

VI. SYNTAX AND CRITERION PHRASE PROCEDURES

Alan Lemmon

1. Introduction

The syntactic procedure incorporated into the SMART system may be separated into three distinct parts. The first is an editing program (BTØKUN) which converts binary data produced by the SMART dictionary look-up process into the binary coded decimal (BCD) form required by the syntactic analysis programs. The second is the Kuno Syntactic Analyzer, described in Reports NSF-8 and NSF-9 by the Computation Laboratory of Harvard University, and not further discussed in the present report.^{1,2} The last and functionally most important part of the syntactic procedures deals with the comparison of selected, syntactically analyzed sentences extracted from the documents with a dictionary of so-called "criterion" trees or phrases. These phrases are preanalyzed and furnished with concept indicators. If the proper match is obtained between a given sentence in a document and a criterion phrase, the corresponding concept indicators can then be used as additional concept indications for the document.

In the present section the input editing (BTØKUN), as well as the "criterion" routines are described in detail. The actual sentence matching procedure and the methods used to setup the criterion tree dictionary are covered in later sections by Sussenguth, Evslin and Lewis, respectively.

2. The Input Editing Problem

The SMART system processes data largely in binary form. However, the syntactic analyzer used by the system expects to receive input data in BCD form. For this reason, a code conversion operation is necessary. Additional format changes are, however, also required: in the first place, a selection must be performed to obtain those sentences from a document which are to be syntactically analyzed. Second, and more importantly, a code construction must be effected. Indeed, the dictionary supplies syntactic codes for the stems of words, and also identifies the suffix (if any); the analyzer, however, requires single codes giving information assembled from both the stem (i.e. part of speech) and from the suffix (tense and number). It is thus necessary to correlate the syntactic information pertaining to stem and suffix before the analyzer takes over. BTØKUN is a routine to handle these conversion tasks.

The details of the input format required by the analyzer are given in NSF Reports 8 and 9^{1,2} and in the manual of operating instructions supplied with the syntactic analyzer. Basically, the form is as follows:

- (1) A "start" sentinel (one record).
- (2) Sentences:
 - (a) A count of the words in the sentence (one record).
 - (b) The words of the sentence. Several records may exist for one word, showing different possible syntactic types (codes) for that word.
 - (c) An "end of sentence" sentinel (one record).
- (3) An "end of text" sentinel (one record).
- (4) End of file.

The syntactic codes used by the syntactic analysis programs are chosen from a list of about 180 BCD symbols. Each of these indicates a part of speech, and, if the word is a noun, pronoun, or verb, then the tense, number, and/or case are also given. Fortunately, as will be seen, the arrangement in the latter case is simple and direct (see Table 1), and corresponds rather closely to the organization of the output of the lookup program used in SMART.

Three different inputs are needed by BTØKUN. The first is, of course, the text as recorded on magnetic tape in text order as nine-word binary items. Each item corresponds to one text word, and has the following format:

- (1) Four binary words, consisting of 24 BCD characters which store the text word itself;
- (2) One binary word:
 - (a) In the decrement (position 3-17), the number of the sentence;
 - (b) In the address (position 21-35), the position of the word in the sentence.
- (3) Two binary words, consisting of six 12-bit bytes, each of which represents a "concept number" or "semantic value";
- (4) Two binary words, divided into nine 8-bit bytes:
 - (a) Eight bytes (64 bits), part of speech code;
 - (b) One byte (eight bits), identifying the last suffix of the word, if any.

These nine-word items are recorded consecutively on magnetic tape. BTØKUN does not require any special blocking. However, at most 1,000 nine-word items may appear on the text tape (i.e. 9,000 binary words).

Type		Infinitive (I)	Finite Forms			Past Part.	Present Part.	Gerund
			Singular	Plural	Either			
Intransitive	1	II1	VI1S	VI1P	VI1C	PI1	RI1	GI1S
	2	II2	VI2S	VI2P	VI2C	PI2	RI2	GI2S
	3	II3	VI3S	VI3P	VI3C	PI3	RI3	GI3S
Transitive	1	IT1	VT1S	VT1P	VT1C	PT1	RT1	GT1S
	2	IT2	VT2S	VT2P	VT2C	PT2	RT2	GT2S
	3	IT3	VT3S	VT3P	VT3C	PT3	RT3	GT3S
	4	IT4	VT4S	VT4P	VT4C	PT4	RT4	GT4S
	5	IT5	VT5S	VT5P	VT5C	PT5	RT5	GT5S
	6	IT6	VT6S	VT6P	VT6C	PT6	RT6	GT6S
	6-1	IT6 1	VT6S1	VT6P1	VT6C1	PT6 1	RT6 1	GT6S1
	7	IT7	VT7S	VT7P	VT7C	PT7	RT7	GT7S
	7-1	IT7 1	VT7S1	VT7P1	VT7C1	PT7 1	RT7 1	GT7S1

Nouns

Type	Singular	Plural	Either
Any noun use	NØUS	NØUP	NØUC
Subject or object	NØVS	NØVP	NØVC
Numbers	NUMS	NUMP	NUMC

Organization of Syntactic Word Class Codes for Input
to Syntactic Analyzer

TABLE 1

Following the text, a file of up to 50 sentence numbers is required, preceded by a load command to load the file. The load command stores in its decrement a count of the number of sentences ($N \leq 50$) which are to be processed by the syntactic analyzer, and the address part of the load command contains $77461_8 - N$. Each sentence number, within the table, refers to one of the sentences to be analyzed, and the program assumes that one sentence is included in the text for each number in the list. In order to be properly processed, the numeric codes must appear on the list in increasing numeric order.

The last input file required by BTOKUN is a file of suffix codes, furnished as part of the SMART library tape (tape B5). This file (the second on the tape) consists of two records:

(1) Directory:

- (a) 256 words, one for each suffix number $n(0 \leq n \leq 255)$, to indicate (in the decrement) how many codes pertain to that suffix, and to refer (in the address) to locations above octal 67372.
- (b) One word, whose decrement is the sum of the decrements of the preceding 256 words, and whose address is 67372. This word serves as an I-Ø command for the next record.

(2) The list of suffix codes.

This record contains as many words as are referred to by the directory. Each of these suffix codes consists of one binary word. Each binary word is interpreted as five BCD characters, followed by five bits, followed by one

blank bit. A given suffix code is derived from a complete syntactic code by judiciously replacing some of the characters of the original syntax code by zeros, and by indicating the pattern of replacement in the five-bit byte which is appended. Each bit within the five bit part is zero if and only if the corresponding BCD character which precedes is zero. The final (sixth) bit is always set equal to one. The construction of the suffix code is illustrated by the examples of Fig. 1.

Suffix	Codes
No suffix	VOOP0 10010 1, IOO 0 10010 1, 000S0 00010 1
- ED	VOOCO 10010 1, POO 0 10010 1
- S	VOOSO 10010 1, 00OP0 00010 1
- ING	GOOSO 10010 1, ROO 0 10010 1
- AL	ADJ 11111 1

Sixth
bit

(a) Suffix codes

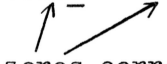
Example and Interpretation of Partial
(Suffix and Stem Codes)

Figure 1

Stem	Codes
ADD	OT10 01101 0
BE	BEO 11011 0
CAPIT-	N UO 11101 0, OT10 01101 0
ACT	N UO 11101 0, OI10 01101 0
M USE	NOUS 11111 1
MICE	N UP 11111 1

(b) Stem codes

Original code: VT1S_ 11111 1
 Zeros inserted: 0..0. 0..0.

 Partial code
 resulting: OT10 01101 1

 zeros correspond

(c) Meaning of parts of a partial code

Figure 1 (continued)

3. Internal Processing - The BT~~O~~KUN Routine

The input files are first read in, including the text, the list of sentences to be processed, and the suffix code list. An initial sentinel is then written on the output tape.

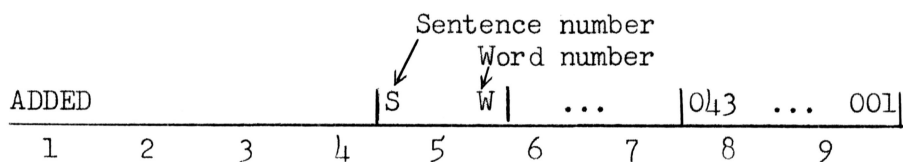
The routine now takes the first entry from the list of sentence numbers, and retrieves the corresponding sentence (using the fifth word in

the nine-word items as key). When that sentence is found the number of words contained in it is counted and the count is written on the output tape. The words in the sentence are then processed one by one, to construct the corresponding complete syntactic code.

Each of the first eight 8-bit bytes (in the last two words of a nine-word item) represents one of about 180 stem syntax codes. Their format is similar to that used for the suffix codes: five BCD characters, followed by six bits. Again the five characters are derived by substituting zeros for certain of the characters in a complete code. The next five bits have the same meaning as before. The final bit, however, is "one" only if no substitutions have been made in the construction of the code.

The program can now capitalize upon the fact that information can be extracted to some extent independently from stem and suffix codes. Consider, for example, the procedure pertaining to verbs. Reference to Table 1 shows that the first and fourth characters of a given syntax code indicate the mode and number of a verb, and can thus be derived exclusively from the suffix (see Fig. 1(a)). On the other hand, the second, third, and fifth characters indicate the type of stem and can thus be formed independently of the suffix.

For a word such as ADDED, appearing in the input, the dictionary lookup program (Sec. IV) produces a nine-word item of the following form:



where 043 is the numeric code for T1 verbs, and 001 is the code of the suffix-ED. Under 043 in the table of stem codes (which is a part of the BTØKUN program), BTØKUN finds "OT10 01101 0". Under suffix number 001 the program finds "VOOCO 10010 1" and "POO 0 10010 1". The stem code is checked against each of the suffix codes to see if they are exactly complementary, as in the example shown:

Stem code denoting T1 verb	} → OT10_01101 0	Suffix code for past	OT10_01101 0
Suffix code showing finite form	} → VOOCO 10010 1	participle	POO_0 10010 1
	<u>VT1C_11111 1</u>		<u>PT1_11111 1</u>

When such complementary pairs are found, they are combined, and the result is taken as a complete syntax code. For the example given, BTØKUN would supply the (correct) codes VTIC, PT1 (finite T1 verb, past participle of T1 verb) for "ADDED".

The processing of nouns is similar. For nouns, the first character is always N; the class occupies the first three character positions. The number (singular, plural, or "common"-irrelevant or indeterminate, e.g. "DEER") occupies position four, and character number five is always blank. Thus stem codes will be NØUO 111010, NØVO 11101 0, etc.; suffix codes are of the form 000S0 00010 1, 000P0 00010 1, 000C0 00010 1. Note here that the suffix/stem partition of complete codes is different for nouns (4/1235) and for verbs (14/235). Thus noun stem codes cannot by mistake be combined with verb suffix codes or vice versa.

A large number of words exist which do not require partial codes. Among these are all those which exhibit no suffixes -- prepositions, irregular forms, pronouns, forms of "be", "have", "do", and so on. Also, the codes for adjectives and adverbs are not divided. Many suffixes are included in the suffix table which give rise to complete syntax codes by themselves, no matter what stem code is supplied. For instance, -NESS always gives rise to the code "NØUS". Under these circumstances the code juxtaposition operation can be skipped.

If no codes are determined by the procedure previously described, because stem and suffix codes do not match, or because no partial codes were provided by dictionary lookup, an arbitrary code list is assigned as shown: VTIC, VIIC, NØUC, ADJ, AVI. This set of codes covers most word classes except pronouns, prepositions and other common words which are entered in the dictionary. BTØKUN writes out the list of syntactic codes generated for each word on the output tape in the following order: the code, the word and "homograph" numbers, the text word (24 characters), and text, sentence, and word number. (The word number thus appears twice).

The sixth and seventh words of a nine-word item contain semantic information. This information is required by the criterion routine described later in this section, and is therefore transferred into COMMON storage where it is not disturbed. (COMMON 451, i.e. octal 76556 and preceding locations).

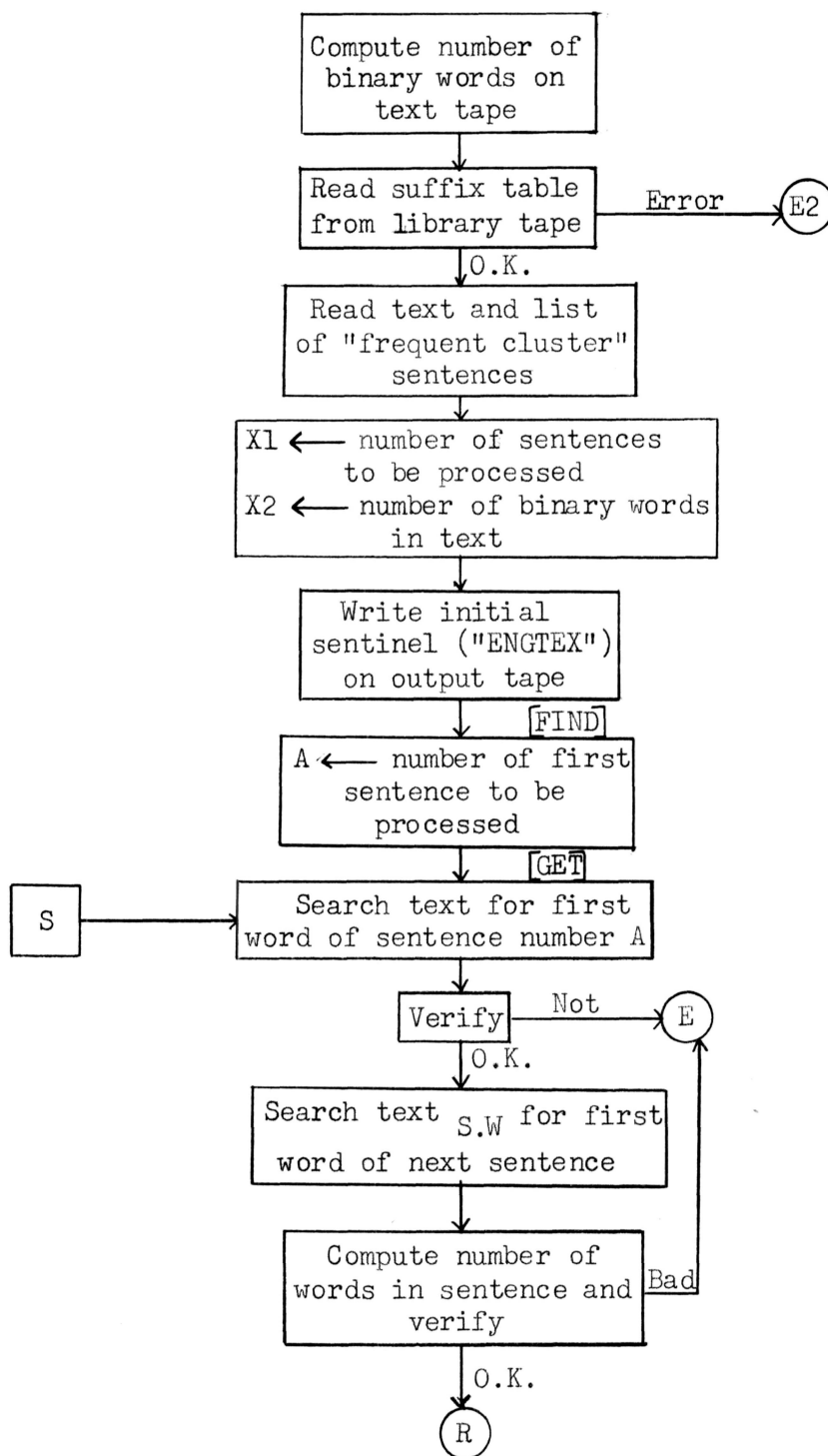
When every word in the sentence has been processed as described an end-of-sentence sentinel is written on the output tape and the next

sentence is processed. If the sentence-number list is exhausted, the routine writes an end-of-text sentinel and an end-of-file on the output tape, the tape is rewound, and the first section (SETUP) of the syntactic analyzer is called. The complete process is detailed in Flowchart 1. The square brackets near the flowchart box indicate the symbolic name of the program areas that perform the corresponding operations.

4. Organization and Operation of the Criterion Routine (CRITER)

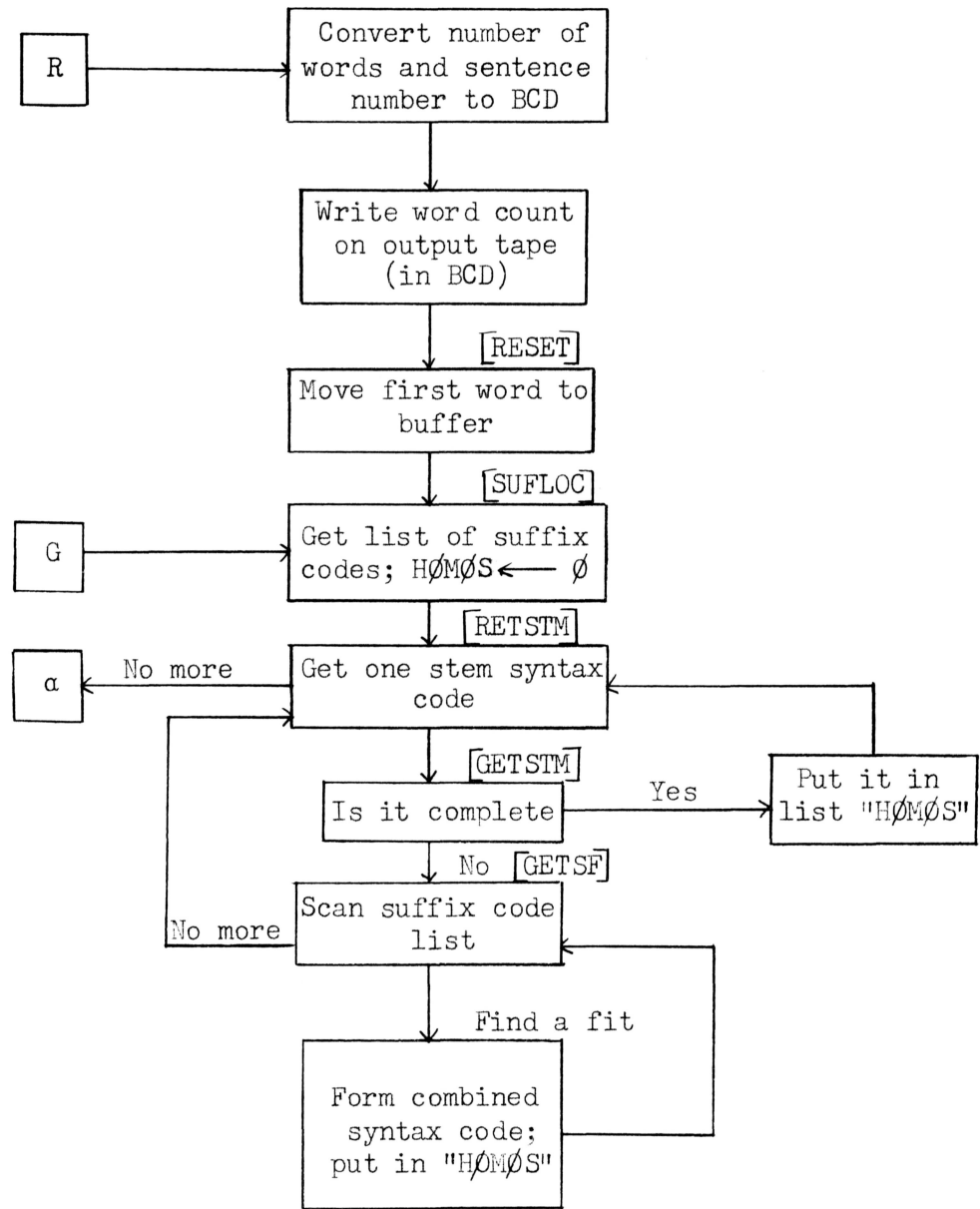
Following the syntactic analysis of the texts, the criterion routine performs three functions. First, it accepts the output of the syntactic analyzer, and compute from it a set of binary connection matrices. Secondly, it lists all semantic and syntactic values appearing in a sentence, and the words to which they pertain. Finally, it supervises the matching of this data against each member of the library of criterion trees. The overall organization of this routine is shown in Flowchart 2.

In the output of the syntactic analysis program, one record is formed for each text word, containing among other information a string of up to 24 BCD characters (see Fig. 2). Each character of such a string represents one structure or substructure in the sentence. The type of structure (e.g. subject, phrase, object, etc.) is indicated by the character: "S" for subject, "P" for phrase, etc. The structure indicated by each character is contained in, depends on, or modifies that indicated by its predecessor in the string (if any). Thus in Fig. 2, "one" in each string denotes a declarative clause standing alone, the "S" in strings one and two indicates a subject of that clause, and so on.

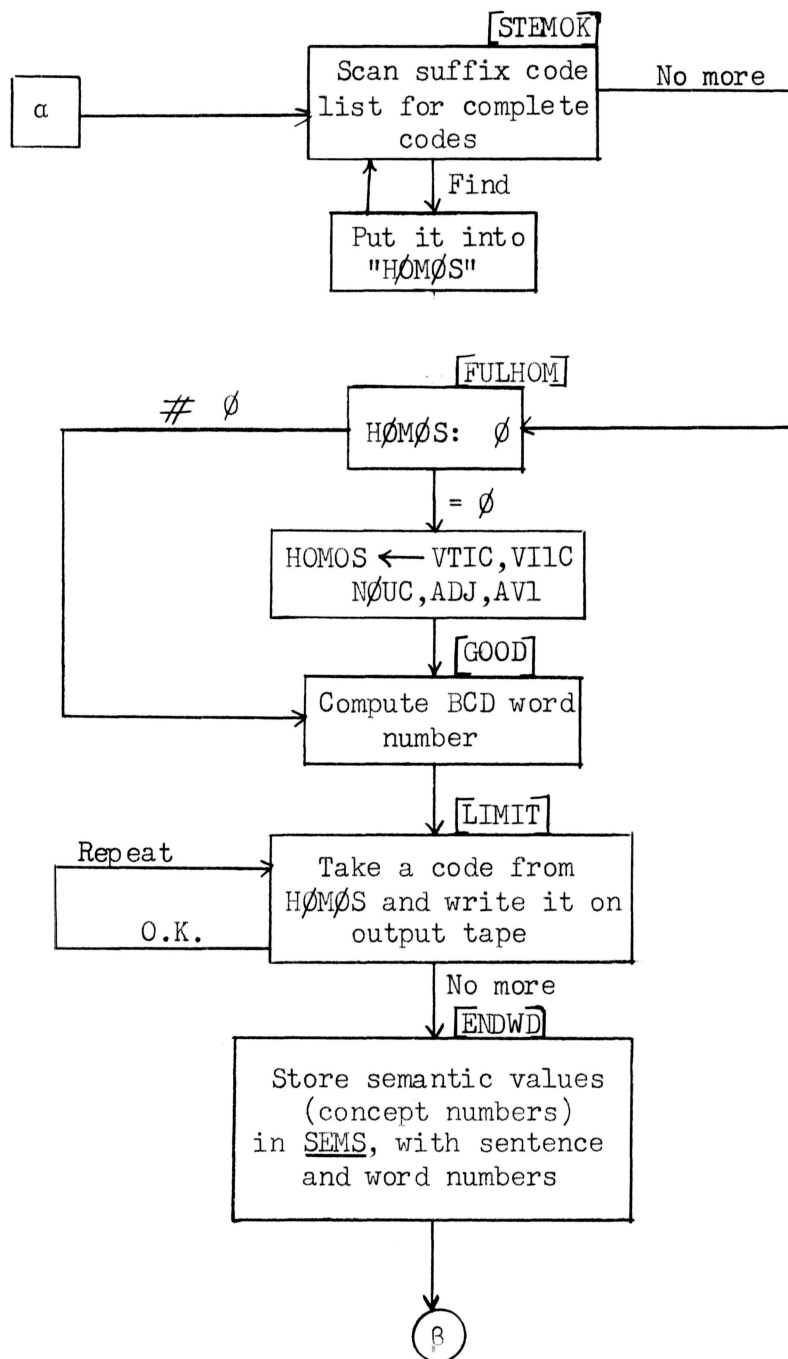


Operations of BT/KUN Editing Routine

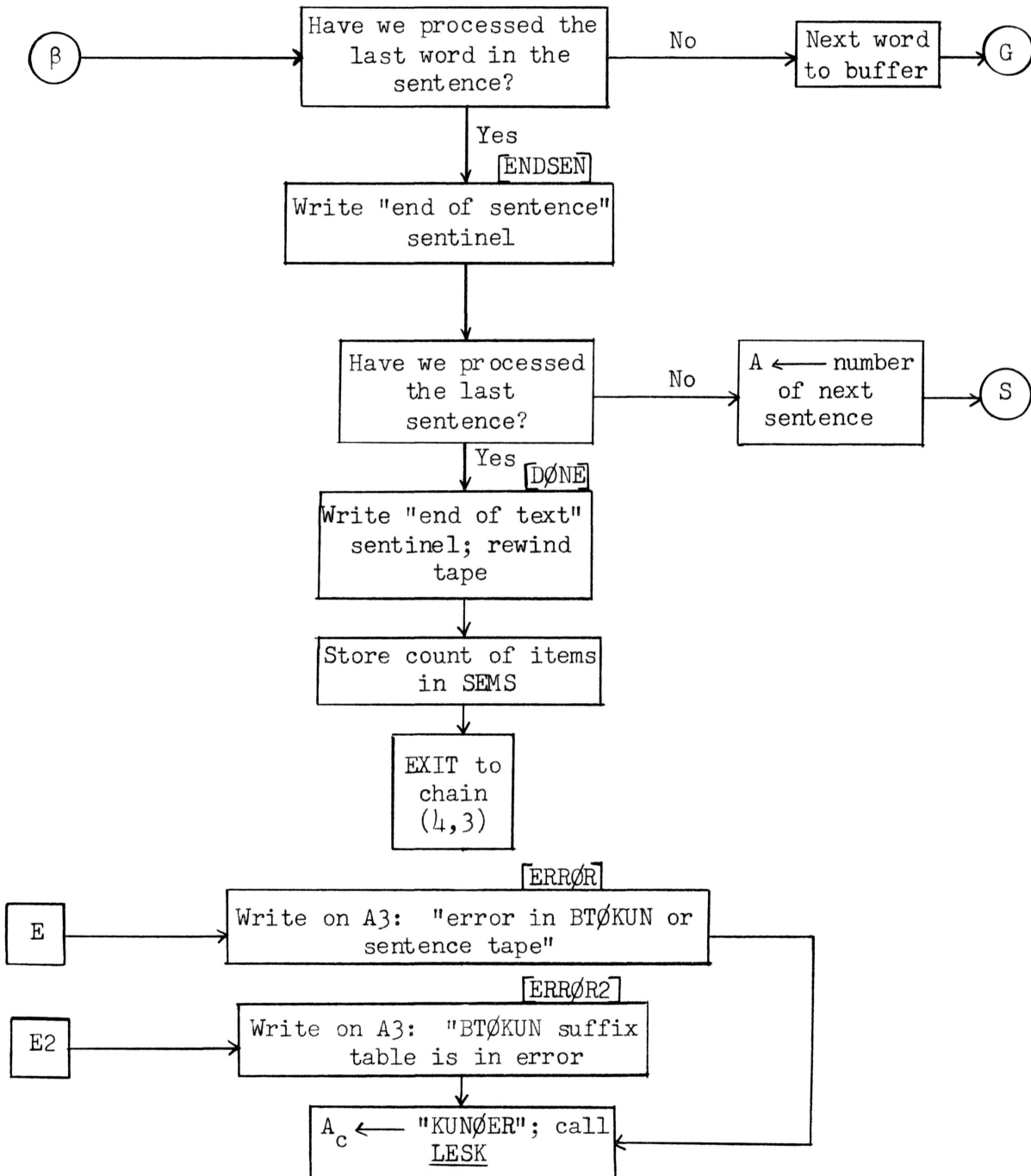
Flowchart 1



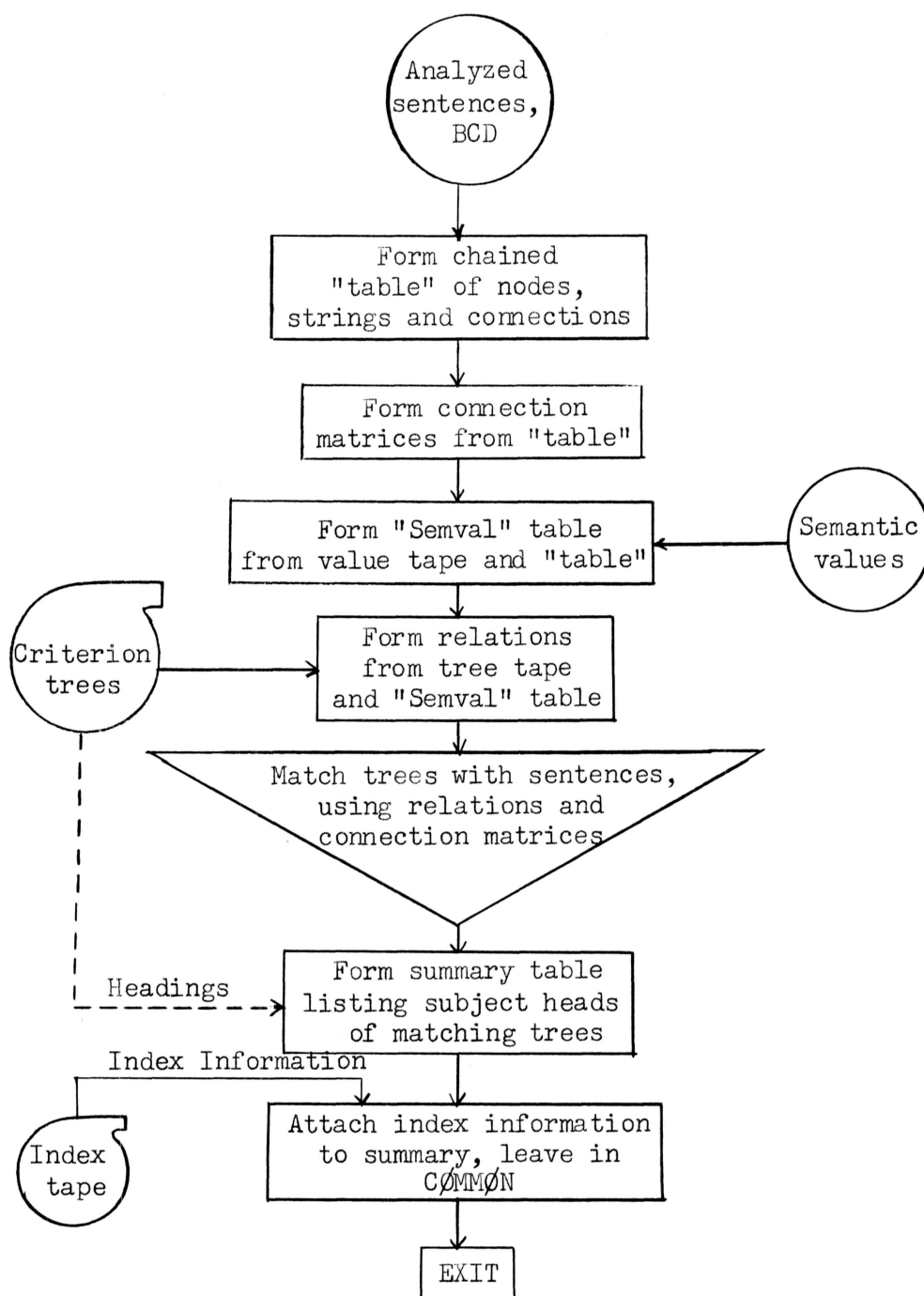
Flowchart 1 (continued)



Flowchart 1 (continued)

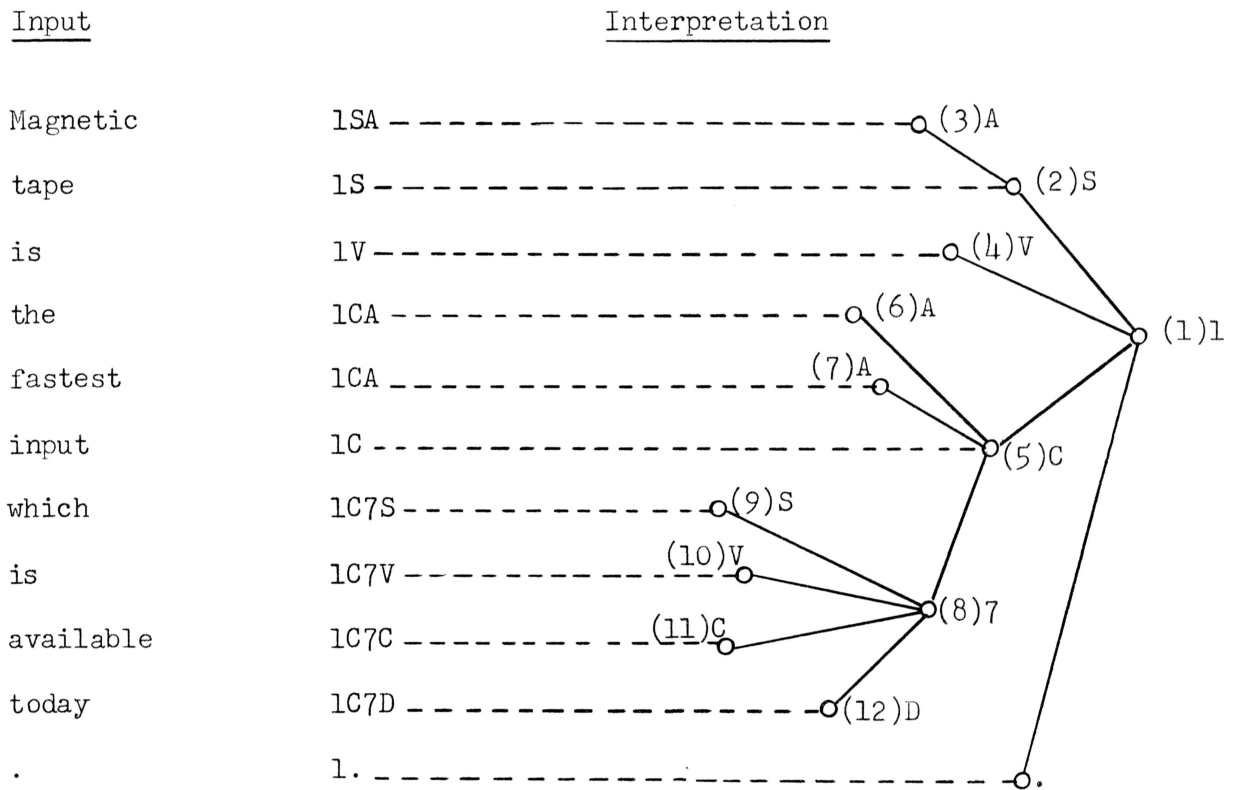


Flowchart 1 (continued)



Organization of Criterion Routine

Flowchart 2



Format of Syntactic Analysis Used as Input

Figure 2

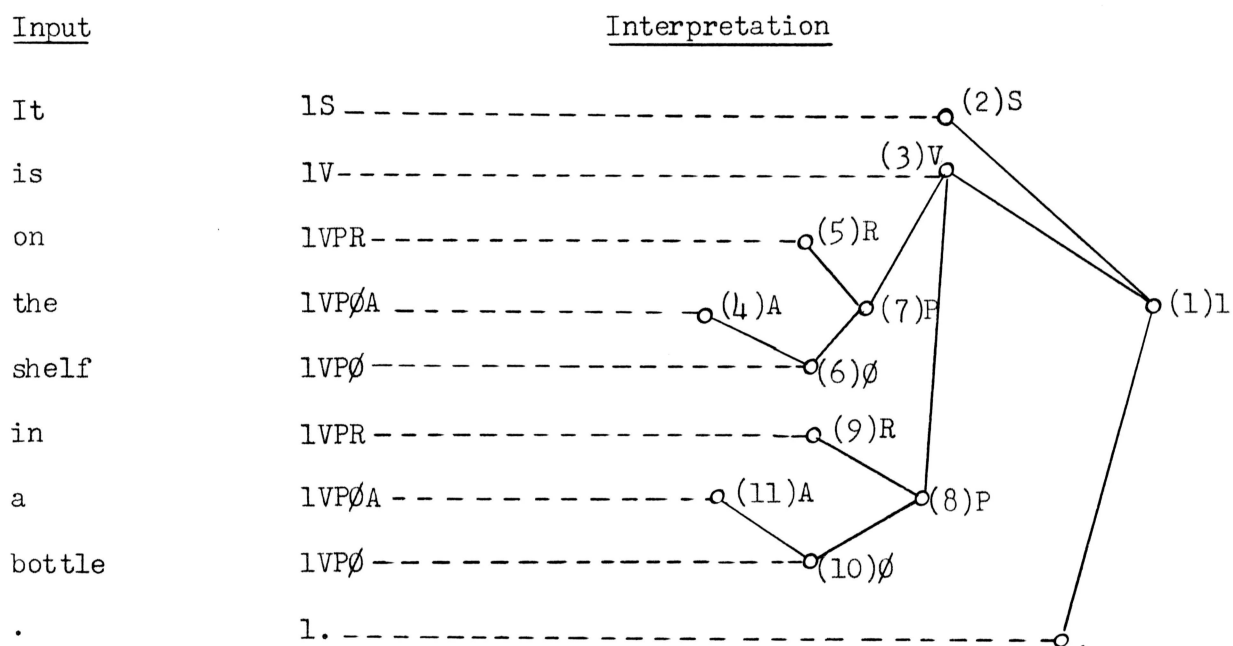
It may be seen from Fig. 2 that the strings may easily be constructed from the corresponding syntactic tree simply by working up the tree from any given node. For example, the string for "magnetic," which corresponds to node three in Fig. 2, may be found by taking the predecessors of node three, that is, node two, and then again of node two, i.e. node one. The characters for these nodes are "A," "S," "1" respectively, and correspond to the string "1SA."

The criterion routine must perform the inverse transformation; that is, it must determine a dependency tree from a set of strings. This task is complicated by several factors. First, distinct nodes may have the same strings; e.g., nodes six and seven in Fig. 2 both have the BCD character or syntactic value "A" (adjective) and the string "lCA." Secondly, the strings contain no explicit information to distinguish between phrases modifying the same word. Thus, in Fig. 3, there are two distinct trees exhibiting the same set of strings. Only one of them (Fig. 3(a)) is correct.

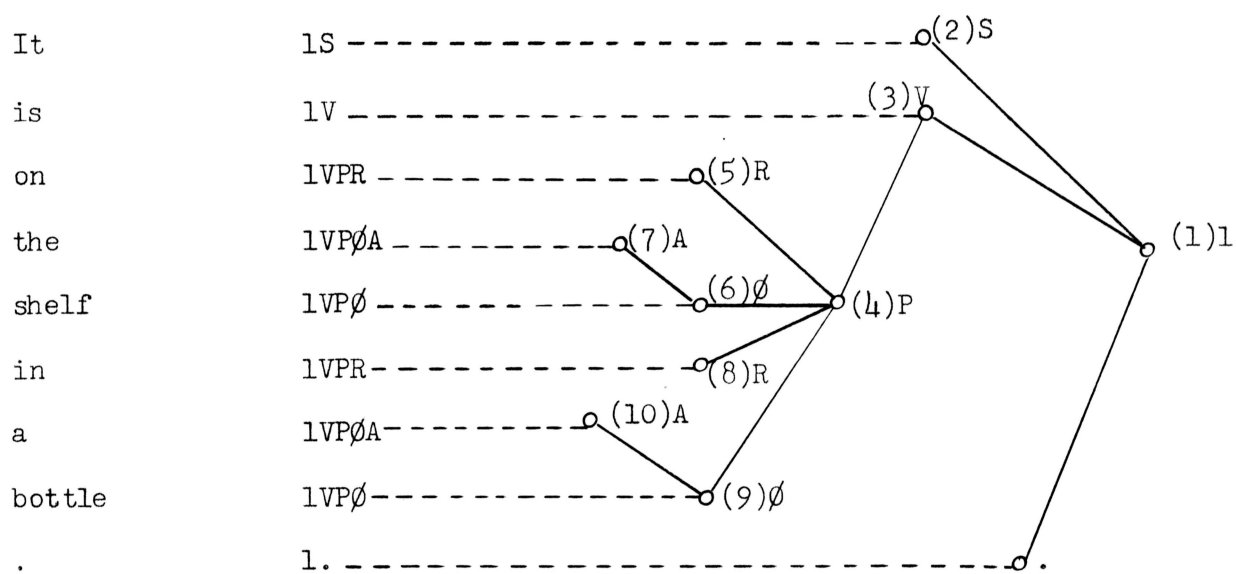
The criterion routine processes the strings one at a time, starting with the first word. Each string is scanned from left to right. The program maintains a list, called TABLE, of sentence nodes and their dependencies. TABLE is initially empty. As each string is scanned, the routine traces through this TABLE (see Fig. 4). When the string specifies a node not recorded in TABLE, a new entry is generated. Thus, when all strings have been scanned, TABLE contains a tree corresponding to the strings. Various refinements are added to the rules for generating new nodes; these refinements handle the various ambiguities discussed in the last paragraph.

The criterion routine uses several tables as part of the process. Two of these, "SEMS" and "SUMM" are used throughout a run; the others are reset for each sentence.

"SEMS" contains the semantic information for all the sentences to be processed. It is stored by the BTOKUN program in COMMON 451 (octal 76556). CRITER makes no alterations in it, but moves it into the program proper for safekeeping. This is necessary because GRAPH uses that area of COMMON for other purposes (see Sec. VII by Sussenguth). The semantic information is



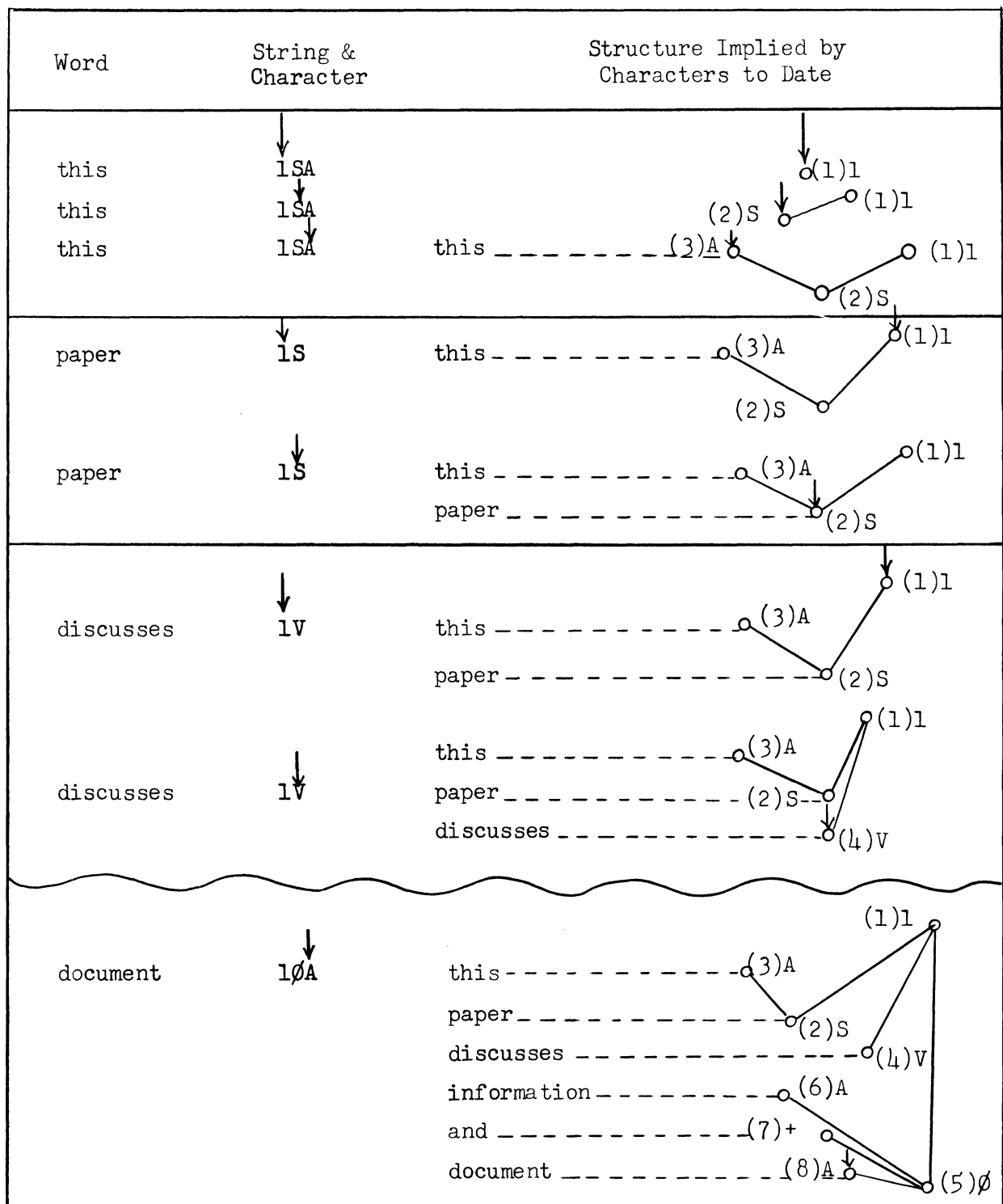
(a) Correct interpretation



(b) Incorrect interpretation

Interpretations of an Analysis

Figure 3



Note: The vertical arrows indicate the character and node being scanned at each step.

Scanning of Syntax Strings

Figure 4

stored in consecutive three-word items. The first word contains the sentence and word numbers originally assigned by the itemizer. The next two words (72 bits) consist of six semantic values (or concept numbers), packed as three 12-bit bytes per word. These are obtained by the thesaurus lookup.

"SUMM" holds at any time the list of matches found between all sentences so far processed, and the criterion trees in the library. SUMM is initially cleared; every time a tree is found to match a sentence, the table is updated. Each entry in SUMM consists of two machine words. The second of these contains a subject heading or "index" specified by six BCD characters. The other word contains a count specifying the number of times that the corresponding heading has been found to match a sentence.

When a match is found between a sentence and a criterion tree, SUMM is scanned for an entry that matches the heading of the tree. If such an entry is found, the count in that entry is incremented. If no entry is found, a new entry is made, with a count of one.

"(ØUT)" is a table constructed from SUMM. It is located in octal 61717 (COMMON 7010) and lower locations. Octal 61717 contains, in the decrement, the number of items in the table. These items are listed in successively lower locations in reverse alphabetical order. Each item comprises five machine words. The first word consists of a subject heading from SUMM; the fifth contains in the decrement, the appropriate count (also from SUMM). The sign of this count is positive. The remaining three words are extracted from an "index" file on the SMART library tape. This file contains a three-

machine-word correspondent for each subject heading; these three words are simply copied into the appropriate locations without any processing.

"TABLE" is the most important table in the criterion routine. It is stored in ~~COMMON~~ 10010 (54027₈). It is the primary active storage during input of a sentence. When an entire sentence has been read-in, TABLE contains all of the structural information provided by the analyzer.

During the processing of a sentence, TABLE holds all the nodes (i.e. substructures) found as of that moment.

Each item in TABLE corresponds to one node of a tree. In memory an item consists of 16 machine words. Of these, the first contains a chain link, and a flag to indicate whether an actual word corresponds to that node. The next five contain up to 30 BCD characters from the analysis of the sentence, indicating the role of the node. The next two words are not used. The remaining eight words in each item form two 144-bit binary vectors. One of these (the first) is called the "indirect" vector, the other the "direct" vector. The "direct" vector is either 0, indicating that the node has no antecedents (i.e. corresponds to the main structure (clause) of the sentence), or contains a single "1" bit whose position corresponds to that of the immediate antecedent. The "indirect" vector contains a "1" bit for each antecedent in the tree all the way up to the main clause.

The chain links referred to above, link together those entries corresponding to strings of equal length. Thirty words, labeled TABLE+1 through TABLE+30, are provided to hold the initial links of these chains. Each link points to the seventh word in a 16-word item. The link at the end of the chain is 00000.

The last table, called "SEMVAL", is reset for each sentence. It lists all the semantic values and syntactic structure types found to pertain to any node of the sentence. In memory, it consists of a sequence of five-word items. The first of these contains one 12-bit value, and a chain link by which the entire table is chained so that the values are in increasing numerical order. The remaining four words comprise a 144-bit binary vector; the "one" bits indicate the nodes to which the given value pertains. Only one entry is made for each distinct value: if two nodes have the same value this is represented by two "one" bits in a single entry.

5. Internal Processing of the Criterion Routine

After a few initializations, CRITER processes the input, one sentence at a time. The input tape is first scanned for a sentinel indicating the beginning of an analysis; the sentinel record then provides the number of the sentence. The sentence is then scanned, and the information extracted from each word is stored in TABLE and SEMVAL. When the sentence is completely read-in, CRITER matches it against each of the criterion trees. The criterion tree tape is then backspaced and another sentence processed. Finally the results are summarized and left in memory for the next program.

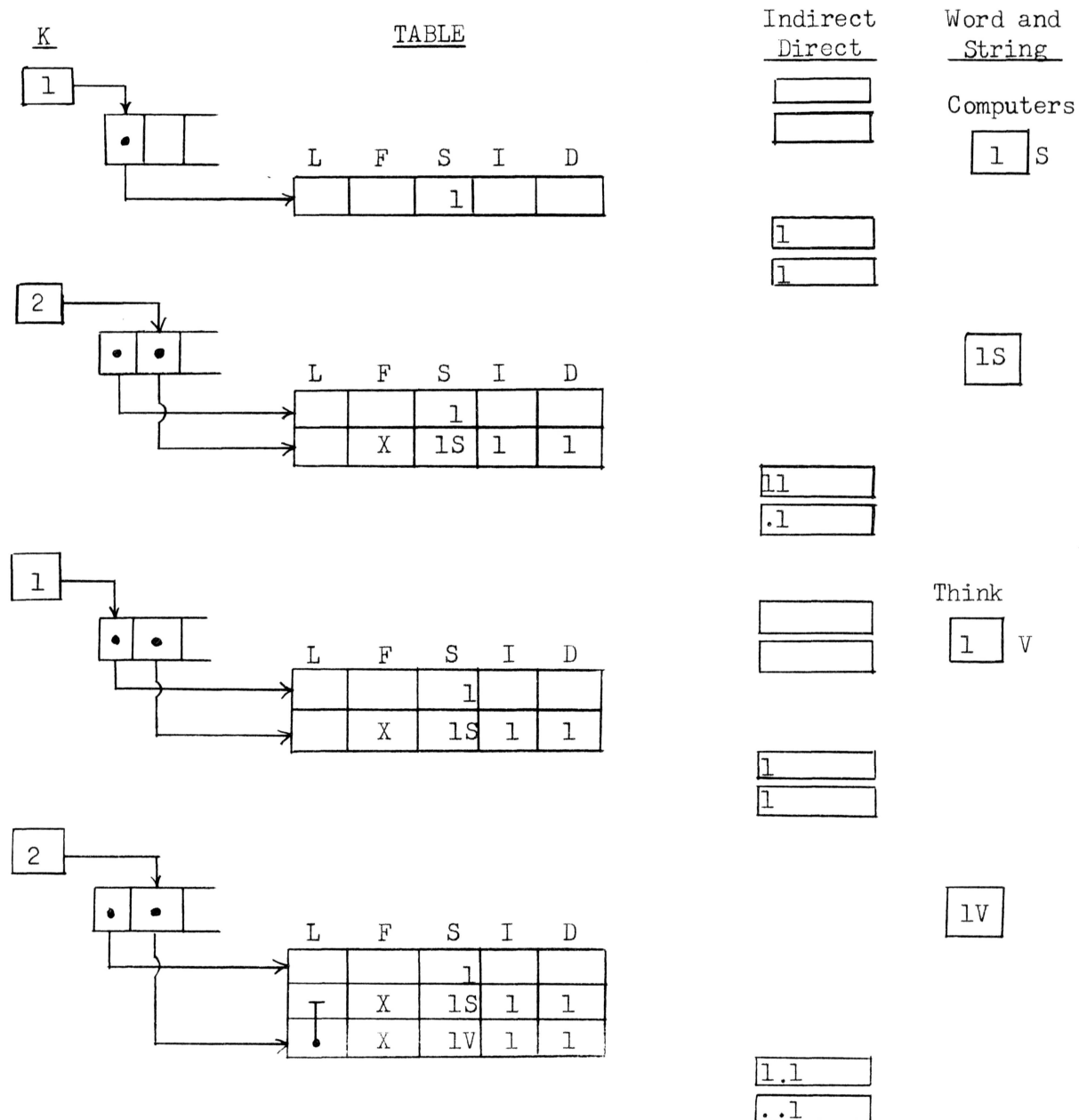
When a sentence word is read from tape, its record contains among other things a string of BCD characters denoting the syntactic function of that word in the sentence.¹ The criterion routine keeps a count, "K", in index register two. For each word, this count starts at one, and then is incremented. The routine takes the first K characters of the character string

representing the BCD syntactic function code and scans through TABLE by means of the chain (starting at TABLE+31-K). The scan stops when a TABLE entry is found whose string is equal to the first K characters of the BCD code string of the word. If the chain is exhausted before such a match is found, or if the match is not permitted (e.g. where a match indicates that the code entry corresponds both to the present and to some previous text word), ENTERK generates a new entry in TABLE. This entry will be chained into the beginning of the Kth chain, and its string will consist of the first K characters of the code string for that word.

When an entry in TABLE is either found by the matching process or generated, the number of the entry (represented as a single "one" bit) is placed in a special vector, called DIRCT; the updated DIRCT vector is then used to alter another "INDRCT" vector by "ØRing" the contents of DIRCT into INDRCT.

Figure 5 shows a simple example of this process. The sample sentence is "Computers think." The BCD code string for "Computers" is "1S", and the code string for "think" is "1V". Here "S" means "subject" and "V" means "verb". The "1" in both strings is the clause indicator for the single clause to which both words belong.

Both DIRCT and INDRCT are set to zero at the start of the process. Then with K = 1, TABLE is scanned for entry whose string field is equal to "1". Since TABLE is empty, none is found, so a new entry is generated as shown at the top of Fig. 5. (The arrows represent the successive links of the chains connecting the entries). The first entry has a string field



Construction of TABLE

Figure 5

of "l", blank I and D fields, because INDRCT and DIRCT are blank, and a blank link (L) because this entry is the first for $K = 1$. DIRCT and INDRCT are now updated. Since the entry just generated is the first entry in TABLE, a "one" bit is placed in the first position of DIRCT, and zeros elsewhere. Then the new DIRCT vector is ØRed into INDRCT.

The process is repeated for $K = 2$. The string is now "lS" and again no entry is found. A new entry is generated; this time the I and D fields each contain a "one" bit in position one, derived from INDRCT and DIRCT respectively. The flag bit is set because "lS" is the full string for the word "computers."

The situation is similar when "think" is processed. When $K = 1$, however, the code string is "l"; and the first entry in TABLE already contains the string "l". Thus no new entry is generated, and DIRCT will have a "one" in the first position. INDRCT will also have a single "one" bit in position one, since it was previously reset to zero before the new word "think" was read-in. The process described above is repeated again for $K = 2$, string = "lV". A new entry (the third) is generated and placed at the beginning of the second chain. DIRCT has a "one" in position three; INDRCT has "ones" in positions one and three.

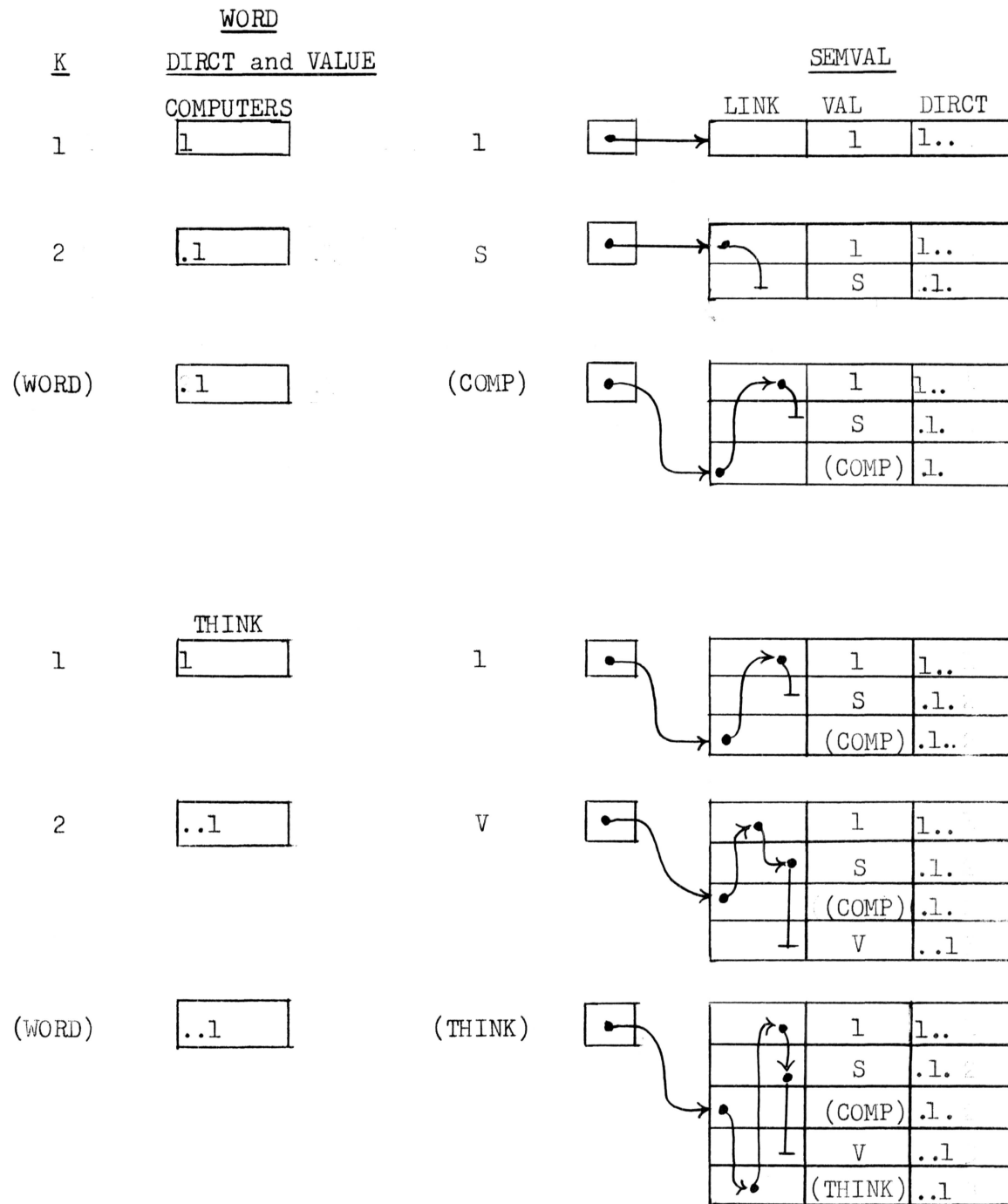
Thus after the processing of the entire sentence, TABLE contains a representation of the corresponding syntactic structure in tree form. Each entry in TABLE represents a node, and the D(direct) field in a given entry indicates the node upon which the given node depends. At any time during the process, DIRCT indicates the most recent node found (or generated),

i.e. the node upon which the next node will depend directly. INDRCT exhibits all the nodes upon which the next node will depend, either directly or indirectly, up to and including the topmost node of the tree.

When the entire code string for a word has been processed, TABLE contains a complete record of the syntactic relations of the given word to the earlier words in the sentence, and SEMVAL records the syntactic functions of the word. The semantic values (from SEMS) are now added to SEMVAL. As before, DIRCT serves to indicate which nodes in the syntax tree are being processed. When a given word is completely processed, DIRCT and INDRCT are cleared to zeros for the next word.

Figure 6 shows how the concept numbers associated with the sentence "Computers think" are entered into SEMVAL. The DIRCT vector is shown as it is generated during the construction of TABLE. The steps which update SEMVAL alternate with those that update TABLE; thus DIRCT always shows the node presently being processed.

After the first node is entered in TABLE, DIRCT contains a single "one" bit as shown. At this point an entry is made in SEMVAL which indicates that the first node has the (syntactic) value "1". Similarly, after node two is entered, an entry is made into SEMVAL, to indicate that node number two has the value "S". Then a third entry is made into SEMVAL, showing that the value "(COMP)" pertains to node number two. This is a semantic value obtained from SEMS; the entry is made only after the complete code string for the text word has been processed.



(Note: The stages of processing shown here are not contiguous, but alternate with those shown in Fig. 5).

Construction of SEMVAL, Using DIRCT as Developed in
Construction of TABLE (see Fig. 5)

Figure 6

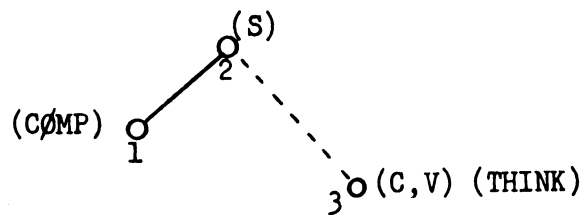
The same procedure is followed for "think". Note that when the first value, "1" is to be entered, a "1" already appears in SEMVAL; in this case DIRCT is ØRed into the entry. This has no effect in the present example, because there is already a "one" in the first position of the SEMVAL entry.

When a word is read which is coded "period" or "question mark", the above processing is not performed. Instead, the INDRCT and DIRCT fields are extracted from each entry in TABLE and reassembled to form two binary matrices, one made up of all the INDRCT fields collected together, the other of the DIRCT fields. Together these matrices are called CØNMAT (CØNnection MATrices); each is an incidence matrix indicating internode dependencies.

The complete preceding process takes place exactly once for each sentence. The "criterion trees", i.e. the model phrases, are now read-in from tape. Each is compared against the sentence as shown in Fig. 8.

Each tree consists of one record on tape, which is divided logically into several fields (see Fig. 7). The first field is the prefix, which contains the number of nodes, the number of "relations" (see below), the number of "relation generators", and a BCD heading and an I.D. (identification) number. The next two fields are indirect and direct connection matrices showing the structure of the tree.

The fourth field consists of a series of "relation generators". Each of these is composed of a 12-bit value (syntactic or semantic), a node number, and a relation number. The latter refers to one of a set of "relations", (denoting restrictions which affect the matching process between the given tree and the sentence). Each relation is represented by a pair of binary



(a) Symbolic representation

<div><div>3</div><div>196</div><div>D</div><div>S</div><div>C</div></div>						<div><div>4</div><div>5</div><div>R</div><div>I</div><div>P</div></div>						P (Prefix)
...						...						
<div><div>1</div><div>.1</div><div>...</div></div>						<div><div>...</div><div>...</div><div>...</div></div>						I (Indirect)
...						...						
<div><div>...</div><div>...</div><div>...</div></div>						<div><div>...</div><div>...</div><div>...</div></div>						D (Direct)
...						...						
<div><div>(COMP)</div><div>(THINK)</div><div>C</div><div>S</div><div>V</div></div>						<div><div>1</div><div>3</div><div>3</div><div>2</div><div>3</div></div> <div><div>2</div><div>4</div><div>3</div><div>1</div><div>3</div></div>						V (Values)

(b) Structure of binary record

A Sample Criterion Tree

Figure 7

vectors, one for the tree and one for the sentence, and its function is to ensure that each of the tree nodes indicated by the "tree" vectors correspond to one of the sentence nodes indicated by the "sentence" vector.

These relations are set up by a simultaneous scan of the set of relation generators (sorted in numeric order) and of SEMVAL (chained in order of value). When an entry in SEMVAL is found to have the same value as a relation generator, a relation is set up.

A space is reserved in storage near each connection matrix (CØNMAT for the sentence, and "TREEBF" for the tree), to store the respective relation vectors. The relation number in the current relation generator specifies a location in each of these reserved storage spaces. In TREEBF, the relation vector is cleared to zero, and a single "one" bit inserted corresponding to the node number found in the relation generator; in CØNMAT, the SEMVAL entry is ØRed into the relation vector. This process is shown in Fig. 8.

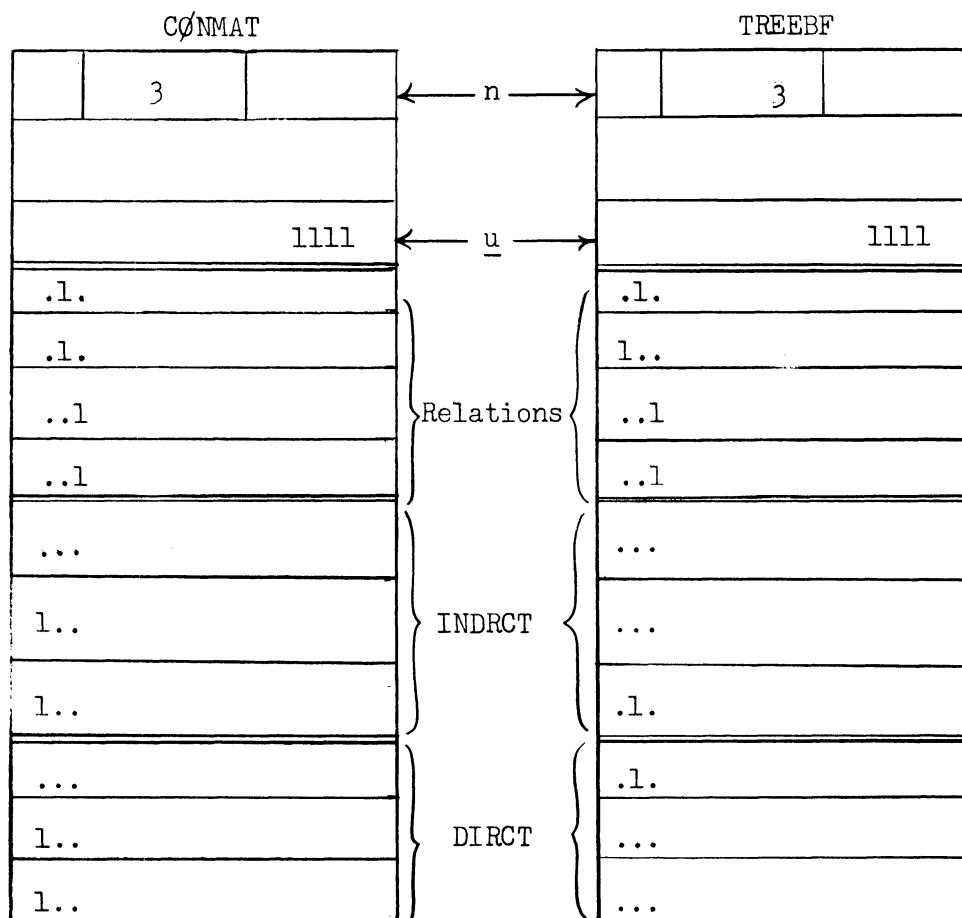
The first SEMVAL entry scanned in Fig. 8 has the value (CØMP), as does the first relation generator. Therefore a relation is set up. This will be relation number two, corresponding to the number given by the relation generator. Thus the second vector in each reserved space is updated: the one in TREEBF is replaced by a vector having a "one" bit in position one (as indicated by the relation generator) and the data from the SEMVAL entry (which has a "one" in position two) is ØRed into the CØNMAT half of the relation.

When this has been done, the routine examines the next relation generator. If the value is the same, the process is repeated; if greater, a new SEMVAL entry is scanned. Then the comparison is repeated. If the SEMVAL entry is less than the relation generator value, SEMVAL is scanned; if greater, a new relation generator is examined. Thus the routine continues its scan until it runs out of entries of either kind.

At this point the situation illustrated in Fig. 9 obtains. Two separate areas are set up in memory, CØNMAT and TREEBF, each containing the data for one syntactic structure. The first word in each area contains the number of nodes in that structure. (In the example of Fig. 9 this is three in both cases; in general, the criterion tree will have fewer nodes -- usually considerably so.) The next word is blank. The third word contains a number of "one" bits, right justified, i.e. a string of "zero" bits followed by a string of "ones". There are as many "ones" as there are relations. This number is, of course, the same in CØNMAT and TREEBF -- four in the example.

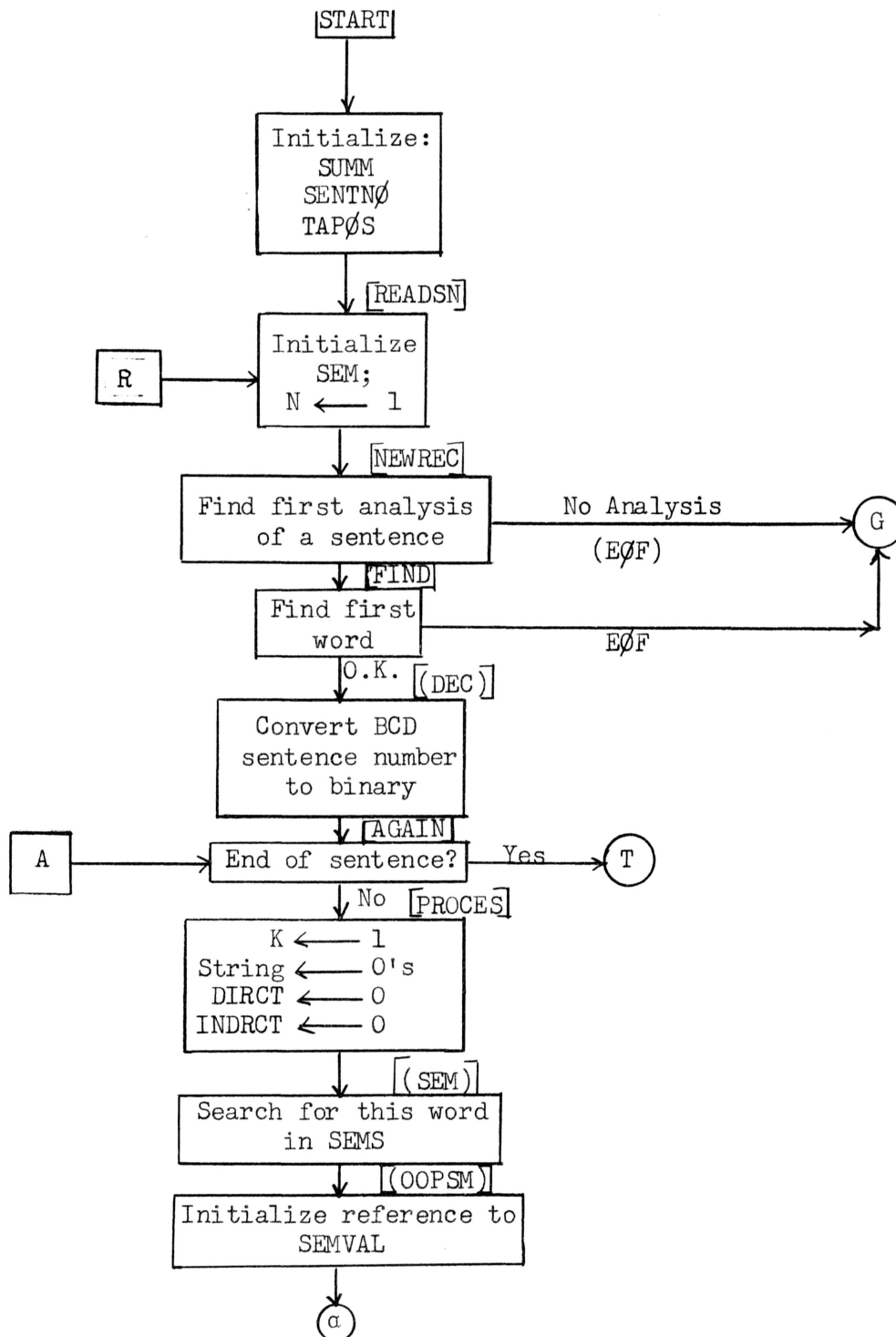
The relations are stored next in memory; these are paired: the first relation vector in CØNMAT goes with the first one in TREEBF, etc. They instruct the graph matching routine GRAPH (see Sec. VII) to permit only certain correspondences.

Finally come the two connection matrices. Those in CØNMAT originate in TABLE; those in TREEBF come from the original tape record. In the example of Fig. 9, they indicate that nodes two and three of the sentence each depend, directly (and hence indirectly) upon node one, while node one depends on nothing; whereas the tree matrices show that the correspondent of node one must depend directly on that of node two, while the correspondent of node three need



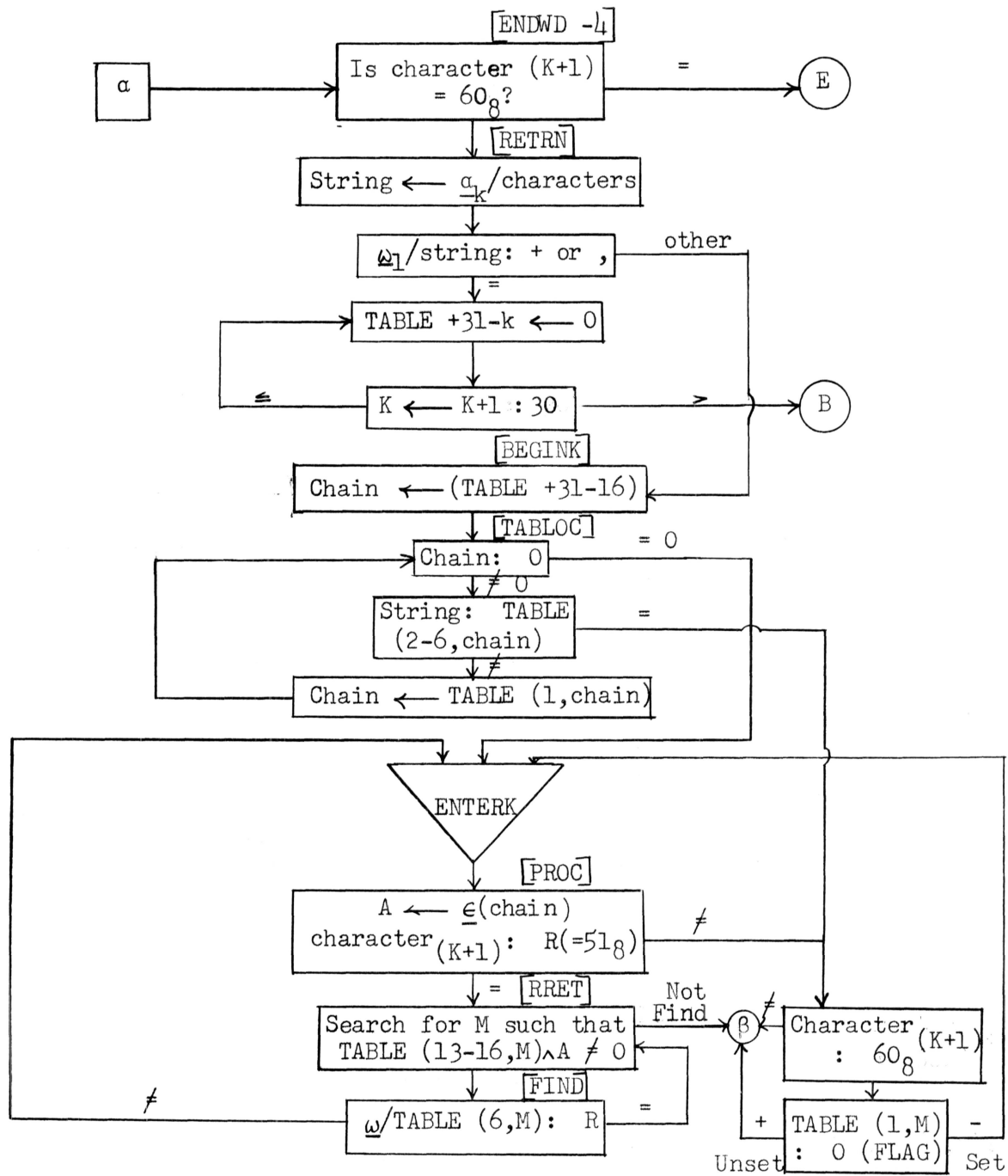
CØNMAT and TREEBF Just Before GRAPH
is Called

Figure 9

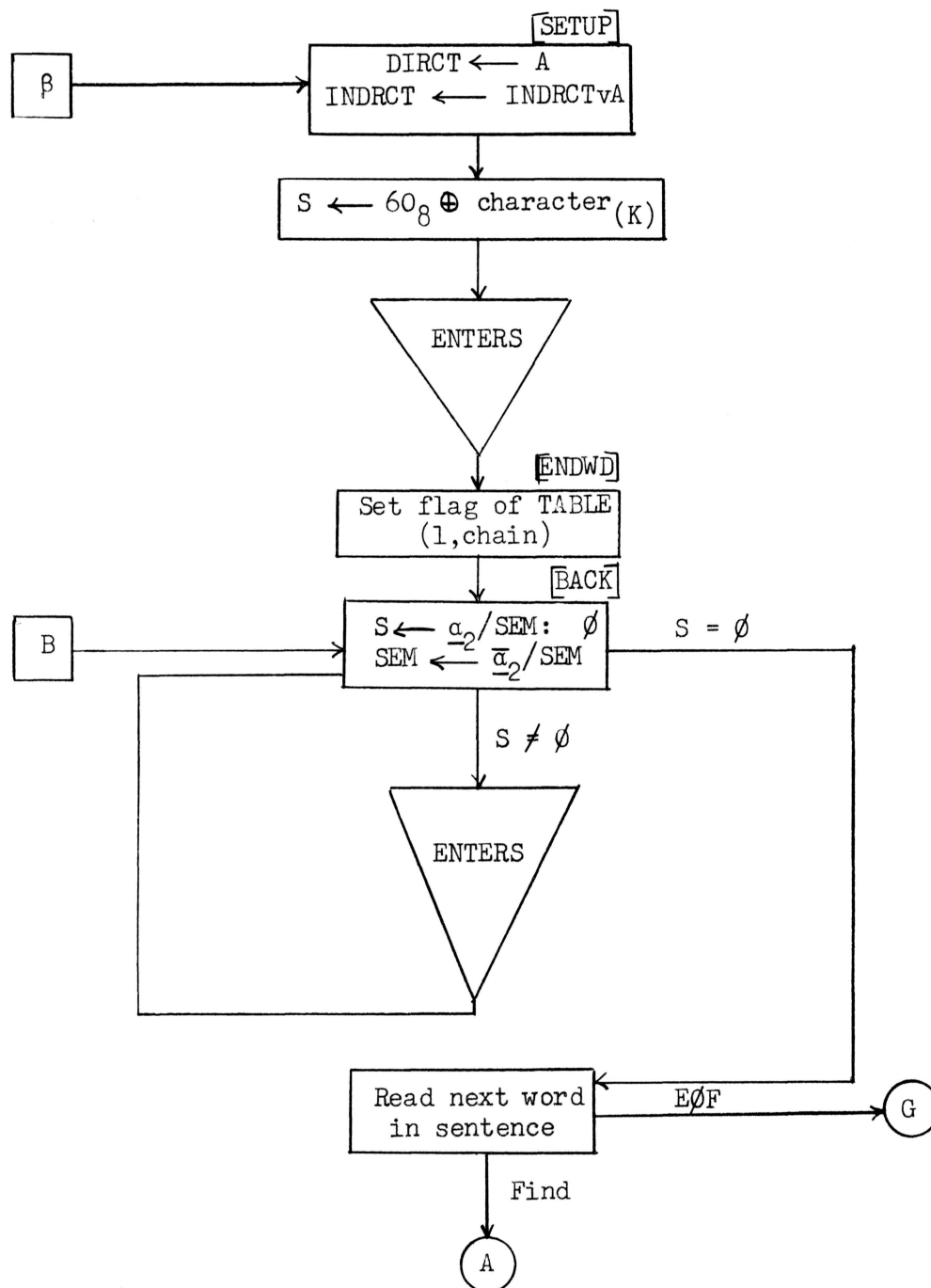


Criterion Routine

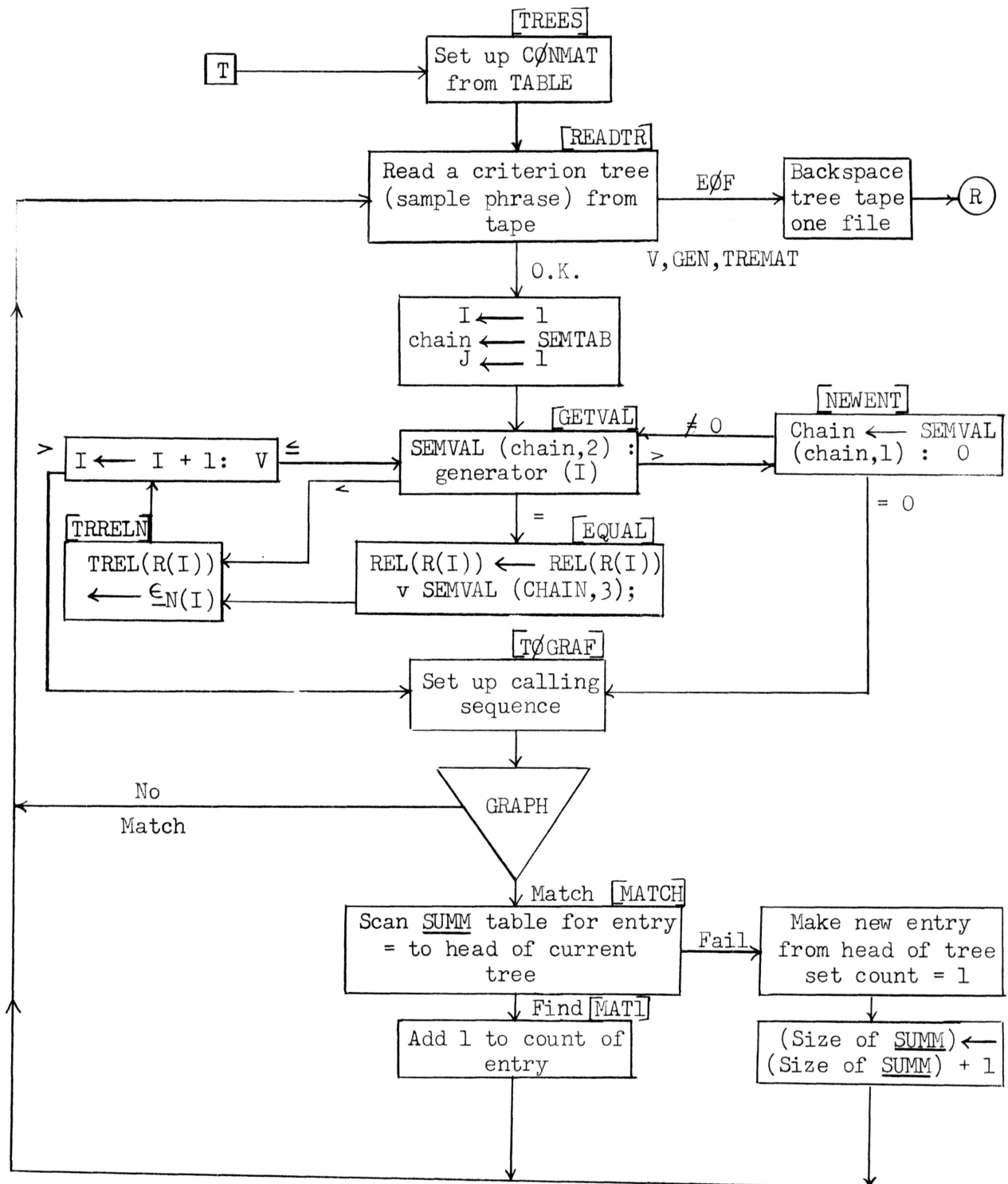
Flowchart 3



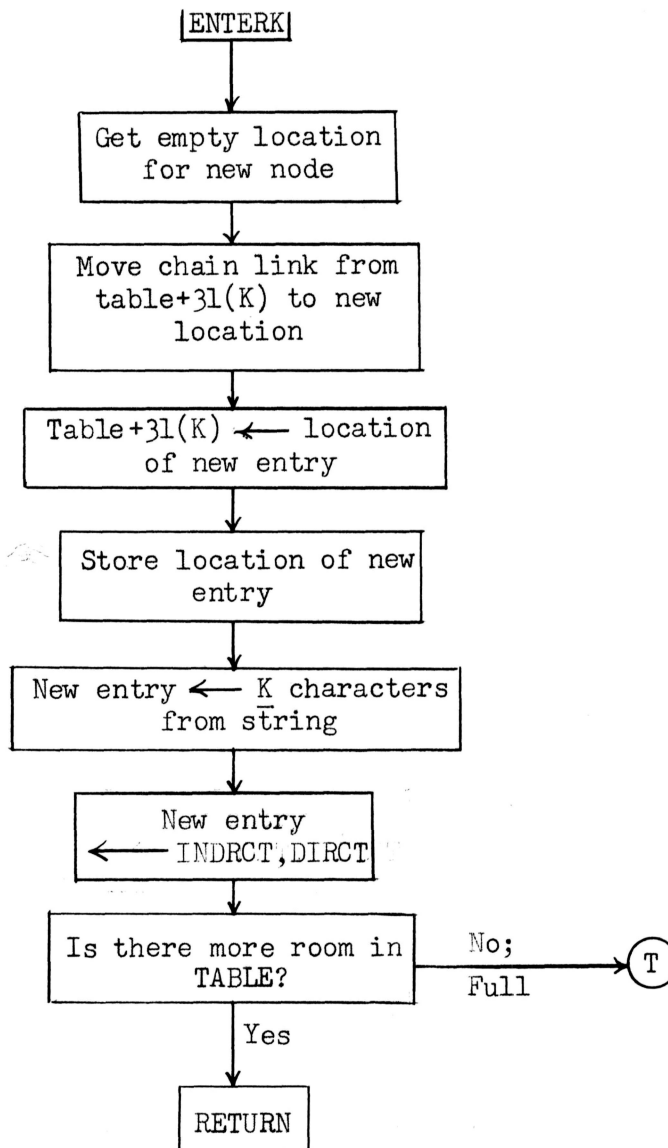
Flowchart 3 (continued)



Flowchart 3 (continued)

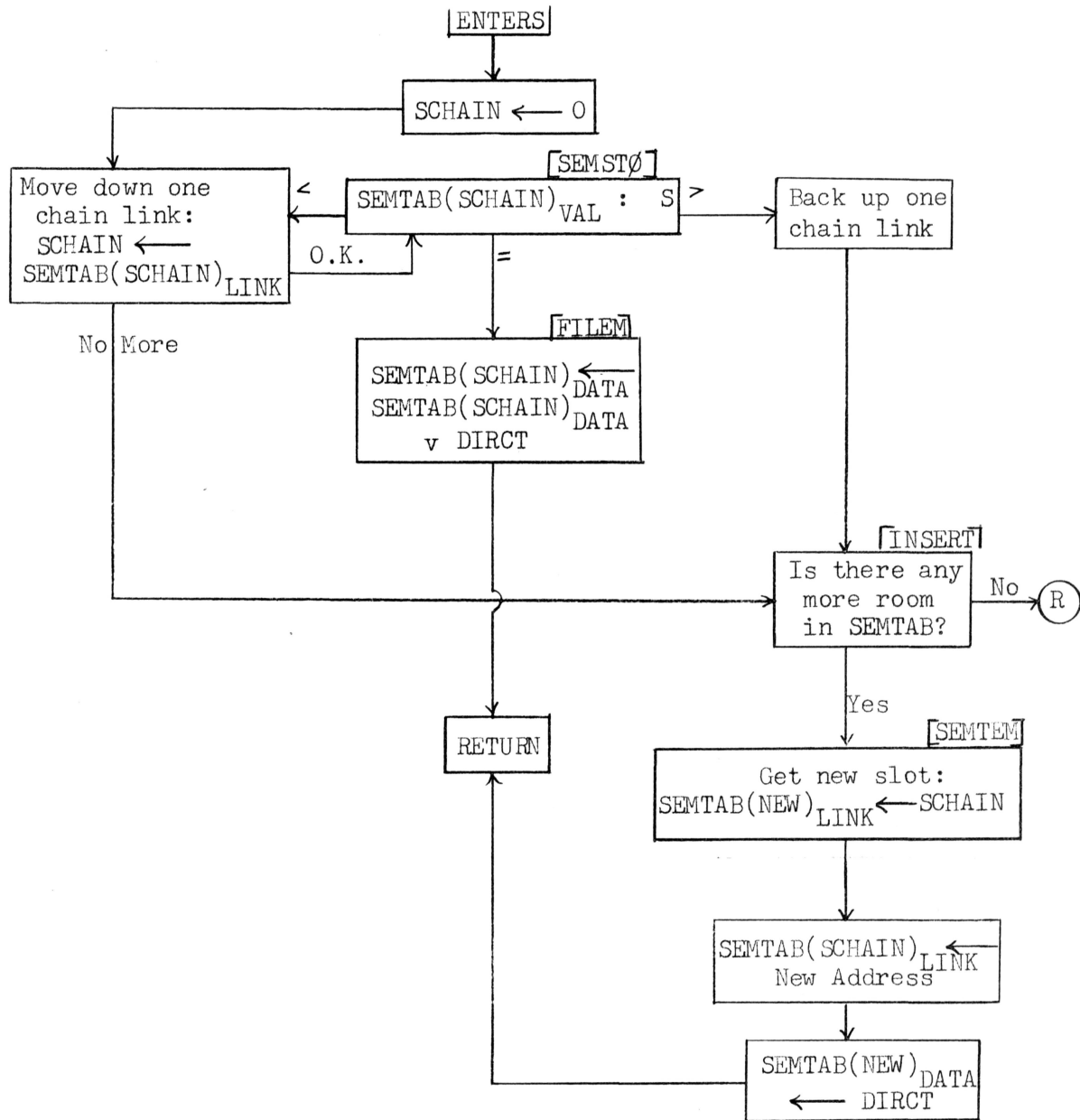


Flowchart 3 (continued)



Subroutine ENTERK

Flowchart 4



Subroutine ENTERS

Flowchart 5

only have an indirect dependence on that of node two. The correspondence $1 \longrightarrow 2, 2 \longrightarrow 1, 3 \longrightarrow 3$ satisfies these dependency requirements but not the relations, because tree node two must correspond (by relation number one) to a sentence node marked S, i.e. to sentence node two. But also by relation two, tree node two must correspond with sentence node one (the only one with value (COMP)); thus no correspondence is possible here.

These two sets of data serve as input to GRAPH (Sec. VII). They are set up as described, and GRAPH is given the addresses of the corresponding areas in memory. Then GRAPH evaluates all the relations, connections, etc. and either decides that no one-to-one correspondence is possible (as in the example given), or finds such a correspondence. The answer is returned in the accumulator: zero if no match, one if a match is possible. If a match is possible, GRAPH leaves in memory a list of binary words, one for each node in the tree. Each of these words contains the number of the sentence node corresponding to the respective tree node.

Flowcharts 3, 4 and 5 describe the various steps used by CRITER in greater detail.

REFERENCES

1. Mathematical Linguistics and Automatic Translation, Report No. NSF-8
The Computation Laboratory of Harvard University (January 1963).
2. Mathematical Linguistics and Automatic Translation, Report No. NSF-9
Vol. I and II, The Computation Laboratory of Harvard University
(June 1963).