ISR-9
August 1965

XII. THE TREE MATCHING PROGRAM - MATCH

M. Razar

1. Introduction

In order to determine precisely what concepts are expressed by a given sentence, one must first know the exact interdependencies of the various words making up the sentence. For otherwise, one would be unable to group two or more words to generate a more complex concept than that expressed by either word alone. As an example of dependencies within a sentence, consider the following three typical sentences.

(a) "Charged particles interact strongly."

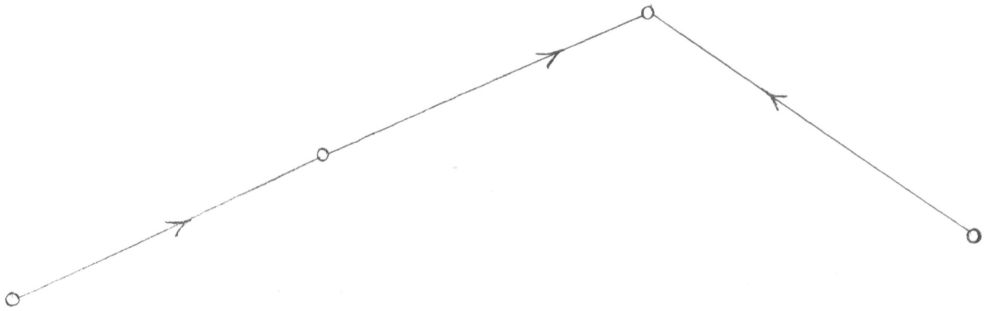
(b) "When charged, particles interact strongly."

(c) "Particles with a charge interact strongly."

Clearly (a) and (c) are completely synonymous. That is, the concepts expressed by the phrases "charged particles" and "particles with a charge" are identical. Note that the actual positions within the sentence of the words "charged" and "particles" are irrelevant. What is important is their syntactic relation - that of an adjective modifying a noun. Conversely, an adjacent position tells one nothing. For in (b), although the phrase "charged particles" appears, there is no direct connection between the two words. The word "charged" is part of the adverbial phrase "when charged" which modifies the verb "interact."

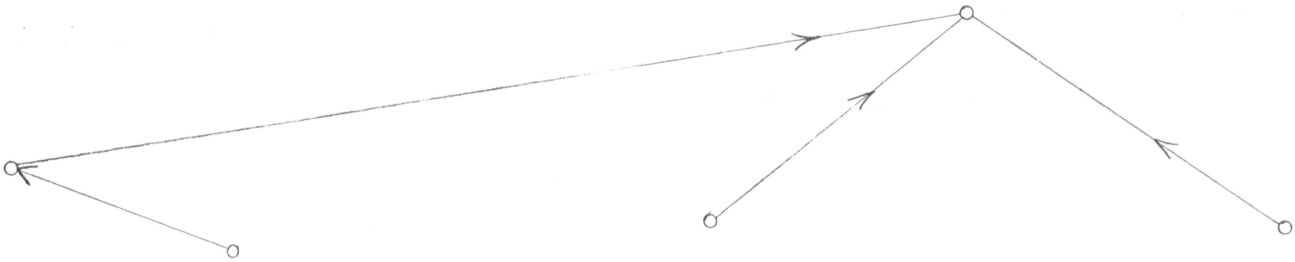
In order to display the syntactic relations of words in a sentence conveniently, it is customary to write a sentence in tree form. (See Fig. 1.)

(a)



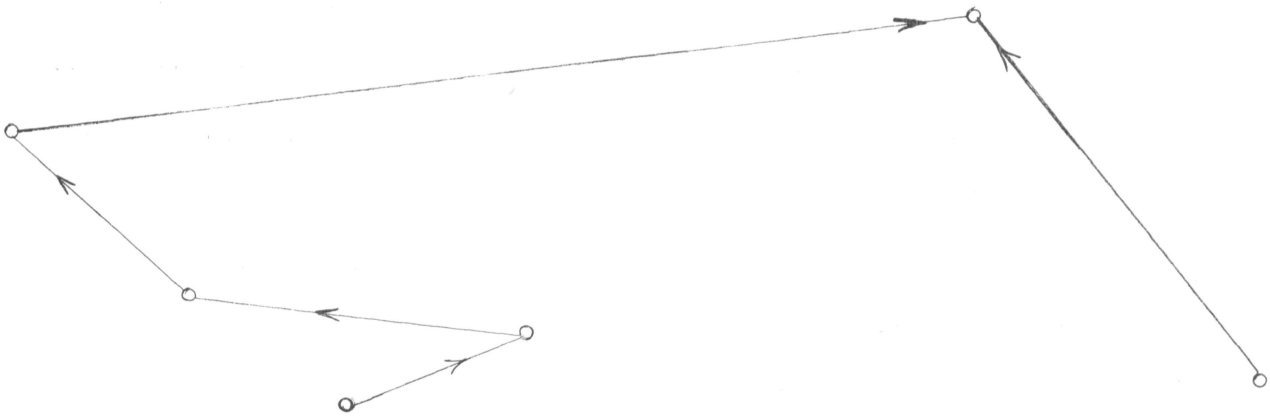
Charged particles interact strongly.

(b)



When charged particles interact strongly.

(c)



Particles with a charge interact strongly.

Typical Sentence Trees

Figure 1

Once a sentence is in tree form, the various word-dependencies become clear. Two nodes, A and B are directly dependent if a corresponding branch AB exists in the tree. They are indirectly dependent if there is a path in the tree leading from A to B. Thus in Fig. 1(a) "charged particles" actually constitute a subtree while in (c) only an indirect dependency exists. In Fig. 1(b), there is no connection, direct or indirect, between the words "charged" and "particles." Moreover, there may be many irrelevant nodes which are connected like "a" and "with" in Fig. 1(c). The point is that given a phrase like "charged particles," it is necessary to determine if it is included (directly or indirectly) in a given sentence. This task is performed by the subroutine MATCH.

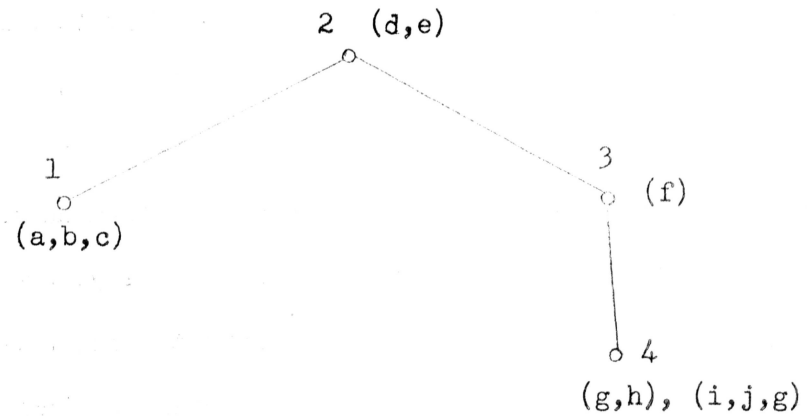
2. General Considerations

It was seen that a sentence can be put into tree form to exhibit word-dependencies. Similarly, any phrase (for example, "charged particles") can be thought of as a tree. To determine whether the phrase is included in the sentence, it is sufficient to determine if the phrase tree is a subtree of the sentence tree. Mathematically, one would like to set up an isomorphism from the phrase tree into the sentence tree. The routine MATCH will find any such isomorphism, or determine that none exists.

MATCH is designed to be primarily value-oriented rather than structure oriented. In SMART, one deals mainly with trees of simple

sentence structure for which only fairly simple dependencies exist. However, the values attached to each node can be quite varied. Just the opposite would be the case if one were dealing with the graphs determined by the structures of organic chemical compounds. Then extremely complex structures would occur but only a relatively few values (e.g. carbon, hydrogen, oxygen, nitrogen) would be attached to the nodes. Therefore, MATCH will first determine possible correspondences without structural isomorphism, and only then see if the structure is correct.

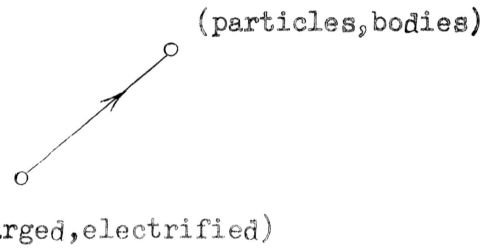
In general, MATCH will take a tree in a form similar to that of the tree in Fig. 2, and determine whether any correspondence is possible with a tree referred to as SENT which is actually, (though not necessarily) the tree generated by some sentence. The multiple



Sample of Labeled Tree
Figure 2

values at each node in Fig. 2 are called relation generators. The totality of associated relation generators attached to a given node is called a relation. More than one relation may be attached to a node and in order to have this node correspond to a node in SENT with value x, x must be a relation generator of each relation attached to the node. Thus the labeled tree in Fig. 2 could correspond to any one of six trees. Node 1 must correspond to a node with value a or b or c; node 2 to a node with value d or e; node 3 to a node with value f; node 4 to a node with value g.

The relation generators of a given relation can be thought of as essentially synonymous words which are part of a common concept. For example, one might not wish to distinguish among the phrases, "charged particles," "electrified particles," "charged bodies," and "electrified bodies." Therefore, one would construct a tree as follows:



In practice, of course, all words are represented by (binary) numbers and this makes possible quite convenient representation of all the relations in a tree. This representation is performed with the help of a vector RELN, each word of which contains a relation generator and a node number as follows:

S	DECREMENT	TAG	ADDRESS
±	VALUE OF RELATION GENERATOR	0	NODE NUMBER

The sign bit determines the position of a relation generator within a relation. In particular, the generators of a given relation are stored in adjacent positions (see Fig. 3) within RELN, a plus sign being used to indicate that a generator is the last of its relation and a minus sign to indicate that it is not.

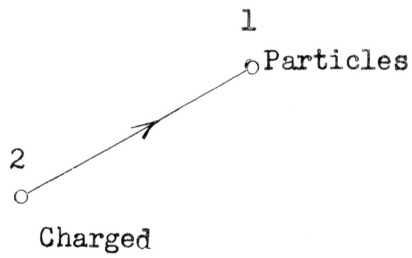
S	D	T	A
-	a	0	1
-	b	0	1
+	c	0	1
-	d	0	2
+	e	0	2
+	f	0	3
-	g	0	4
+	h	0	4
-	i	0	4
-	j	0	4
+	g	0	4

Vector RELN for Tree of Figure 2

Figure 3

In the last few paragraphs, the storage format for the values attached to a node was described. The structure of the tree is even simpler to determine. A vector TPAR is set up so that TPAR(n) contains the parent of node n in its decrement. If the sign of TPAR(n) is plus,

the corresponding parent is a direct parent and only correspondences with direct dependencies will be considered. If the sign is minus, indirect dependencies are also considered. As an example, consider the tree:



The decrement of TPAR(2) contains 1, indicating that node 1 is the parent of node 2. If the sign is plus, only direct dependencies are possible matches and this form corresponds to a subtree of the tree in Fig. 1(a) but to no subtree of the tree in Fig. 1(c). With a minus sign in TPAR(2) however, indirect dependencies would be permitted and a match would also be obtained with the tree in Fig. 1(c).

The node values and tree structure of the phrase tree which is being matched into a sentence constitutes half the information needed by MATCH. The other half obviously pertains to the sentence tree into which the phrase tree is being matched. The information about the sentence tree structure is very simply stored in a vector SPAR. SPAR(n) contains the parent of node n in its decrement and no other information. A somewhat useful fact is that the vector SPAR is arranged so that SPAR(n) $<$ n for all nodes n.

The values attached to the sentence nodes are determined quite indirectly from a table to be described below. For each

possible value under consideration, four words of core are set aside. The first, VALU, contains the actual value representing an English word which may appear in a sentence. The other three, VEC1, VEC2 and VEC3 are considered as a single 108-bit logical word referred to hereafter as VEC. The bits are consecutively numbered from 1 to 108 and a 1-bit in position n indicates that node n in the sentence tree has the value contained in VALU. A zero in bit position n indicates that node n has a value other than that in VALU. Note that this limits the size of the sentence to 108 words, which in practice is no real limitation.

The motivation behind this format is that one can determine all possible correspondents of a given node in the phrase tree as follows. Form a 108-bit word for each relation attached to the node by or-ing together the respective contents of VEC corresponding to each relation generator. When this is done for each relation attached to the node, the results are "and-ed" together to give a 108-bit logical word PCORR with 1-bits in those and only those positions which correspond to sentence nodes whose values make them possible correspondents of the phrase node in question. Thus if the value attached to nodes 6 and 10 in a sentence is "charged" while the value "electrified" is attached to node 19, then a node in the phrase tree having the single relation ("charged," "electrified") attached to it will generate a word PCORR with 1-bits in positions 6, 10 and 19.

The algorithm used by MATCH is fairly simple and is separated into two distinct parts. The first part has already been discussed extensively above. We determine, for each node in the phrase tree, all possible nodes in the sentence tree to which it could correspond. If any node in the phrase-tree can correspond to no node in the sentence, then clearly no match is possible and the subroutine MATCH returns this information. If every node has possible correspondents in the sentence, the second part of MATCH is activated to consider structural constraints. We choose the node with the fewest possible correspondents and tentatively assign to it one of its possible correspondents in the sentence tree. (This assigned node in the sentence is referred to as its image or correspondent.) The structural constraints can then be checked by seeing if the parent of the correspondent is a correspondent of the parent of the node under consideration. If it is, and if the parent of the node has not yet been tentatively assigned to something else, one of the remaining possible correspondents of the parent is assigned to it. This procedure is repeated until all nodes of the phrase tree are assigned images in such a way that the image of TPAR(n) is always equal to SPAR(image of n) or until all possibilities of such an assignment are exhausted. A more detailed description of the algorithm is found in the flowcharts which are appended to this report.

3. Calling Sequence for MATCH; Returning from MATCH

The calling sequence to MATCH is quite simple. It reads:

CALL MATCH, SPAR, SENSIZ, TPAR, TRESIZ, RELN, RELSIZ

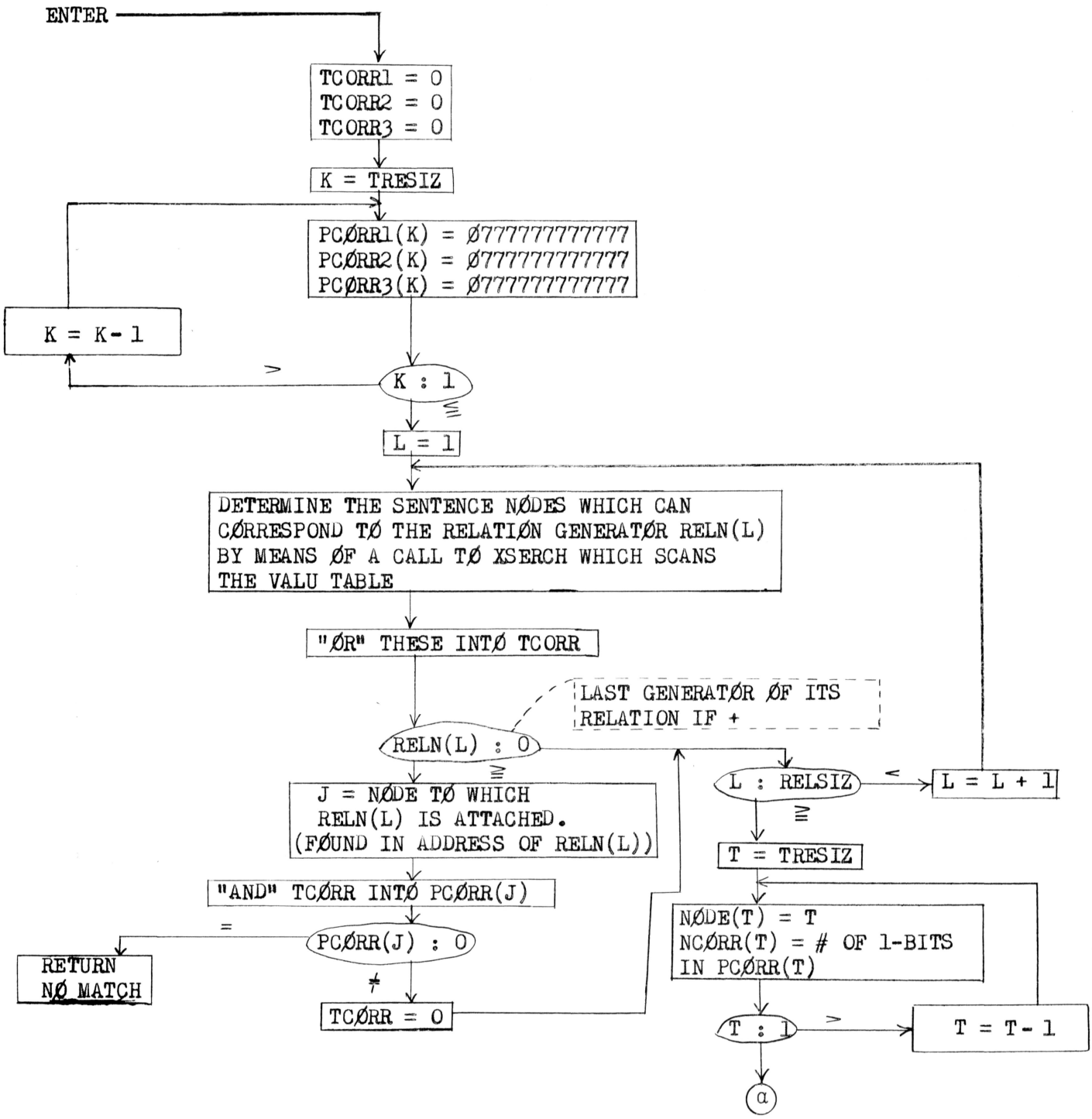
where SPAR, TPAR and RELN are the FORTRAN names of each of these previously described vectors. That is, the vectors SPAR, TPAR and RELN are stored backwards in core so that SPAR(n) is in location SPAR-n; TPAR(n) is in TPAR-n; RELN(n) is in RELN-n. The lengths of the vectors SPAR, TPAR and RELN are found in the decrements of SENSIZ, TRESIZ, RELSIZ respectively as FORTRAN type integers.

When a match is found, the actual correspondence is communicated to the main program in the following way. A vector CØRESP of length TRESIZ is set up (as a Fortran array) in such a way that CORESP(n) contains the sentence node corresponding to node n of the phrase tree. This information is contained as a decrement integer in CØRESP(n). Clearly then, the relation 1 = CORESP(n) = SENSIZ = 108 must hold for all nodes n.

Once the array CORESP has been set up, MATCH returns the information by calling a subroutine HIT with a single argument, CORESP (that is, CALL HIT, CØRESP). Then MATCH proceeds as if it had failed to find this last correspondence to find any other possible ways of mapping the phrase tree into the sentence tree. When all possibilities for such full correspondences have been exhausted, irrespective of whether any matches at all have been found, MATCH simply returns control to the main program.

APPENDIX A

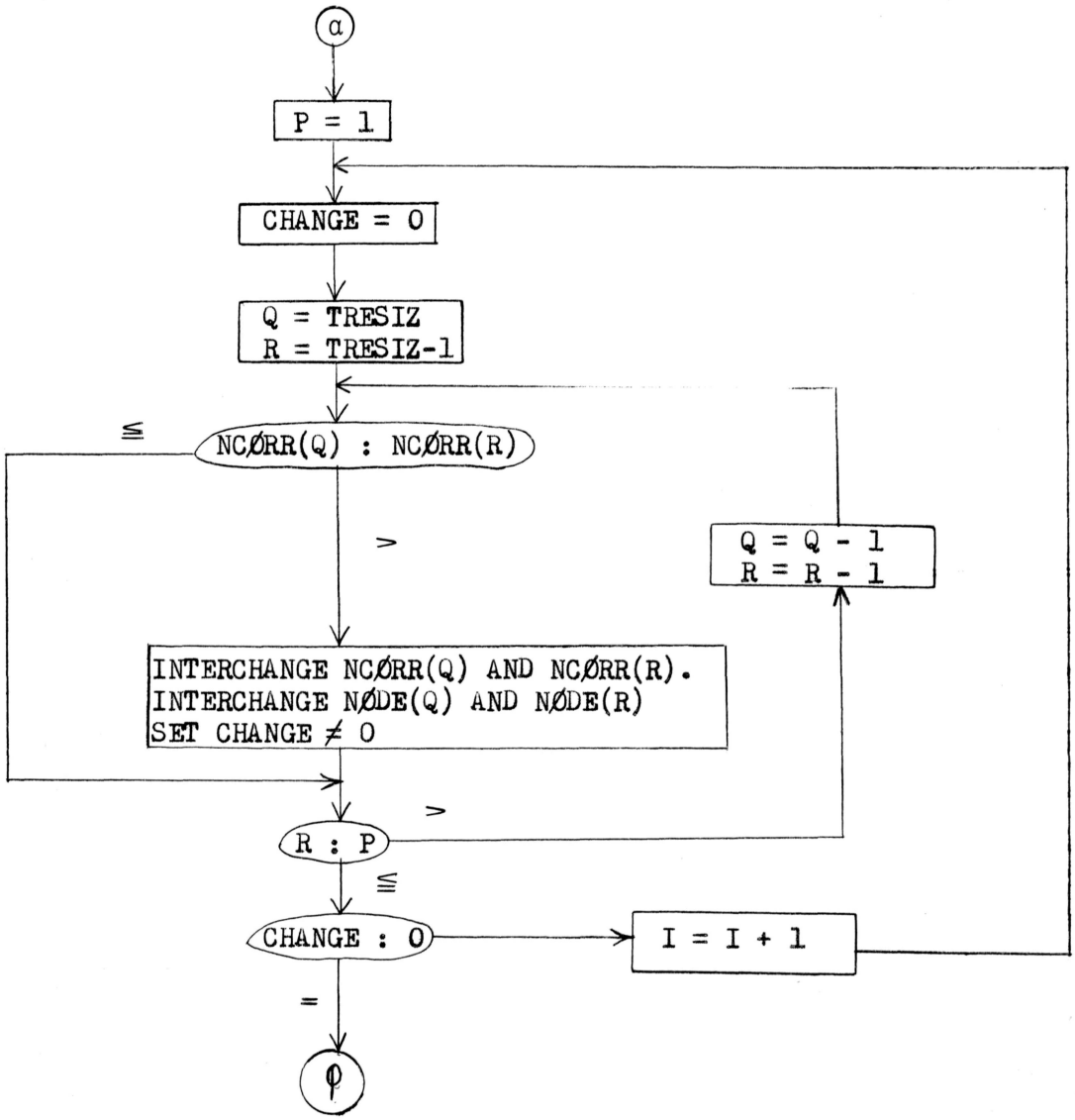
ALGØRITHM FØR DETERMINATIØN ØF VALUE CØNSTRAINTS
ØN PØSSIBLE MATCHES



Flowchart 1

APPENDIX B

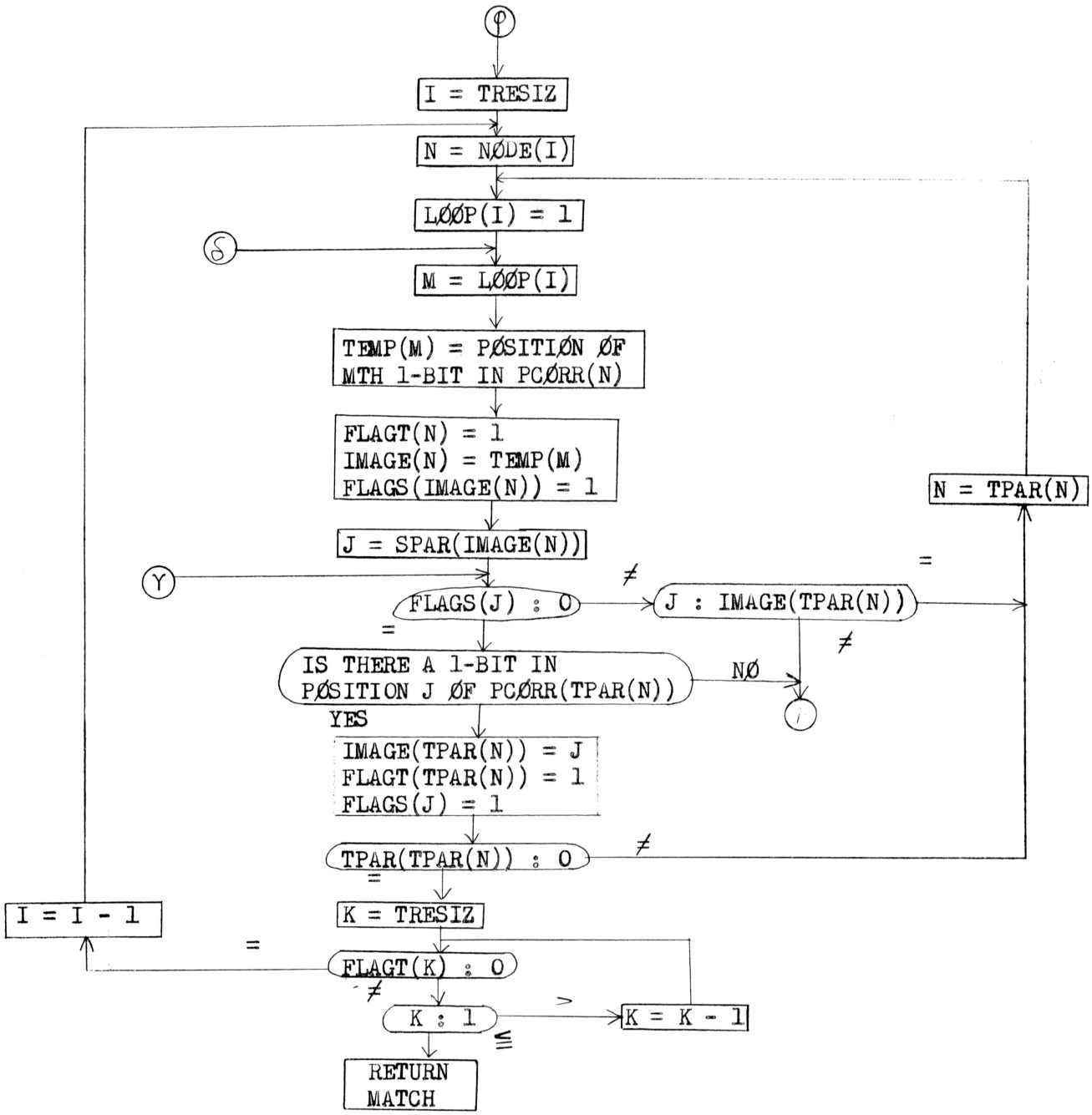
SØRTING ØF NØDES IN ASCENDING ØRDER ØF NUMBER ØF PØSSIBLE CORRESPØNDENTS



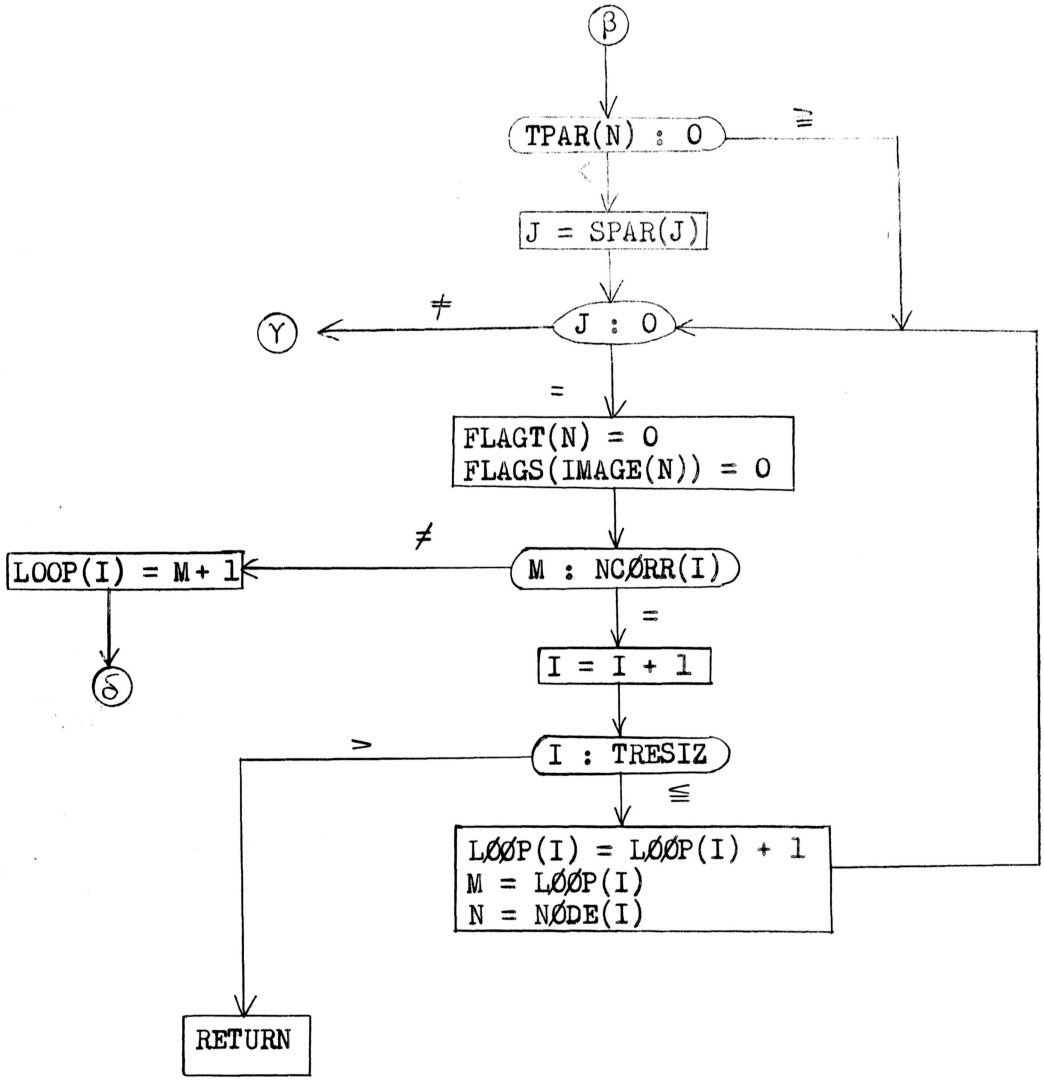
Flowchart 2

APPENDIX C

ALGØRITHM FØR DETERMINATIØN ØF STRUCTURAL CØNSTRAINTS
ØN POSSIBLE MATCHES



APPENDIX C (continued)



Flowchart 3