X. THE REVISED SYNTACTIC PROCESSING FOR SMART

A. Lemmon

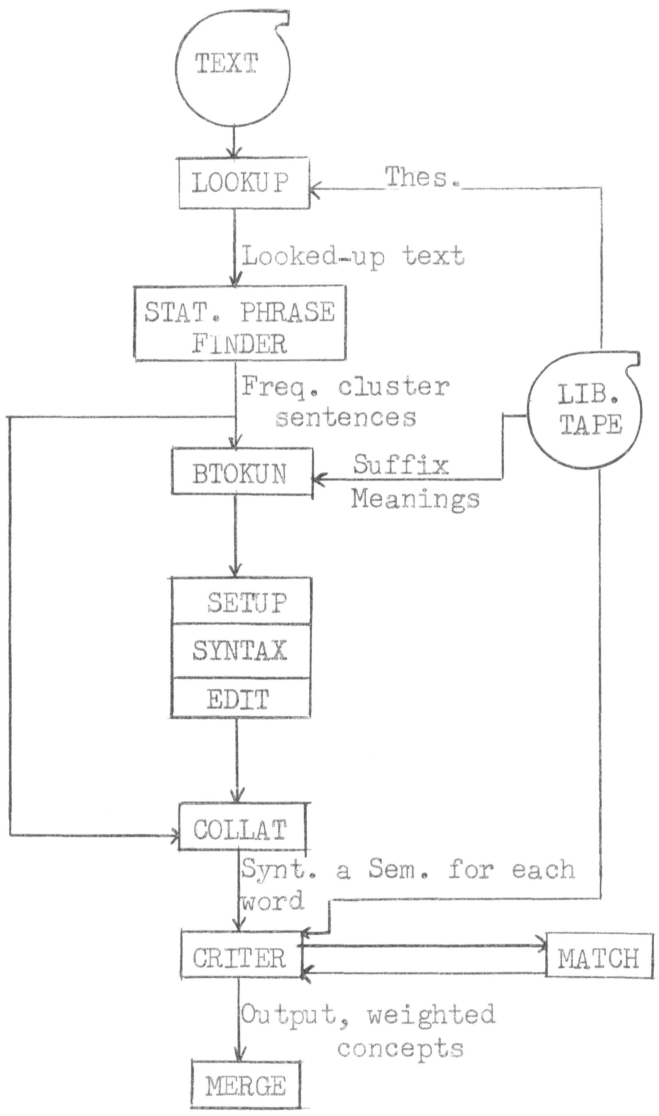
1. Introduction

The purpose of this section is to give a brief description of the use and operation of the syntactic processing section of the new SMART system. The philosophy and principles involved are the same as those previously described in Information Storage and Retrieval Report No's. ISR-5, 7, and 8. However, due to changes in the SMART system as a whole, the various routines of the syntactic section have been rewritten and expanded, and their operating instructions changed. The new syntactic section is thus not compatible with the old one. Figure 1 shows the structure of the complete syntactic section.

2. Use and Function of System Routines

A. BTOKUN (Binary-TO-KUN Converter)

This BTOKUN routine has been left relatively unchanged. Its function is to convert from the new SMART text format to the format accepted as input by the Kuno Multiple-Path English Syntactic Analyzer. Two specific changes have been made in BTOKUN. The first is to accommodate the new binary tape format. The new BTOKUN routine expects a tape in which documents are introduced as 12-word records, and each sentence consists of one record of 10-word items. All sentences received by BTOKUN are prepared for the Analyzer by this routine.



Overall Organizations of the Syntactic Section

Figure 1

The other major change consists in modifying BTOKUN for use with the INOUT routine to effect all input and output processing. This routine permits the computer simultaneously to compute and perform input and output operations.

B. CRITER (CRITERion tree routine)

The purpose of CRITER is to find occurrences of certain standard phrases (known as "criterion trees") in the sentences of a text. CRITER makes use of the output of the English Syntactic Analyzer. A phrase is said to "occur" in a sentence if, for each "node" in the phrase, a corresponding node may be found in the sentence, such that:

(1) No two phrase nodes have the same correspondent in the sentence;

(2) If a phrase node depends upon another, then the sentence correspondent of the first must depend on that of the second, and;

(3) The correspondent of each phrase node must satisfy certain restrictions on semantic values and syntactic function.

When a match has been found, the fact is recorded in an output table. This table contains a list of concept numbers and also a weight for each one. Each of the concept numbers refers to a concept embodied, not necessarily in one word, but in a phrase. The weight indicates how frequently, and how significantly the corresponding concept appears. Ordinarily a weight of twelve is allotted for each appearance of a concept, The user may, however, select any of the numbers 0,3,4,6,12,18,24 or 48 as the weight allotment of a given occurrence.

These options, as well as others to be described, are controlled by special indicators stored with the tree, called "output concept numbers" (or "OCN's"). Each OCN specifies (1) a concept number, and (2) the weight with which the number is to be recorded. These specifications are decided by the user when he makes up his library of criterion tree phrases.

The user has another option for specifying output concept numbers. Instead of specifying a specific number, for example "280," he may say "The concept number(s) already attached to the correspondent of tree node "1." Thus the number recorded in the output table will come directly from the sentence. This option is used in connection with contexts devoted to important ideas, e.g., "the central problem is ..." or "the subject of this paper is ...". Thus, a criterion tree phrase might be arranged to match "the central problem is ...," with node 1 (say) corresponding to the blank. Then the concept numbers of the correspondent of node 1 (that is, the word in the blank) could be recorded in the output table. In this way important concepts can be recorded according to this context alone.

It is also possible to attach any concept number to the correspondent of any node. In this way idioms may be translated and important information moved from place to place in the sentence. Appendix 1 gives an example of this option. The phrase "error correction" is assigned concept number 280. Whenever this phrase appears in a sentence, the number 280 will be attached to the appropriate node in the sentence, so that it can participate in later processing.

In the above discussion the phrase "later phrase" has been used. This is a reference to the fact that the phrases are stored on tape in an ordered sequence. Each sentence is matched against all the phrases in turn. The next sentence is then matched against all the phrases, and so on. Successive sentences do not affect each other in any way. Successive phrases do, however, influence each other's matches as described above.

Matches in themselves result in no permanent record. The final output of CRITER is taken from a table (mentioned briefly above), initially empty, in which are recorded the desired output concept numbers. The output table contains only one entry for each such number; a total point count is kept for each entry and updated whenever required. At the end of each document, the data in the output table is sorted into concept-number order and written out on tape, together with the name of the document.

The output concept numbers thus come from two different sources: either they are specified by the user, or else they consist of numbers already attached to the correspondents of certain phrase nodes. Similarly, the concept numbers specified in these ways may be used in either of two ways: they may be inserted into the output table, or attached as new concept numbers to the correspondents of certain phrase nodes. The same output concept number may be used in either or both of these ways.

A given sentence node may correspond to more than one concept number. This may arise because of ambiguity of the word represented by that node, or because other concept numbers have been attached by previous trees.

If the user specifies that the "concept number" of such a sentence node should be recorded in the output table, all of the numbers will be recorded, but with a weight (point count) equal to the specified weight divided by the number of recorded concept numbers. Thus the total weight added to the output table will be as specified by the user.

C. TRECND (TREe maker, CoNDensed format)

This routine writes user-specified phrases (criterion trees) on magnetic tape, in the format expected by CRITER. TRECND accepts cards (or, more precisely, card images on magnetic tape) and interprets the user's specifications, writing out trees according to those specifications.

A given tree has four main parts or specifications:

(1) individual identification;

(2) structure;

(3) node restrictions in terms of semantic and syntactic values;

(4) action to be taken when match is found (OCN's).

Of these the program usually supplies (2) and parts of (3) and (4). The user may overrule these standard interpretations at any time. The program uses a file of standard phrase types, at present 14 of them. These standard types include all the structual information and whatever syntactic values are required. Also, unless otherwise specified, all of the OCN's will be attached to the correspondent of one particular node, known as the "head" of the phrase. In the case of a noun phrase, for instance, the head is the noun. Under user control the numbers can be attached to other nodes as well.

The user supplies two kinds of information: (1) a six-character alphabetic name; and (2) a serial number, less than 32,768. The user may elect to have the program assign a number one greater than the last serial number, or to omit the serial number entirely. All other information must, however, be supplied by the user. If it is omitted, the routine will reject the tree by not writing it out, and by giving an error message.

The format of the card images accepted by TRECND is as follows. There are four fields:

(1) The BCD Name field, columns 1 through 6.

(2) The OCN field, extending from column 7 up to but not including the first column containing I,D, (, or /. This field consists of subfields, each introduced by = .

(3) The Relations field, beginning with I,D, (, or / , and extending to, but not including, the first $. It is composed of one or more node fields, separated by / .

(4) The Specification field, beginning after the first $ and extending up to, but not including, the first blank. It is composed of subfields separated by commas.

The BCD name field may consist of any six characters, except that it may not be filled completely with zeros, or blanks or five blanks followed by an asterisk. Also it may not begin with a slash.

The output concept number field may not be empty. Each subfield is composed of up to four types of sub-subfields:

(1) A decimal integer. This is the concept number which is to be written out. It must be the first sub-subfield, if present. There may be only one sub-subfield of this type.

(2) The letter N, followed by a decimal integer less than 16. There may be any number of these, subject to the restriction that no two may have the same integer. These sub-subfields identify the nodes whose correspondents specify the OCN's. No sub-subfield of this type may appear with one of type 1; at least one must appear if there is none of type 1.

(3) A period, followed by nothing or by a decimal integer less than 16. There may be any number of these, providing that they are all specified by different integers. Each one refers to a node to whose correspondent the concepts of this OCN are to be attached. If the period is not followed by an integer, it means that the standard "head" node is to be ignored.

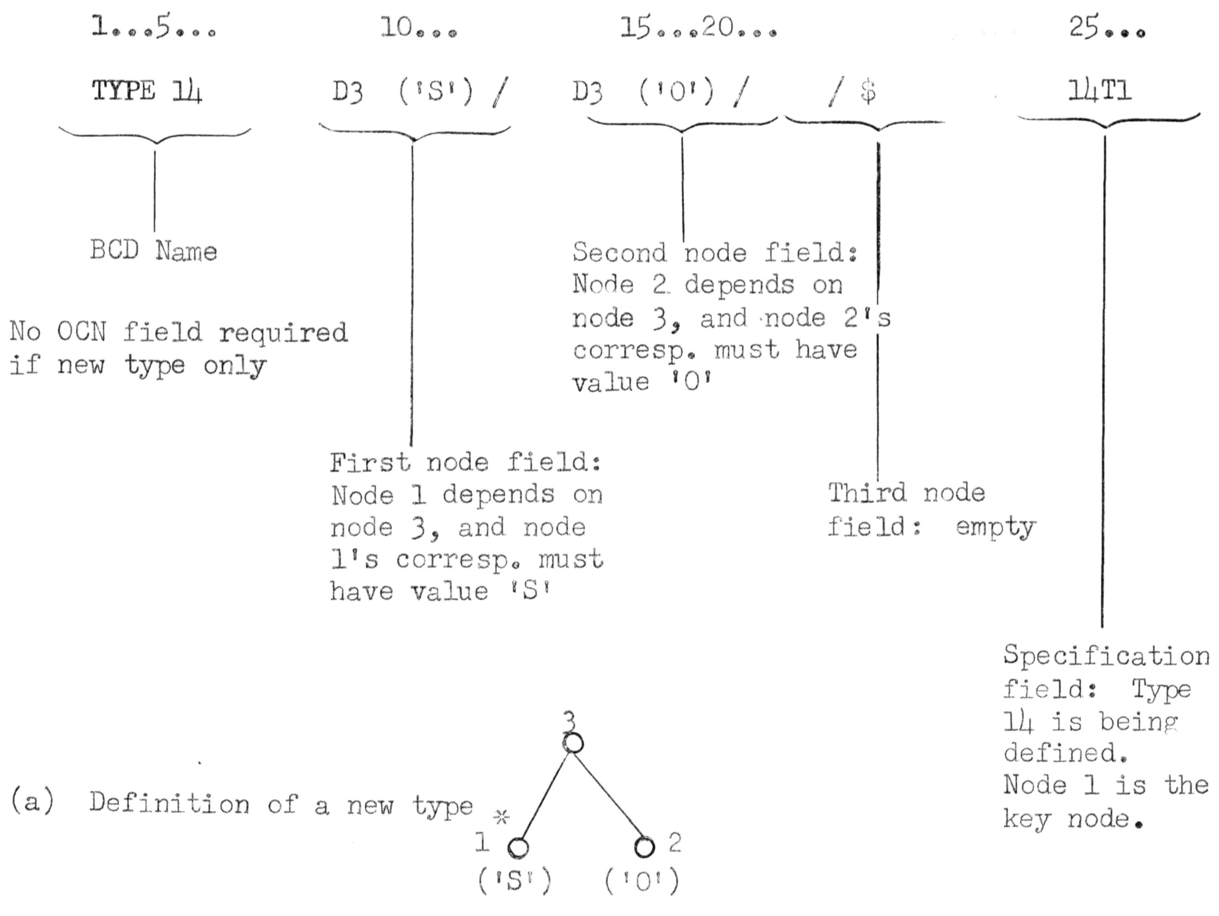
(4) An asterisk, followed by a decimal integer or nothing. Only one of these may appear. The integer becomes the point count (weight) for this OCN. If this field is absent, a weight is automatically assigned so that the total weight of all OCN's is 12. All unspecified weights will be automatically assigned the same weight. If the integer is absent but the asterisk is present, a weight of zero will be assigned.

The program is designed to automatically make the most logical interpretation whenever no option is specified by the user. Thus it is sufficient to have an OCN field consisting simply of one or more integers, each preceded by an "equals" sign (as in the old TRECND format). Each of these will be given a point count of 12/ (number of OCN's) and will be attached to the head node of each phrase generated by this card image.

The relations field consists of "node" subfields. Each node subfield consists of an optional "relation" part. The connection part consists of the letter I or the letter D, followed by a decimal integer denoting the parent (D) or ancestor (I) of the present node. This part will ordinarily be omitted and the information supplied by the prestored types. The relation part consists of one or more "relations" enclosed in parentheses. Each relation consists of one or more "generators" separated by commas. Generators are of two types: (1) unsigned decimal integers, interpreted as concept numbers; and (2) any BCD character, enclosed between apostrophes, the character being interpreted as a syntactic value. There must be at least one relation in the relations field.

The specifications field is as described in report ISR-8, Section VI, except that type 0 may be specified, in which case the tree is constructed entirely from data given on the card, and that new tree types may be constructed from the description in the relations field of the card, including both the structural and syntactic information. This is specified by the last subfield, which consists of a decimal integer, followed by the letter T or P, followed by from 0 to 15 distinct integers less than 16, separated by commas. The integer before the letter indicates the number of the tree type being (re) defined; the numbers after the letter denote the key ("head") nodes. The program will write a message on the output tape indicating that the new type has been defined. Also if the letter is P, binary card images are written on tape B4, suitable for loading the new types with the program the next time.

This description closes with some examples of criterion tree formats and interpretations of the various card fields. The motivation behind these fields has been discussed earlier (ISR-8, Section VI), and is also taken up in Appendix 1 to this section.


1...5...
10...
15...20...
25...
TYPE 14
D3 ('S') /
D3 ('O') /
/ $
14T1
BCD Name
No OCN field required if new type only
Second node field: Node 2 depends on node 3, and node 2's corresp. must have value 'O'
First node field: Node 1 depends on node 3, and node 1's corresp. must have value 'S'
Third node field: empty
Specification field: Type 14 is being defined. Node 1 is the key node.
3
*
1
2
('S')
('O')

(a) Definition of a new type

Illustration of TRECND Input Formats

Figure 2

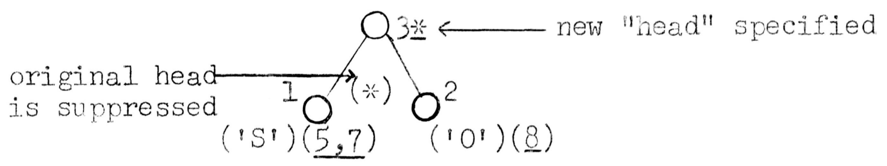
```
1 ...    5 ...    10...    15...    20...    25...

 E X A M P L = 95.3.       (5 ; 7 )/ (8) $ 14 / 9
 \_________/
  BCD Name
```

OCN Field. One OCN; 95 will be attached to node 3 but not the head node, and also given a weight of 12

First node field, one relation: the corresp. of node 1 must have value 5 or value 7

Second node field. Corresp. of node 2 must have value 8

Specification field. Tree is of type 14 and has serial number 9



(b) Use of a defined type. Data from this card are underlined; other data comes from storage.

Figure 2. (continued)

3. Technical Considerations

The present part of this report covers some special problems in connection with the criterion tree routine, CRITER. The basic function of CRITER is the interpretation of syntactic-function strings. With one exception (the inclusion of conjunctions by the new CRITER), the interpretation of a sentence by the present CRITER version is identical with that of the previous version. However, the representations are different, and the algorithm has been considerably streamlined.

The new algorithm is based on the fact that each string in the analysis corresponds to a path in the tree, ending at a unique node. (See ISR-5, Secs. II and III.) Moreover, the tree is considered to be ordered in such a way that the path corresponding to the first string is at the left edge of the tree. Successive strings (one per word) then correspond to paths further to the right.

Thus it is sufficient to determine where each path deviates from its predecessor.

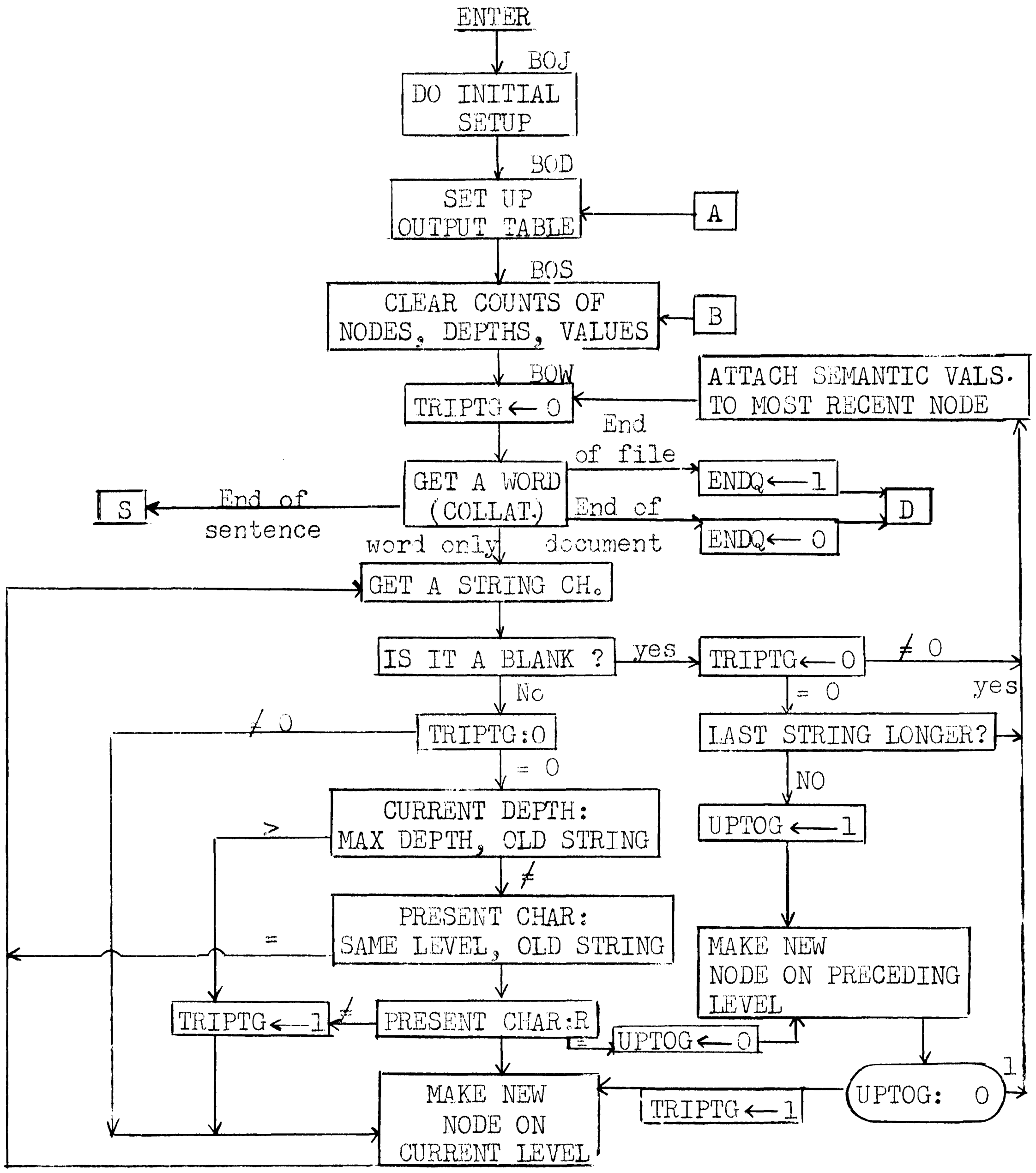
The strings are processed one by one in the order in which they appear in the sentence. The tree structure of the sentence is built up as the necessary information becomes available. Thus each string corresponds to the right-hand edge of the structure as it exists at the time the string is processed.

When a string is processed, it is examined one character at a time, starting from the left. The path corresponding to a string deviates from its predecessor at the first character that differs from the corresponding character in the preceding string, and thereafter becomes a new branch to the tree structure. A deviation also occurs at the last character if the present string is not an R, a deviation is made at the preceding character. The reasons for these actions are discussed in ISR-5, Sec. II, p. 12.

The new CRITER algorithm replaces the cumbersome TABLE of the old version, by three vectors: OLDBUF, DEPVEC, PARENT. OLDBUF holds the preceding string, broken up into characters. DEPVEC keeps track of the right-hand edge of the tree; the ith entry in DEPVEC is the number of the most recent node at depth (or level) i. Finally, PARENT shows the parent (if any) of each node.

Once a deviation is detected, new nodes are created, forming a new path. The numbers of a new node is one more than the total number of nodes found so far — that is, the numbers are taken in serial order. The parent of each node is the most recent node assigned one level back; that is, the appropriate entry in DEPVEC. When a node has been created in this fashion, its number is recorded in DEPVEC, since it now becomes the most recent node on its level.

These various actions are shown in the flowchart of Fig. 3. Note first that many boxes are abbreviated, e.g. "Made new node"; also note the double subroutine linkage to MATCH, which actually finds occurrences, and from MATCH to HIT for each occurrence found. This scheme is used to disconnect MATCH from the rest of the program, so that it can be programmed separately. HIT is the entry to CRITER which causes information to be recorded from matches.



CRITER Flowchart

Figure 3

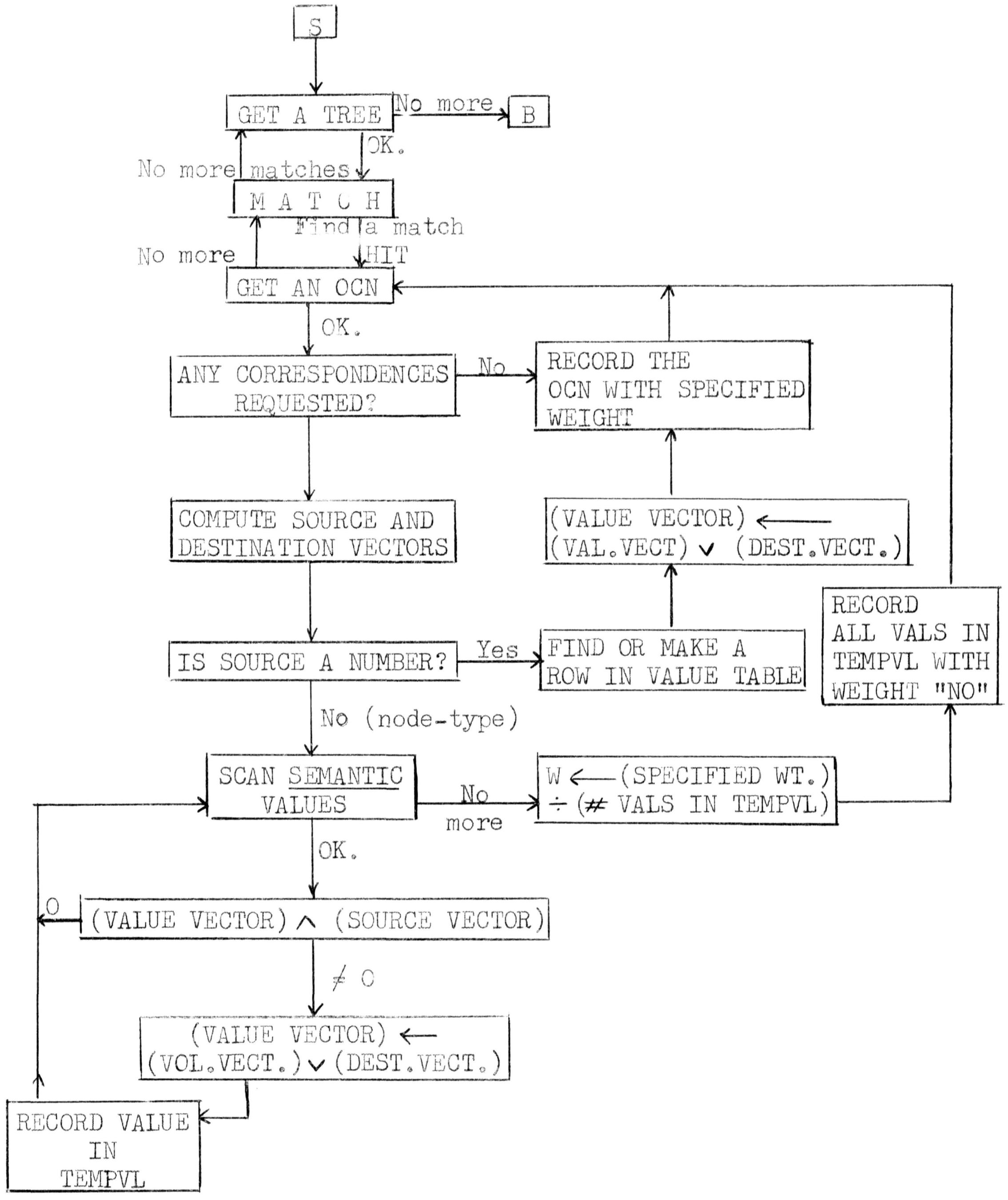

S
GET A TREE
No more
B
OK.
No more matches
M A T C H
Find a match
No more
HIT
GET AN OCN
OK.
ANY CORRESPONDENCES REQUESTED?
No
RECORD THE OCN WITH SPECIFIED WEIGHT
COMPUTE SOURCE AND DESTINATION VECTORS
(VALUE VECTOR) ← (VAL.VECT) ∨ (DEST.VECT.)
IS SOURCE A NUMBER?
Yes
FIND OR MAKE A ROW IN VALUE TABLE
RECORD ALL VALS IN TEMPVL WITH WEIGHT "NO"
No (node-type)
SCAN SEMANTIC VALUES
No more
W ← (SPECIFIED WT.) ÷ (# VALS IN TEMPVL)
OK.
0
(VALUE VECTOR) ∧ (SOURCE VECTOR)
≠ 0
(VALUE VECTOR) ← (VOL.VECT.) ∨ (DEST.VECT.)
RECORD VALUE IN TEMPVL

Figure 3 (continued)

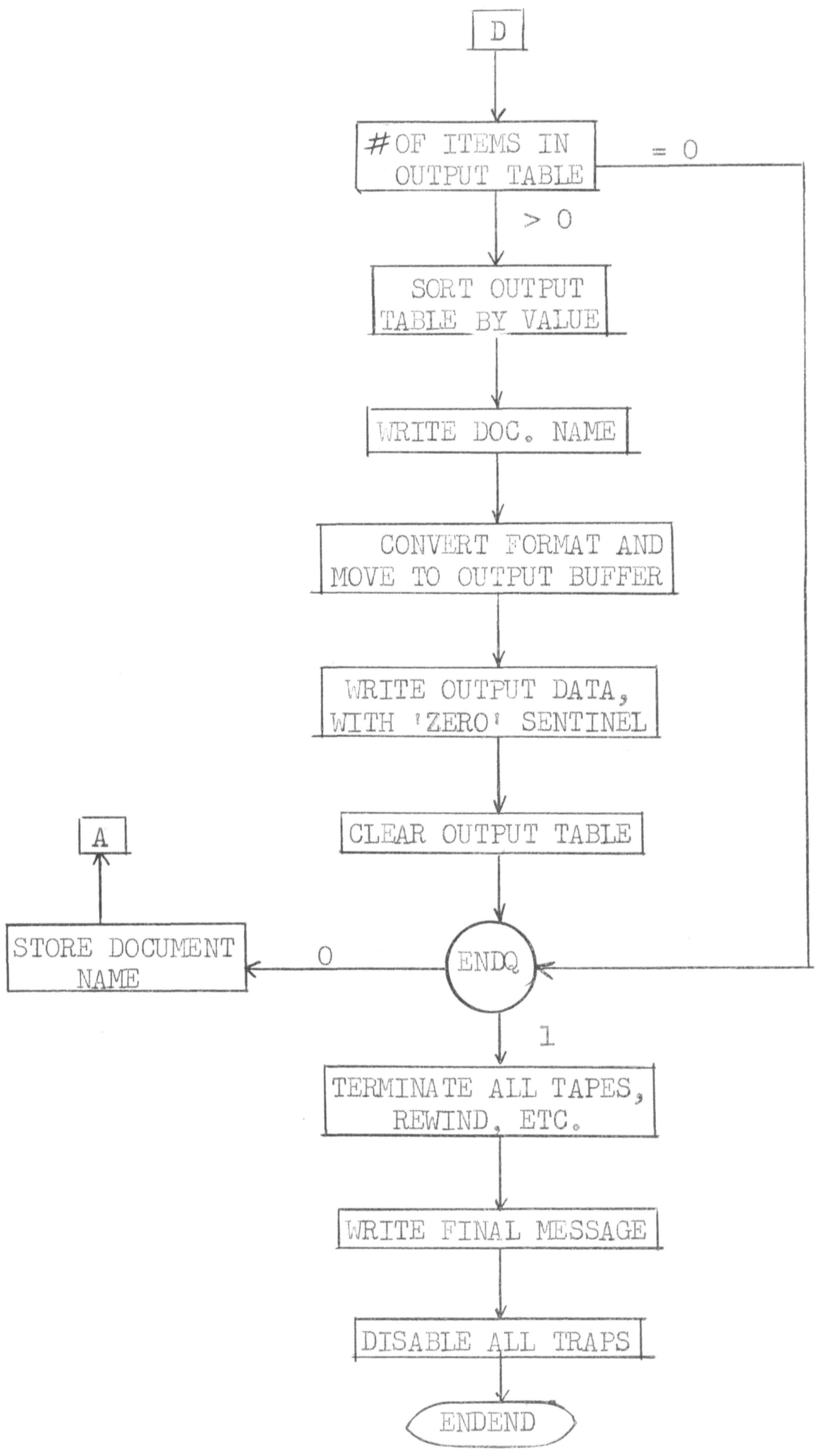
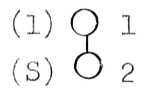
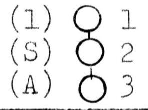
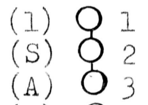
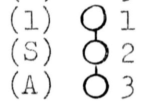
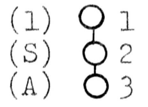
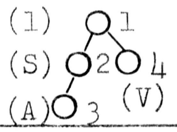
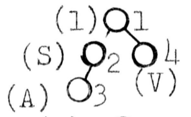
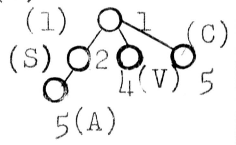

D
#OF ITEMS IN OUTPUT TABLE
= 0
> 0
SORT OUTPUT TABLE BY VALUE
WRITE DOC. NAME
CONVERT FORMAT AND MOVE TO OUTPUT BUFFER
WRITE OUTPUT DATA, WITH 'ZERO' SENTINEL
CLEAR OUTPUT TABLE
A
STORE DOCUMENT NAME
0
ENDQ
1
TERMINATE ALL TAPES, REWIND, ETC.
WRITE FINAL MESSAGE
DISABLE ALL TRAPS
ENDEND

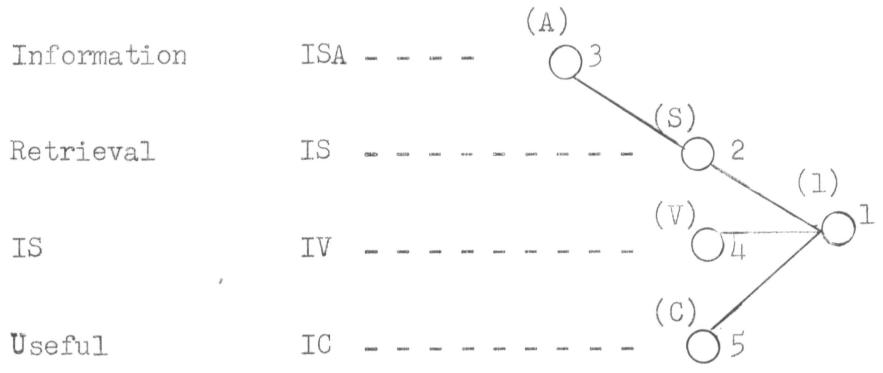
Figure 3 (continued)

STRING POINTER		OLDBUF	DEPVEC	PARENT	PARTIAL STRUCTURE
INFORMATION	1SA	-	-	-	-
	↑	1	1	0	(1) 01
	1SA ↑	1,S	1,2	0,1	
	1SA ↑	1,S,A	1,2,3	0,1,2	
RETRIEVAL	1S ↑	1,S,A	1,2,3	0,1,2	
	1S ↑	1,S,A	1,2,3	0,1,2	
IS	1V ↑	1,S,A	1,2,3	0,1,2	
	1V ↑	1,V,A	1,4,3	0,1,2,1	
USEFUL	1C ↑	1,V,A	1,4,3	0,1,2,1	
	1C ↑	1,C,A	1,5,3	0,1,2,1,1	

Stages in Interpretation of Strings in a Sentence by CRITER

Figure 4

(a) Sequence of processing



(b) Correspondence of tra with sentence

Figure 4 (continued)

Figure 4 shows the successive contents of OLDBUF, DEPVEC, and PARENT during the analysis of a small sentence.

4. Summary

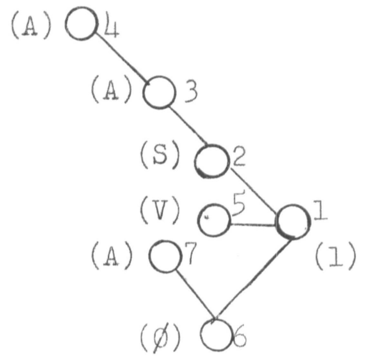
The syntactic section of the SMART system has been revised and expanded to make it compatible with changes being made in the system as a whole. These revisions do not require fundamental changes in the algorithms, but rather consist of changes in format details. While the routines were being rewritten, the opportunity was taken to streamline cumbersome procedures and to provide new and more general capabilities.

APPENDIX 1

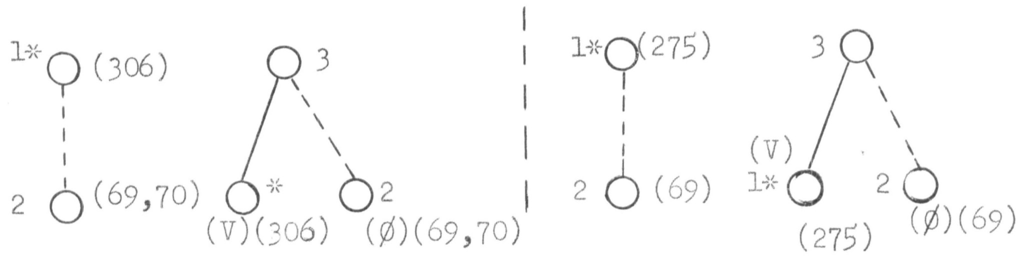
This appendix demonstrates the operations of CRITER using a simple sentence, to show the addition of concepts to sentence nodes.

Word	String	Conception
Error	1SAA	69
Compensating	1SA	275
Codes	1S	281
Have	1V	1019
Many	1∅A	1086
Applications	1∅	614

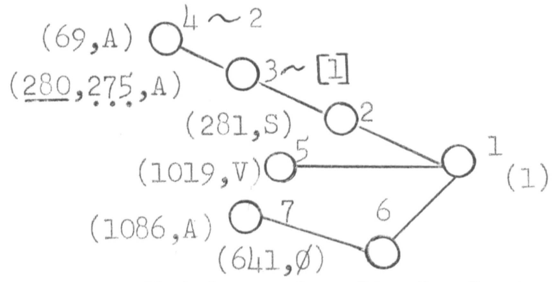
Structure



(a) The sentence and its analysis



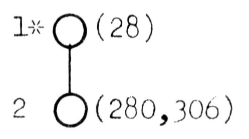
(b) Trees for "error corections"



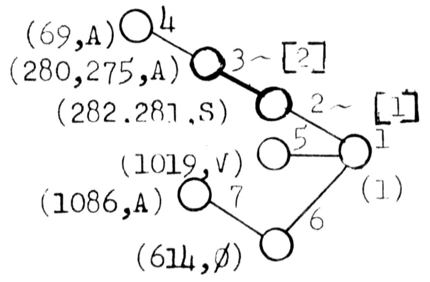
Matches of a Simple Sentence

Figure 5

(c) Match with third tree, and new additions resulting.



(d) Trees for "correction code"



Key: Concept numbers already in the sentence, allowing a match to occur, are underlined - - -

Concept numbers attached to the sentence as a result of a match are underlined ———

Matches are shown by bold lines ○—○ tree nodes to which sentence nodes correspond are shown in brackets [ ] Count 12 points for concept 282.

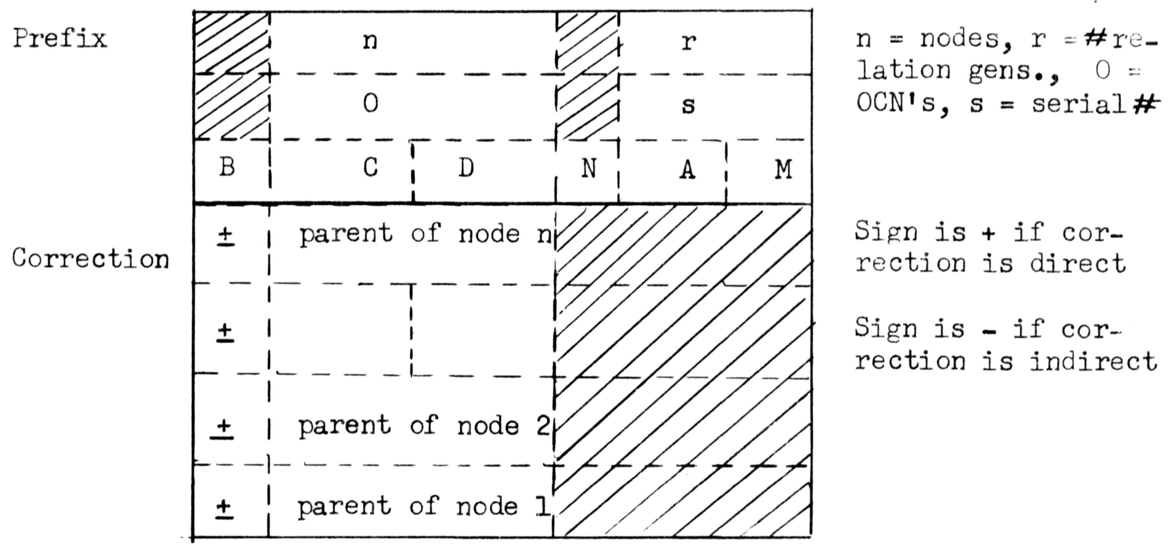
(e) Final match, <u>with</u> added concept no.

Note that the match described in (e) above would not have occurred had the concept number 280 not been attached to node 3 of the sentence.

Figure 5. (continued)

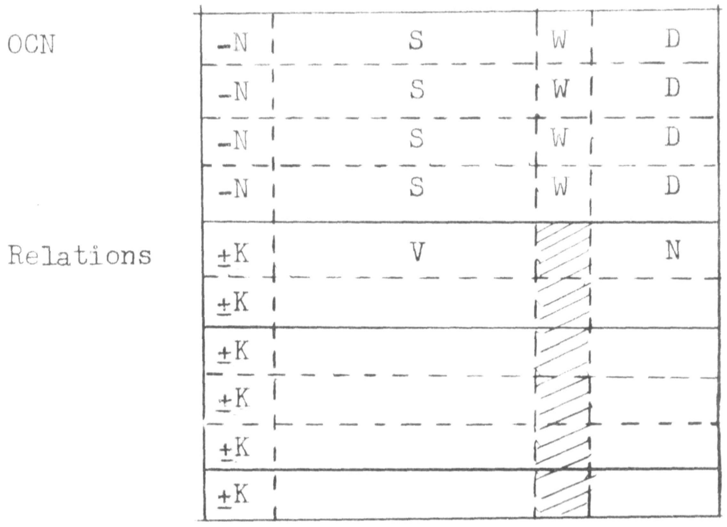
APPENDIX 2

The binary format of the criterion tree phrases has been radically changed. The correction matrices have been replaced by a vector of pointers, identifying the parent (direct corrections) or ancestor (indirect), if any, of each node. This reduces the space required, by one half. Also, relations generators are now recorded in chronological order (because of new table-searching policies in CRITER), and no longer bear relation numbers, which are now implicit in context. Finally, the OCN's have been greatly expanded. Each OCN now occupies a full word bearing all the information specified. Examples of the new format follow:


Prefix
n
r
O
s
B
C
D
N
A
M
Correction
±
parent of node n
±
±
parent of node 2
±
parent of node 1
n = nodes, r = #relation gens., O = OCN's, s = serial#
Sign is + if correction is direct
Sign is - if correction is indirect

Binary Tree Format

Figure 6



N is 0 if S is a specific number.

N is 1 if S is a vector of nodes whose correspondents supply the OCN.

D is a vector of nodes to whose correspondents the OCN(s) will be attached.

W is a weight code:

0	1	2	3	4	5	6	7
0	3	4	6	12	18	24	48

The generators ofeach relation are consecutive. The sign of the last (or only) generator in each relation is + ; other generators are -.

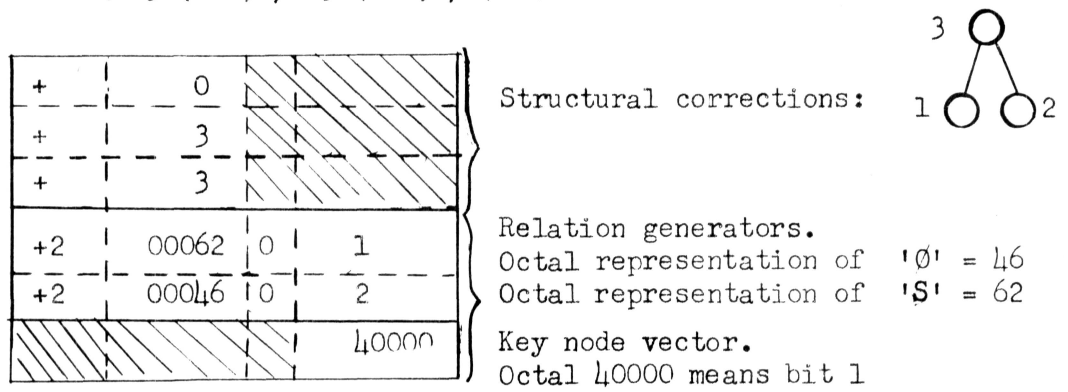
K is 0 if V is a concept number .

K is 2 if V is the actual representation of a syntactic character.

N is the number of the node to which the relation applies.

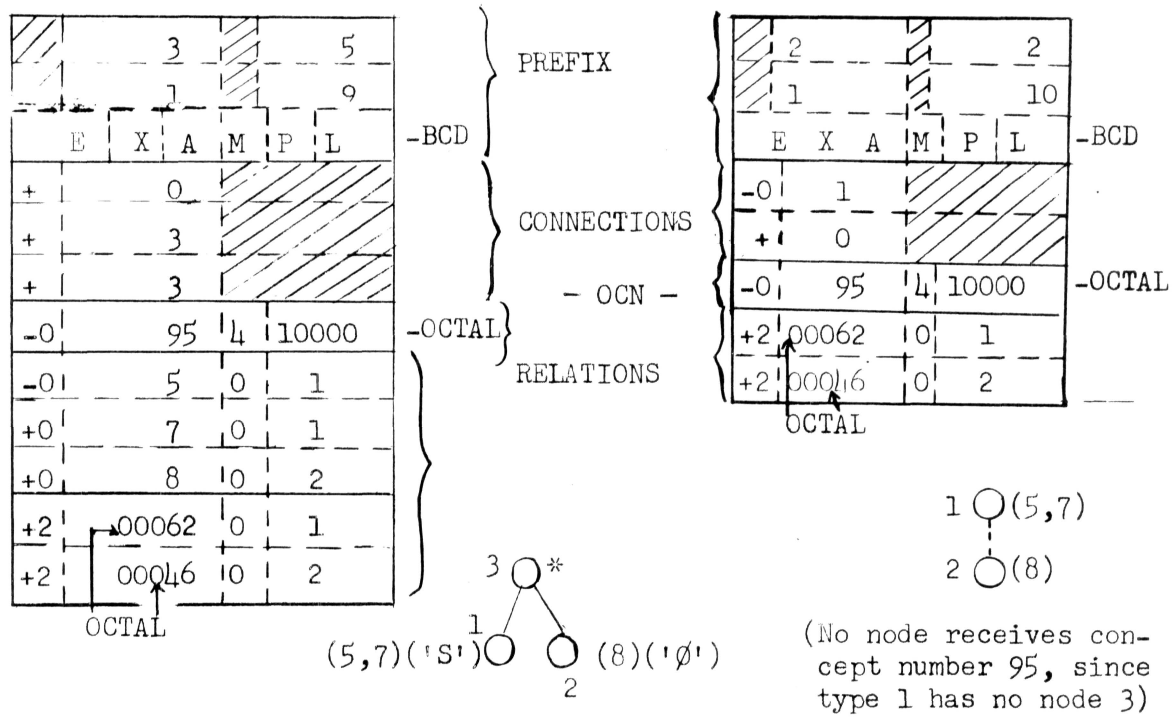
Figure 6. (Continued)

TYPE 14 D3 ('S') / D3 ('0') / $ 14 T 1



(a) Type definitions as recorded in memory

EXAMPL = 95.3. (5,7) / (8) $ 14 / 9, 1+



(No node receives concept number 95, since type 1 has no node 3)

(b) Binary formats as written on tape

Specific Examples of Binary Tree Formats

Figure 7