TRANSFORMATIONS AND DISCOURSE ANALYSIS PROJECTS

17. Resolution of Variable Classification

Preliminary version

Naomi Sager

LINGUISTICS DEPARTMENT
University of Pennsylvania
Philadelphia, Penna. 19104

1. Multiply classified words:

The alternatives program is concerned with those words which have more than one possible syntactic classification, e.g. words like study (noun or verb), total (noun, verb, or adjective), these (pronoun or pronominal adjective), little (noun or adjective); also words whose suffix determines their classification in more than one way, e.g. words ending in ed like compared (past tense verb or passive adjective). These words are received from the dictionary with special markings which call out decision procedures for resolving the alternative classification.

The dictionary entries, of course, must indicate all possible syntactic classifications of words, since at the level of dictionary look-up each word is considered independently and hence without reference to the restrictions derived from its occurrence in the sentence in a particular construction. It is important to decide as many of these cases of multiple classification as possible, as early in the program as possible, since each unresolved alternative gives rise to an ambiguity which complicates and multiplies the work of later stages, e.g. identifying local strings and higher level bracketing. On the other hand, the decisions among alternatives rest on features of the syntactic environment which in some cases is cumbersome to state in terms of strings of class symbols only, i.e. before some of the local bracketing

has been performed, and simple to state afterwards. Thus linguistic considerations up to a certain level of complexity only are included in the decision procedures, and it is expected that some cases of multiple classification will not be solved, but will be carried as ambiguities to be solved at a later stage in the program.

II. Code Numbers:

The input to the alternatives program (aside from the tape containing the program itself) is a tape containing the sentences of the text(s) with the dictionary and other inromation for every sentence word occupying set positions within a block of six machine words. As with other sections of the program, operations are performed with respect to this fixed format. A description of the relevant parts of the entry and two examples are shown in Fig. 1. In this program, only the second and third entry words are involved.

If an alternative exists in the sentence word, digits 1-3 of the second UNIVAC word of the entry will be occupied by a code number indicating the choice to be made, e.g. 130: V/N, 350: N/A, 153: V/A/N, 650: R/B, 140: W/S* in place of the usual alphabetic class mark, N, V, etc. The class mark and subclass information for each of the alternatives is then found in the locations shown in Fig. 1. In the case of success in deciding among the alternative class marks, the code number is erased and the appropriate block of class symbols is moved into the initial digit positions of the second UNIVAC word. Subsequent programs will

*Some of the main class marks are: V (verb), N (noun), A (adjective), R (pronoun), B (pronominal adjective, e.g. this), W (verb with tense mark, e.g. studied), S (passive adjective, e.g. studied).

look for the syntactic classification in this position.

It will be noticed that the individual digits of code numbers like 130, 153 bear a simple relation to the alternative classifications making up the choice: 1:V, 3:N, 5:A, 6:R, 4:S. Thus 130 stands for the V/N choice, 153 for V/A/N, etc. There were several reasons for choosing this non-compact representation aside from the convenience of immediate interpretability. At various points in this and subsequent programs, it is of use to know the possible outcomes of a decision even if the decision could not be made. For example, the situation may arise where in scanning the sentence for a particular class mark which will help us decide one alternative, we encounter another. Here we should like to know if the desired symbol is among the possible outcomes of deciding the second alternative, since in that case we would want to repeat the operation at a later time when the second alternative may be solved. E.g. one of the tests to distinguish between N and V calls for a check that there are no preceding verb forms in the sentence, i.e. no W, S, V, G. Suppose in scanning back we encounter a 130. This could be a V, but so far was not decided. It may be that in a future run through the sentence, it will have been decided, and this will enable us to complete the present test. An economical rerun procedure would repeat only those tests which have a chance for success as a result of further changes. In this example, we would ask whether the code number of the alternative encountered contains the digit 1, and if so, record the test for rerunning.

Another feature of the 3-digit code is the possibility of changing

the order of the digits to control the order of application of parts of the decision procedure. The tests (more fully described below) eliminate one alternative when successful, and the order of the digits stands for the order of attempted elimination of alternatives, e.g. 130 calls first for tests eliminating V and then for tests eliminating N. Often we know in advance which alternative is likely to be the case, e.g. value is more likely to be a noun, and show is more likely to be a verb. In the preferred-order system, value would be listed as 130 and show as 310 with the corresponding difference in order of application of tests. This feature has not been utilized in the present program, but could become important when the decision procedures become long with many parts.

III. Linguistic tests:

As indicated above, linguistic considerations which distinguish among alternative syntactic classifications are applied in the form of tests. (Flow charts for the tests used in the UNIVAC program are appended. In addition, some of the tests are described below.) The tests ask whether certain other class marks occur in the sentence environment of the word in question with the hope of uncovering a construction which makes one of the alternative readings impossible, i.e. rejects one classification because it gives rise to an unacceptable string. An example is Test 02 which asks if the preceding class mark is T1 (a, an) or T2 (the). If so, the word in question cannot be a verb as constructions like the acquire or a value (value taken as a verb) do not occur.

Success in resolving an alternative depends on having the right question to ask for the given construction. In this it differs from the tree-following activity of, e.g., the local bracketing program and resembles the trouble-shooting program for non-well-formed sentences in which pointed questions are asked about the results of previous analysis. The number of tests is arbitrary, but the number of simple tests is limited. In addition to Test 02 above, other simple tests are:

Test 01: asks if P09 (of) follows. If so, the word in question cannot be a verb, except for a special class of verbs which take of as part of their object, e.g. consist of. These verbs have the subset 29 (20 plus the P subset number 09). The tests asks, then, that if a P09 was found, the verb subsets be checked to see that the number 29 does not occur. If no 29 occurs, the test is successful, and the verb possibility is eliminated.

Test 03: asks if B (this, as adjective) precedes. If so, the verb possibility is eliminated.

Test 04: asks if a sentence breaker precedes, i.e. certain conjunctions or a $ (sentence beginning or end). If so, it asks if the word in question has the plural present tense mark, "S," e.g. studies. If this is the case, the verb possibility is eliminated, since words like goes, studies (as verb) do not occur in the sentence initial position.

Test 09A (for N/A): asks if the following class mark is N; if so, the noun possibility is eliminated.

Test 11 (for W/S): asks if the preceding is a verb with one of the following subset marks: a (the verb have), b (the verb be), 03 (verbs like seem, appear which take predicate objects). If so, the tense-marked verb possibility (W) is eliminated.

In connection with Test 11, it is worthwhile pointing out that the alternatives tests are restricted to cases which can be decided definitely, because the rejected information is erased. The bracketing program, on the other hand, when it encounters an alternative, may make a temporary decision to class the word as one or another of the possibilities on the basis of the most probable reading of the entire string, but a mark is left indicating another reading is possible. Test 11 is an exception in that it may give rise to a wrong result; e.g. in the sentence All he had appeared worthless, appeared would be decided in favor of S because of the preceding had, whereas it is W, the main verb of the sentence. The loss of information is only apparent, however, since the trouble-shooting program will have to examine the $\{ W\ S \}$ verb string in order to fit it into the sentence as a whole. So many W/S occurrences are solved by Test 11 that it was decided to retain it with the provision for error as described.

The more complex tests usually involve checking the sentence for higher level constructions, but in terms of class marks only. The

tests could be stated more efficiently in terms of bracketed segments, but these are not available yet, the purpose of the alternatives program being to assist the bracketing program in advance by eliminating as many multiple readings as possible.

Test 05: asks if the word in question is preceded by A. If so, the verb possibility is eliminated unless the A is the final symbol (i.e. the grammatical center) of a preceding noun phrase. An A cannot be the center of a noun phrase, however, if the noun phrase begins with B, T1, or R/B; e.g. study can be a verb in the wise study, but not in this wise study or a wise study. An A also cannot be the center of a noun phrase which follows a sentence breaker (C5 or $) or a preposition (P) without beginning with the, e.g. because wise study, from wise study. Thus the test asks that we skip back as long as the symbols encountered are those permitted in a noun phrase until we come to a T1, B, 650, $, C5, or P. Any one of these means success; the A cannot be a noun phrase head, hence the word in question cannot be a verb. If instead we come to a symbol other than those stated, we cannot tell the function of the A, hence the test fails.

Test 07: asks if there are any other possible main verbs between the nearest left and right sentence breakers. K (which,

etc.) and C5 (because, etc.) are permitted boundaries on the left, but not on the right, because of the nesting property of clauses, e.g. which of the conditions which were imposed...

Test 08: asks if the word in question is possibly an N1 and singular because these nouns (house, chair) require that the noun phrase begin with an article or quantifier (T, Q, or L). Otherwise the noun possibility is eliminated.

IV. Structure of the program:

There are a number of ways the various tests could be arranged and called upon to resolve alternative classifications. The structure of the UNIVAC program was influenced by the small amount of active sotrage available, too small to accommodate all the tests simultaneously. Therefore, some form of tape look-up was indicated, for which the most efficient unit seemed to be the complete subprogram for handling alternatives of a given type, e.g. 130, 153, etc., despite the fact that this entailed duplicate coding where a test applied to more than one type of alternative. Each program then consists of a series of parts (the tests) interspersed with instructions as to what to do in case of success or failure. It is possible that with a larger machine and a larger number of tests, a program based on a test library, with the subprograms (e.g. 130, 153) acting as compilers would be more economical. A program of this type was flowcharted, but proved impractical for UNIVAC I.

The operation of the present program can be followed with the aid of the appended flow charts. The main program (Fig. 2) contains two loops, a sentence loop (upper half) and a word-processing loop (lower right). Its operation is as follows. At the start of the program, we input the first sentence of the test by means of the general input routine (G I R) common to all the programs. If we are not at the end of the tape, we set the initial conditions for running the sentence and enter the word loop. The initial settings are 1) the first sentence (1 → 1) and 2) clear the rerun-indicator (0 → N). The function of the rerun-indicator will be discussed later. In the word loop, if we are not at the end of the sentence, we examine each sentence word in turn to see if its class mark is a decimal digit, indicating the presence of an alternative to be decided. If it is, the subprogram corresponding to the 3-digit code (e.g. 130) is located on tape and copied into the memory and control is transferred to the subprogram (all the subprograms return to the same point in the main program). If no code number was found, the next sentence word is examined, and so on until the end of the sentence. At the end of the sentence, a decision is made whether or not to rerun the sentence (see below). If yes, we do so; if not, we output the sentence by means of the general output routine (G O R) and begin with another sentence.

The general form of a 3-digit subprogram is shown in the flow chart of Fig. 3 . Tests are taken up one at a time. If a test fails, the next one is tried, and so on. If all tests fail, control is returned

to the main program. If any one test succeeds, the appropriate changes in the class mark are made, and control is returned to the main program without reference to the other tests (now obviously unnecessary) unless it is a 3 alternative choice. Before exiting from a success path, the sentence is marked for rerunning. This is a device for controlling the iteration of the sentence loop, since as long as some change has been made in the class marks of the sentence, it is possible that a test which previously could not be completed because of a second interfering alternative could now be solved. Other ways of handling the incomplete tests could obviously have been used. In the test-library version of the program previously mentioned, individual tests were marked for rerunning instead of the entire program, thus saving the repetition of tests which already failed and the handling of entries for which no incomplete tests were recorded.

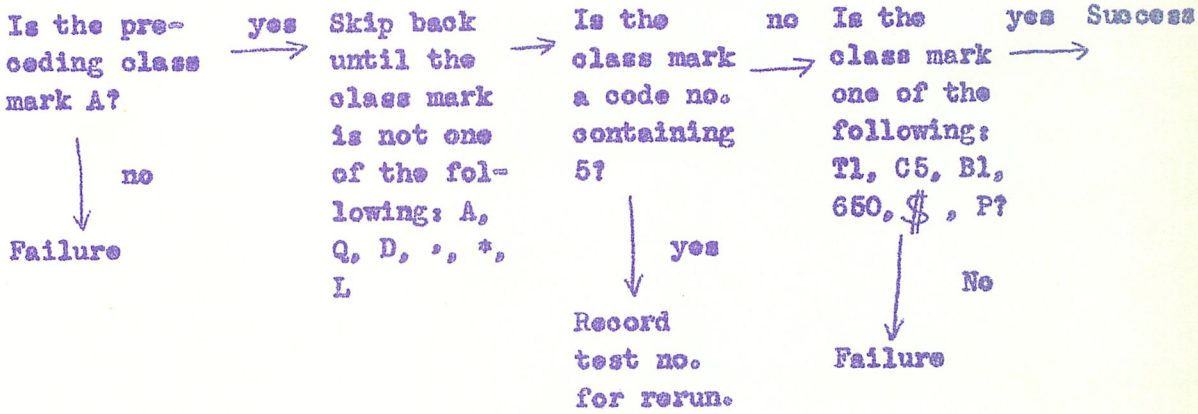
The tests themselves have been described briefly in Section III. An idea of how the whole program works may be gained by following an example taken from a sample text. The description follows the test-library program, which uses the same tests as the present UNIVAC program but differs in that the subprograms initiate jumps to the tests and store the numbers of individual tests which are to be repeated in what is called the "word space" of the particular entry (entry = 6 word format corresponding to each sentence word).

V. Example: the word total in the construction:

a higher	total		amount	of	enzyme	is		formed.
T1 AA	153 N	A	130 N1	P09	N	W		140 S
	V 02		V 21			Vb031121		W 0102
						200		300

On the first run through the sentence we arrive at the word total and find digits 153 in the class mark position. We search F.T. 1 for 153 and are directed to program 153 where we load initial test numbers: 01, 02, 03, 04, 05, 06, 07, 08, 09 into a register and make several success settings. The program initiates a jump to test 01 (test 01: Is the following class mark P09, i.e. the word of?); test 01 fails and we return to the program for a jump to test 02 (test 02: Is the preceding class mark T2, i.e. the word the?); test 02 fails and we return for a jump to test 03. Tests 03 and 04 also fail and we jump to test 05.

TEST 05:



It can be seen that test 05 succeeds, i.e. the verb possibility is eliminated. This leaves the N/A classifications still to be decided. The success activity is to change code number 153 to 350. The program is also set to rerun the sentence since a change was made. A jump is made to program 350 in accordance with the new code number.

Program 350 includes tests 08 and 09. Test 08 fails and test 09

asks if the following class mark is N, and if not, whether it is a code number containing 3 (noun possibility), in which case the test is recorded as incomplete. It can be seen that this is the case here. No other test succeeds, so the class mark for total remains the code number 350, the result of eliminating V from V/A/N, and the work space contains the test number 09.

The next word is amount, also an alternative. We jump to program 130 via F.T. 1 and there find initial test numbers: 01, 02, 03, 04, 05, 06, 07. Test 01 succeeds since the following class mark is P09 (see above). Success settings of program 130 and test 01 cause the code number to be replaced with the class mark N. We now return to the main program and proceed to the next word of the sentence, etc.

When all the words of the sentence have been examined, we get a positive answer to the END-SENTENCE? query and a positve answer to the RERUN? question, since at last one success was achieved. We begin the program again, but this time we examine the work space of each entry to see if test numbers are stored instead of examining the class mark position. When we reach total we find test number 09 stored and again after transferring control to the code number program (now 350), we apply test 09. This time the test succeeds since in the meantime amount was solved giving us the class mark N in the position following the alternative, as called for. Success settings for program 350 and test 09 direct the erasure of code number 350 and the writing of A in the class mark position for total.

Notational Conventions

(i, j, k) = k^{th} digit of j^{th} UNIVAC word of entry i

entry i = entry being examined or tested for alternative class marks. NOTE: i - 1 means Address of i - 6 UNIVAC words.

digit 1 in (i, 2, 1) = class mark beginning with digit 1; may be more than 1 digit (will be clear in the context)

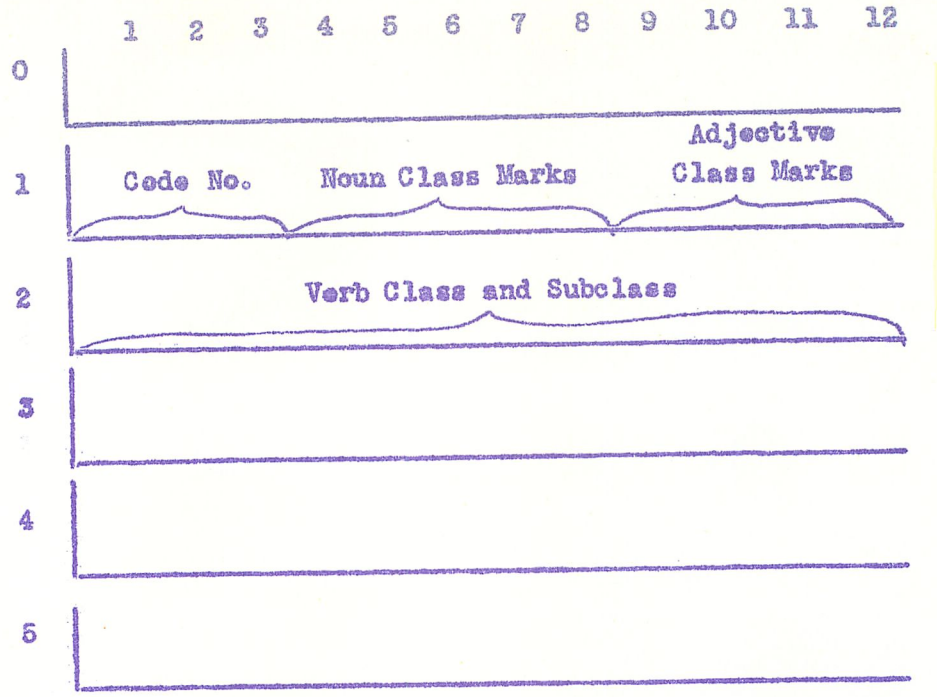
x, y = x or y

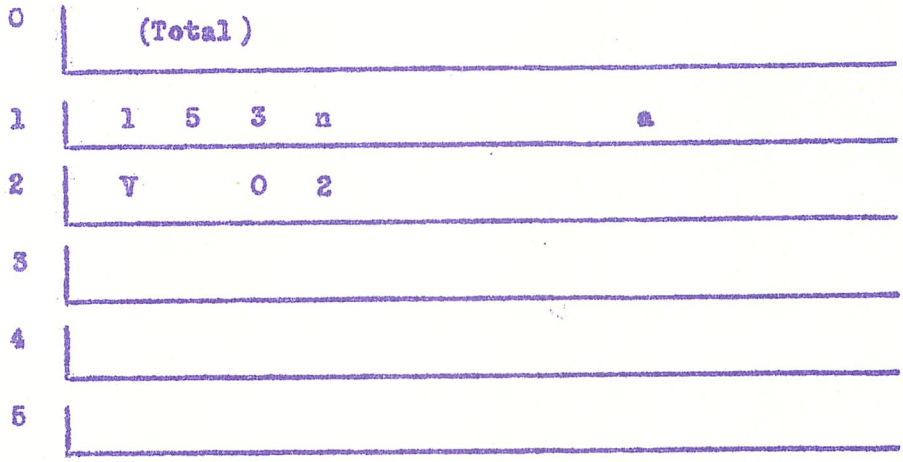
"Skip back..." = Examine i - 1, i - 2, i - 3...i - 1

d_{m-n} = m^{th} to n^{th} digits of (i, 2)

 = Omit intervening operations but leave room in code for adding later.

$X/_{y}$ = member of class X, but not subset y


1 2 3 4 5 6 7 8 9 10 11 12
0
1
Code No.
Noun Class Marks
Adjective Class Marks
2
Verb Class and Subclass
3
4
5


0
(Total)
1
1 5 3 n a
2
V 0 2
3
4
5

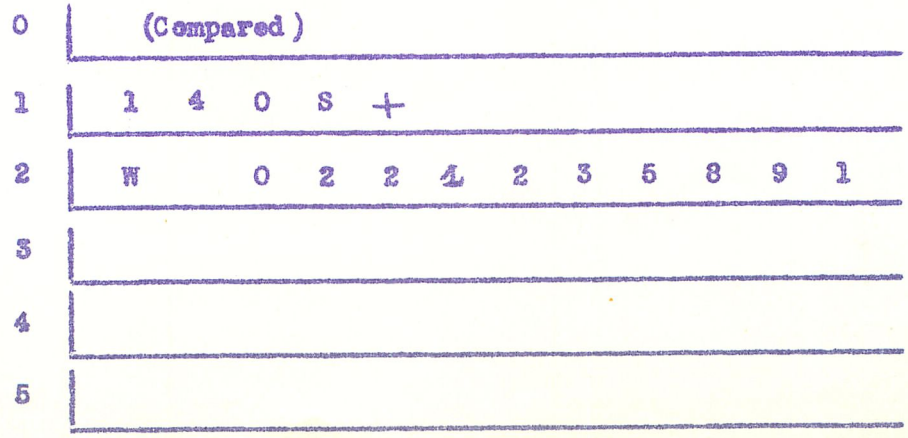
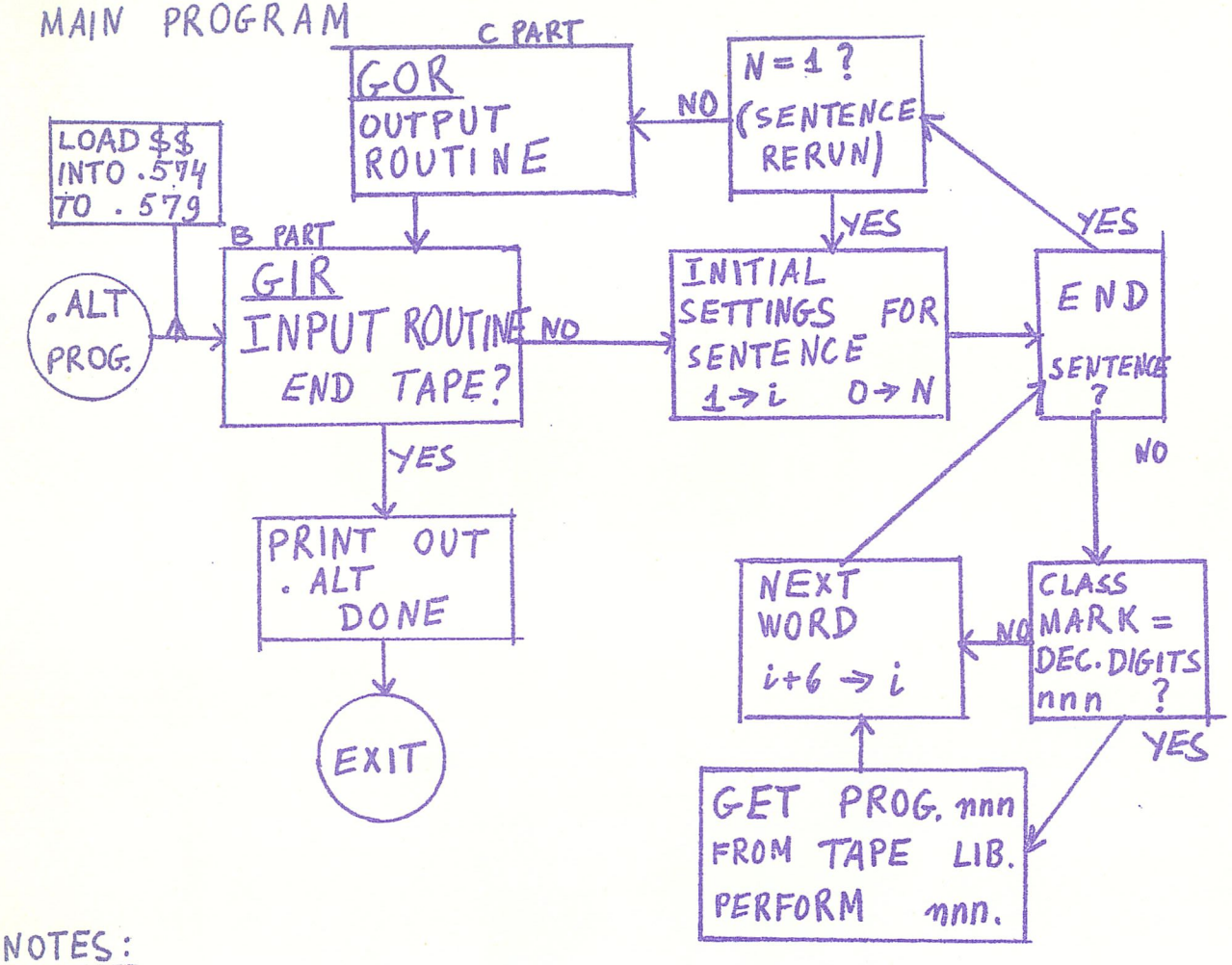

0
(Compared)
1
1 4 0 S +
2
W 0 2 2 4 2 3 5 8 9 1
3
4
5

Fig. 1

TDAP 17

ALT
MAIN PROGRAM

nnn PROGRAM



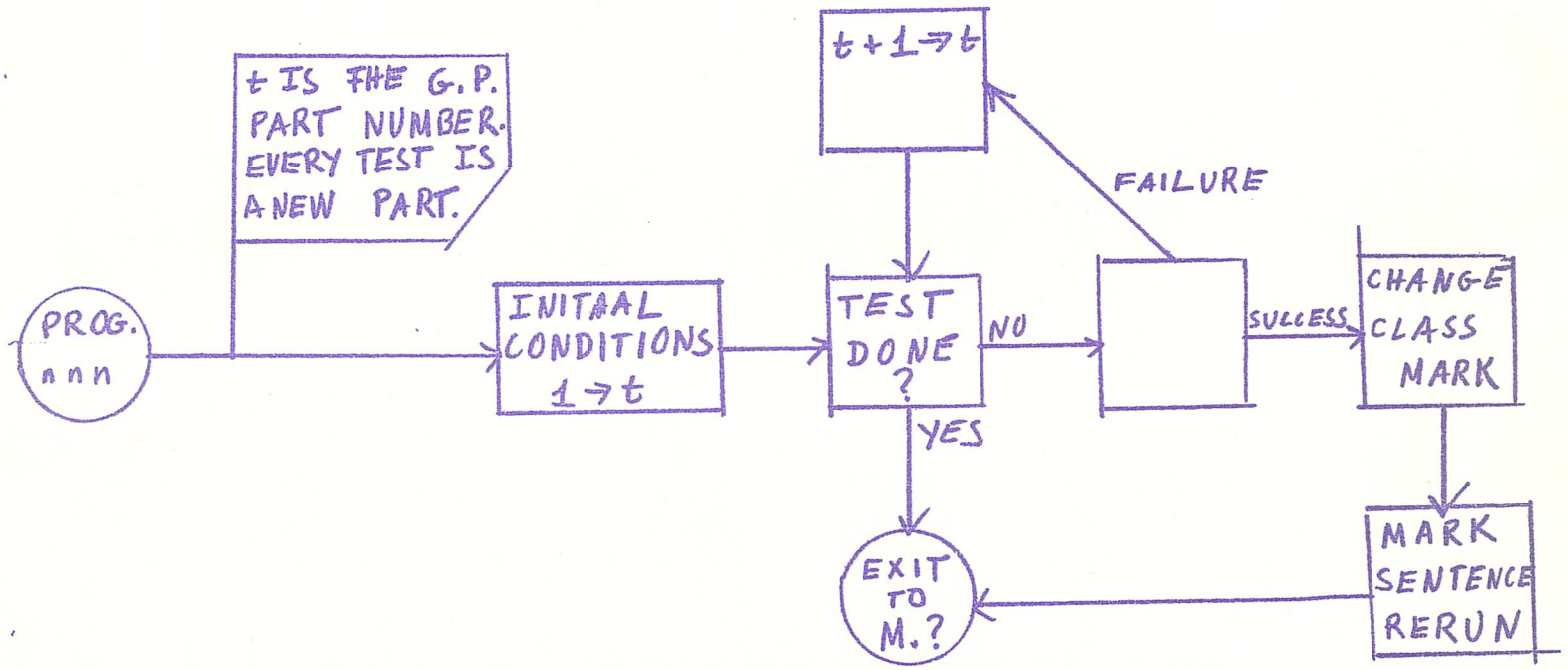
NOTES:

1) .ALT STORES i in .020 FOR THE nnn PROGRAMS.
2) nnn PROGRAMS ARE RELATIVIZED TO L.120; AND RETURN TO .ALT L56 = 22+34
3) N is A 501 = L.73.
4) .581 IS IN A9 = L.31

FIG. 2

TDAP 17

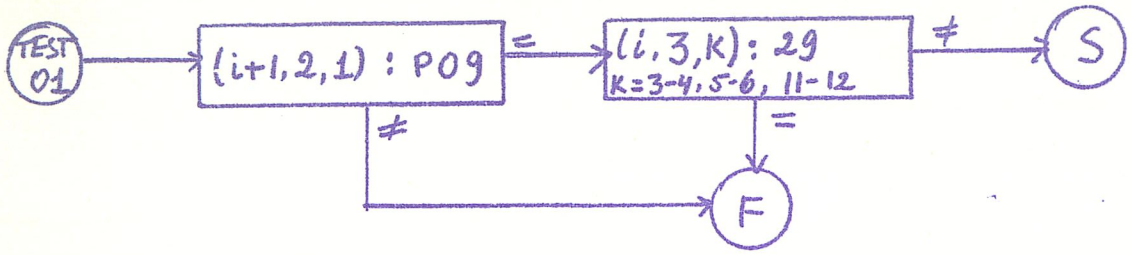
. ALT
GENERAL FORM OF PROGRAM nnn

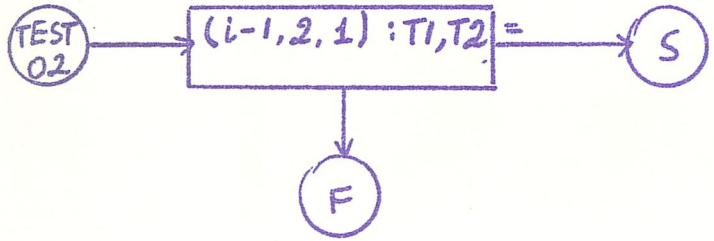


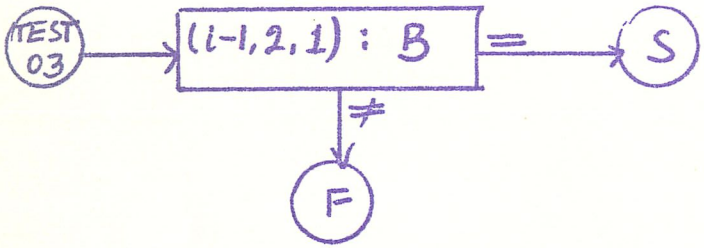
RETURN TO L.A 28 = L.50

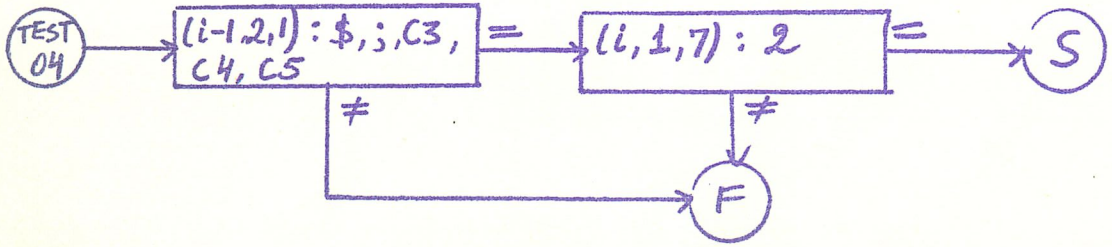
FIG. 3

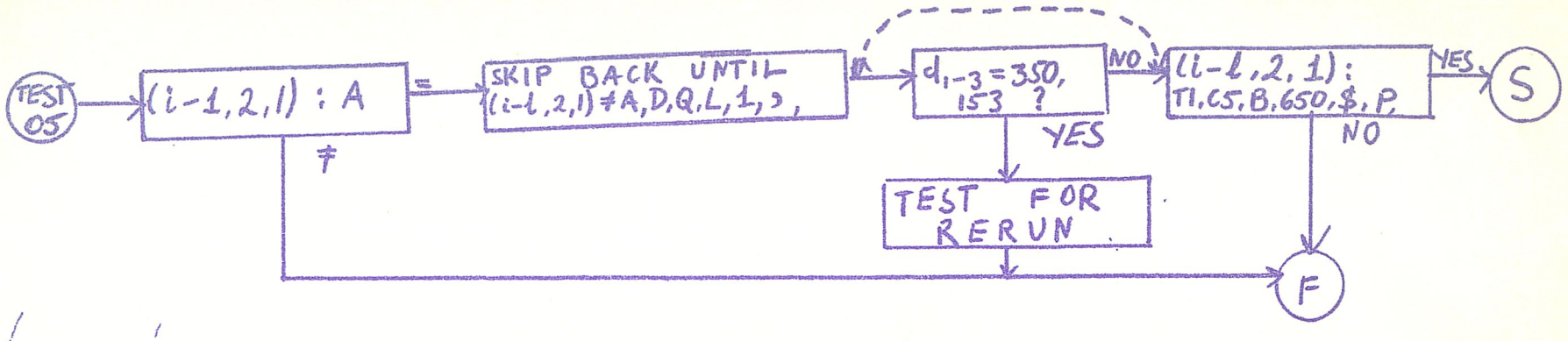
TDAP 17

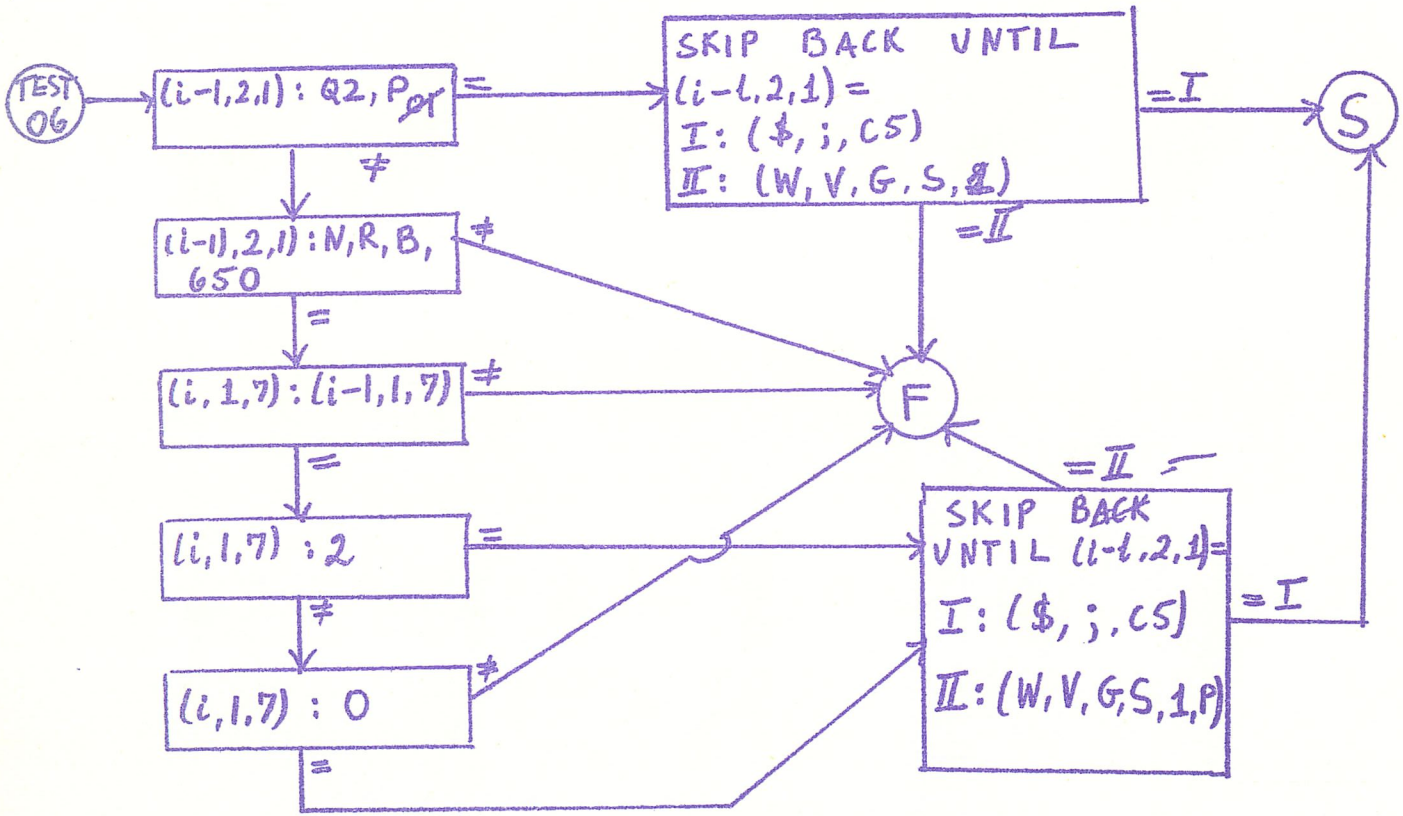

TEST 01
(i+1,2,1) : POQ
=
(i,3,K) : 29
K=3-4, 5-6, 11-12
≠
S
≠
=
F

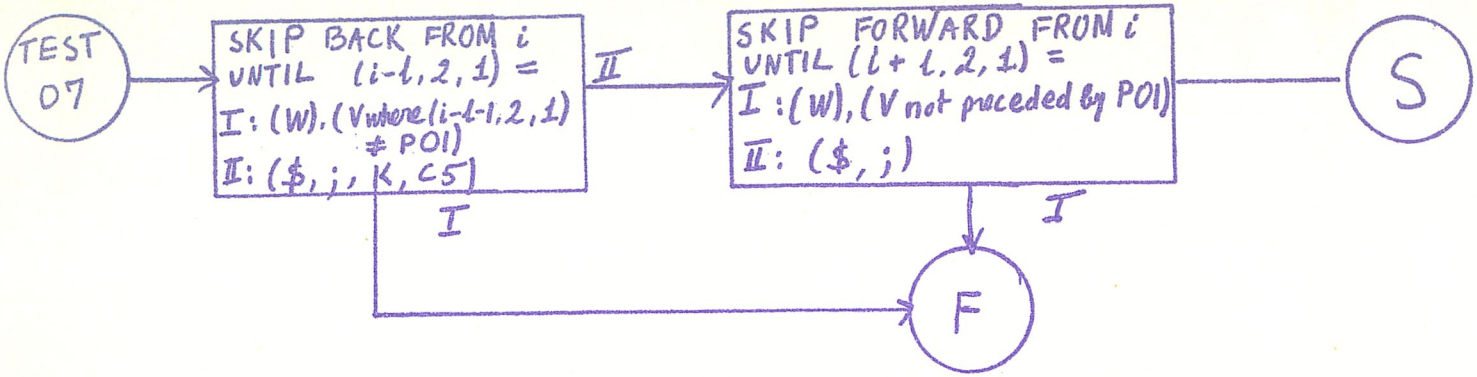

TEST 02
(i-1,2,1) : T1,T2
=
S
F

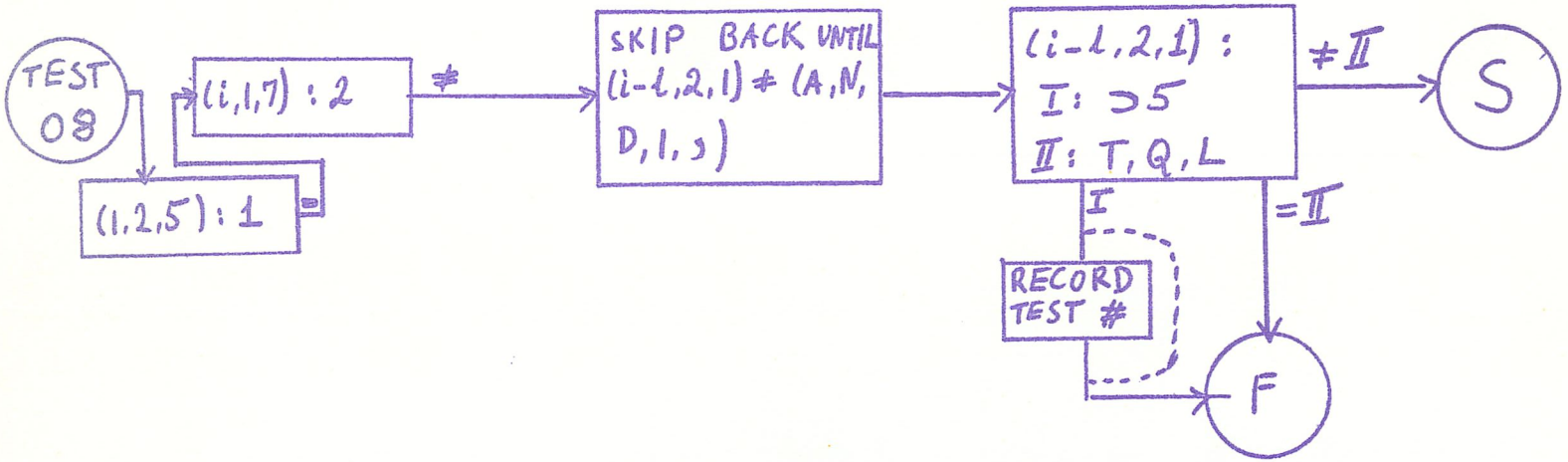

TEST 03
(i-1,2,1) : B
=
S
≠
F

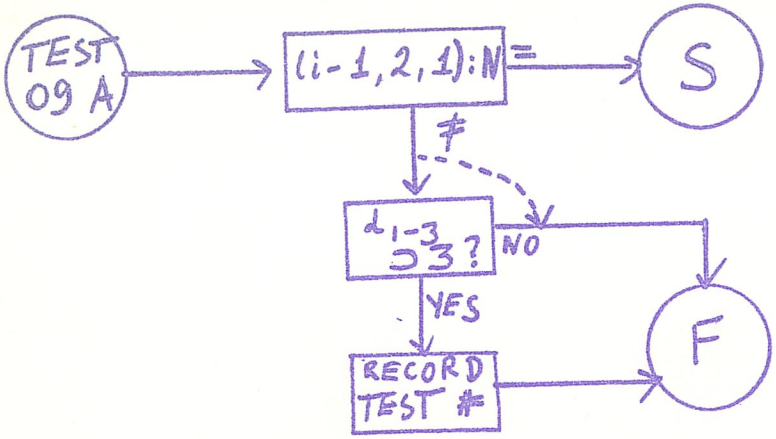

TEST 04
(i-1,2,1) : $,;,C3, C4, C5
=
(i,1,7) : 2
=
S
≠
≠
F

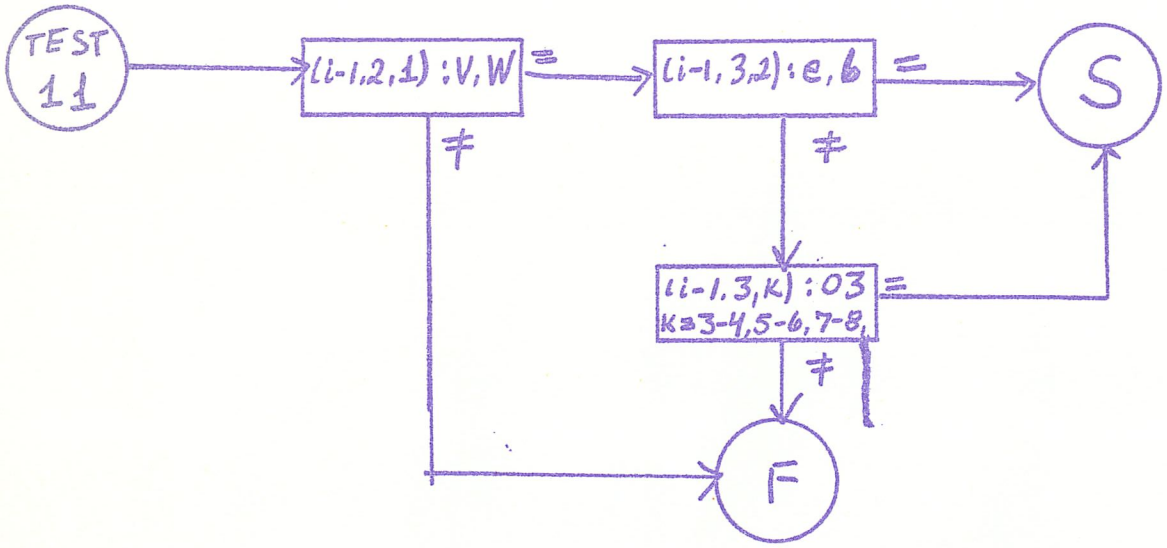

TEST 05
(i-1,2,1) : A
=
SKIP BACK UNTIL (i-1,2,1) ≠ A,D,Q,L,1,>,
IR
d1,-3 = 350, 153 ?
NO
YES
(i-1,2,1): T1,C5,B,650,$,P,
YES
NO
S
TEST FOR RERUN
F
≠

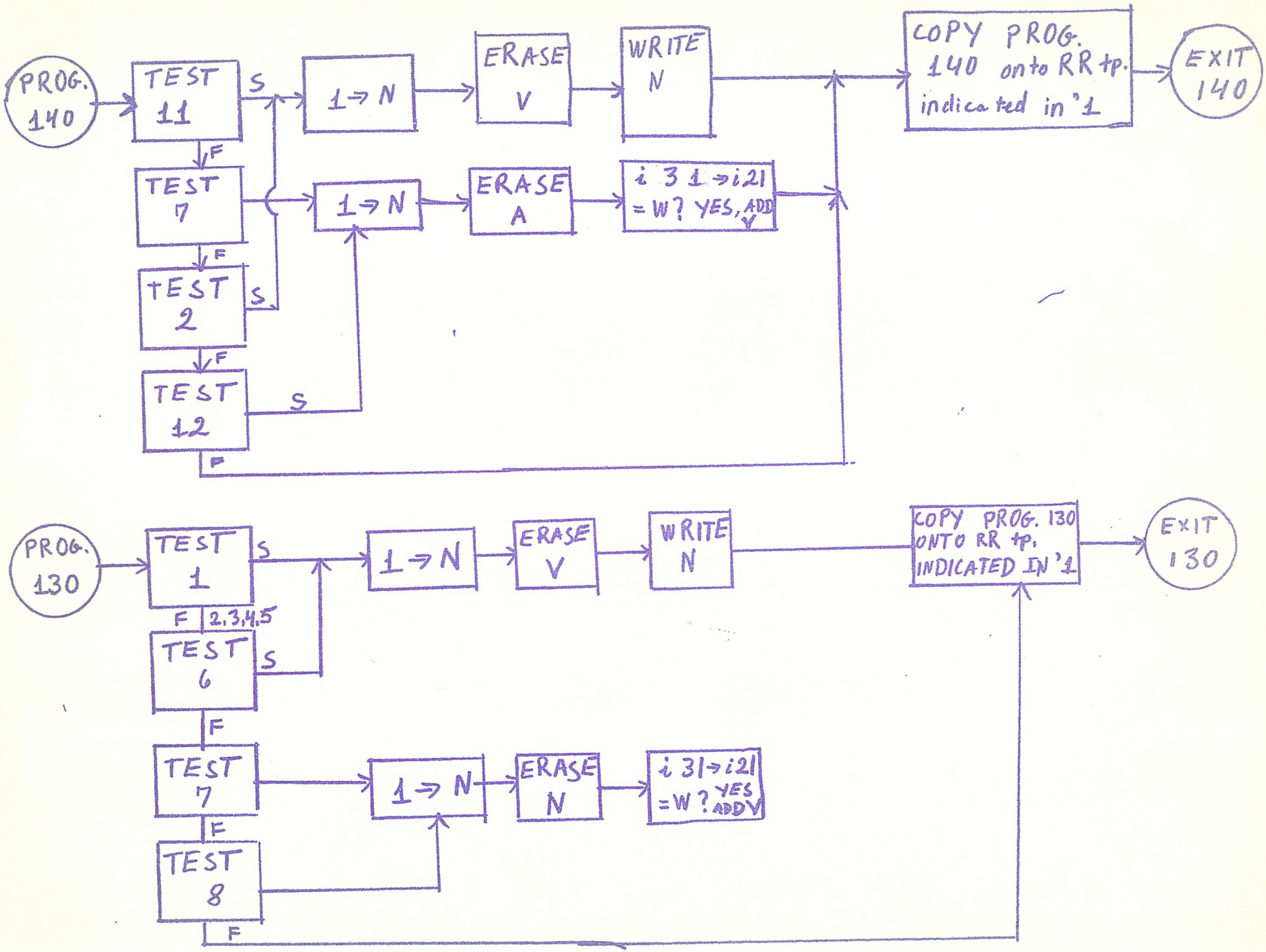

TEST 06
(i-1,2,1) : Q2, Por
=
SKIP BACK UNTIL (i-1,2,1) =
I: ($, ;, C5)
II: (W, V, G, S, 1)
=I
S
≠
=II
(i-1),2,1) : N,R,B, 650
≠
=
(i, 1, 7) : (i-1,1,7)
≠
F
=
=II
(i,1,7) : 2
=
SKIP BACK UNTIL (i-1,2,1)=
I: ($, ;, C5)
II: (W,V,G,S,1,P)
=I
≠
(i,1,7) : 0
≠
=


TEST 07
SKIP BACK FROM i
UNTIL (i-1,2,1) =
I: (W), (V where (i-1-1,2,1) ≠ POI)
II: ($, ;, K, C5)
II
SKIP FORWARD FROM i
UNTIL (i+1,2,1) =
I: (W), (V not preceded by POI)
II: ($, ;)
S
I
I
F

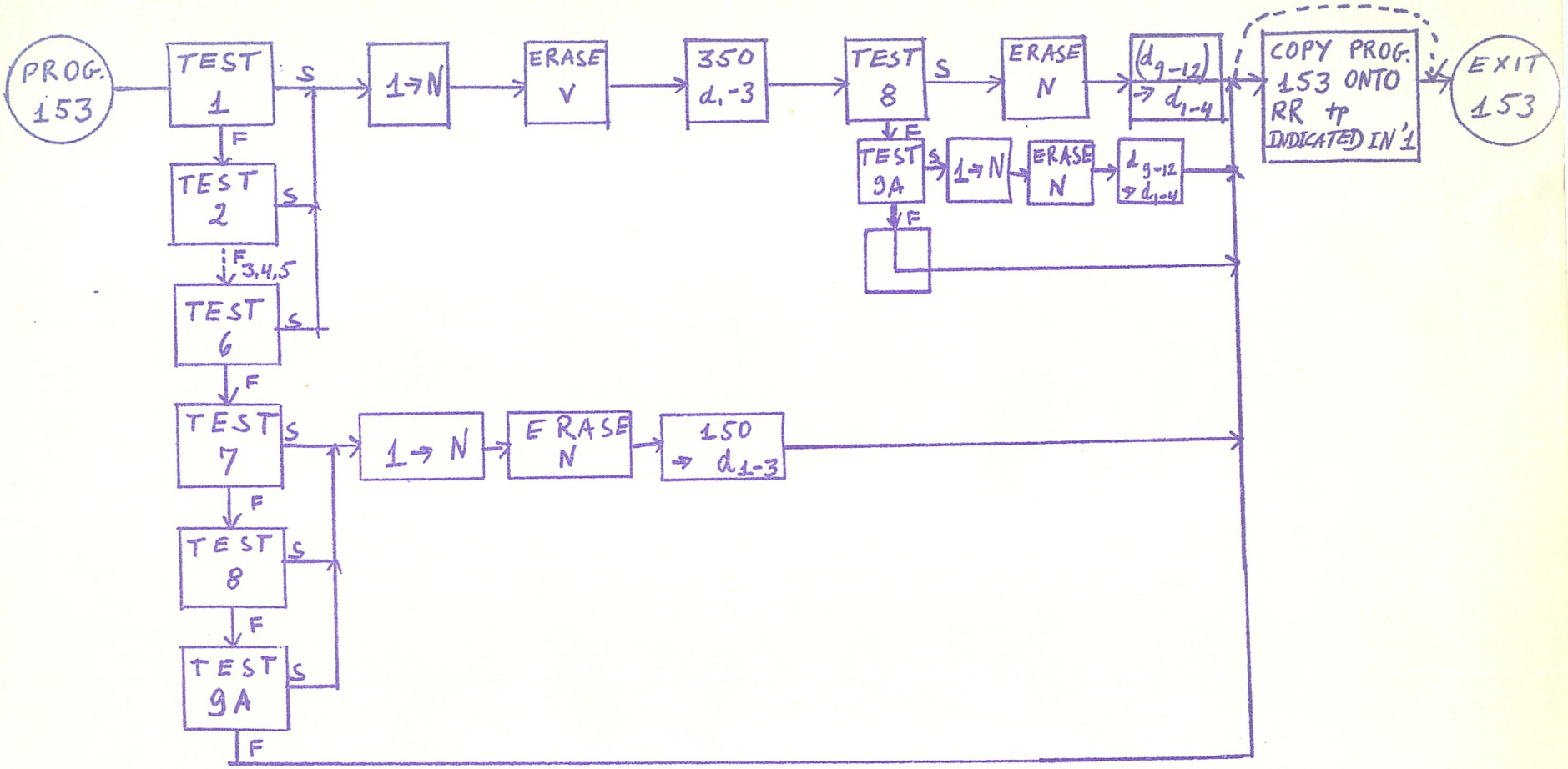

TEST 08
(i,1,7) : 2
≠
(1,2,5) : 1
SKIP BACK UNTIL
(i-1,2,1) ≠ (A,N, D,1,;)
(i-1,2,1) :
I: >5
II: T, Q, L
≠II
S
I
=II
RECORD TEST #
F

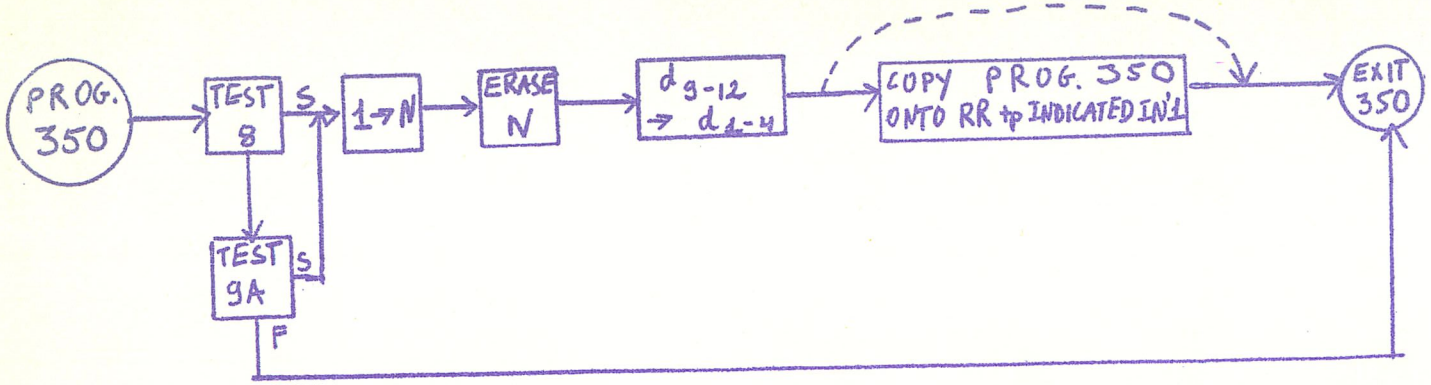

TEST
09 A
(i-1,2,1):N
=
S
≠
d_{1-3}
$\neq 3$?
NO
YES
RECORD
TEST #
F

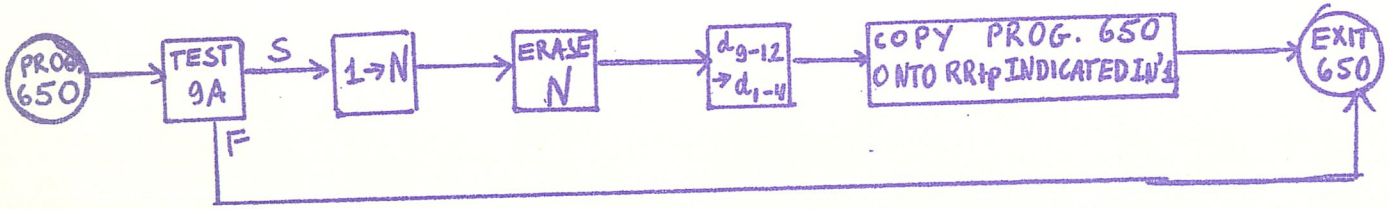

TEST
11
(i-1,2,1):V,W
=
(i-1,3,1):e,b
=
S
≠
≠
(i-1,3,K):03
K=3-4,5-6,7-8,
=
≠
F


PROG. 140
TEST 11
S
1 → N
ERASE V
WRITE N
COPY PROG. 140 onto RR tp. indicated in '1
EXIT 140
F
TEST 7
1 → N
ERASE A
i 31 → i21 = W? YES, ADD V
F
TEST 2
S
F
TEST 12
S
F
PROG. 130
TEST 1
S
1 → N
ERASE V
WRITE N
COPY PROG. 130 ONTO RR tp. INDICATED IN '1
EXIT 130
F
2,3,4,5
TEST 6
S
F
TEST 7
1 → N
ERASE N
i 31 → i21 = W? YES ADD V
F
TEST 8
F

TDAP 17


PROG. 153
TEST 1
S
F
1→N
ERASE V
350 d1-3
TEST 8
S
F
ERASE N
(d9-12) → d1-4
TEST 9A
S
F
1→N
ERASE N
d9-12 → d1-4
TEST 2
S
F
3,4,5
TEST 6
S
F
TEST 7
S
F
1→N
ERASE N
150 → d1-3
TEST 8
S
F
TEST 9A
S
F
COPY PROG. 153 ONTO RR +p INDICATED IN '1
EXIT 153





ERASE N
SHIFT (i,2)
L8
then R8

ERASE V
d_5 or d_6 = t ?
YES : GO ON
NO : 00... → (i,3)

ERASE A
SHIFT (i,2)
R4, then L4