DEFENSIVE PUBLICATION

UNITED STATES PATENT AND TRADEMARK OFFICE

Published at the request of the applicant or owner in accordance with the Notice of Dec. 16, 1969, 869 O.G. 687. The abstracts of Defensive Publication applications are identified by distinctly numbered series and are arranged chronologically. The heading of each abstract indicates the number of pages of specification, including claims and sheets of drawings contained in the application as originally filed. The files of these applications are available to the public for inspection and reproduction may be purchased for 30 cents a sheet.

Defensive Publication applications have not been examined as to the merits of alleged invention. The Patent and Trademark Office makes no assertion as to the novelty of the disclosed subject matter.

PUBLISHED OCTOBER 7, 1975

939 O.G. 9

T939,010

PROGRAM TEST METHOD

Richard Anthony Bender, Hyde Park, and Earl Lee Pottorff, Poughkeepsie, N.Y., assignors to International Business Machines Corporation, Armonk, N.Y.

Continuation of application Ser. No. 249,650, May 2, 1972. This application Mar. 25, 1974, Ser. No. 454,047

Int. Cl. G06f *11/04, 9/06;* **G05b** *17/02*

U.S. Cl. 444—1

26 Sheets Drawing. 127 Pages Specification

A method of analyzing computer programs which are to be tested. The method determines the data flow patterns through a program and uses the data flow patterns to identify a set of paths in the program which must be exercised in order to verify a program's correctness. The method includes the steps of: partitioning the program source statements into groups (each of which always executes as a serial unit, i.e. serit), determining the execution sequences of the different serial units, determining the data input and output activities of the serial units across their boundaries and the program boundaries, locating the sources of input data to each serial unit, and identifying test paths through the program such that every first-order dependency (each direct source of an input data item provided to each serial unit) is exercised by test data designed to exercise this set of test paths. The method can be applied to programs written in different programming languages.

FIG. 1

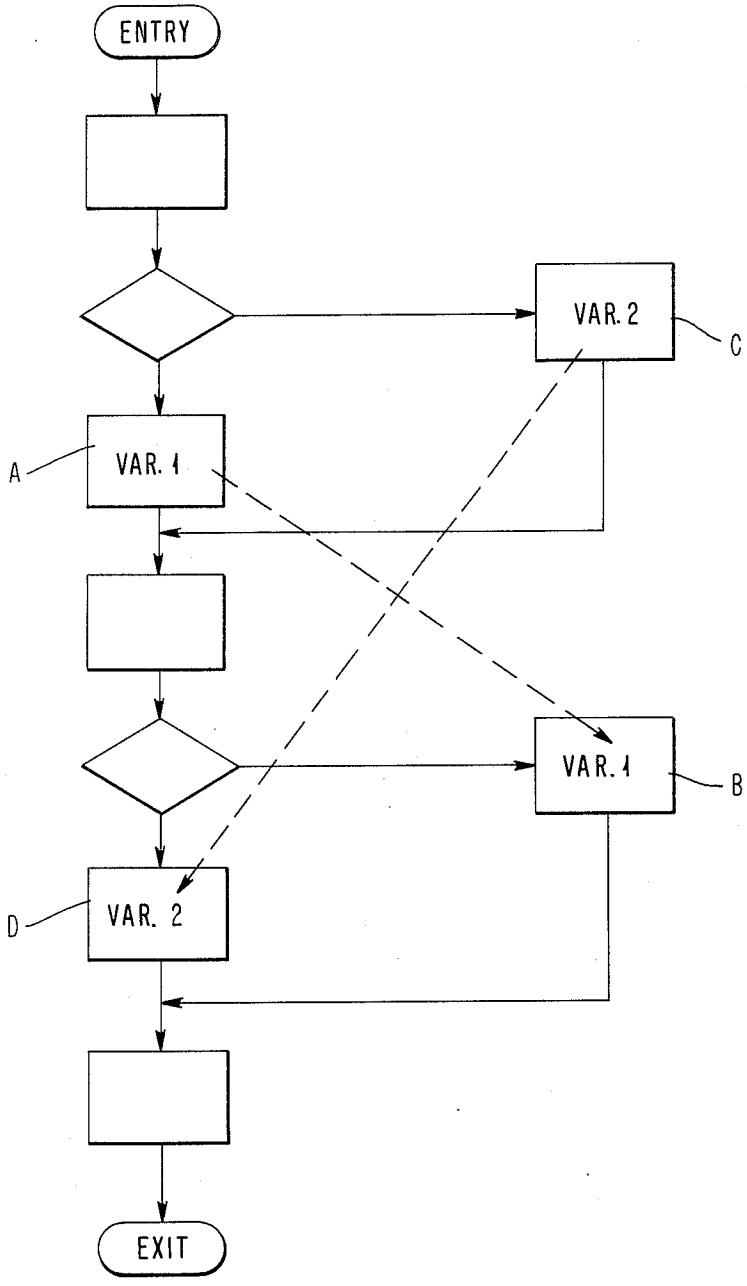


FIG. 2A

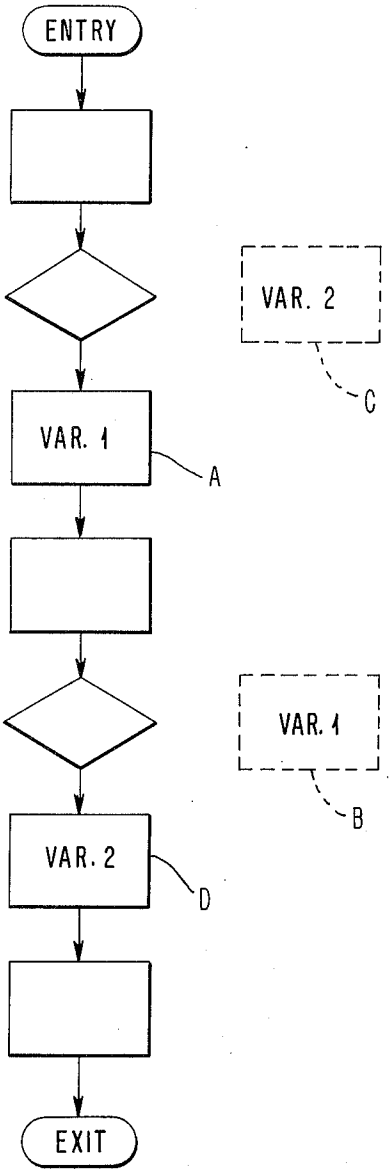

ENTRY
VAR. 2
C
VAR. 1
A
VAR. 1
B
VAR. 2
D
EXIT

FIG. 2B

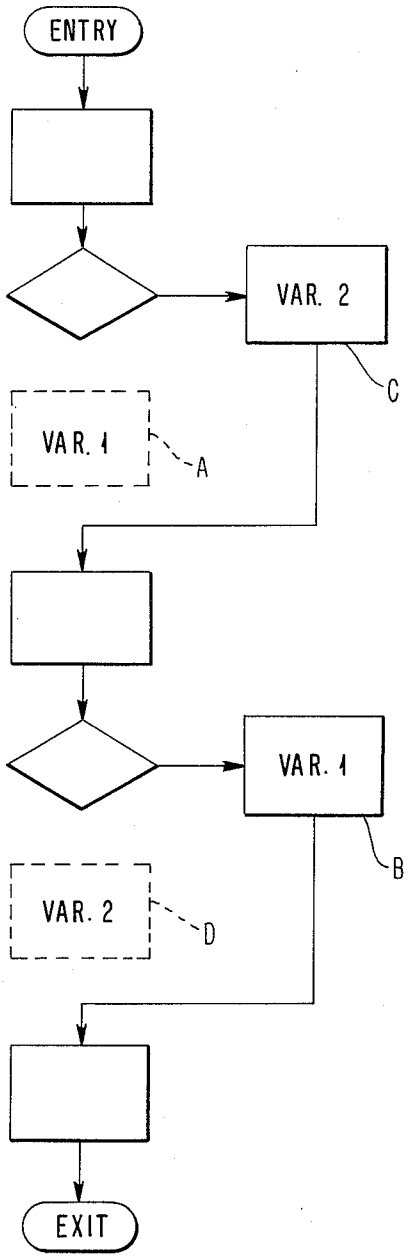

ENTRY
VAR. 2
C
VAR. 1
A
VAR. 1
B
VAR. 2
D
EXIT

FIG. 3A

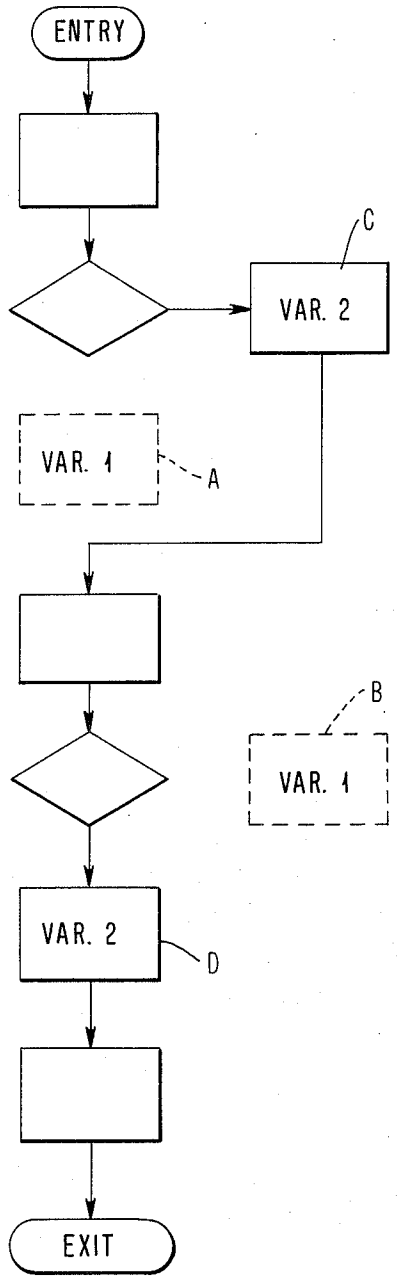

ENTRY
C
VAR. 2
VAR. 1
A
B
VAR. 1
VAR. 2
D
EXIT

FIG. 3B

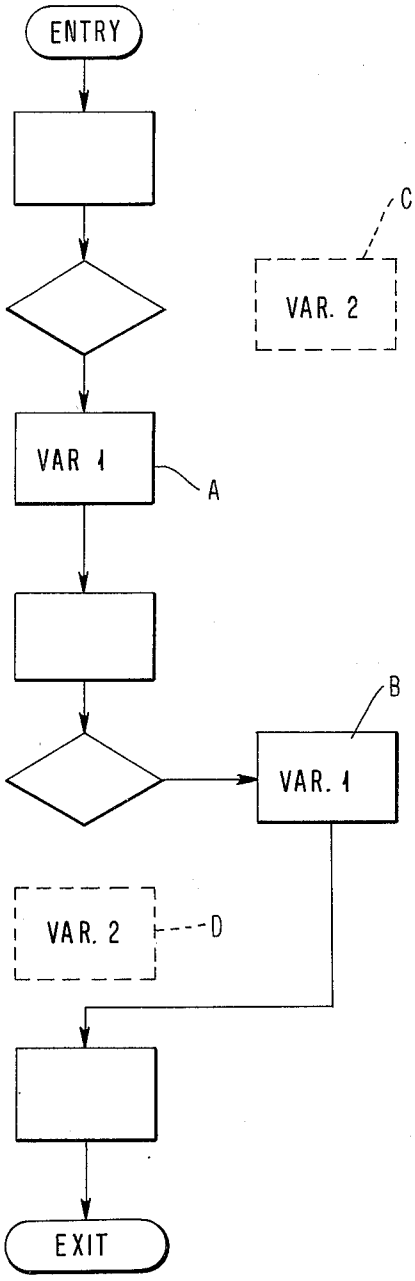

ENTRY
C
VAR. 2
VAR 1
A
B
VAR. 1
VAR. 2
D
EXIT

FIG. 4 (SERIT FLOWCHART)

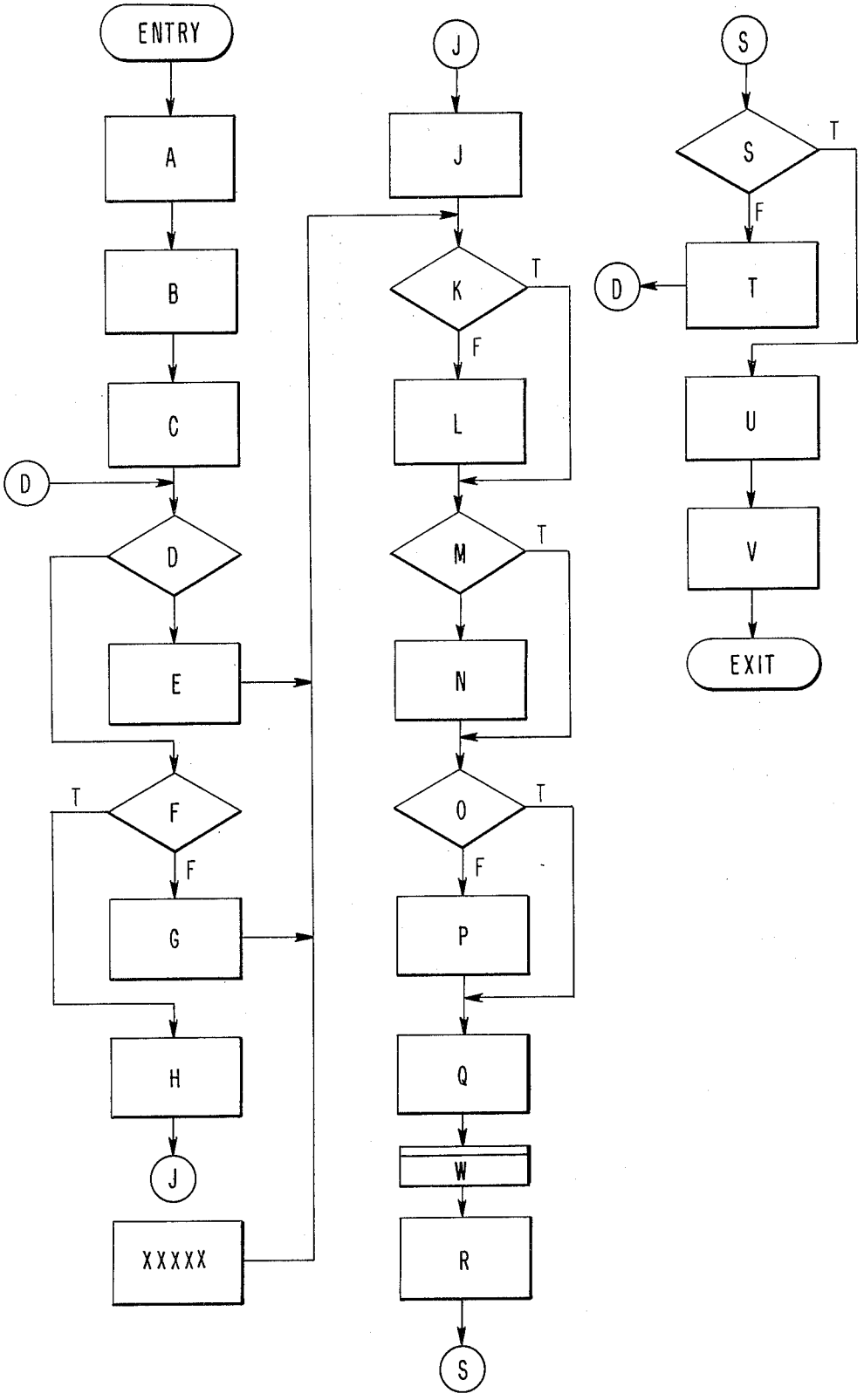


FIG. 5A (PROGRAM FUNCTION)

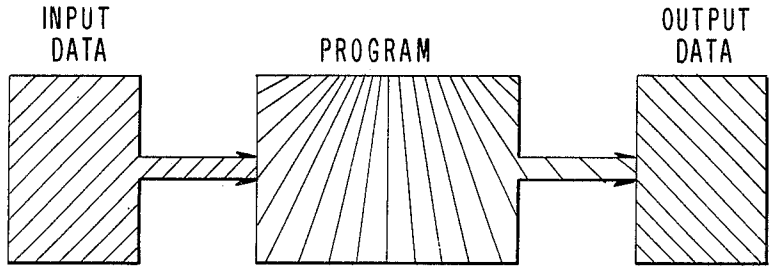
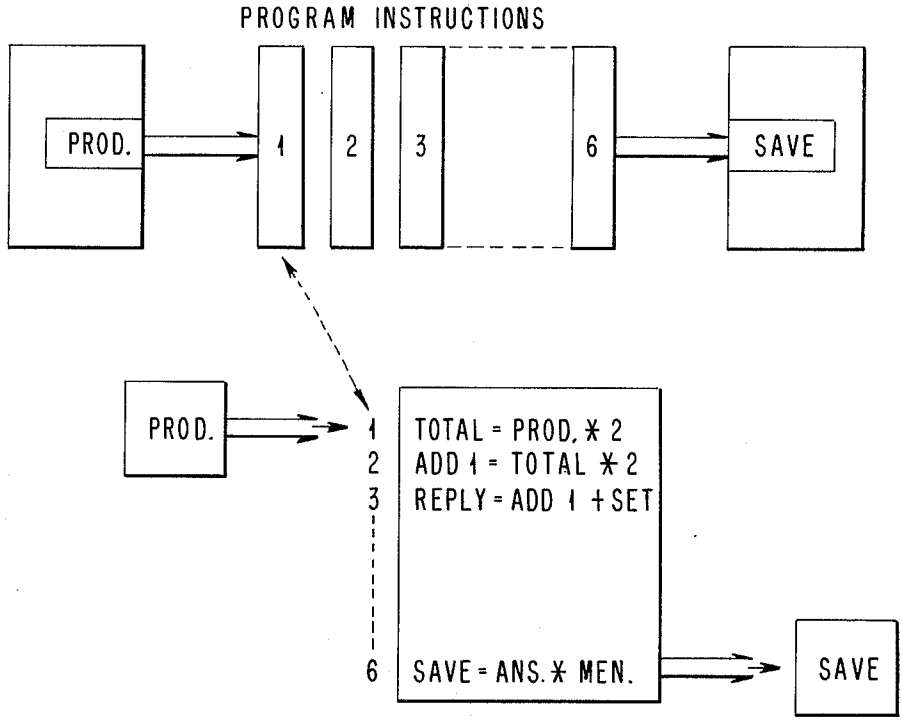


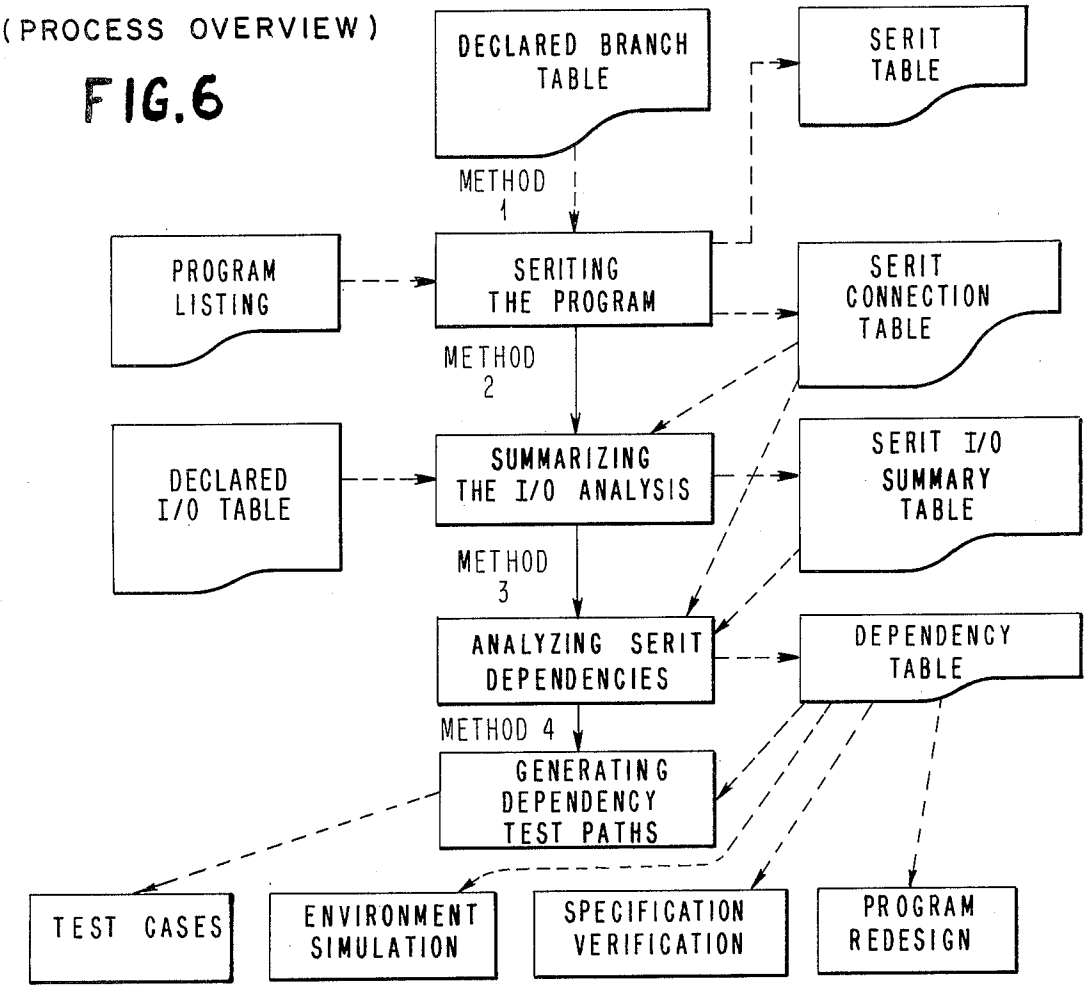
FIG. 5B

(TRANSFORMATION BY PROGRAM INSTRUCTIONS)



(PROCESS OVERVIEW)

FIG.6



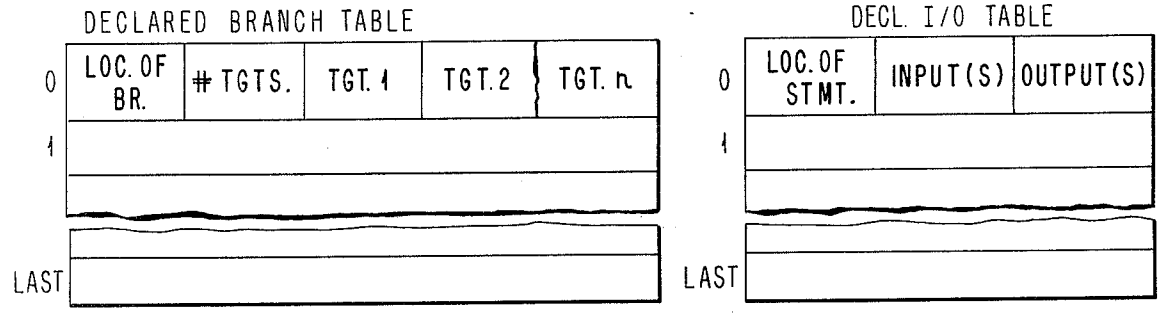
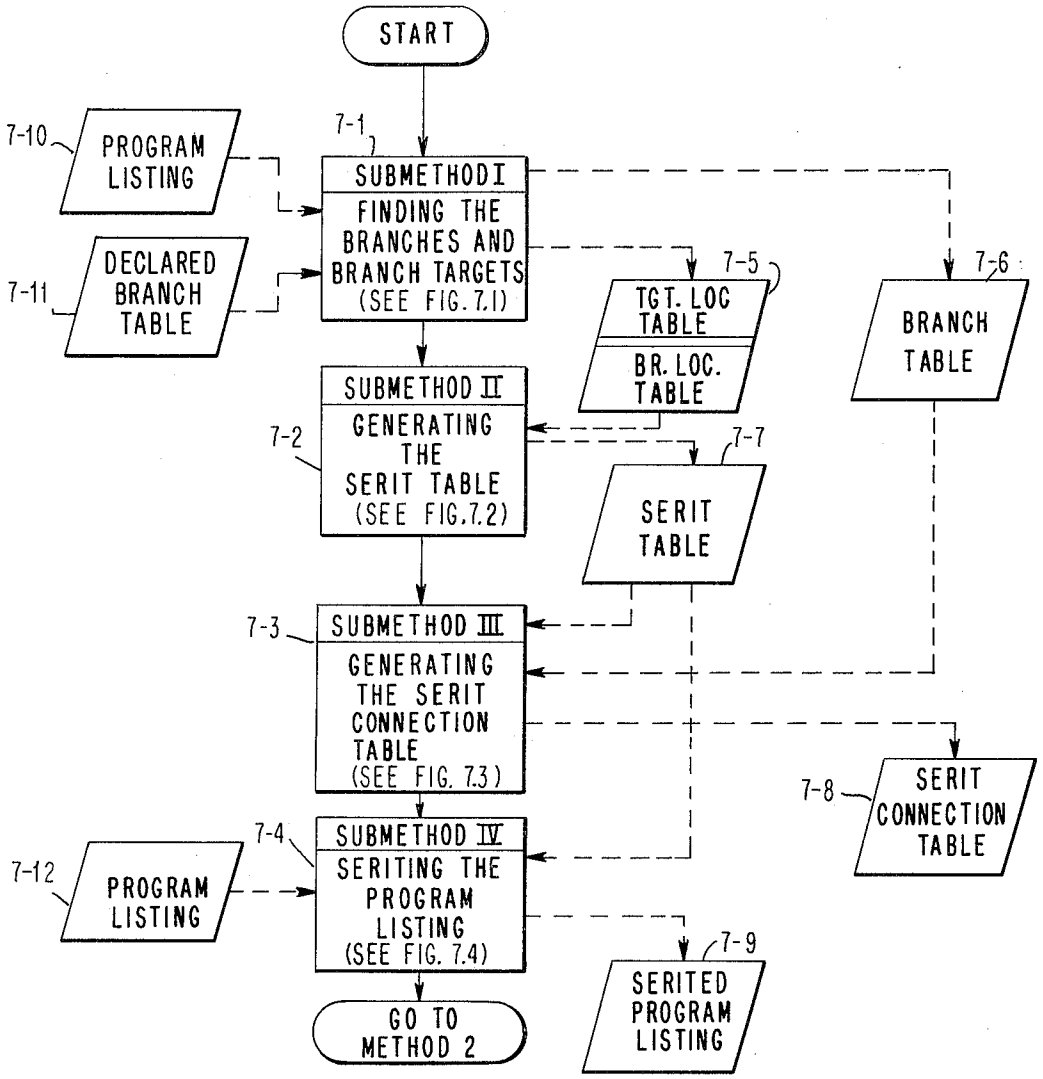
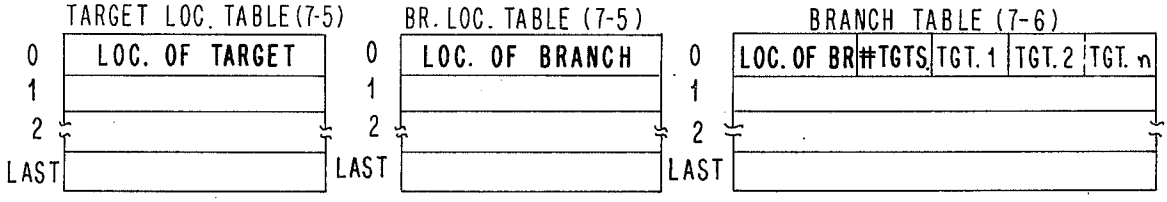


FIG. 7 (METHOD 1: SERITING OVERVIEW)





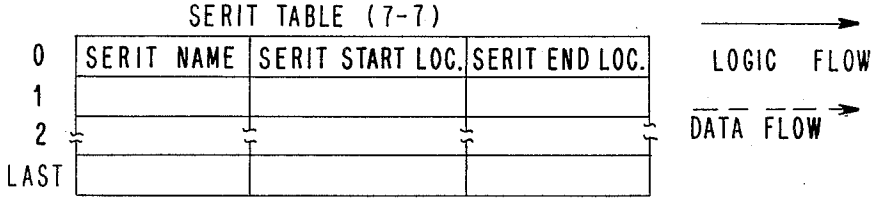


FIG. 7.1 (METHOD 1, SUBMETHOD I: FINDING BRANCHES & BRANCH TARGETS)

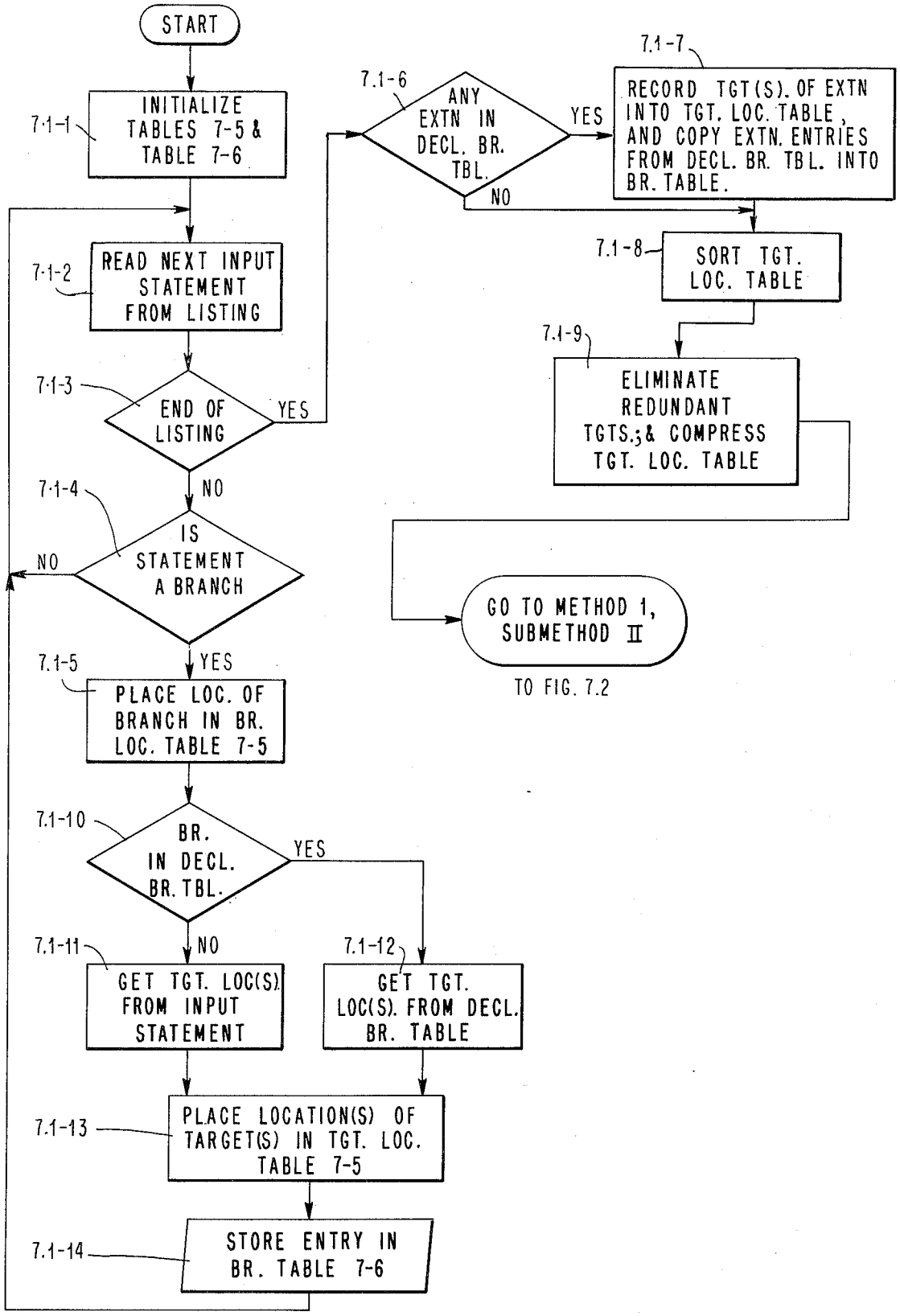


FIG. 7.2 (METHOD 1, SUBMETHOD II: GENERATING THE SERIT TABLE)

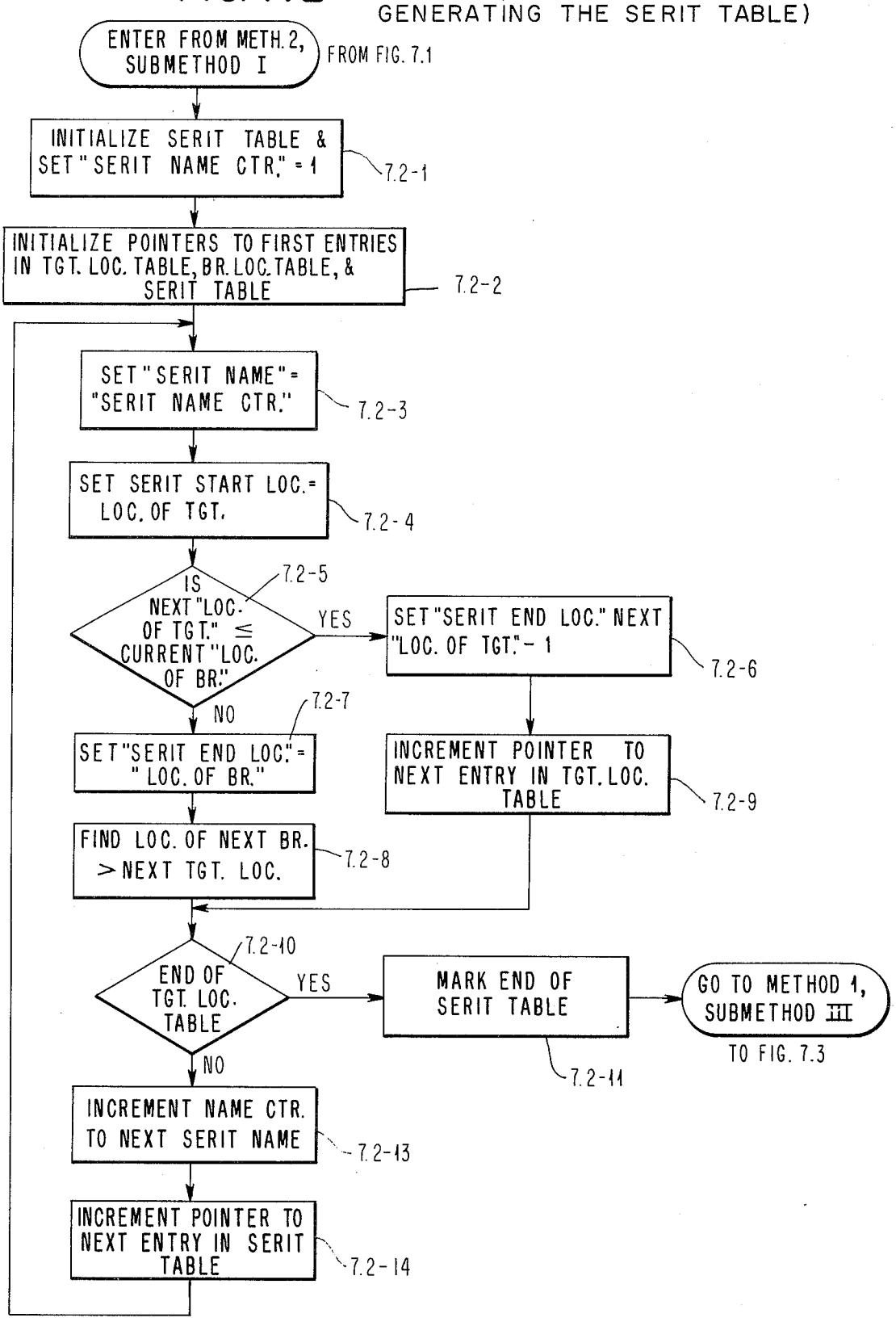


FIG. 7.3 (METHOD 1 SUBMETHOD III: GENERATING THE SERIT CONNECTION TABLE)

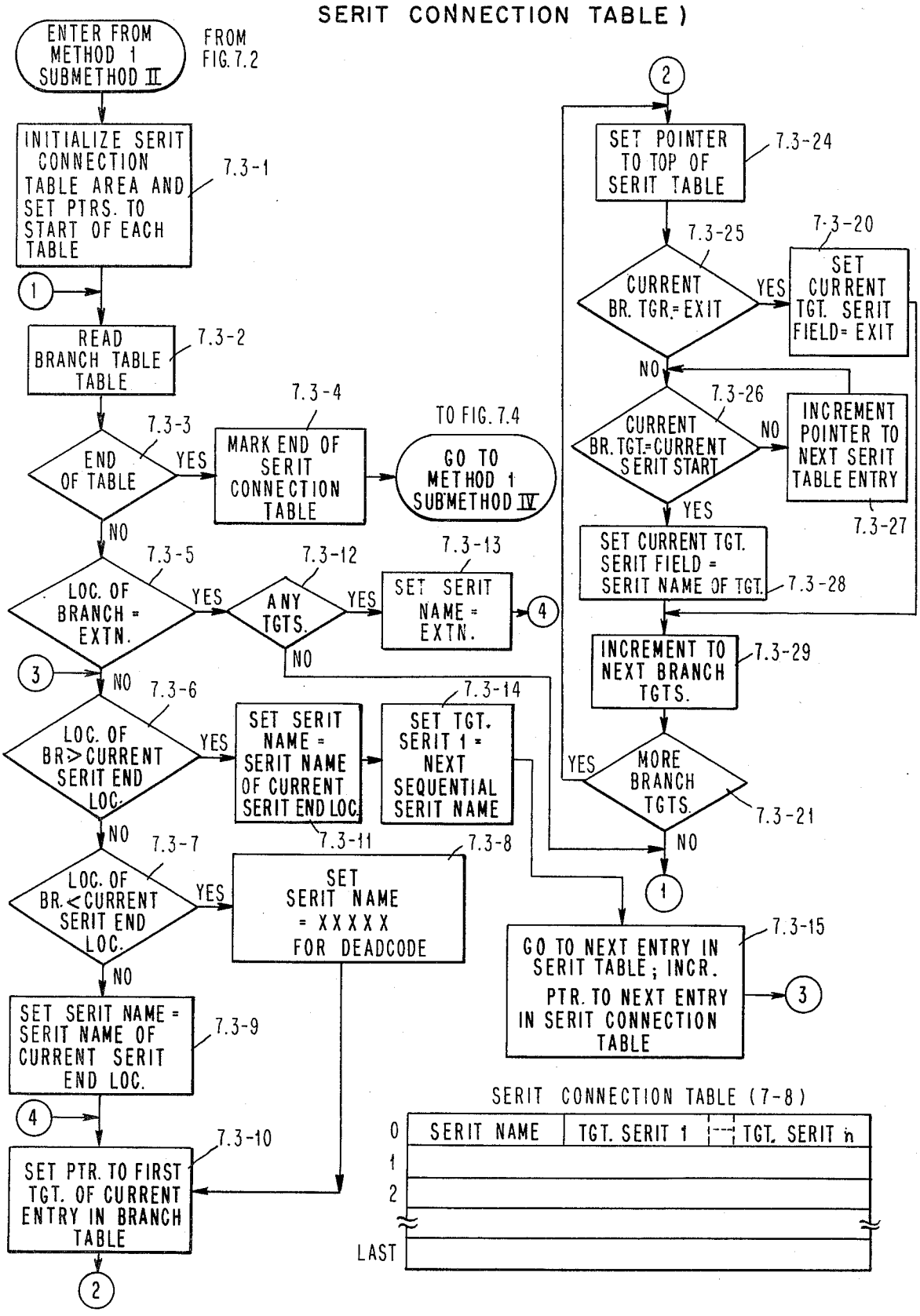


FIG. 7.4 (METHOD 1, SUBMETHOD IV: SERITING THE PROGRAM LISTING)

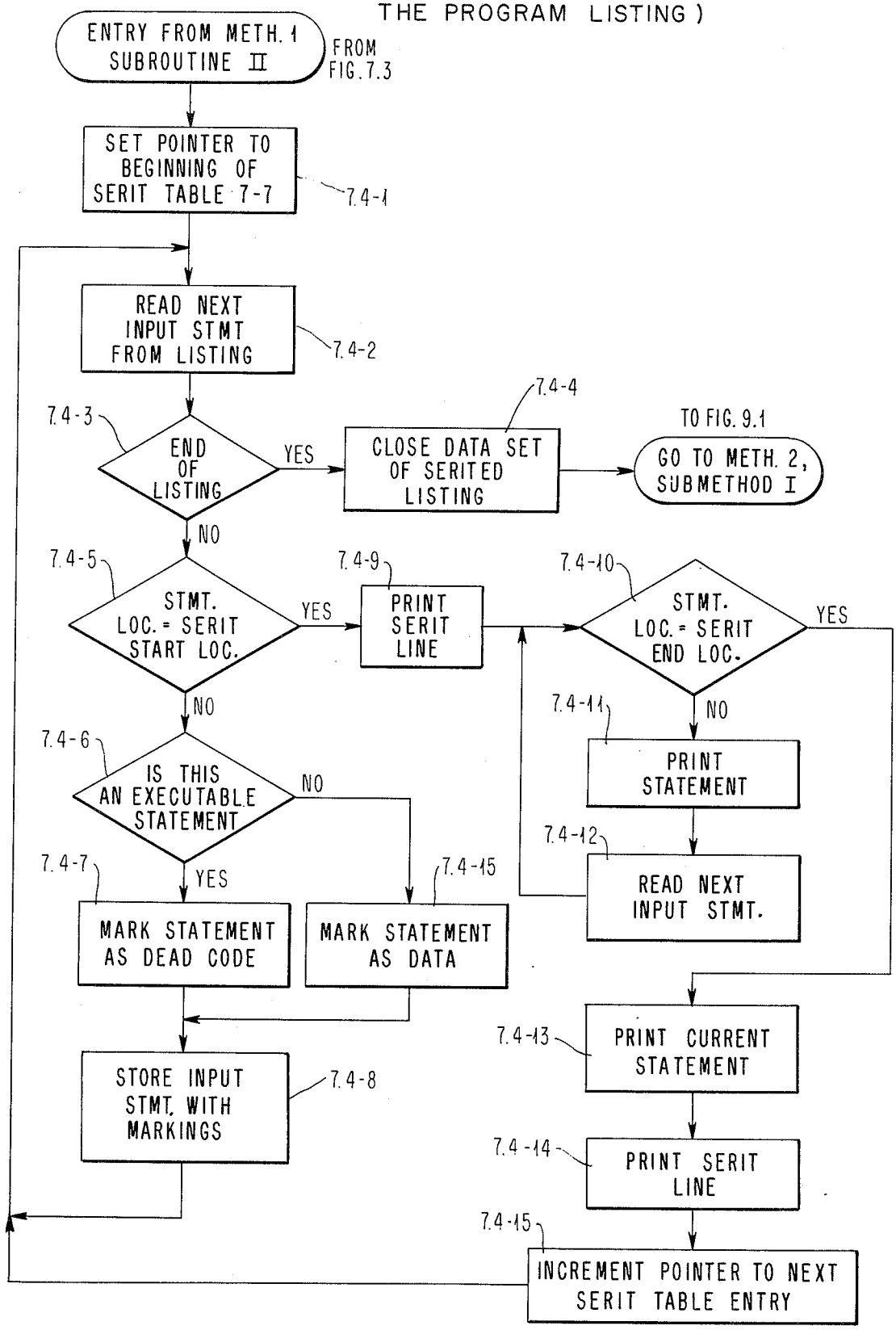


FIG. 8 (SERIT FLOWCHARTING)

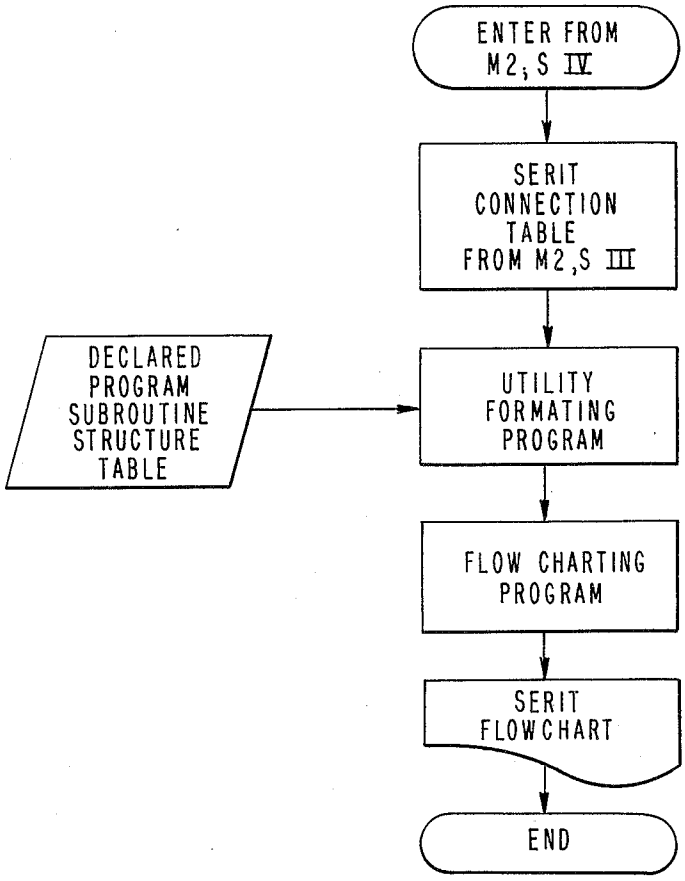


FIG. 9 (METHOD 2: SUMMARIZING OVERVIEW)

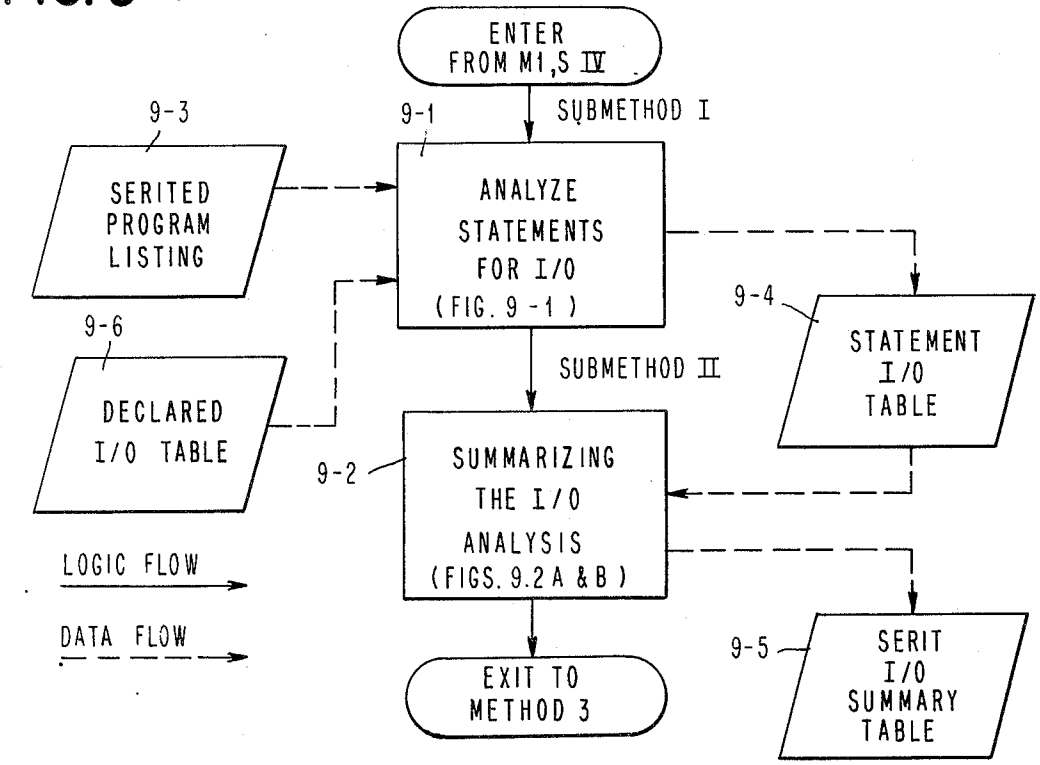


FIG. 9.1 (METHOD 2, SUBMETHOD I: ANALYZING THE STATEMENTS FOR I/O)

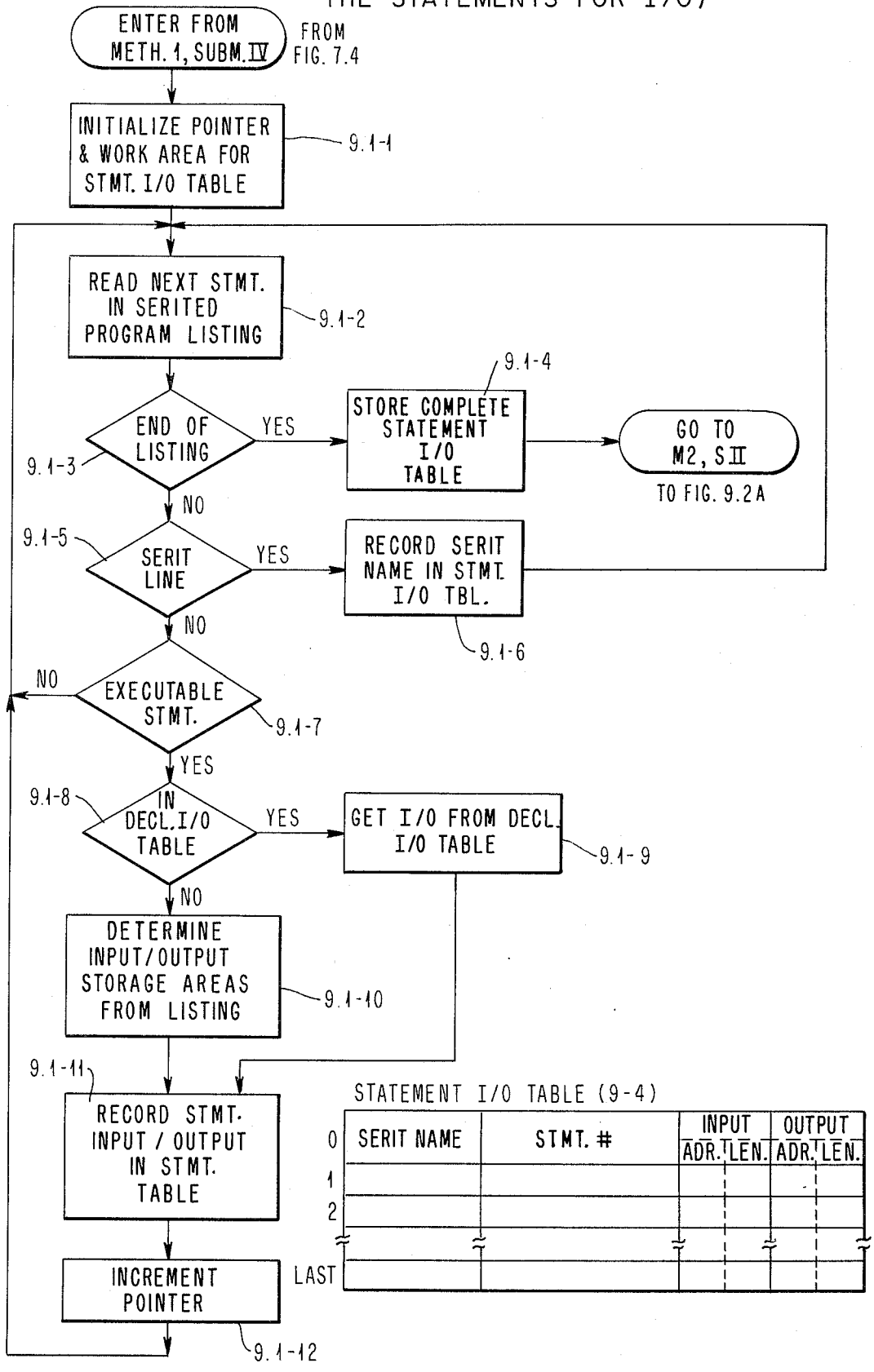


FIG. 9.2A

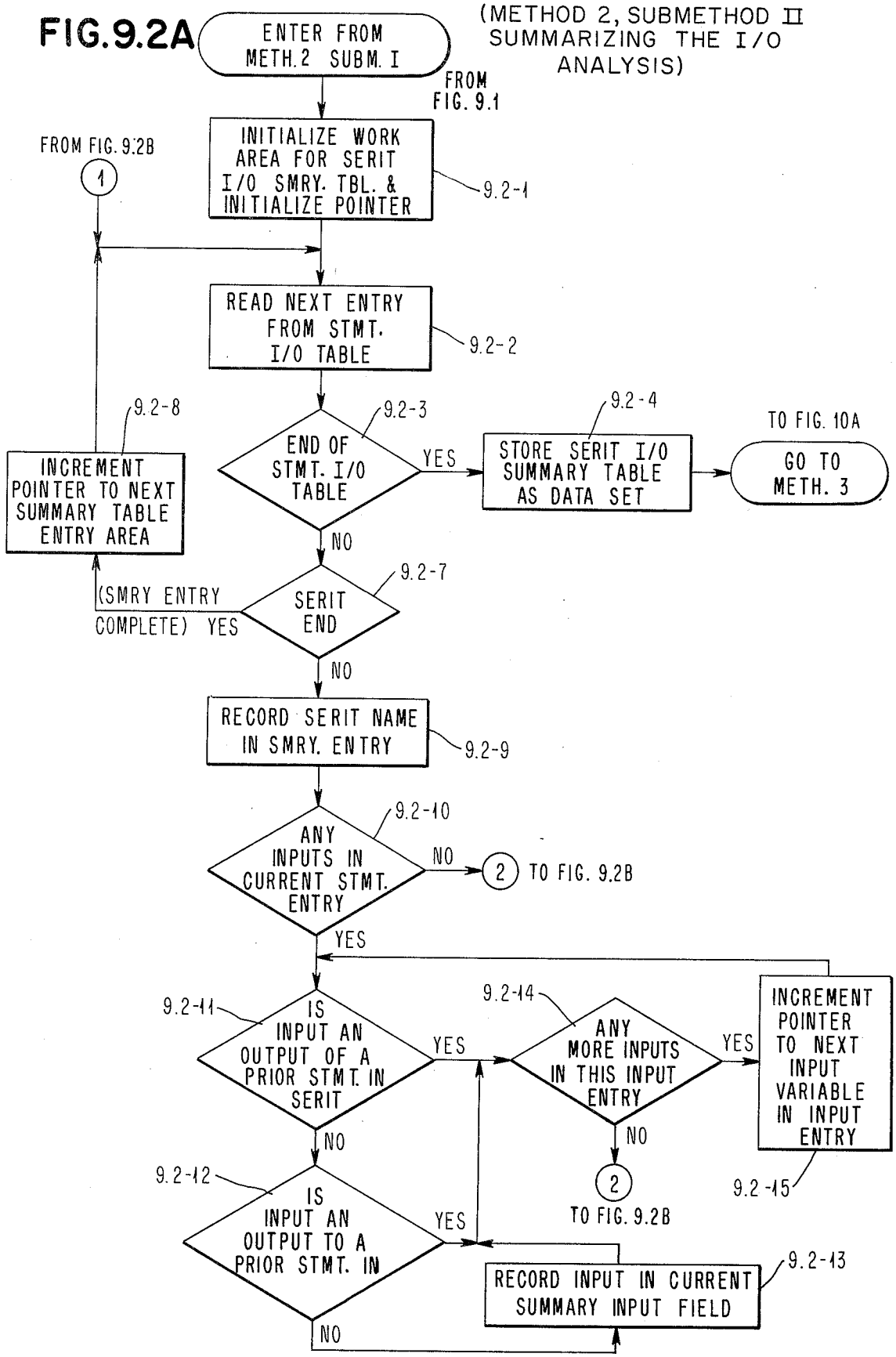
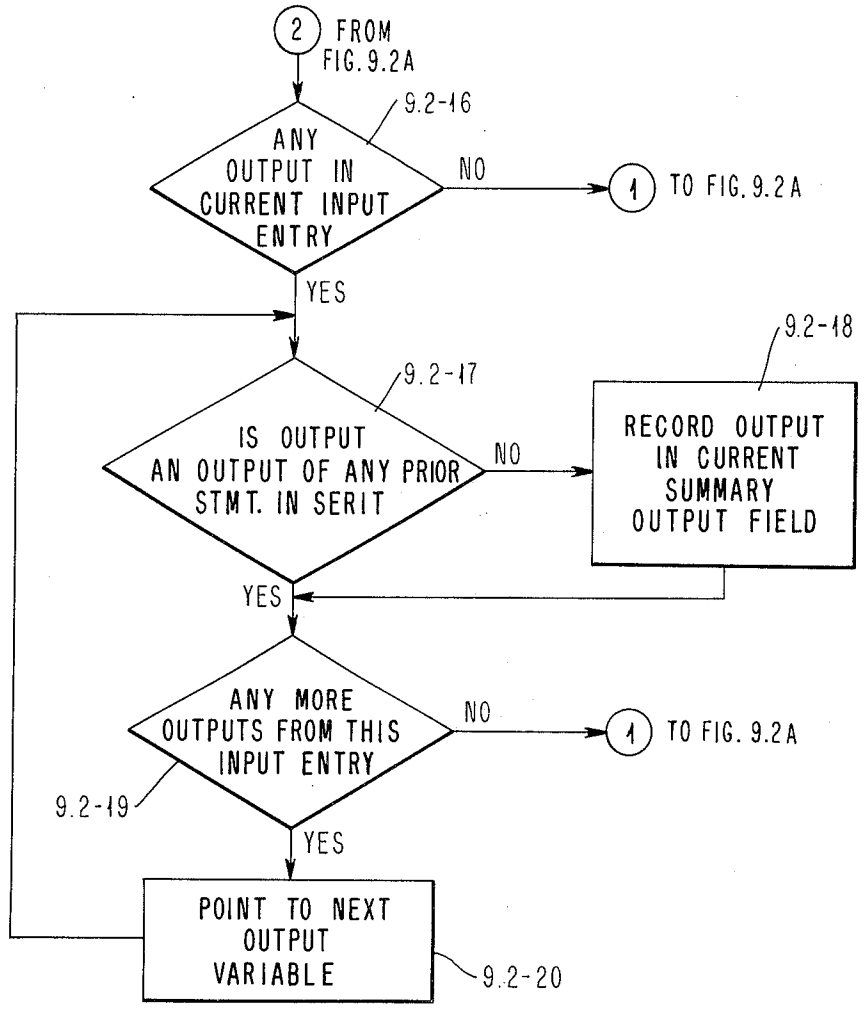

(METHOD 2, SUBMETHOD II SUMMARIZING THE I/O ANALYSIS)
ENTER FROM METH. 2 SUBM. I
FROM FIG. 9.1
FROM FIG. 9.2B
1
INITIALIZE WORK AREA FOR SERIT I/O SMRY. TBL. & INITIALIZE POINTER
9.2-1
READ NEXT ENTRY FROM STMT. I/O TABLE
9.2-2
9.2-3
END OF STMT. I/O TABLE
YES
9.2-4
STORE SERIT I/O SUMMARY TABLE AS DATA SET
TO FIG. 10A
GO TO METH. 3
NO
9.2-8
INCREMENT POINTER TO NEXT SUMMARY TABLE ENTRY AREA
9.2-7
SERIT END
(SMRY ENTRY COMPLETE) YES
NO
RECORD SERIT NAME IN SMRY. ENTRY
9.2-9
9.2-10
ANY INPUTS IN CURRENT STMT. ENTRY
NO
2
TO FIG. 9.2B
YES
9.2-11
IS INPUT AN OUTPUT OF A PRIOR STMT. IN SERIT
YES
9.2-14
ANY MORE INPUTS IN THIS INPUT ENTRY
YES
INCREMENT POINTER TO NEXT INPUT VARIABLE IN INPUT ENTRY
9.2-15
NO
2
TO FIG. 9.2B
NO
9.2-12
IS INPUT AN OUTPUT TO A PRIOR STMT. IN
YES
RECORD INPUT IN CURRENT SUMMARY INPUT FIELD
9.2-13
NO

FIG. 9.2B (M2, S II CONT.)



SERIT I/O SUMMARY TABLE (9-5)

	SERIT NAME	OUTPUT VARIABLE(S)	INPUT VARIABLE(S)
0			
1			
LAST			

FIG. 10A (METHOD 3: ANALYZING SERIT DEPENDENCY)

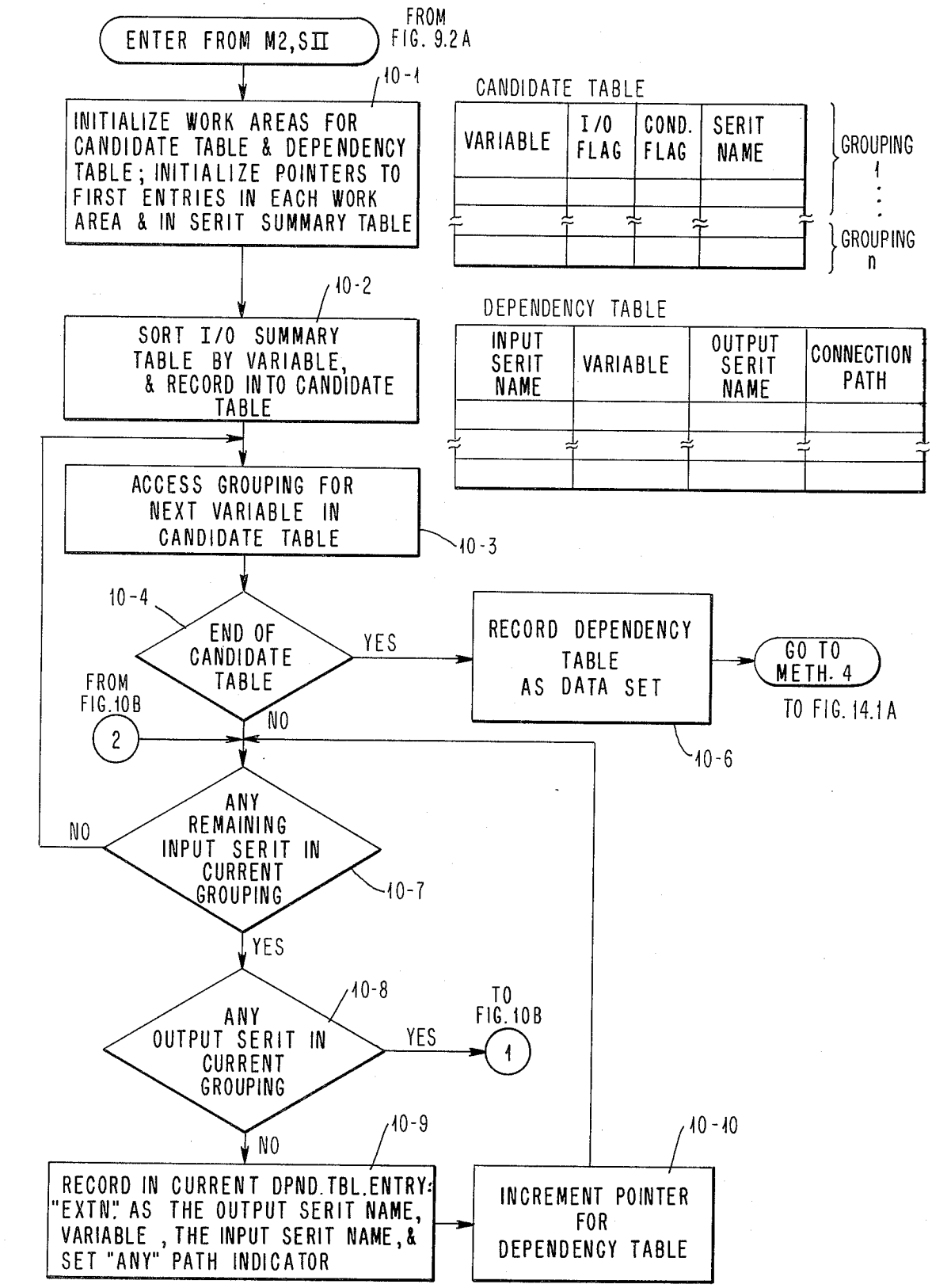


FIG. 10B (METHOD 3 CONT.)

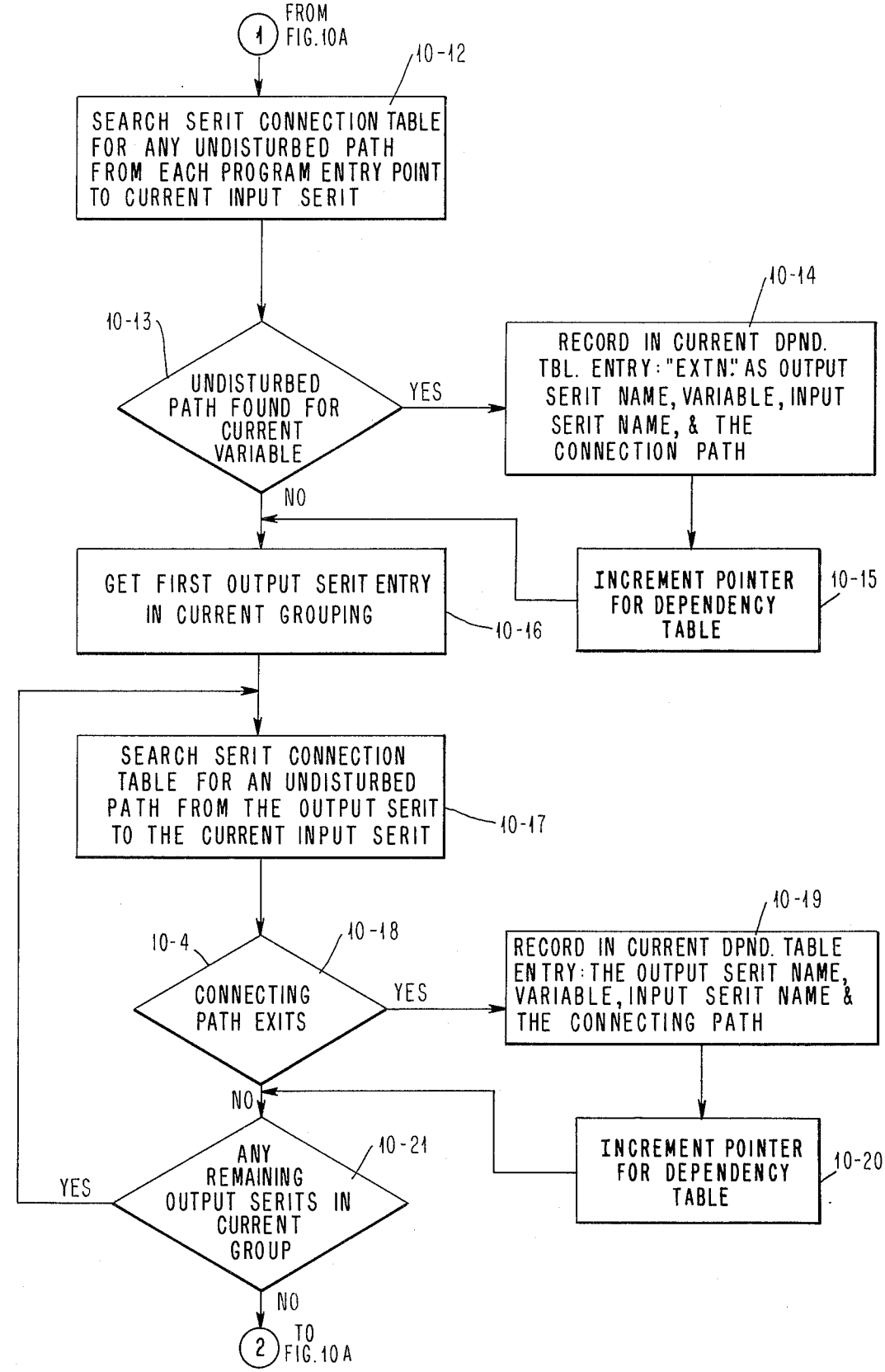


FIG. 12

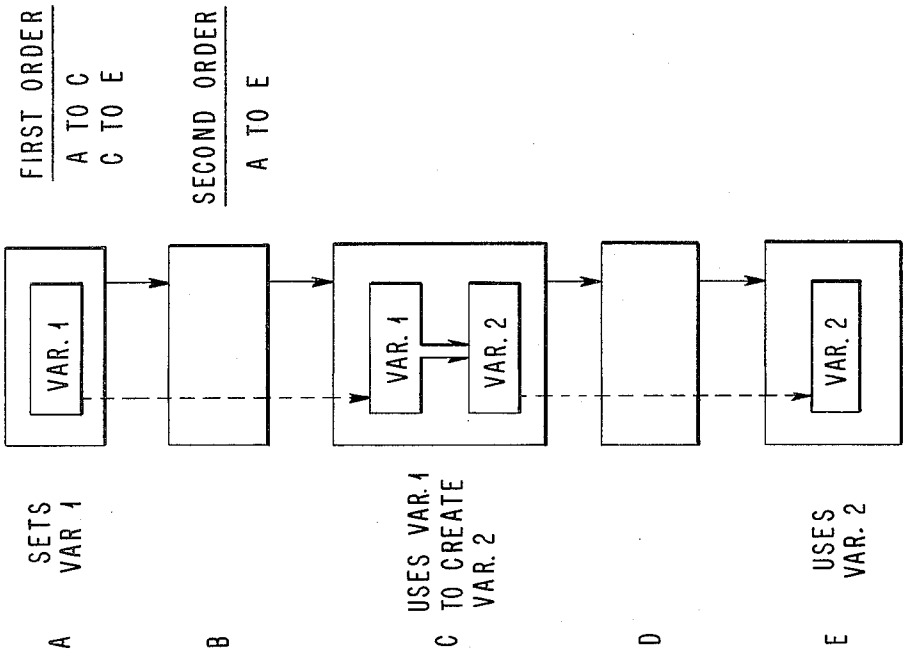


FIG. 11

SUBROUTINE SERIT FLOWCHART

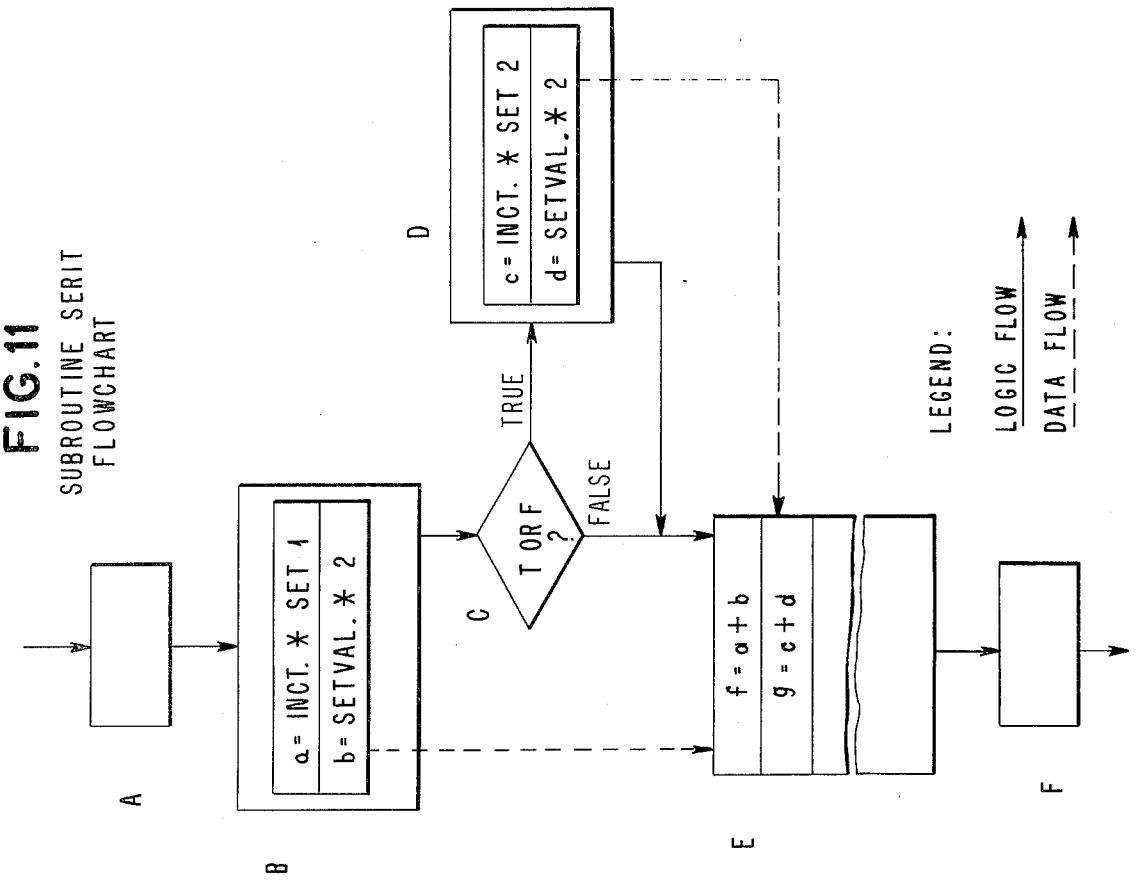
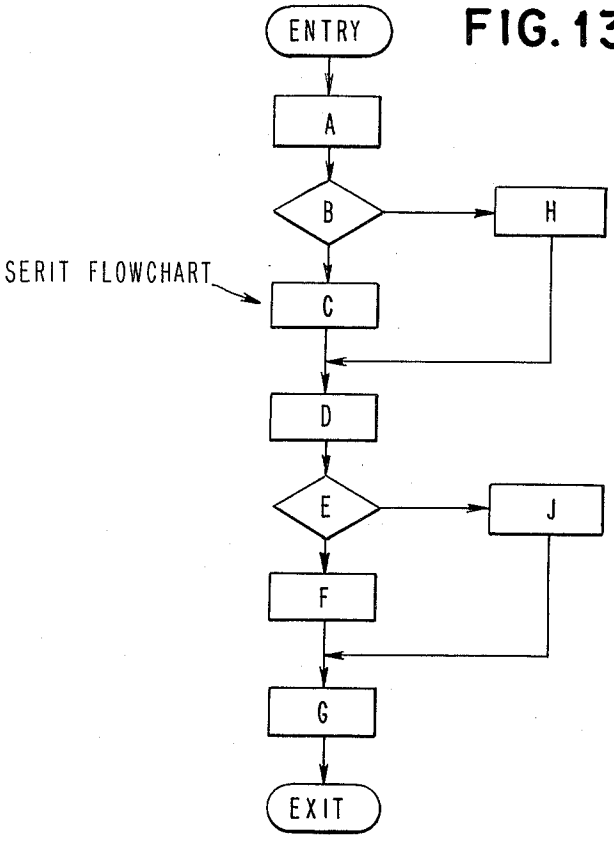


FIG. 13 (GENERATING TEST PATHS)



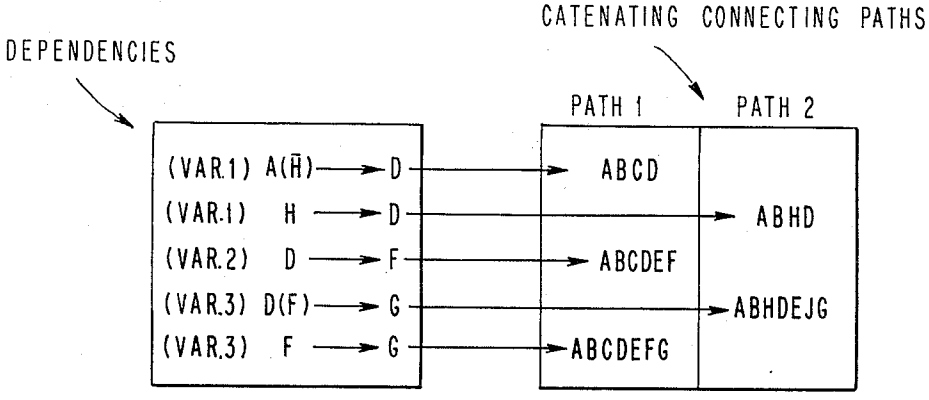


FIG.14 (METHOD 4: GENERATING DEPENDENCY TEST PATHS)

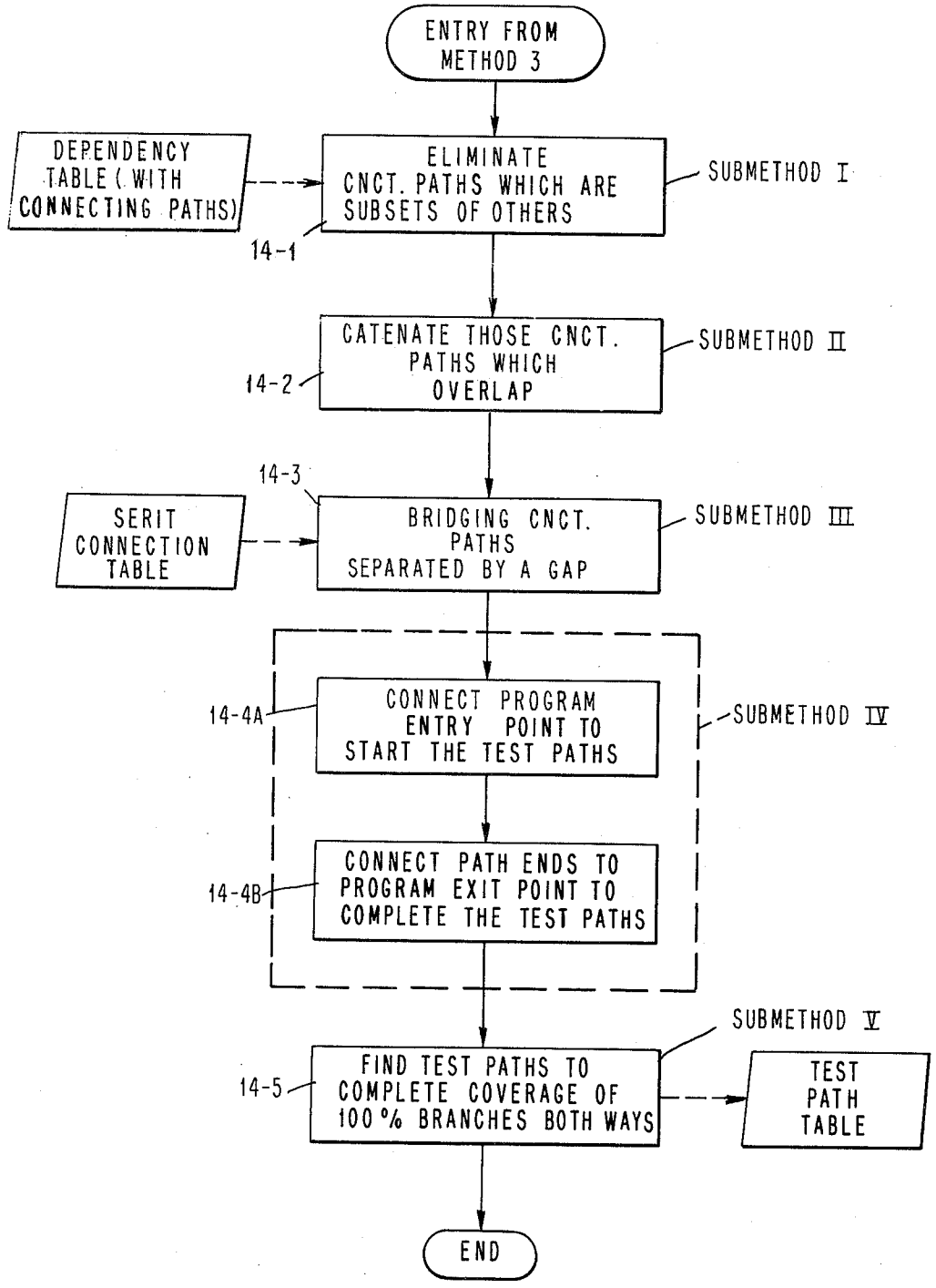


FIG. 14.1 (METHOD 4, SUBMETHOD I: ELIMINATING SUBSET CONNECTING PATHS)

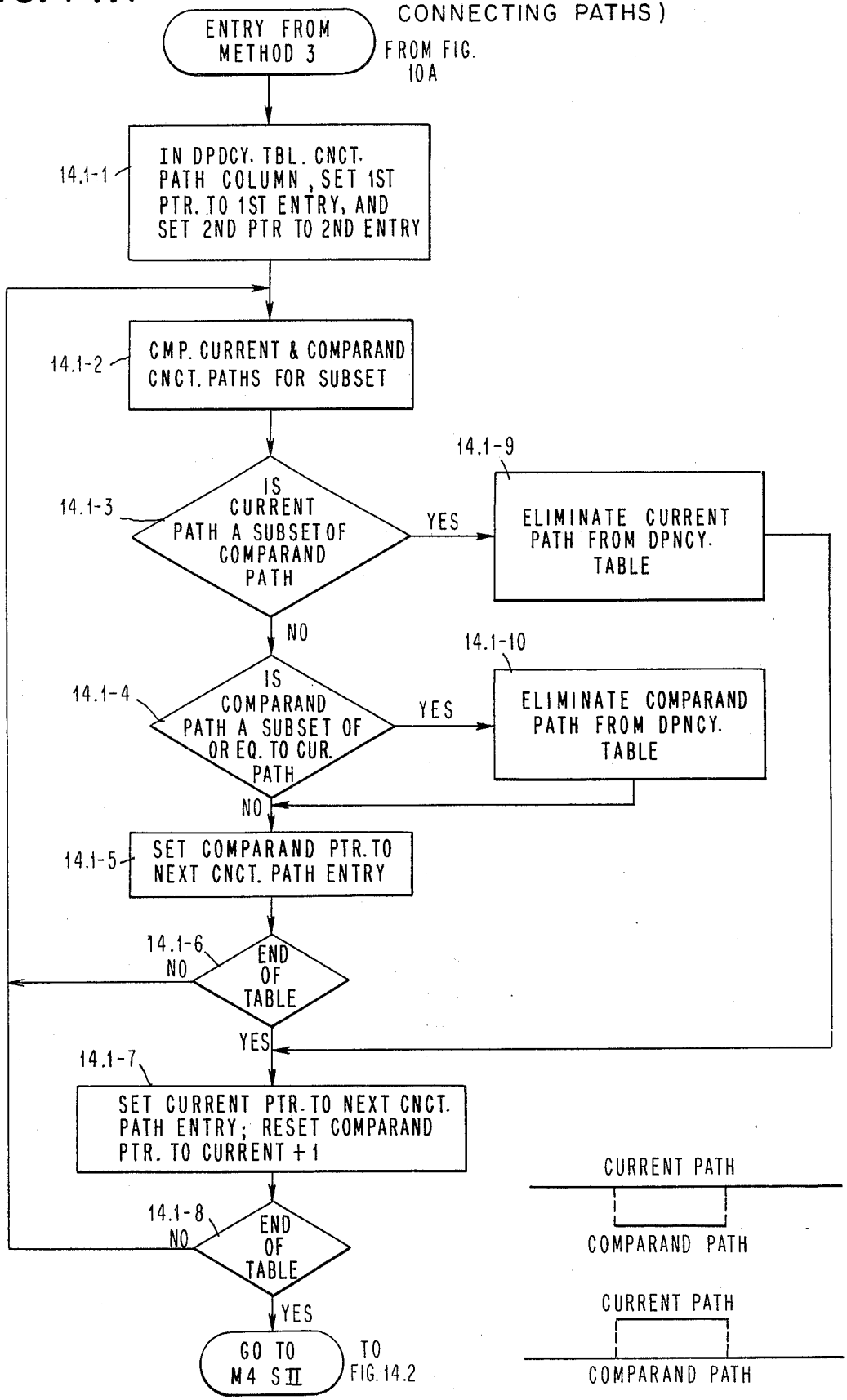


FIG. 14.2

(METHOD 4, SUBMETHOD II: CATENATING OVERLAPPING CONNECTING PATHS)

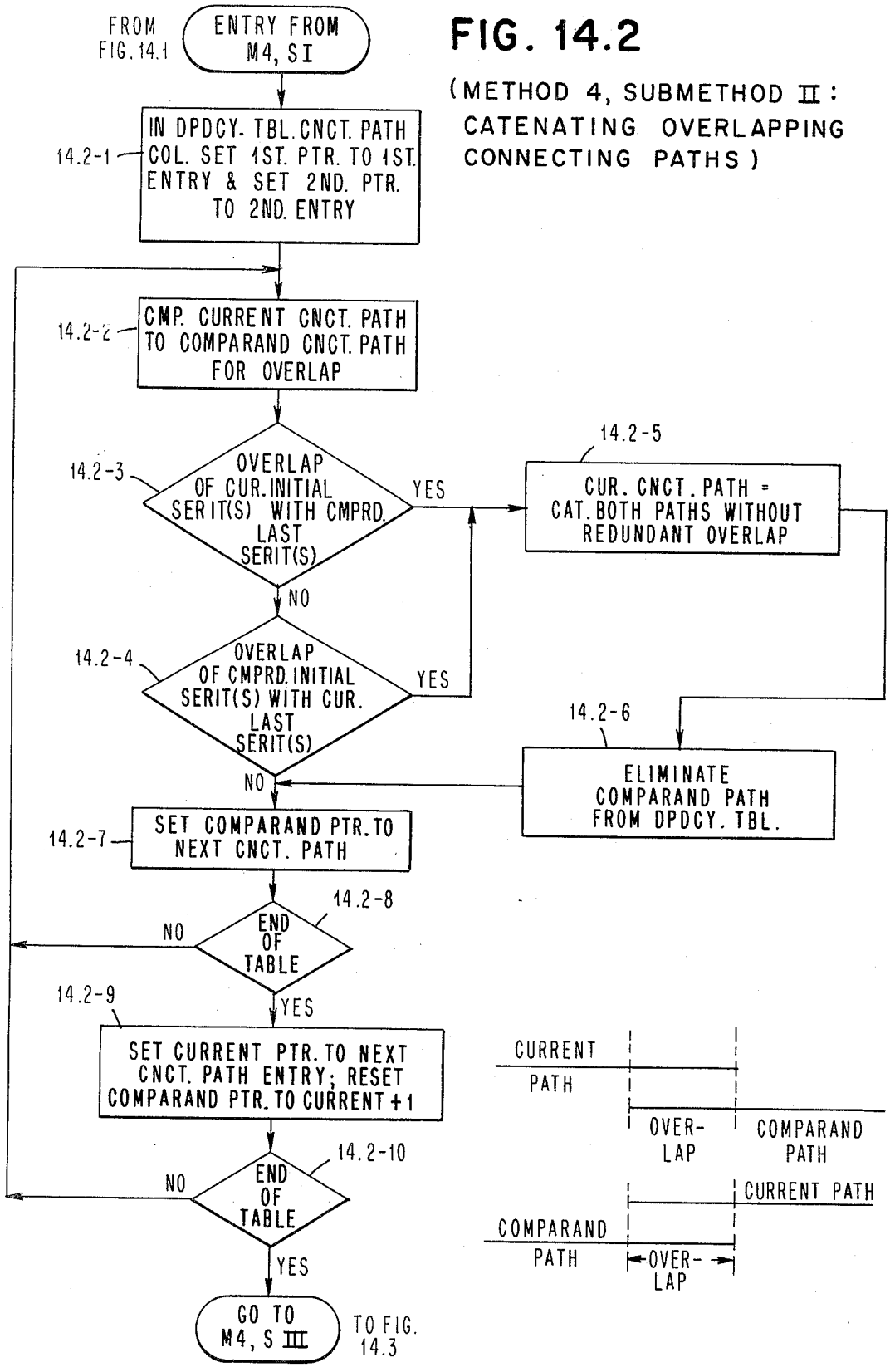


FIG. 14.3 (METHOD 4, SUBMETHOD III: BRIDGING THE CONNECTING PATHS SEPERATED BY A GAP)

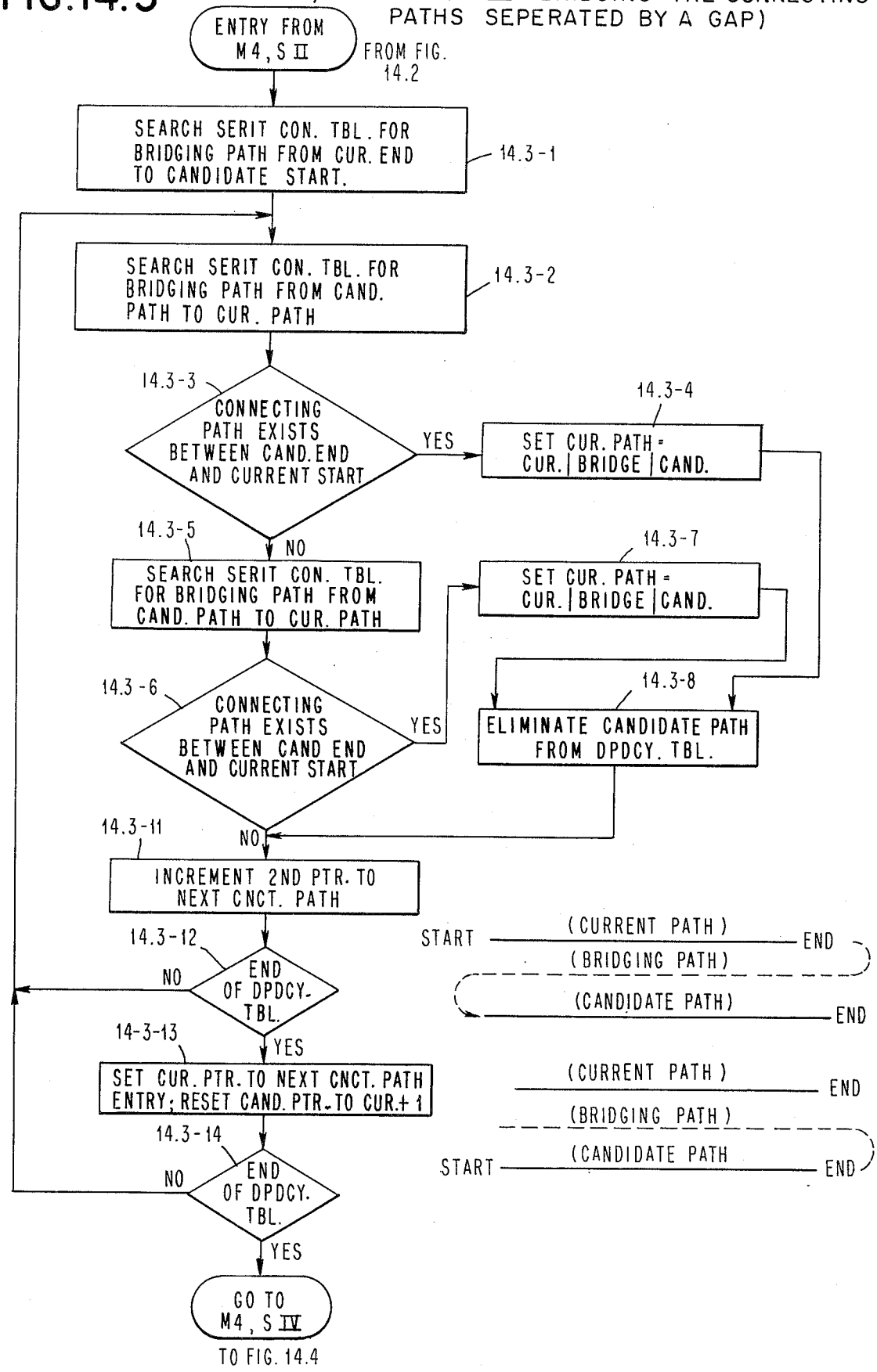


FIG.14.4 (METHOD 4, SUBMETHOD IV: GENERATING TEST PATHS FOR EXECUTING 100% OF THE DATA VARIABLES IN A PROGRAM)

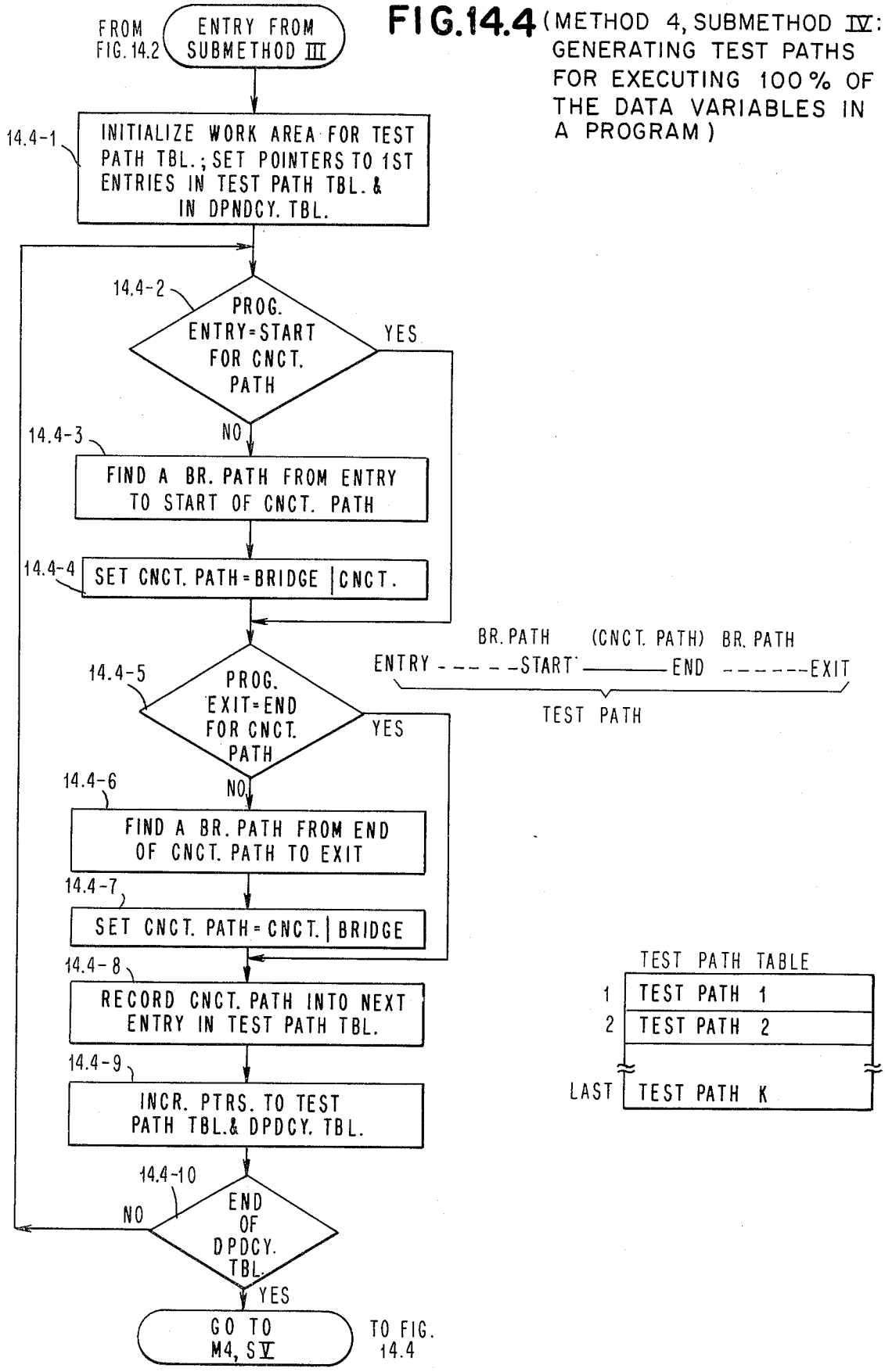


FIG. 14.5A

METHOD 4, SUB. Ⅴ: (COMPLETING THE TEST PATHS TO ALSO EXECUTE 100% OF THE BRANCHES IN A PROGRAM)

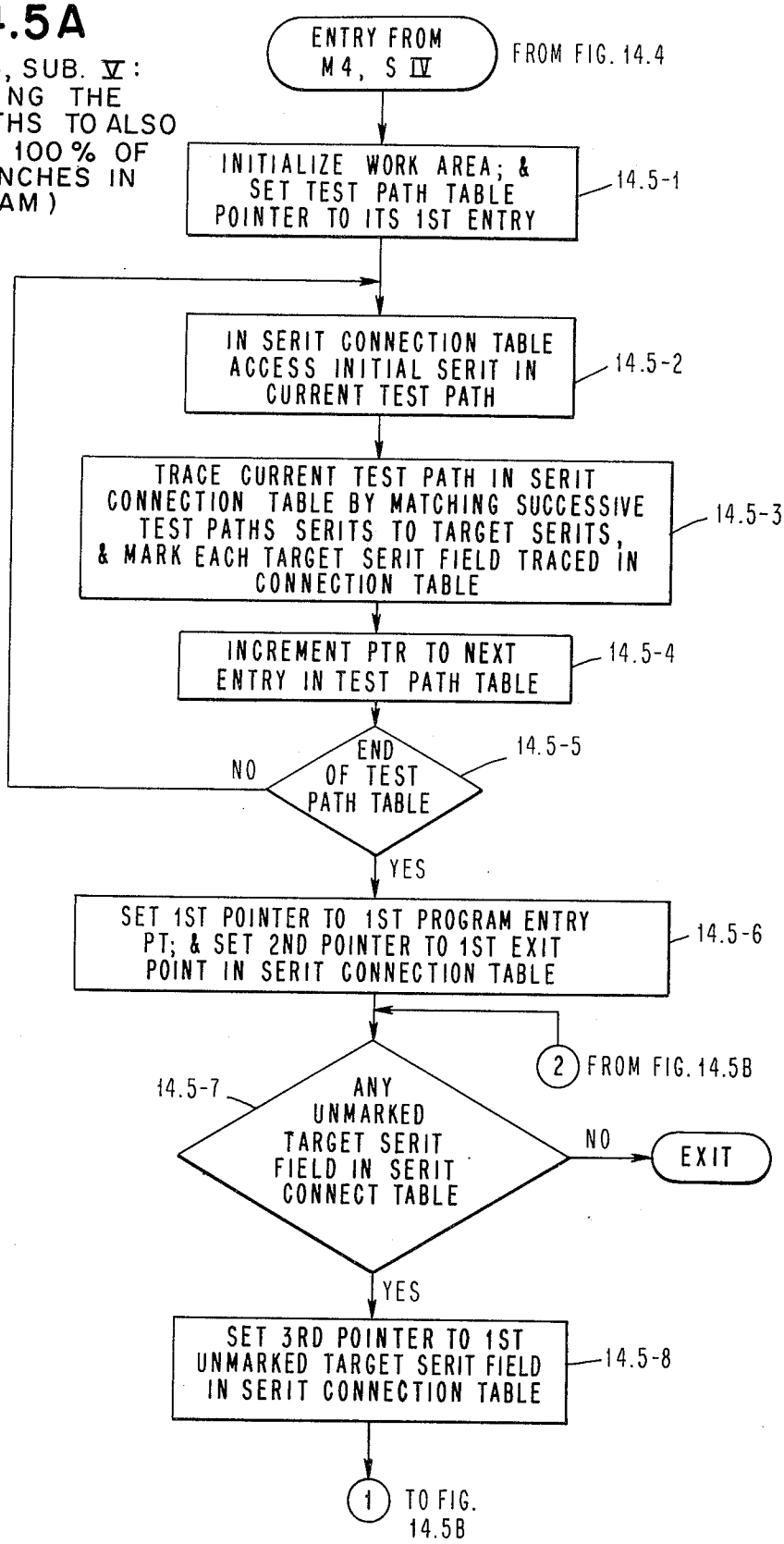
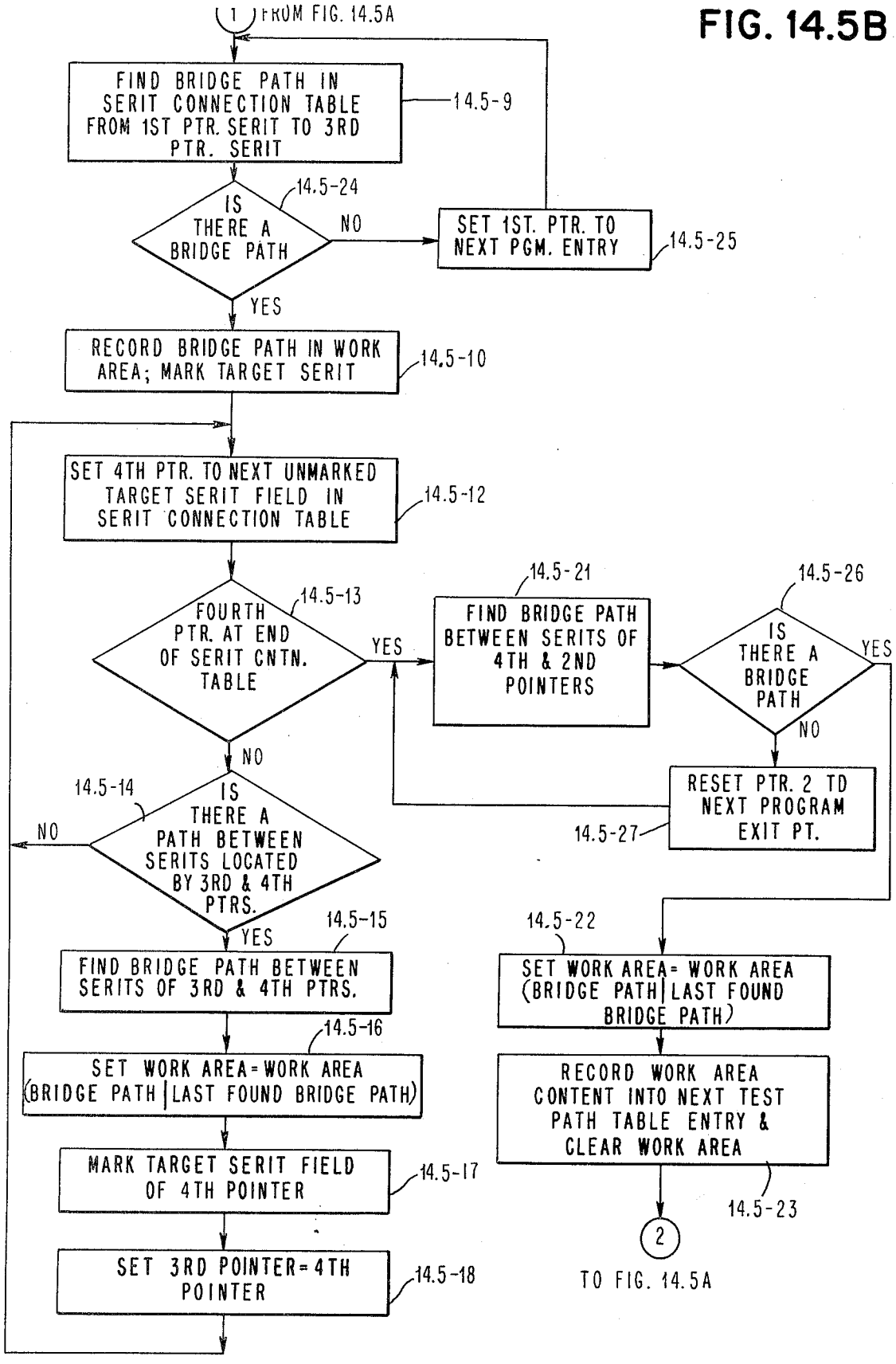


FIG. 14.5B


1 FROM FIG. 14.5A
FIND BRIDGE PATH IN SERIT CONNECTION TABLE FROM 1ST PTR. SERIT TO 3RD PTR. SERIT
14.5-9
IS THERE A BRIDGE PATH
14.5-24
NO
SET 1ST. PTR. TO NEXT PGM. ENTRY
14.5-25
YES
RECORD BRIDGE PATH IN WORK AREA; MARK TARGET SERIT
14.5-10
SET 4TH PTR. TO NEXT UNMARKED TARGET SERIT FIELD IN SERIT CONNECTION TABLE
14.5-12
FOURTH PTR. AT END OF SERIT CNTN. TABLE
14.5-13
YES
14.5-21
FIND BRIDGE PATH BETWEEN SERITS OF 4TH & 2ND POINTERS
14.5-26
IS THERE A BRIDGE PATH
YES
NO
RESET PTR. 2 TO NEXT PROGRAM EXIT PT.
14.5-27
NO
IS THERE A PATH BETWEEN SERITS LOCATED BY 3RD & 4TH PTRS.
14.5-14
NO
YES
FIND BRIDGE PATH BETWEEN SERITS OF 3RD & 4TH PTRS.
14.5-15
SET WORK AREA = WORK AREA (BRIDGE PATH | LAST FOUND BRIDGE PATH)
14.5-16
MARK TARGET SERIT FIELD OF 4TH POINTER
14.5-17
SET 3RD POINTER = 4TH POINTER
14.5-18
14.5-22
SET WORK AREA = WORK AREA (BRIDGE PATH | LAST FOUND BRIDGE PATH)
RECORD WORK AREA CONTENT INTO NEXT TEST PATH TABLE ENTRY & CLEAR WORK AREA
14.5-23
2
TO FIG. 14.5A