past)
$V_{(b)}$	be, have, do
W	auxiliary, e.g., will, can (but not have, be, do)
Z	color name
¢	sentence conjunction of the non-C type
$¢_o$	yet
$¢_1$	because, where, whereas, so that, except that, provided that, in order that
$¢_2$	unless, until, if, when
$¢_3$	while, since, once, though, although, as if, as, whether
#	sentence initial mark, head of center string
&	conjunctional adverb: moreover, however, nevertheles

NOTE: Some of the above symbols may also appear with superscript letters which refer to dependence instructions: e.g., N^{PRON} states that this N carries instruction PRON; in the case of verb-object instructions, the letters OBJ are omitted.

Appendix A-2

STRINGS

The following are the main axiomatic strings.

A	e.g., grisly
B	his
C	DO CONJ instruction (III 5.1)
D	hopefully
${}_1\text{For } N \text{ to } V^+$	For Brutus to stab Caesar
${}_2\text{For } N \text{ to } V^-$	for children to twist
G	skidding
G^+	scraping furniture
${}_1H\ N\ V^+$	that the house is on fire
${}_2H\ N\ V^-$	that I went through college with
${}_3H\ V^+$	that likes mother
$K_1\ V^+$	who assassinated the Grand Duke
$K_2\ N\ V^-$	whom the Grand Duke saw
${}_1K_3\ V^+$	which came to light
${}_2K_3\ N\ V^-$	which he drank
${}_1K_4\ V^+$	what succeeds
${}_2K_4\ N\ V^-$	what I think
${}_1K_5\ N\ V^-$	whose goose is cooked
${}_2K_5\ N\ N\ V^-$	whose barn we painted
$K_6\ V^+$	whoever steps forward
$K_7\ N\ V^-$	whomever I chide
${}_1K_8\ V^+$	whichever survives
${}_2K_8\ N\ V^-$	whichever they choose
${}_1K_9\ V^+$	whatever turns green
${}_2K_9\ N\ V^-$	whatever you say

Appendix A-2 (continued)

${}_{3}K_{9}\ N\ N\ V^{-}$	whatever choice you make
${}_{1}K_{10}\ N\ V^{+}$	whosever salary increases
${}_{2}K_{10}\ N\ N\ V^{-}$	whosever hand you squeeze
$K_{11}N\ V^{+}$	where maples grow
$K_{12}\ N\ V^{+}$	whether fish learn
${}_{1}N$	silicon (diode)
${}_{2}N\ \bar{A}$	book buying (spree)
${}_{3}N,$	[apposition]
${}_{4}N\ V^{-}$	(people) you may meet
${}_{5,m}N\ P\ K$	[${}_{5}N$ strings are K strings with NP preceding, for K = ${}_{1}K_{3}$, ${}_{2}K_{3}$, ${}_{1}K_{5}$, ${}_{2}K_{5}$]pages of which are torn
${}_{6}N\ V^{+}$	(the reason) it is here
${}_{7}N\ G^{+}$	people wearing hats, other things being equal
${}_{8}N\ S^{+}$	[sentence minus <u>is</u>] his presence assumed
${}_{9}N\ A$	[" " "] the room free
${}_{10}N\ D_{2}$	[" " "] the painters out
${}_{11}N\ P\ N$	[" " "] the boat on its way
${}_{12}N\ N$	[" " "] the destination an island
${}_{13}N\ G$	elephant hunting
${}_{1}N's\ G^{+}$	children's playing games
${}_{2}N's\ G^{GOFN}$	children's playing of games
${}_{3}N's$	children's
${}_{1}P\ N$	in the mountains
${}_{2}P\ K_{2}\ N\ V^{P}$	to whom it may concern
${}_{3}P\ K_{3}\ N\ V^{P}$	beyond which I can't see
${}_{3a}P\ K_{3}\ N\ N\ V^{+}$	at which position one is rejected
${}_{4}P\ K_{4}\ N\ V^{P}$	of what we dream

Appendix A-2 (continued)

${}_{5}P\ K_{5}\ N\ N\ V^{P}$	into whose den he walked	
${}_{7}P\ K_{7}\ N\ V^{P}$	on whomever one relies	
${}_{8}P\ K_{8}\ N\ N\ V^{P}$	on whichever page it is	
${}_{9}P\ K_{9}\ N\ N\ V^{P}$	to whatever heights you soar	
${}_{10}P\ K_{10}N\ N\ V^{P}$	on whosever word you take it	
${}_{11}P\ A$	in general	
Q	much	
R	we	
S^{+}	prepared	
T_{q}	many	
T	the, a	
$T_{2}\ \bar{A}$	the good, the rewarding, the undecided, the grey	
${}_{1}To\ V^{+}$	to tell them to wait	
${}_{2}To\ V^{-}$	(the person) to see	
W	will, can	
Z	vermilion, chartreuse	
$\#\ N\ V^{+}$	[center string] we were first.	
&	however	
${}_{1}\not{C}_{0-3}\ N\ V^{+}$	[Note: ${}_{1}C_{0}$ is normally blocked (III 5.32)]	We will write when it's finished.
${}_{2}\not{C}_{0-3}\ N\ W$	[" ${}_{2}C_{0}$ " " " "]	I wouldn't say so but you may.
${}_{3}\not{C}_{0-3}\ N\ V_{(b)}$	[" ${}_{3}C_{0}$ " " " "]	He will go, since we haven't.
${}_{4}\not{C}_{3}$ having V_{pp}^{+}	We changed our view once having seen them.	
${}_{5}\not{C}_{2-3}\ O_{B}$	We changed our plans while driving.	

Appendix A-2 (continued)

OBJECT STRINGS

Following are the main object strings of English verbs. (The listing in two sections has no meaning for the substring program; it is for transformational reasons). P_i indicates an individual preposition specified by the verb. Not all the noun replacement strings (App. A-3) can replace N in object strings; a slightly reduced *rn* list would have to be stated for this position (see Fn 6).

∅	[zero object, e.g., for *exists*]
$\bar{A}$, N's	(seems) fine, (is)John's
D_2	(are) here
G^+	(like) digging clams
N	(found) sea shells
N_2	[V N_2 has no passive, e.g., *he slept a long sleep*]
N_{pl}	(correlate) these facts
N and N	(correlated) this and that
N with N	(compare) alpha with beta
N N	(give) him a book
N P_i N	(refer) him to me
N P'_i	[P' in a subclassification of prepositions, e.g., *broke the game u*
P'_i	(broke) up
P'_i N	(broke) up the game [R is blocked in the list of replacement strings of this N, e.g., *broke up the game* but not *broke up it*]
P_i N	(rely) on their results
P_i G^+	(refrain) from going
S^+	is (gone)
to V^+	want (to swim)
A H N V^+	(it is) certain that none will follow
A to V^+	(it is) possible to state the difference
A for N to V^+	(it is) possible for him to speak
A to V_o^-	(it is) easy to do

Appendix A-2 (continued)

H N V^+	(hoped) that it would work
N A	(thought) him wise
N D_2	(") him there
N N	(thought) him an Englishman
N P N	(") " in Philadelphia
N as D_2	(consider) him as here
N as G^+	(") " " being fair
N as N	(") " " a candidate
N G^+	(start) people talking
N H N V^+	(told) him that nothing would result
N P_i G^+	(restrain) him from going
N V^+	(know) the thing was rigged
N^* V_o^+	(demand) they appear
	[Agreement instructions are not applied to strings with V_o when N^* is matched.]
P N H N^* V_o^+	(ask) of him that he not leave
N to V_o^+	(want) them to succeed
V_o^+	(let) go

Appendix A-3

LISTS OF STRING HEADS

The following lists include the heads of the main adjuncts, replacement strings, etc., for each symbol.* The strings referred to are listed in full, pp. 37-39.

__lan__
(left adjuncts of N)

1^N
2^N
$\bar{A}$**
laa
N's
Q
laq
T
B
T_q
T_D

__lav__
(left adjuncts of V)

D
1^P
W

__laa__
(left adjuncts of $\bar{A}$)

D

__laq__
(left adjuncts of Q)

D

__ran__
(right adjuncts of N)

A_v
D_2
-1^{For}
2^{For}
G^+
-1^H
2^H
3^H
K_1
K_2
1^{K_3}
2^{K_3}
1^{K_5}
2^{K_5}
3^N
4^N
$5,m^N$
-6^N
1^P
2^P
3^P
5^P
S
1^{to}
2^{to}

__rav__
(right adjuncts of V)

D
1^P

__raa__
(right adjuncts of $\bar{A}$)

1^P
2^{to}

__rn__
(replacement strings of N)

1^{For}
G
G^+
1^H
1^{K_4}
2^{K_4}
3^{K_4}
K_6
K_7
1^{K_8}
2^{K_8}
1^{K_9}
2^{K_9}
3^{K_9}
$1^{K_{10}}$
$2^{K_{10}}$
K_{11-12}
7^N
13^N
$1^{N's}$
$2^{N's}$
Q
R
T_2
T_q
$1^{to}T_h$

__sa__***
(sentence adjuncts, called by all symbols except P and heads or members of left adjuncts of N.)

A
D
1^{For}
G^+
K_6
K_7
1^{K_8}
2^{K_8}
1^{K_9}
2^{K_9}
3^{K_9}
$1^{K_{10}}$
$2^{K_{10}}$
3^N
$7\text{-}12^N$
1^P
11^P
S^+
1^{to}
&
$-1\text{-}3^{¢}{}_0$
$1\text{-}3^{¢}{}_{1-3}$
$4^{¢}{}_3$
$5^{¢}{}_{2-3}$

__c__
(center string)

#

__lap__
(left adjuncts of P)

D

*The indication of dependence instructions is omitted from the lists. Left subscripts are not included in the matching process.

**$\bar{A}$ stands for the symbols A, S, G, Z

***A number of sentence adjuncts, depending on particular words or subsets, have not been included in the list, e.g., the type of __I think__, in __He went away, I think__. More refined distinctions of subsets would show more sentence adjuncts after subject N than after other N, i.e., the particular list of sentence adjuncts depends on position. (See Fn 6)

Appendix B

EXAMPLES

Each sentence symbol in the following examples is analyzed with respect to a particular match at that point. Only on some occasions is it pointed out, as an example, where other matches are possible. (The program itself could be used to investigate how the number of alternative analyses varies from left to right through the sentence formula.) Also, the only section of the list which is itemized is the one containing the match which is being considered for this analysis.

Sentences were selected which would require many dependence instructions. Other sentences, which may be complicated for English, are straightforward for the method, e.g., sentences with many nestings, such as the one on p. 12, or with repeated application of a single instruction, such as _The guide we told the man to tell the people to wait for never came_ (three uses of OBJ-), and _The people we told the man to tell to wait for the guide have left_ (two uses of OBJ-). Further detailed examples of the application of dependence instructions to various complicated situations in English will be given in a forthcoming paper of TDAP.

Form of the examples: The computation of successive sentence symbols is shown in successive frames. The combined list appears under the sentence symbol, and the match (not necessarily the only match) is indicated by an arrow. (An example of how the combined list was assembled is given once, in frame 2, p. 46.)

The SIPs are shown in the state they are in before the match takes place, unless the notes indicate otherwise. E.g., new SIPs which are added as a result of a given match are shown in the same frame as the match, if they do not replace existing SIPs; and the placing of object string heads in the SIP by OBJ+ is shown in the same frame as the match of the verb which carries OBJ+. In the first example, the notes covering dependence instructions precede those describing the OUTPUT and SIP procedures. In the second example, the notes are in the opposite order (and probably easier to follow).

Appendix B (continued)

Index of Procedures and Instructions

Procedure or Instruction	Page Used	Page Defined
Assembly of combined list	45 - 60	15
Match	" "	15
OUTPUT Procedure	" "	15
SIP Procedure	" "	16
Partial Ordering	" "	19
NOTC	45	32f
CPDN	46,47	25
CPDR	47	25
NMBR	47	25
OBJ+	49	20
Symbols in a sublist	49	13,15,17
PDN+	50	27
PDND	51	28
NOTC	53	32f
LFTE	53	26
CNTN	53	25
NMBR	55	25
PRON	55	26
OBJ-	56	21
CONJ	56,57	30
OBJB	58	23
OBJ+	58	20
OBJE	59	23
OBJP	59	22
OBJS	60	22

Example sentence:

#Several different radio astronomy stations will make fully systematically planned measurements.

```
   Q          A        Ni/V     Nsg          Npl       W      Vo      P            D             Vp/S        Npl
```

```
Sentence                                      |      [1]
Symbol                   #                    | Output: <NV+;
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

                  *
SIP(0)       c    #     N        V+

                        lan      lav       sa
                 [2]
                        rn       sa        rav

                        sa       ran
```

1 Sentence symbol # opens the center string bracket, and
2 sets up SIP(0) (initial conditions, II 4.5). "C" for center string is noted(SIP 3b).
3 # blocks C from occurring on the next combined list (NOTC). See next frame.

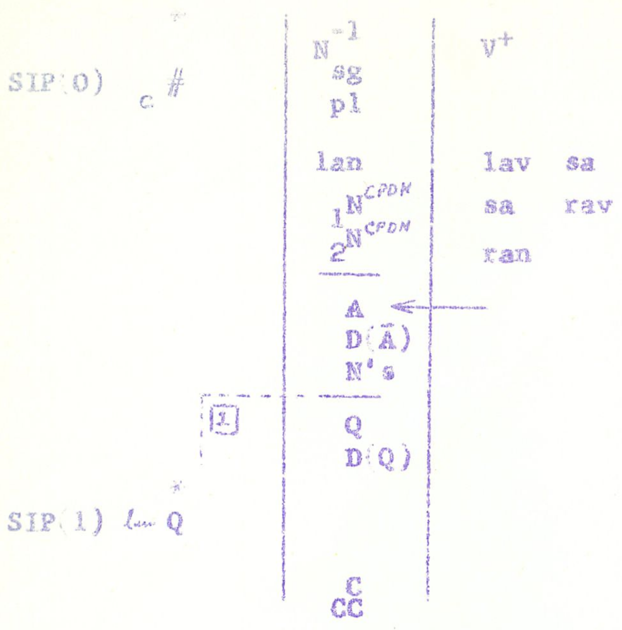
```
Sentence                                      |                [2]
Symbol                   Q      (several)     | Output: <NV+; l_an (Q:  several
- - - - - - - - - -*- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
SIP(0)       c    #   |  -1    |   V+
                      | Nsg    |
                      |  pl    |
                      | lan    |   lav      sa
                      |  ,N    |
                      |  ,N    |   sa       rav
                      | ----   |
                      |  Ā     |   ran
                      |  B(Ā)  |
                      |  N's   |
                      | ----   |
                      |  Q   <-|---
                      |  D(Q)  |
             - - - - -| ----   |
             [1]      |  T     |
                      |  B     |
                      | ----   |
                      |  Tq    |
                      | ----   |
                      |  Td    |
                      | rn     |
                      | sa     |
                 *    | CC [4] |
SIP(1)     l_an  Q [3]
                    △
```

Example of LIST Assembly (4.1,p.15): Since the previous symbol, #, was assigned the depth 0, the combined list for Q contains:

1. The position list following # in SIP(0). (This is the active list in SIP(0) according to the definition on p.14).
2. Since the list in 1. contains a string member (N), condition 2. doesn't apply.
3. CC (C is blocked.)

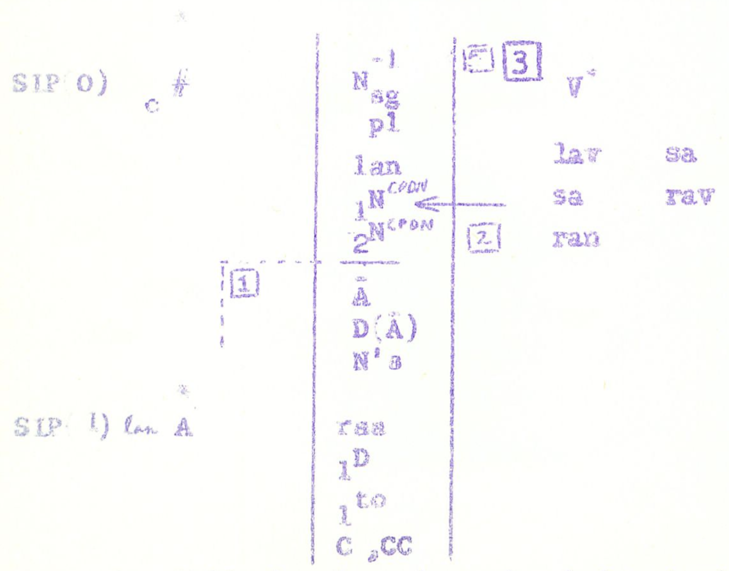
1 Q blocks the sections and subsections below Q in the list (partial ordering, III 1)
2 Q opens an adjunct bracket, and
3 sets up SIP(1); the position list to the right of Q in SIP(1) is empty.

Sentence Symbol $\mathring{A}$ (different) | Output: $\langle NV^{+};_{lan}(Q;\text{several})_{lan}$ A:different $\boxed{2}$ $\boxed{3}$

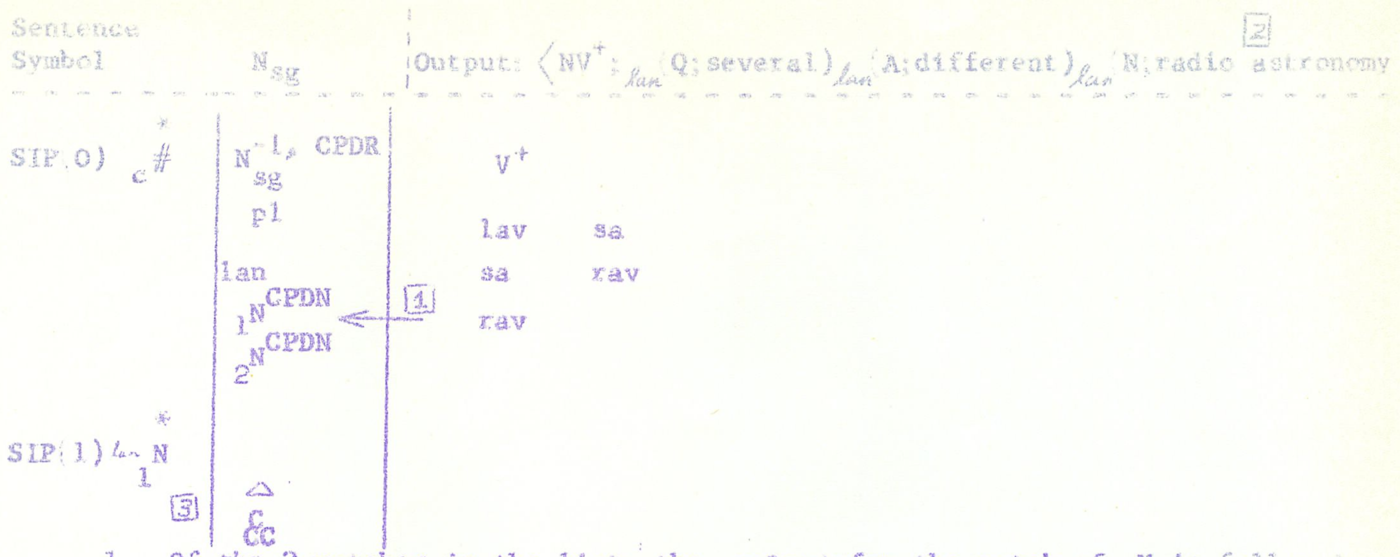


1 A blocks the subsection below it in the list (partial ordering). (Ā stands for symbols A,S,G,Ȧ)
2 The match of Ā in the list from SIP(0) closes the bracket on Q (depth 1), (Output procedure, step 1) and erases SIP(1), (SIP procedure, step 1).
3 A opens an adjunct bracket (Output procedure, step 3) and sets up a new SIP(1), shown in the next frame (SIP procedure, step 3).

Sentence Symbol $\boxed{3}$ $N_1/V_\emptyset$ (radio) | Output: $NV^{+};_{lan}(Q;\text{several})_{lan}(A;\text{different})$ $_{lan}(N;\text{radio}$ $\boxed{4}$



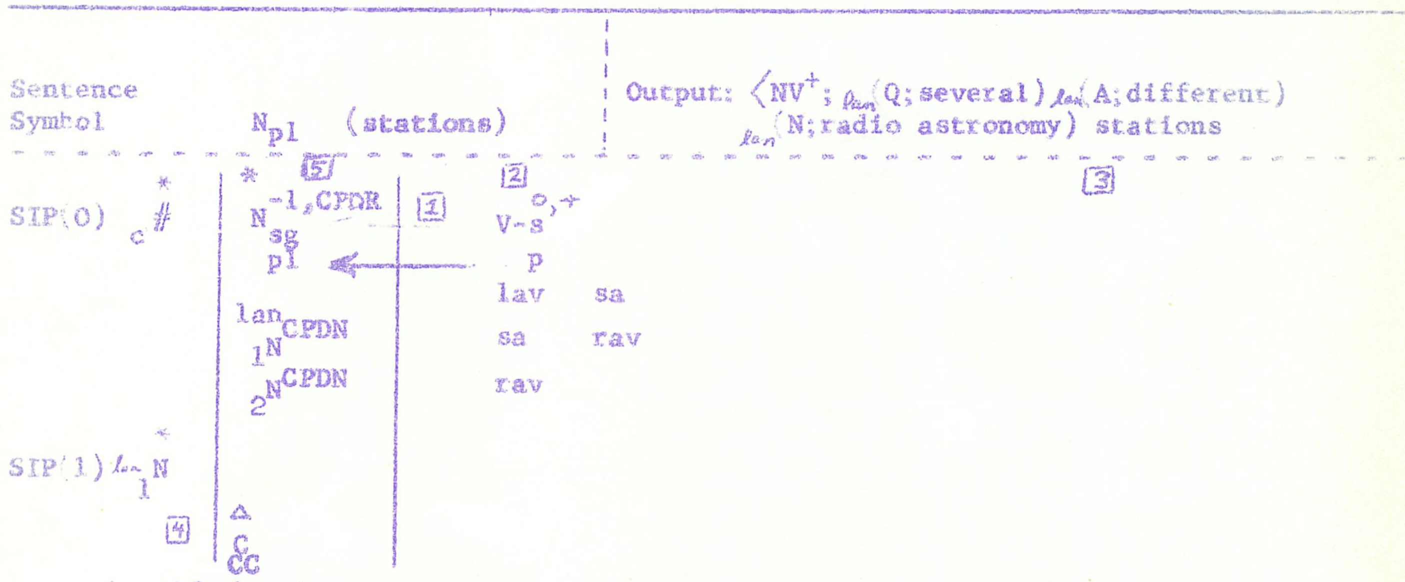
1 $_1N$ blocks the subsection below it in the list (partial ordering).
2 $_1N$ and $_2N$ are both matched; the reading for $_1N$ is followed in the example.
3 Note that the restriction on N_1 (count noun) prevents a third match of N and the fact that there is no V in the list decides the classification of radio(N/V) in favor of N.
4 The match of $_1N$ (depth 0) closes the bracket on A (depth 1) and erases SIP(1). N_1 opens an adjunct bracket and sets up a new SIP(1) (shown in the next frame).
5 CPDN attaches CPDR to string member N in the list. (See next frame) This will release the block on the Output Procedure which CPDN imposes (on the next match)


Sentence Symbol
N_{sg}
Output: $\langle NV^{+};\ _{lan}(Q;\text{several})_{lan}(A;\text{different})_{lan}(N;\text{radio astronomy}$
$\boxed{2}$
SIP(0) $_{c}\overset{*}{\#}$
$N_{sg}^{-1,\ CPDR}$
pl
V^{+}
lav sa
sa rav
rav
lan
$_{1}N^{CPDN}$
$\boxed{1}$
$_{2}N^{CPDN}$
SIP(1) lan $_{1}\overset{*}{N}$
$\boxed{3}$
△
C
CC

1 Of the 3 matches in the list; the analysis for the match of $_{1}N$ is followed in the example.

2 The placing of all brackets is blocked by the previous CPDN instruction.

3 The match of $_{1}N$ (depth 0) erases SIP(1) and sets up a new SIP(1) (SIP procedure); in this case the new string is the same as the previous one.


Sentence Symbol
N_{pl} (stations)
Output: $\langle NV^{+};\ _{lan}(Q;\text{several})_{lan}(A;\text{different})\ _{lan}(N;\text{radio astronomy})$ stations
$\boxed{5}$
$\boxed{2}$
$\boxed{3}$
SIP(0) $_{c}\overset{*}{\#}$
$\overset{*}{N}{}_{sg}^{-1,CPDR}$
pl
$\boxed{1}$
$V\text{-}s^{o,+}$
p
lav sa
sa rav
rav
lan_{CPDN}
$_{1}N$
$_{2}N^{CPDN}$
SIP(1) lan $_{1}\overset{*}{N}$
$\boxed{4}$
△
C
CC

1 Of the three matches in the list, the analysis is the example is for the match of the string member N.

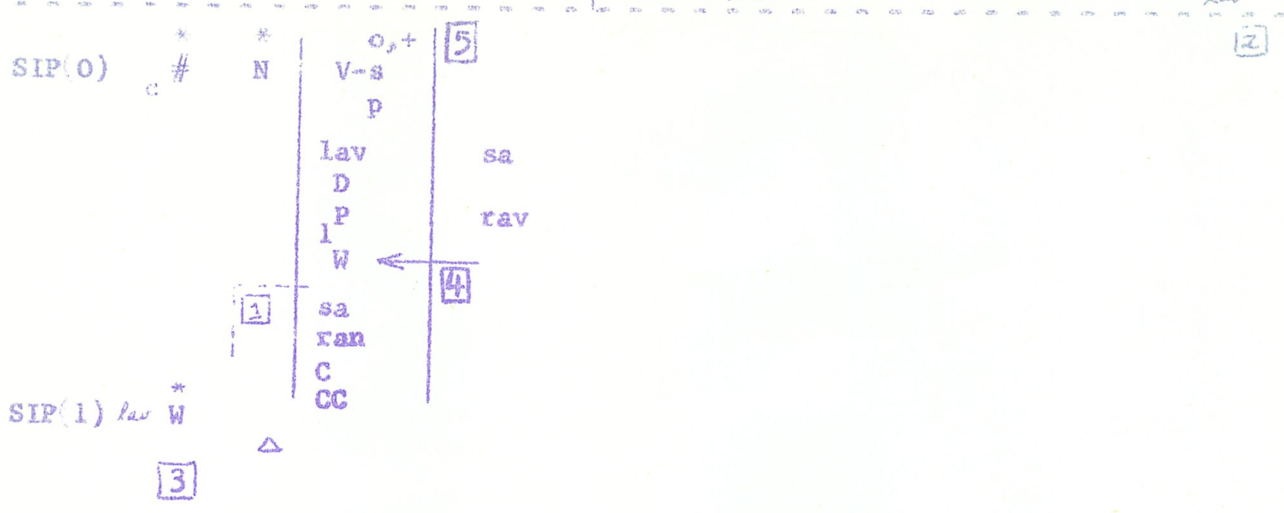
2 N_{pl} blocks V_{s} (NMBR)

3 CPDR releases the block on the Output procedure. Accordingly the match of N at depth 0 closes the bracket on $_{1}N$ (depth 1) (OUTPUT, step 1)

4 The match of N at depth 0 erases SIP(1) (SIP procedure, step 1).

5 The list at N is starred (SIP procedure, step 2).

Sentence Symbol W (will) | Output: $\langle NV^{+};{}_{lan}(Q;\text{several})_{lan}(A;\text{different})\ {}_{lan}(N;\text{radio astronomy})\ \text{stations}_{lav}\ W;\text{will}$



1 W blocks sections below it in the list (partial ordering, p. 19).
2 W opens an adjunct bracket (Output procedure, step 2) and
3 sets up SIP(1), (SIP procedure, step 3).
4 W blocks V_p (TENS, p.23) (See next frame).
5 Had V_o been blocked, W would have released the block (NMBR, p. 25).

```
Sentence                                |Output: ⟨NV+; lan(Q;several) lan(A;different) law(N;
Symbol                V    (make)       |        radio astronomy)stations law(W;will)  mak
                       o                |
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
                      [2]  *                                                          [1]
              *     *        o, + ←  [3]
SIP(0)     #      N    |  V-s     |os                  Object Register
          o            |   -p     |sa
                       |          |                     N         e.g., make measurements
                       |  lav     |rav
                    *  |          |                     NN        e.g., make him chairman
SIP(1)     law    W    |          |
                       |  △       |                     NV+       e.g., make it work
                       |  C       |                     ...
                       |  CC      |
```

1 The match of V_o (depth 0) closes the bracket on W (depth 1), (Output procedure, step 1), and erases SIP(1), (SIP procedure, step 1).

2 The list at V is starred (SIP procedure, step 2).

3 OBJ+ posts the list of object string heads in the string member section of the next position (condition 1 of OBJ+). The sublists lan and rn are written under each N. E.g., we mark the first, second, and third object string heads which are N with 1, 2, and 3 primes, respectively. The list os will then appear:

```
N'
lan(N')
rn(N')
N''
lan(N'')
rn(N'')
N'''
lan(N''')
rn(N''')
.
.
.
```

The list expanded in the manner of Section II, 2.2 becomes:

```
N'
lan(N')
  .
  .
  .
  Ā(N')
  laa(Ā(N'))
  D(Ā(N'))
  .
  .
  .
 rn
  .
  .
  .
 7N^R(N')
   lan(7N^R(N'))
   rn(7N^R(N'))       Note: 7N^R is excluded from
                             rn(7N^R), (E,    p.13).
 N''
  .
  .
  .
```

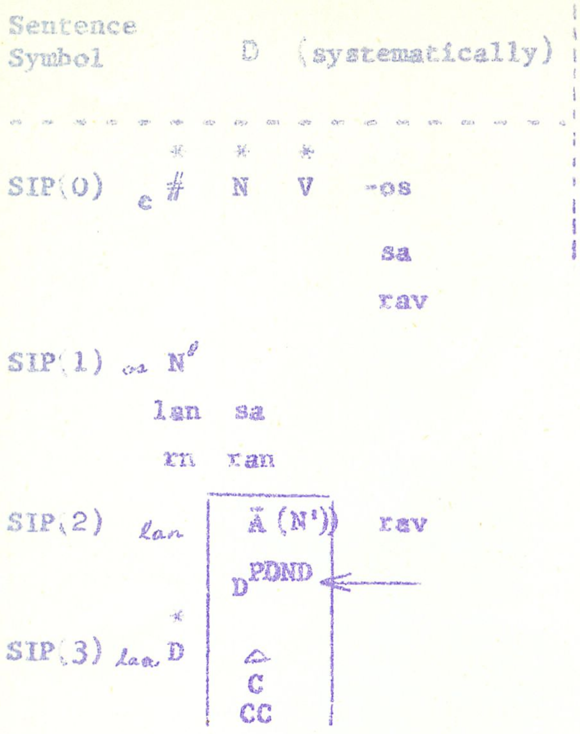
```
Sentence                                   |Output: (NV+; lan(Q;several) lan(A;different) lan(N;
Symbol                         D (fully)   |        radio astronomy)stations lav(W;will)  make
                                           |        (N;  (Ā;  (D;fully
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
              *    *    *     | -os  [3]   |        [1]
SIP(o)      c#    N    V      |  N'        |
                              | lan(N')    |
                              |  .         |
                              |  .         |
                              |  .         |
                              |  Ā(N')     |
                              |  laa(Ā)(N'))
                              |    D(Ā(N'))  <--------
                              |  N's(N')   |
                              |  ________  |
                              |  Q(N')     |
                              |  laq(Q(N'))|
                              |  T(N')     |
                              |  B(N')     |
                              |  ________  |
                              |  .         |
                              |  .         |
                 - - - - -    |  .         |
                         |    | rn(N')     |
                         |    | N''        |
SIP(1)  os   N'          |    |  .         |
             lan   sa    |    |  .         |
             rn    ran   |    |  .         |
                         |    | sa         |
                         |    | rav        |
                         |    | C          |
                         : _ _  CC - - - - - - - -
SIP(2)  lan  Ā (N')
             laa(Ā(N')) raa
             D^PDND(Ā(N')) [4]

             *
SIP(3)  lua  D(Ā(N'))
               [2]
```

1 D opens an adjunct bracket; then, since D is in the sublist laa(Ā), an adjunct bracket for Ā is opened to the left of the bracket for D; then, since Ā is in the sublist lan(N'), where N' is in os, an object string bracket for N' is opened to the left of the bracket for Ā (Output procedure, step 3d).

2 Correspondingly, D brings in the position lists for the D string as SIP(1); then, since D is in the sublist laa(Ā), the SIP for D is pushed down to SIP(2) and Ā brings in the position lists for the Ā string as the new SIP(1); then, since Ā is in the sublist lan(N') the stack of new SIPs is pushed down (D→ SIP(3) Ā→ SIP(2) and N' brings in the position lists for the object string N' as SIP(1) (SIP procedure, step 3f).

3 The list of object strings is blocked (SIP Procedure, Step 3e). (Next frame)

4 D is one of the symbols which carries PDN+, which associates PDND with the occurrence of D in the list. Since D is in a sublist, the list in question is the one containing string member Ā, in the SIP set up for Ā, i.e. SIP(2). (See p. 18)



Output: (NV⁺; lan(Q;several) lan(A;different) lan(N;
radio astronomy)stations lan(W;will)make
(N; lan(Ã; lan(D;fully)(D;systematically

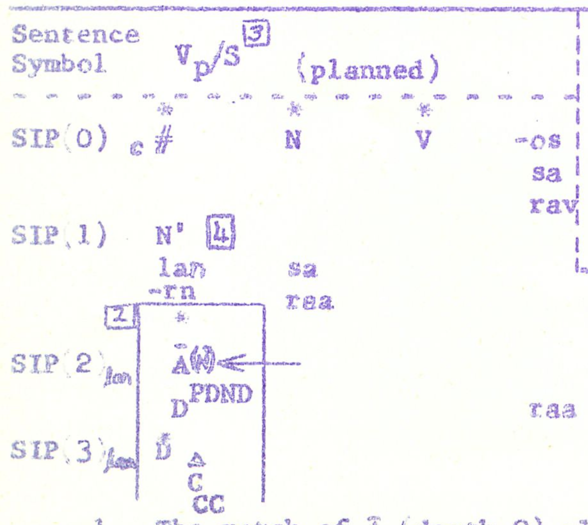
[2] [3]

... (N; lan(Ã; lan(D; lan(D;fully) sytematically.
[4] [2]

[1]

D COUNTER = 1

1 The match of D^{PDND} increments the D counter.
2 Brackets are closed as usual; the match of D (depth 2) closes the bracket on D (depth 3) and SIP(3) is erased.
3 Executing PDND for q=o, the bracket for D is opened as usual.
4 For q=1, another reading is recorded in which the bracket for D is opened to the left of 1 pair of brackets.
5 A new SIP(3) for D is set up.



Output: (NV ; (Q;several) (A;different) (N;
radio astronomy)stations (W;will)make
(N; (Ã; (D;fully)(D;systematically)planned
[1]

(NV ; (Q;several) (A;different) (N;
radio astronomy)stations (W;will)make
(N; (Ã; (D;(D;fully)systematically)planned

[1]

($\bar{A}$ = A or S or G or Z)

1 The match of Ã (depth 2) closes the bracket on D(depth 3) and erases SIP 3).
2 The list at Ã is starred (SIP Procedure, step 2)
3 Now that V_p/S is decided in favor of S because there is no V_p in the list.
4 The replacement strings rn of N' in SIP(1) are blocked, so that we eliminate, e.g., fully systematically planned what to measure. This comes about as follows: 1) Since Ã is a sublist of N', instructions carried by Ã apply to the position list which has N' as string member (p. 18); 2) Associated with the match of Ã in lan is the instruction to block sections of the position list below lan (partial ordering).

```
Sentence                                 Output: ⟨NV⁺; lan(Q;several) lan(A;different) lan(N;
Symbol         N_pl (measurements)          radio astronomy)stations lan(W;will)make
                                            {N; lan(Ā; lad(D;fully) lad(D;systematically
                                            planned)measurements
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
                                                    [1]
             *    *    *
SIP(0)      c#    N    V  |  -os      |
                          |  sa       |
                          |  rav      |
                       [2]| *    -1   |
SIP(1)                    |  N' sg  <---------
                          |     pl    |
                          |  lan      |  sa
                          |           |  ran
                  *       |           |
SIP(2)            Ā       |           |
                          |  raa      |
                          |  C        |
                          |  CC       |
```

1 The match of N' (depth 1) closes the bracket on Ā (depth 2) and erases SIP(2). (Only one analysis will be ~~mm~~ carried further in this example.)

2 The list at N' is starred, since in SIP(1), N' is a string member (SIP Procedure, Step 2).

```
Sentence                                 Output: NV ; (Q;several) (A;different) (N;
Symbol                                      radio astronomy)stations (W;will)make
                                             N; (A; (D;fully) (D;systematically
                                            planned)measurements⟩
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
                                                              [1]
             *    *    *
SIP(0)      c#    N    V  |  -os      |
                          |  sa       |
                          |  rav      |
                  *       |           |
SIP(1)            N'      |           |
                          |  sa       |
                          |  ran      |
                          |  C        |
                          |  CC       |
```

1 The intervention of the period, the end-sentence mark, terminates the computation. Since there is no string member remaining in the SIPs, the analysis is accepted and a closing center bracket is placed (p. 33).

Example sentence:

The experiment which we started and for which ~~[illegible]~~ you were waiting was interupted.

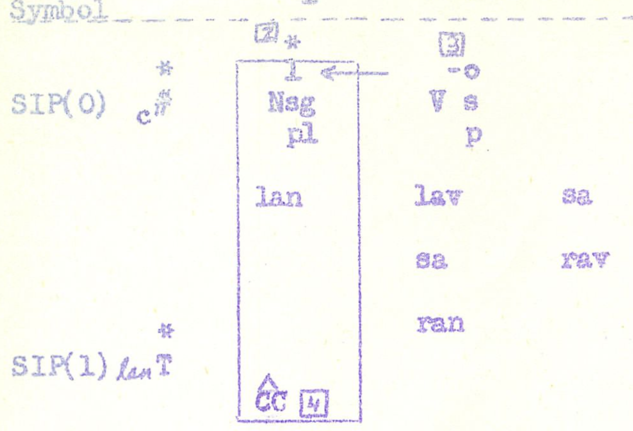
```
#  T2   N1        K3  Rpl  Vp/S  C , P  K3            Rpl  Vp     G-   Vp    Vp/S   .
---------------------------------------------------------------------------------------
                                                    |          [1]
Sentence                                            |
Symbol     #                                        | Output: <NV+;
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
        [3]*
SIP(0)   c#       N     V+

                  lan   lav   sa
             [2]
                  rn    sa    rav

                  sa    ran
```

[1] Sentence initial mark, #, (given as a "match" by initial conditions) opens the center string bracket in the output, and [2] brings in the string #NV+ with its position lists, as SIP(0). (II, 5.1, p. 17, and OUTPUT and SIP Procedures, Step 3, pp. 16, 17).
[3] Position 0 is starred, c for center string is noted (SIP Proc., Steps 3b-c, p. 16).

```
                                                    |                   [1]
Sentence             T2      (the)                  | Output: <NV+;    (T;the
Symbol                                              |             lan
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
                     +-------+
              *      | [5]   |
                     | + =1  |
SIP(0)     c  #      | N sg  |       V+
                     |    pl |
                     |       |
                     | lan   |       lav     sa
                     | 1N    |
                     | 2N    |       sa      rav
                     |-------|
                     | A     |       ran
                     |  laa  |
                     |  N's  |
                     |-------|
                     | Q     |
                     |  laq  |
                     |-------|
                 [3] | -T    | <----
                     | -B    |
         :- - - - - -|-------|
         b       [4] | Tq    |
         l           |-------|
         o           | Td    |
         c           |       |
         k           | rn    |
         e           |       |
         d           | sa    |
         :           |       |
          [2] *      | CC    | [6]
SIP(1)    lan T      +-------+

                       ^
```

[1] T_2 opens a left adjunct bracket in the output, and [2] sets up SIP(1) with position zero starred. The position list following T is empty.
[3] T blocks itself and a (LFTE), and [4] the sections below it in the list (partial ordering)
[5] T releases the block on N_1 (CNTN).
[6] # blocked C from occurring on this combined list and T_2 blocks C from occurring on the next combined list (NOTC).

Sentence Symbol	N_1 (experiment)	Output: $\langle NV^{+}$; ${}_{lan}$(T;the) experiment $\boxed{1}$

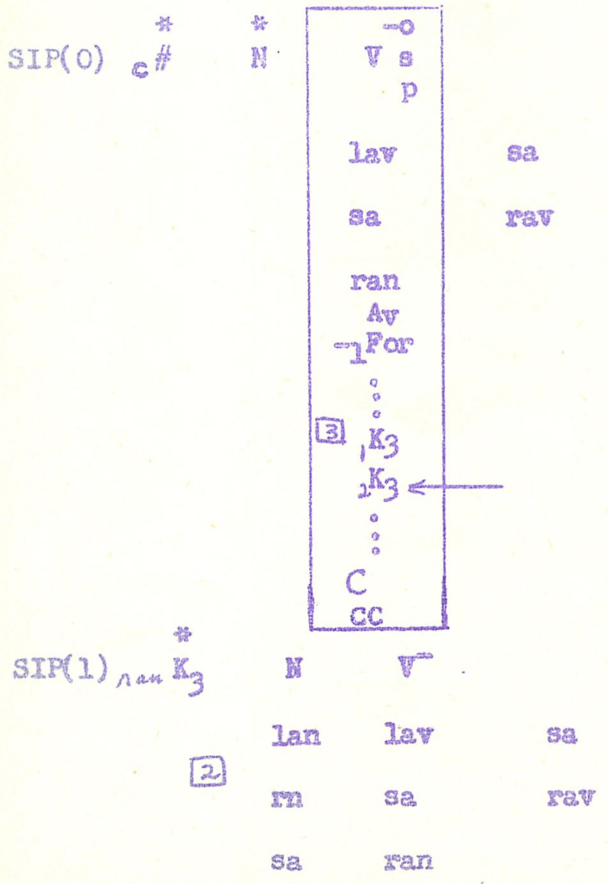


$\boxed{1}$ The match of N_1 at depth 0 closes the bracket on T at depth 1, and erases SIP(1). $\boxed{2}$ The list at N is starred because of the match of a string member.

$\boxed{3}$ N_1 blocks V_o (NMBR).

$\boxed{4}$ Note that CCis added to the list but not C (NOTC).

Sentence Symbol	K_3 (which)	Output: $\langle NV^{+}$; ${}_{lan}$(T;the)experiment ${}_{ran}$($\boxed{1}$ K_3NV^{-};which



$\boxed{1}$ ${}_2K_3$ opens an adjunct bracket for the string K_3NV^{-} and $\boxed{2}$ sets up a new SIP(1) .

$\boxed{3}$ ${}_1K_3$, which heads the string K_3V^{+} (e.g. the experiment which started) is also matched, but only one reading is followed in this example.

```
Sentence                         |                                   [1]
Symbol              R_pl (we)    |Output:<NV+; lan (T;the)experiment nan (K3NV-;which nn [R;we
---------------------------------------------------------------------------------------------
           *     *
SIP(0)  c#       N    V
                      lav      sa
                      sa       rav
                      ran

           *      [3] *        [4] o
SIP(1) nan K3    +---------+
                 |  N      |   V-s
                 | lan     |     p
                 | rn      |   lav      sa
                 |  1For   |
                 |   G     |   sa       rav
                 |   :     |
                 |   :PRON |   ran
                 |   R  <-----     [6]
                 |   :     |
         [5]-----|sa :     |
                 | C       |
                 | CC      |
                 +---------+
           *
SIP(2) nn  R
            [2]   △
```

[1] R opens a replacement string bracket and [2] sets up SIP(2).

[3] The list at the position of the replaced symbol is starred (SIP Page 3d) since the string member requirement is satisfied by the replacement string.

[4] R_{pl} blocks V_s (NMBR), e.g. not we starts, and blocks sections below it. [5]

[6] R blocks various right adjuncts of N (PRON).

```
Sentence                                 | Output:<NV+; lan (T;the)experiment nan (K3NV-;which
Symbol                   V_p/s  (start)  |                                 [4]
                                         |                      nn [R;we ] started
---------------------------------------------------------------------------------------------
           *     *
SIP(0)  c#       N      V
                        lav      sa
                        sa       rav
                        ran
                          *
           *     *     +-------------+
SIP(1) nan K3    N     |    o, -     |      OBJECT REGISTER   (before executing OBJ-)
                       |  V-s        |        G+              e.g. start going
                       |     p <---------     N                    start the experiment
                       | lav         |  os    NG+                  start them working
                       | sa [2]      | [5] G  to V+                start to write
                       | ran         |    to
                       |    [3]      |    N
                       |             |  sa
           *           |             |  rav  [4] OBJECT REGISTER  (after executing OBJ-)
SIP(2) nn  R           |             |            ∅
                       | △           |            G+
                       |             |            to V-
                       |   C         |            NG-
                       | CC          |
                       +-------------+
```

$\boxed{1}$ The match of V_p at depth 1 closes the bracket on R (depth 2) and erases SIP(2).

$\boxed{2}$ There is also a match of S in the list of sentence adjuncts (another reading would be started).

$\boxed{3}$ There would have been another match of S in the list ran, but S was blocked by Instruction PRON carried by R.

$\boxed{4}$ OBJ- operates on the object strings of the verb start as follows:

$N \rightarrow \emptyset$		(step 1)
$NG^+ \rightarrow G^+$		(")
to $V^+ \rightarrow$ to V^-		(step 2)
$NG^+ \rightarrow NG^-$		(")
G^+ is dropped		(step 3)

$\boxed{5}$ OBJ+ (step 4 of OBJ-) causes the object string heads (leftmost symbols in the Object Register) to be posted. Since one of the string heads is $\emptyset$, this symbol is not copied, but causes the list to be placed below the string member section of the next position so that the list for the next sentence symbol will include the active list of SIP(0), (OBJ+, condition 2).

```
Sentence                  C        (and)    |  Output: [1]
Symbol                                      |
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
             *        *      +------+
SIP(0)      c#        N      |  V+  |
                     lan     | lav  |     sa
                     rn      |  sa  |     rav
                     sa      | ran  |
            *      *    *    |      |
SIP(1)ran   K3     N    V-   |      | [2]
                  lan  lav   |  os  |
                  rn   sa    |  sa  |
                  sa   ran   | rav  |
                             |  CC  <----
                             +------+
```

$\boxed{1}$ Associated with C is Instruction CONJ which delays the output at C until after the next match and controls the assembly of the list for the next sentence symbol. (III 5.1)

$\boxed{2}$ Since the string member section in the active position-list of SIP(1) is empty, END-STRING(1) is +, i.e. we are at the end of the depth 1 string when C occurs. The list set up by CONJ will then contain more entries then if C occurred in the middle of a string. (See notes 1 and 2 at the top of the next page.)

```
Sentence                          Output:<NV+;lan(T;the)experimentran(K3NV-;which
Symbol           P (for)               rn[R;we] started)and ran(3PK3NVP;for
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
                                                      [3]     [4]

List set up
by CONJ:        depth 0   |  N    |
                          |  lan  |
          [2]             |  rn   |
                          |       |
                depth 1   |  ran  | [1]
                          |   .   |
                          |   .   |
                          |  3P   <----
                          |   .   |
                          |   .   |
                          |   .   |
                          |       |
                          |  N    |
                          |  lan  |
                          |  rn   |
                          |       |
                          |  V-   |
                          |  lav  |
                          |  D̄    |
      *     *             |  CC   |
SIP(0)c#    N    V
                 lav   sa
                 sa    rav
                 ran

          *
SIP(1)ranP   K3    N    VP
                   lan  lav    sa
                   rn   sa     rav
                   sa   ran
```

[1] we have ran in the list instead of just K_3 (LIST, 2a), and D̄ and CC (LIST 3., 4);
[2] both SIP(0) and SIP(1) are represented in the list, (LIST 2b).

[3] The match of $_3P$ (depth 1) places, to the left of C, the closing bracket on the K_3 string (of the OUTPUT procedure of CONJ). Note: OUTPUT 3a2 is not shown here.

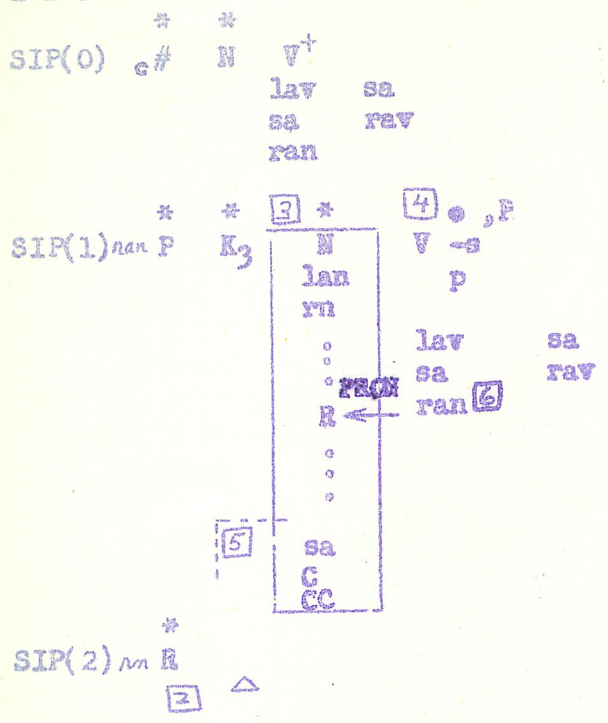
[4] $_3P$ opens a bracket (3a1 of the OUTPUT procedure of CONJ, which is the same as 3 of the usual OUTPUT procedure), correspondingly $_3P$ sets up a new SIP(1) (CONJ SIP 3a1).

```
Sentence                   Output:<NV+;lan(T;the)experimentran(K3NV-;which rn[R;we]
Symbol      K3 (which)           started) and ran(PK3NVP; for which
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
        *    *
SIP(0)c#     N     V
                   lav     sa
                   sa      rav
               [1] ran
          *     *
SIP(1)ranP  | K3 |  N    VP
            |    |       lav     sa
            | C  |       sa      rav
            | CC |       ran
```

[1] The list at K_3 is starred, since in this case K_3 is a string member and not a string head.

Sentence Symbol $R_{pl,sg}$ (you) | Output: ⟨$NV^+;_{lan}$(T;the)experiment$_{ran}$(K_3NV^-;which$_{rn}$ [R; we] started) and$_{ran}$(PK_3NV^- ;for which$_{rn}$ [R;you

$\boxed{1}$



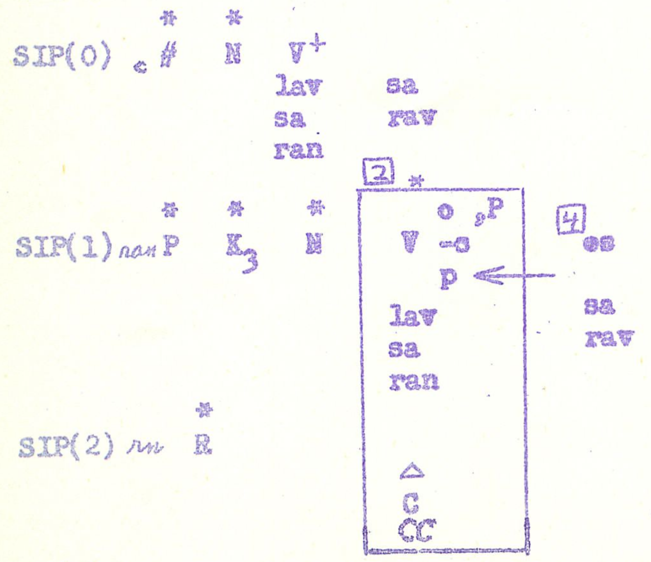
$\boxed{1}$ R opens a replacement string bracket, and $\boxed{2}$ sets up SIP(2).

$\boxed{3}$ The list at N is starred (SIP Procedure, 3e).

$\boxed{4}$ R_{pl} blocks V_s (NMBR), and $\boxed{5}$ the sections below R in the list (partial ordering), and $\boxed{6}$ certain right adjuncts of N (PRON).

Sentence Symbol V_P (be) | Output: ⟨$NV^+;_{lan}$(T;the)experiment$_{ran}$(K_3NV^-;which$_{rn}$[R;we] started) and$_{ran}$(PK_3NV^P;for which$_{rn}$[R;you] were

$\boxed{1}$



OBJECT REGISTER (before executing OBJB)

```
N
A
PN
.
.
G  E,+
[3] S  E,+
to^E  V^E,+
```

OBJECT REGISTER (after executing OBJB)

```
N
A
PN
.
.
G  E,P
[3] S  E,S,P
to^E  V^E,P
```

$\boxed{1}$ The match of V (depth 1) closes the bracket on R (depth 2) and erases SIP(2).

$\boxed{2}$ The match of a string member stars the list at this position.

$\boxed{3}$ OBJB transfers Instruction OBJP to G and S and V in the Object Register; OBJP replaces OBJ+, (shown by replacing + with P).

$\boxed{4}$ OBJ+ (Step 2 of OBJB) posts os, the list of the object string heads, in the string member section of the position list to the right (OBJ+, condition 1).

```
Sentence                                 | Output: <NV+;lan(T;the)experiment_ran(K3NV-;which_rn [R;we]
Symbol                     G             |    started) and_ran(PK3NV^p;for which_rn [R;you] were
                         (wait)          |    waiting
- - - - - - - - - - - - - - - - - - - - -|- - - - - - - - - - - - - - - - - - - - - - - - - -
                                              [1]
          *    *
SIP(0)  c#    N    V+
                   lav    sa               OBJECT REGISTER (before the execution of OBJP)
                   sa     rav
                   ran                       to V+

       *    *    *    *    * [2]             P_for N
SIP(1) ranP  K3   N    V   +------+
                           | os   |  [4]   OBJECT REGISTER (after the execution of OBJP)
                           |  .   |   os
                           |  .   |          to V^p
                           | G^E,P <- sa [2] ø      [3]
                           |  .   |   rav
                           |  .   |
                           |  .   |
                           | sa   |
                           | rav  |
                           | C    |
                           | CC   |
                           +------+
```

$\boxed{1}$OBJE blocks the opening of an object bracket and blocks the setting up of a new SIP (Step 1).

$\boxed{2}$ Also, the list at this position is starred and sa and ray are added to the next position (Step 2).

$\boxed{3}$ OBJP operates on the object strings in the object register as follows:

to $V^{+} \rightarrow$ to V^{p} (Step 1)

P (for) is compared with P (for), the head of the string containing the verb waiting, and the two are found to match. Therefore the adjusted instruction OBJ- is executed (Step 2), giving:

P_{for} N $\rightarrow$ ø

$\boxed{4}$ OBJ+ (Step 3 of OBJP) posts the list of object string heads below the string member section of the next position list since there is a ø in the list (OBJ+,2.); this allows the assembly procedure to include a position list from SIP(0) in the list for the next sentence symbol, i.e. for zero object the sentence returns to the including string.

```
                                          | Output: <NV+;lan(T;the)experiment_ran(K3NV-;which_rn [R;we]
Sentence                                  |    started) and_ran(PK3NV^p;for which_rn [R;you] were
Symbol                       V_P (be)     |    waiting) was
- - - - - - - - - - - - - - - - - - - - - |- - - - - - - - - - - - - - - - - - - - - - - - - -
                              [2] +-*--o-+                 [1]
        *      *                  |       |  [4]
SIP(0) c#      N                  | V  s  |  os
                                  |    p <-------           OBJECT REGISTER
                                  |       |                   N
                                  | lav   |  sa               A
                                  | sa    |  ran              PN
                                  | ran   |                   .
                                  |       |                   .
         *    *    *    *    *    |       |                   .
SIP(1) ranP   K3   N    V    os   |       |                   G^E,+
                                  | os    |              [3]  S^E,+
                                  | sa    |                   to^E V^E,+
                                  | rav   |
                                  | C,CC  |
                                  +-------+
```

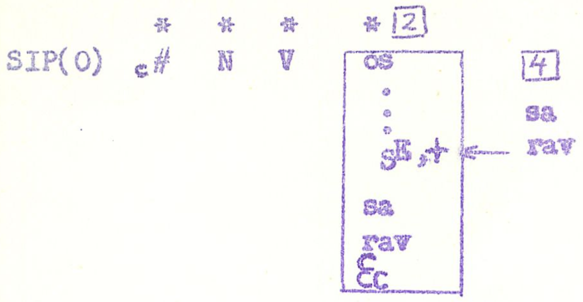
$\boxed{1}$The match of V_p (depth 0) closes the bracket on the PK3-string (depth 1) and erases SIP(1).

$\boxed{2}$The list at V is starred (match of a string member).

$\boxed{3}$There are no instructions for OBJB to transfer to V,G,S in the object register (Step 1).

$\boxed{4}$OBJ+ (Step 2 of OBJB) posts the list os in the string member section of the next position (Condition 1 of OBJ+).

Sentence Symbol V_p/S $\boxed{5}$ (interrupt) | Output: $\langle$NV$^+$; $_{lan}$(T;the)experiment$_{ran}$(K$_3$NV$^-$;which$_{rn}$[R;we] started) and$_{ran}$(PK$_3$NV^P;for which$_{ran}$[R;you] were waiting) was interrupted $\boxed{1}$



OBJECT REGISTER (before executing OBJS)
N

OBJECT REGISTER (after executing OBJS)
Ø $\boxed{3}$

$\boxed{1}$ OBJE blocks the opening of an object string bracket and blocks the setting up of a new SIP (Step 1).

$\boxed{2}$ OBJE stars this list and adds sa and ~~rrr~~ rav to the list at the next position (Step 2).

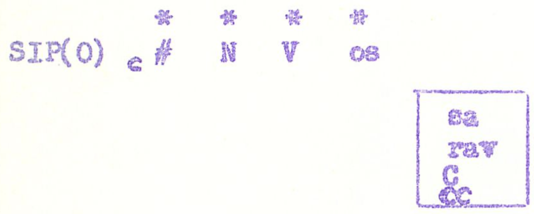
$\boxed{3}$ OBJS operates on the object string in the register:

$N \rightarrow Ø$ (Step 3)

$\boxed{4}$ OBJ+ posts the list of object string heads (which is empty in this case).

$\boxed{5}$ Note that the double classification is decided because no V appears in the list.

Sentence Symbol . | Output: $\langle$NV$^+$; $_{lan}$(T;the)experiment$_{ran}$(K$_3$NV$^-$;which$_{rn}$[R;we] started) and$_{ran}$(PK$_3$NV^P;for which$_{rn}$[R;you] were waiting) was interrupted$\rangle$ $\boxed{1}$



$\boxed{1}$ The occurrence of the period (end-sentence mark), when the list for the sentence symbol contains no string member, closes the center string bracket. This analysis is acceptable (p. 33).

List of Terms and Symbols

Footnotes

1 For helpful conversations and criticisms I wish to thank Robert McNaughton, Aravind Joshi, Henry Hiż and Zellig Harris.

2 Harris, Z. S., Computable Syntactic Analysis, National Science Foundation, Transformations and Discourse Analysis Projects, paper 15.

3 Gleitman, Lila R., Word and Word-complex Dictionaries, National Science Foundation, Transformations and Discourse Analysis Projects, paper 16.

4 For languages with less restricted nesting, rules appropriately different from Rules 1-3 could be formulated; or Rules 1-3 could be retained, and dependence instructions written for the type of intercalation found in the given language.

5 The occurrence of a beginning of a string is given according to Rules 1-3 as the beginning of a string which is either the center string or a string which adjoins or replaces an occurrence of a symbol in a string. The occurrence of a string is continuous from its beginning except for interruption by strings nested within it (Rules 2,3).

6 Differences in occurrence which correlate with position in a string could be handled by an extension of I,2.,2 (p.4). E.g., instead of Instruction SUBJ, which adjusts the list rn for the subject position, we could call the N in the subject position, N', and define a modified rn list, called rn'. Strings which contain N in the subject position would be written with N' in place of N, e.g. $HN'V^-$ rather than HNV^-; then rn' would be called out (by N') rather than rn, when the position lists for the string are assembled.

7 The arrangement inside the bracket is based on a suggestion by John McCarthy.

8 String member and sentence symbol N, V, R, have subclassifications which are used in the matching process: N is either N_1 or N_{sg} or N_{pl}; R is either R_{sg} or R_{pl}. V is either V_o or V_s or V_p or V_{pp}:

V_s(e.g. gives) singular present 3rd person; also includes am (V_s')1st person

V_o(e.g. give) plural present 3rd person; present 1st and 2nd person (except am, are); tenseless verb.

V_p(e.g. gave) singular and plural past; also includes are (V_p')plural

V_{pp}(e.g. given) follows have; is distinguished in its string positions (but not in its form) from the passive S.

9 An example of 8, i.e. a verb with omission due to passive S (see 2.4), plus omission due to the character of the string (using OBJ-) is: The evidence which his conclusions were based on. object of base: N P_{on} N; passive object of base: P_{on} N; passive object of base with omission due to OBJ-: P_{on}.

Footnotes (continued)

10 The following are the object strings of <u>be</u>, <u>have</u>, <u>do</u>, with examples of the bracketing of the object string due to OBJB and OBJE. G^{+} in O_B does not include <u>having</u> V^{+}_{pp}. These forms could be referred to by name with the classification scheme in FN 11.

<u>have</u>

$V_{pp}^{E,+}$ we have picked N;berries

to^{E} $V_{o}^{E,+}$ we have to stop G^{+};dreaming

N

N to V_{o}^{-}

NS^{+}

<u>be</u>

$$O_B\left\{\begin{array}{ll} G^{E,+} & \text{we are reading N;poetry} \\ S^{E,+} & \text{we were amazed} \\ A & \\ N & \\ PN & \\ D_2 & \end{array}\right.$$

<u>do</u>

$V^{E,+}$ They do_{rav}(D,not) accept N;checks

N

11 Linguistically, W is not an adjunct, although it can be computed this way. Alternatively, W, together with <u>do</u>, <u>does</u>, <u>did</u>, and the tense suffixes, can be considered a required part of a center string N W V^{+}. Then <u>have - en</u>, <u>be - ing</u>, and somewhat differently, <u>be - en</u> could be considered right adjuncts of W, e.g. classification U. In the dictionary <u>have</u> and <u>be</u> would be given alternative classifications, e.g. <u>have</u> = U/V.

12 Special provision is required to carry out OBJE before II 4.4 (SIP Procedure). When $V_{(b)}$ is bracketed according to FN 11 (in a later version), OBJB and OBJE will not be required.

13 Linguistically, the article is not an adjunct; but it can be computed as one by having instructions CNTN(3.2) and LFTE(3.5).

14 The pronoun <u>we</u> may occur followed by, e.g. (<u>on the committee.</u>) <u>We</u> has certain properties of the noun not shared by other pronouns, e.g. also <u>We teachers</u> but not <u>I teacher</u>.

15 Additional provision is required for successive <u>N's</u>, e.g. an "N's counter" (starting with the second N's) which causes the first PDN bracket (for q = 0) to be placed to the left of p pairs of brackets, where p is the value of the N's counter. E.g. we may have <u>(young)(boy's)mother</u> and <u>((young)boy's)mother</u> and <u>((boy's)friend's)mother</u> but not <u>(boy's)(friend's)mother.</u>

16 By "repeat" we do not mean that the words of the sentence are repeated but that structurally equivalent elements occur on both sides of the C, i.e. the same string or segment of a string. The term "preceding string" in this context refers to the string which is interrupted by C, if C occurs in an unfinished string, or to any string we are in (as defined on p. 6) if C occurs at the end of a string. In the term "string headed by C" the word "head" appears not in the technical sense of the rest of the paper where "to head" means to open a bracket in the sentence formula, set up a new SIP etc. (II 4.1-4), but in the sense of Instruction CONJ which replaces II 4.1-4 for C. "The string headed by C" means the string set up for computation by C (via CONJ) when C is matched.

17 Alternatively, A C and CC could be added to the lists on every level.

18 A sentence adjunct may intervene between C and "the string headed by C" as defined by the CONJ procedure (FN 16), i.e. when a sentence adjunct occurs after C there are two cases: one covered by the existing CONJ (this is the case where C repeats part of a preceding sentence adjunct, e.g. <u>The outcome was predictable in his opinion and mine</u>); and one for which additional machinery is required (this is the case where the sentence adjunct intervenes, e.g. <u>The soldiers fought well but in my opinion needlessly</u>). To cover the latter case, we would add to LIST:

> 4. A list of sentence adjunct heads, <u>sa</u>, having the depth of the deepest SIP is added to the list at n+1. We can also include here the right adjuncts of particular C: e.g. <u>also</u> for <u>CA</u>, <u>else</u> for <u>CO</u>.

We would then add to MATCH:

> If the matched symbol is in the list <u>sa</u> added in 3. above we save the list at n+1 and revert to II 4.1-4 until the END-STRING indicator of the <u>sa</u> string is +. Then the saved list is substituted for C in the next use of II 4.1. When a symbol from the saved list is matched, the CONJ procedure is resumed.

19 When C occurs at an interior point of a string, we do not provide for a repetition which begins at an interior point (or beginning) of a string of lesser depth, e.g. not <u>We considered a plan which the</u> committee, and proposed a solution which the secretary, rejected. This restriction can be relaxed by removing the condition END-STRING (k) = +, on LIST 2b.

20 If one wishes to include checks of well formedness, one might want to insure that when two verbs share an object string, as in the example of OUTPUT 3a2, the second verb will list for possible computation only those object strings which are common to both verbs, e.g. <u>We arranged and played his music</u> but not <u>We arranged and played on the piano</u>. The correct list of object strings in this case is the intersection of the object strings of the two verbs. One may add to CONJ SIP 2a: "If an inserted position list has V in the string member section, Instruction OBJC is attached to the V as the first OBJ instruction." OBJC would be defined as follows:

Instruction OBJC -

1. Perform the indicated OBJ instructions except OBJ+
2. Retrieve or, if necessary, recompute the object strings of the verb of SIP(k)
3. Take the intersection of the results of 1 and 2, and erase the other object strings.
4. Perform OBJ+

21 A subset of verbs (called "aspectual" in other papers of this series), e.g. like to, are similar to $V_{(b)}$: the main verb is the one which comes after $V_{(b)}$ or the aspectual verb. Thus like to may be included in the element Y which is omitted in the XZ type string: She likes to play tennis and he golf.

22 After C it is possible to repeat parts of two preceding strings, provided that the later string is the object or adjunct of the earlier. In such cases the repetition from the earlier string is XZ and the repetition from the later string is XY (rarely XZ or X).

Examples of XZ - XY repetition after C:

1. Where the later string is the object:
   I claim that the men waited and you that they didn't.
2. Where the later string is an adjunct:
   She writes poems which no one can read and he novels which everyone can.

23 The transformational character of strings headed by CT(than) and CS(as) is close to that of strings headed by CA, except that the list of permitted repetitions from the preceding string is greater after CT, CS. However, the distinction among the various kinds of permitted repetition is lost when we consider only the word classes, as in the present procedure, and not the actual words. Hence in terms of word classes, what can occur after CT, CS is virtually every repetion from the preceding string as well as many other strings provided they are lacking a symbol which is present in the preceding string. Note also footnote 27.

24 In the dictionary, both has two classifications: Tq and CCA; either and neither are T4 and CCO, CCN; as is $\emptyset_3$ and P and CCS and CS; adjectives ending in -er are A plus simultaneously CCT; more, less, and rather are D and T_q, plus simulatenously CCT.

25 "Non-sublisted" is specified to allow e.g. She is either a very smart person or well-read. Otherwise we would be requiring that or follow very.

In some cases, a sentence adjunct intervenes between CC and the string which should be tagged, e.g. The people either, for the most part, approving or applauding his work... . The machinery for this case has not been included. The more complete statement is: If the symbol following CC is a sentence adjunct head, there are two analyses possible:

1) CCα1 is applied as usual, e.g. The people who either having applauded his work or having encouraged him in other ways... .
2) CCα1 is delayed until after the sentence adjunct (and its adjuncts) has been computed, e.g. The people, either for the most part approving or applauding... Compare footnote 18.