THE UNIVERSITY OF MICHIGAN

SYSTEMS ENGINEERING LABORATORY

Department of Electrical Engineering
College of Engineering

SEL Technical Report No. 11

COMPUTER PROGRAMS DEALING WITH FINITE-STATE MACHINES: PART I

by

Thomas F. Piatkowski

May 1967

This research was supported by United States Air Force Contract AF 30(602)-3546.

Acknowledgements

. . . to the Ford Motor Company for making their time-sharing computer facilities available for the development of these programs.

Table of Contents

I. Introduction

During the past several months I have been able to program a number of algorithms dealing with finite-state machines; the purposes of this effort being: (1) pedagogical application, (2) the stimulation of insight, and (3) the generation of new questions and approaches. Out of a set of thirty-eight more or less different problems dealing with finite state machines, I selected seven representatives; namely:

1) simulate a given machine,
2) determine if a given machine is strongly connected,
3) determine the state equivalence classes and minimal form for a given machine,
4) determine the automorphisms and their group for a given machine,
5) determine a shortest simple adaptive diagnosing experiment for a given machine and admissible set,
6) determine an equivalent regular expression for a given machine,
7) exhibit the lattice of SP partitions for a given machine.

This report, the first of two, describes the programs treating problems one through four, a second report treating problems

five through seven will follow.

Concurrent with my own programming efforts, I attempted to document existing programs in this area; the results of my search are included in the bibliography.

The four programs contained in this report are written in the BASIC language for use in a time-sharing environment. A conscious effort has been exerted to make them straightforward and easy to use.

II. Conventions

We will deal with N, P, Q Mealy* machines where

N = the number of states in state set S, $1 \leq N \leq 100$

P = the number of symbols in input alphabet A_I, $1 \leq P \leq 5$, except for the machine minimization program where $1 \leq P \leq 3$.

Q = the number of symbols in output alphabet A_O.

We will only consider machines coded such that

$S = \{1, 2, \ldots, N\}$

$A_I = \{1, 2, \ldots, P\}$

A_O = any set of Q distinct integers.

A machine is a system $M = \langle S, A_I, A_O, FS, FZ \rangle$ where S, A_I, A_O are as above and where FS, the next state function, maps $S \times A_I \rightarrow S$ and FZ, the output function, maps $S \times A_I \rightarrow A_O$.

It is assumed that the user is familiar with the necessary theoretical concepts of finite state machine theory [See 1 and 2] and with the BASIC language and its operation.

The programs are written to run under BASIC, Ford Motor Company Time Sharing System, Dearborn, Michigan, as it existed April 1967.

---

*Moore machines can be handled by the artifice of defining an equivalent Mealy machine in which FZ is independent of the input.

III. Machine Simulation

A. Program Name: SIM

B. Purpose: To simulate the input/output behavior of any specified machine with initial state. An option is available to suppress or not suppress the typing of current state information (in order to demonstrate state identification experiments, for example, one may wish to suppress the current state information).

C. Method: Given current state S1 and input I, the output is FZ(S1, I) and the new current state is FS(S1, I).

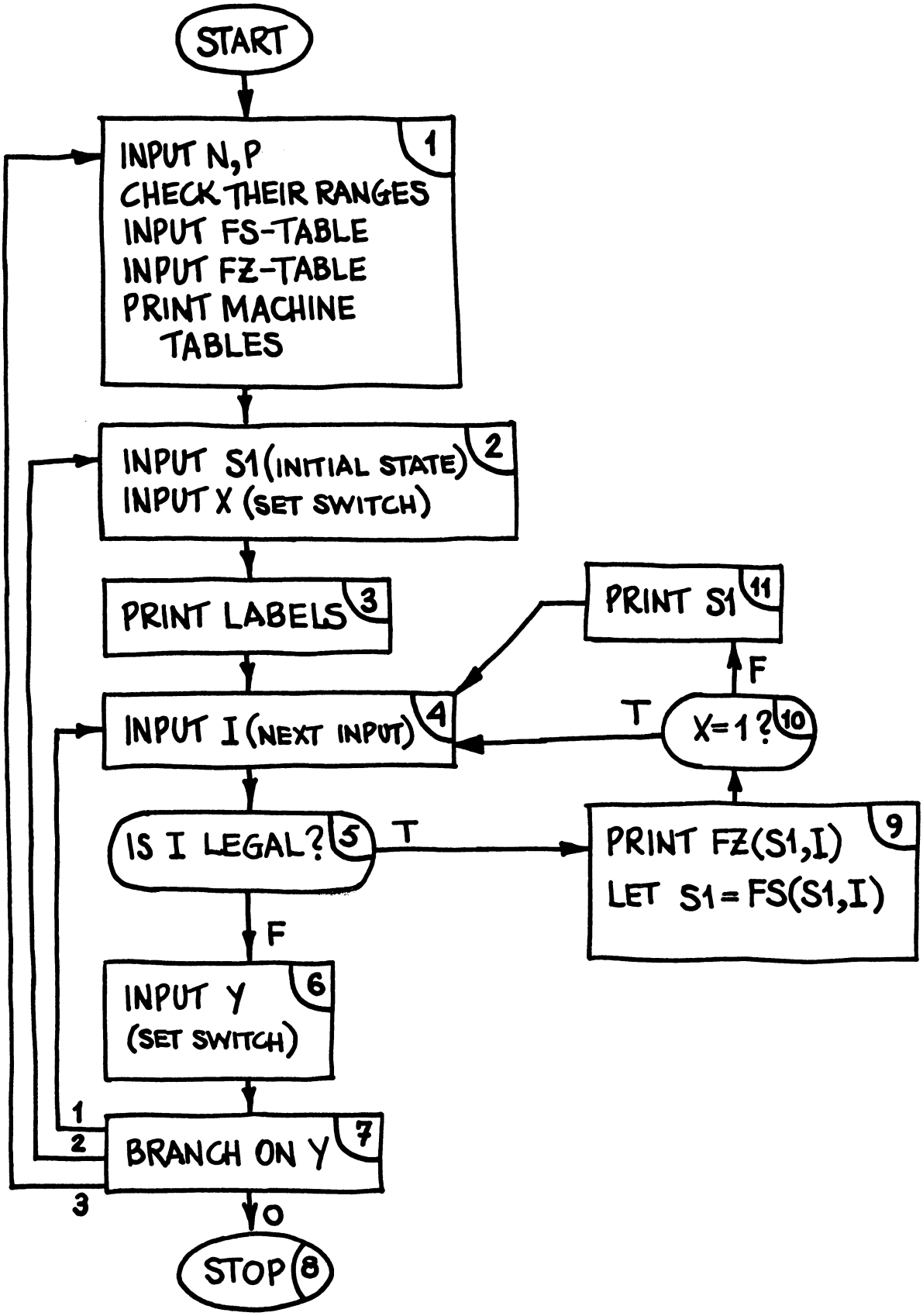
D. Operation: All user input is supplied at program request in the following sequence:

Program Types	User Types
1) "N, P?"	Two integers separated by a comma and followed by a RETURN and corresponding to N and P for the machine to be simulated; $1 \leq N \leq 100$; $1 \leq P \leq 5$. If N or P is out of range, the program will ask for N, P again.

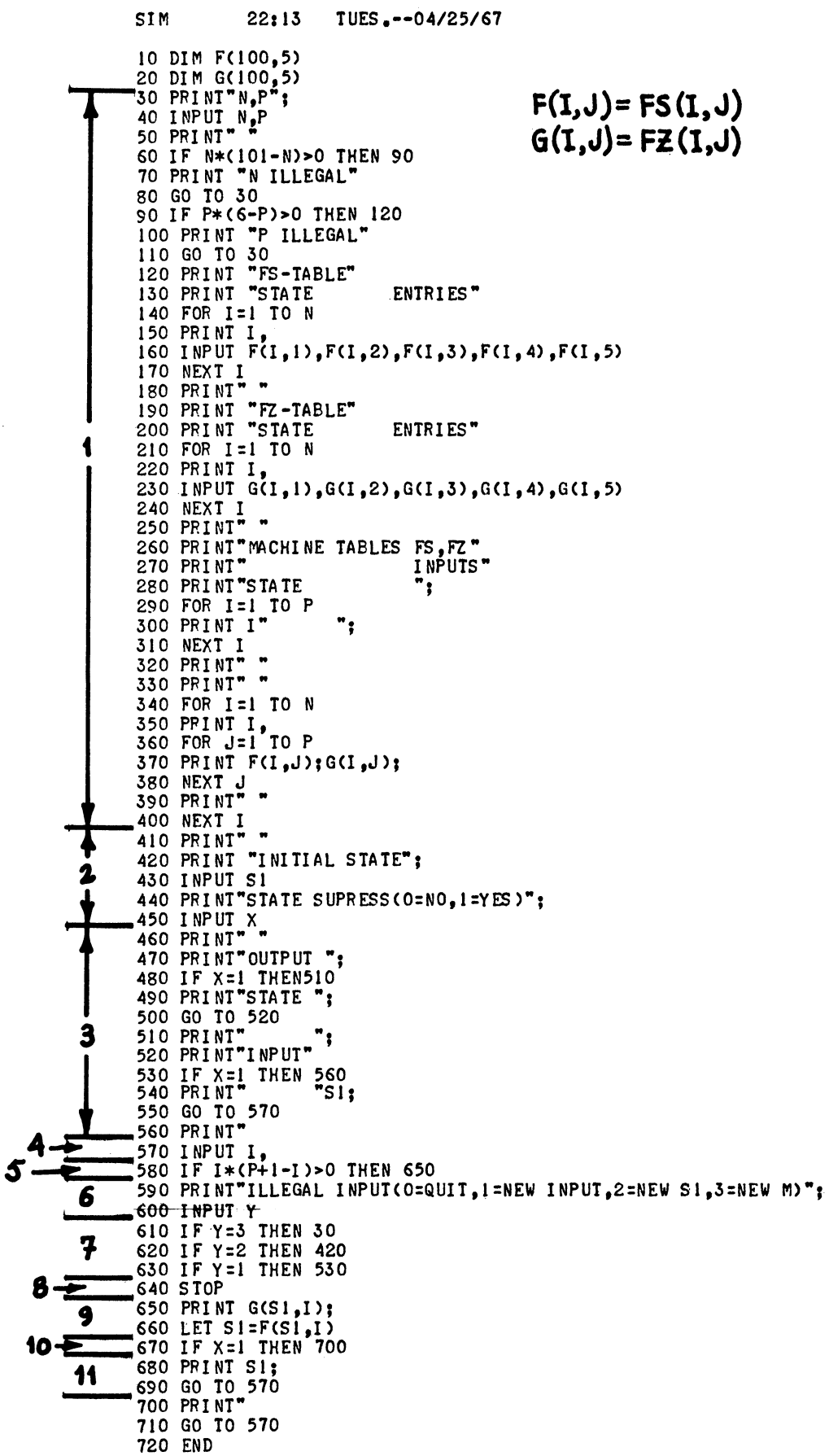
2) After typing two lines of labels for the FS-table , the program will type N lines of the following form: "I?"	After each "I?" type five integers separated by commas and followed by RETURN. The first P integers correspond to FS(I, 1), FS(I, 2), . . . , FS(I, P); the remaining integers are ignored but must be typed. They may be given any values and are needed only to satisfy the inflexible format requirements of BASIC.
3) After typing two labels for the FZ-table, the program will type N lines of the following form: "I?"	After each "I?" type five integers separated by commas and followed by a RETURN. The first P integers correspond to FZ(I, 1), FZ(I, 2), . . . , FZ(I, P); the remaining integers are ignored (See 2 above).
4) Program types full machine table in N lines of form: I, FS(I, 1), FZ(I, 1), FS(I, 2), . . . , FZ(I, N)	
5) "INITIAL STATE?"	One integer, in range 1 to N, followed by RETURN; this specifies the initial state of

	the machine.
6) "STATE SUPPRESS?"	Zero or one followed by RETURN; indicates if current state should be typed by program or not.
7) After typing appropriate labels, the program is ready for a simulation run.	The user has two options: a) type integer in range 1 to P followed by RETURN, in which event the integer is interpreted as a legitimate machine input—the program responds by typing the output and next state (if not suppressed); the user can then type the next input and so on. b) type integer out of range 1 to P followed by RETURN in which event the program notes the illegal input and requests further instruction via a second integer and RETURN. With this second integer the user can indicate that he quits, wants to input to the current machine, wants to reset the state of current machine, or start defining a new machine.

E. Flow Chart


START
INPUT N,P
CHECK THEIR RANGES
INPUT FS-TABLE
INPUT FZ-TABLE
PRINT MACHINE TABLES
1
INPUT S1 (INITIAL STATE)
INPUT X (SET SWITCH)
2
PRINT LABELS
3
INPUT I (NEXT INPUT)
4
IS I LEGAL?
5
T
F
INPUT Y (SET SWITCH)
6
BRANCH ON Y
7
1
2
3
0
STOP
8
PRINT FZ(S1,I)
LET S1 = FS(S1,I)
9
X=1?
10
T
F
PRINT S1
11

F. Annotated Program Listing



G. Sample Run

```
SIM        22:07   TUES.--04/25/67

N,P?4,2

FS-TABLE
STATE        ENTRIES
 1              ?3,4,0,0,0
 2              ?1,2,0,0,0
 3              ?4,4,0,0,0
 4              ?2,3,0,0,0

FZ-TABLE
STATE        ENTRIES
 1              ?0,1,0,0,0
 2              ?1,1,0,0,0
 3              ?0,0,0,0,0
 4              ?1,0,0,0,0

MACHINE TABLES FS,FZ
               INPUTS
STATE           1          2

 1              3     0    4     1
 2              1     1    2     1
 3              4     0    4     0
 4              2     1    3     0

INITIAL STATE?1
STATE SUPRESS[0=NO,1=YES]?0

OUTPUT STATE INPUT
       1    ?1
 0     3    ?2
 0     4    ?1
 1     2    ?2
 1     2    ?1
 1     1    ?5
ILLEGAL INPUT[0=QUIT,1=NEW INPUT,2=NEW S1,3=NEW M]?1
       1    ?2
 1     4    ?-2
ILLEGAL INPUT[0=QUIT,1=NEW INPUT,2=NEW S1,3=NEW M]?2
INITIAL STATE?1
STATE SUPRESS[0=NO,1=YES]?1

OUTPUT       INPUT
             ?1
 0           ?2
 0           ?1
 1           ?2
 1           ?1
 1           ?2
 1           ?6
ILLEGAL INPUT[0=QUIT,1=NEW INPUT,2=NEW S1,3=NEW M]?3
N,P?5,6

P ILLEGAL
N,P?103,4

N ILLEGAL
N,P?STOP

RAN  35 SEC.
```

IV. Checking a Machine for Strong Connectedness

A. Program Name: STR

B. Purpose: To check if a specified machine is strongly connected.

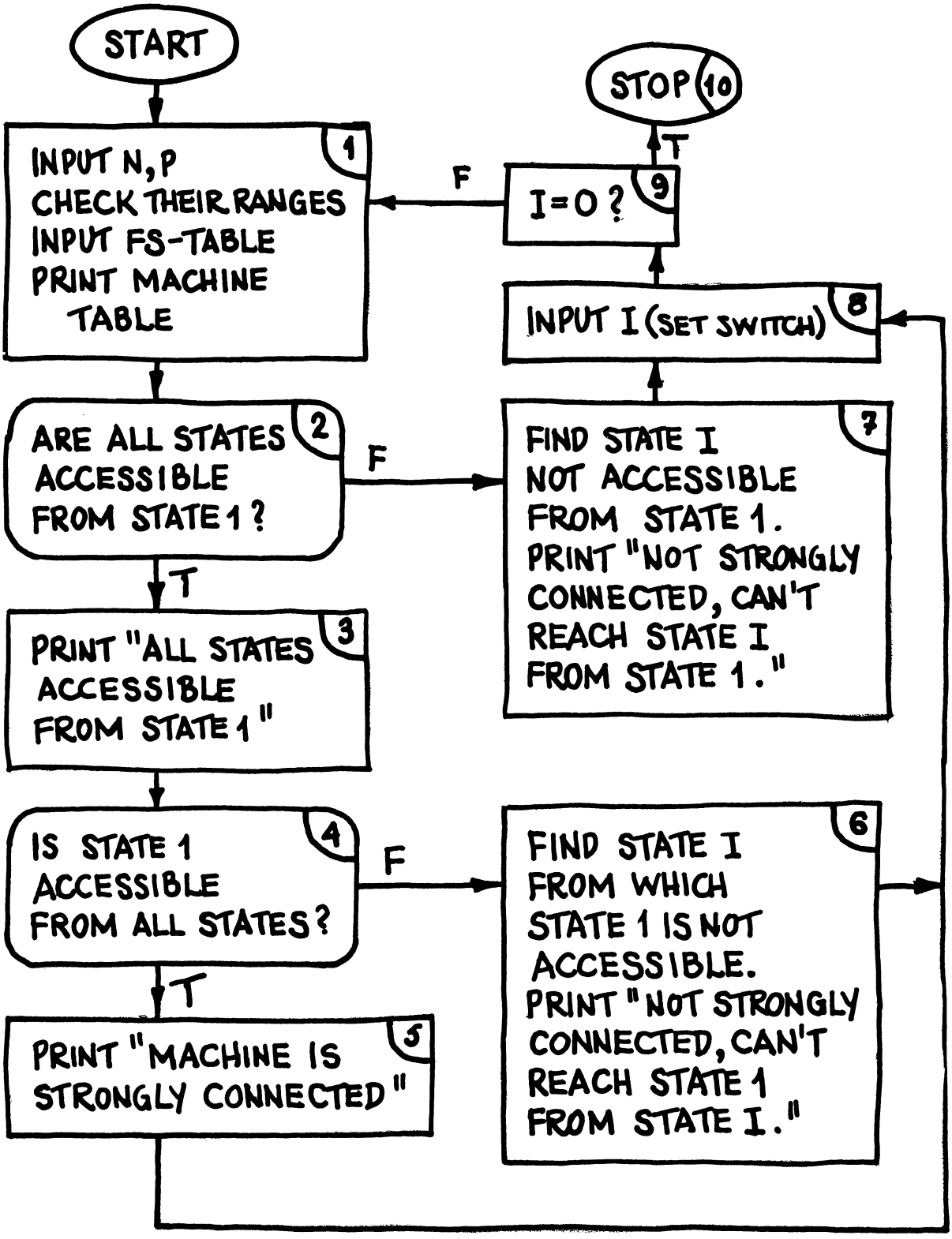
C. Method: Find G(1) the set of all states reachable from state 1 and H(1) the set of all states reaching state 1. M is strongly connected iff G(1) = H(1) = S.

D. Operation: All user input is supplied at program request in the following sequence:

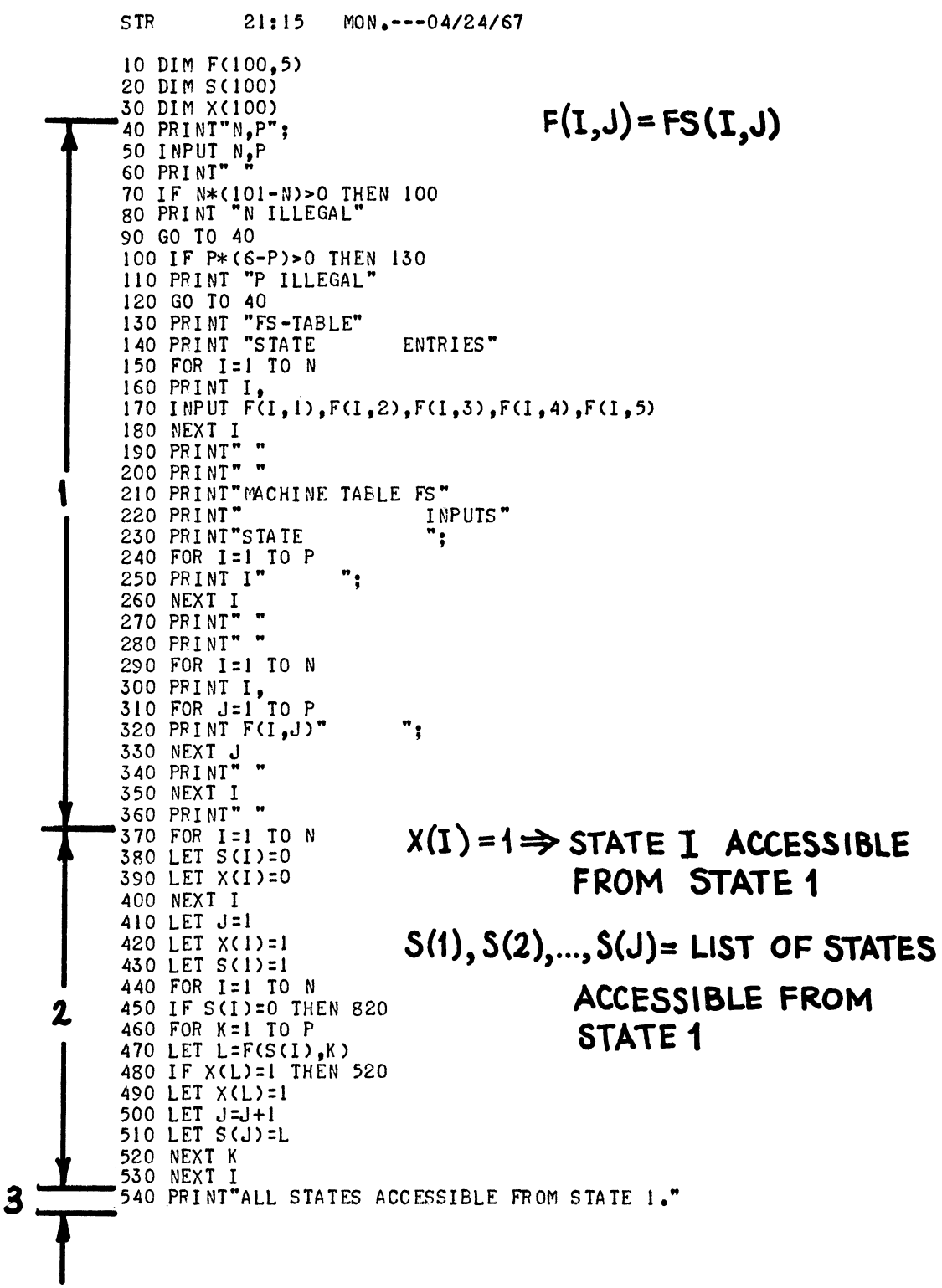
Program Types	User Types
1) "N, P?"	Two integers separated by a comma and followed by a RETURN and corresponding to N and P for the machine to be checked; $1 \leq N \leq 100$; $1 \leq P \leq 5$. If N or P is out of range the program will ask for N, P again.
2) After typing two labels for the FS-table, the program will type N lines of the following form: "I?"	After each "I?" type five integers separated by commas and followed by a RETURN. The first P integers correspond to FS(I, 1), FS(I, 2), . . . , FS(I, P); the remaining integers are ignored but must be typed. They may be given any

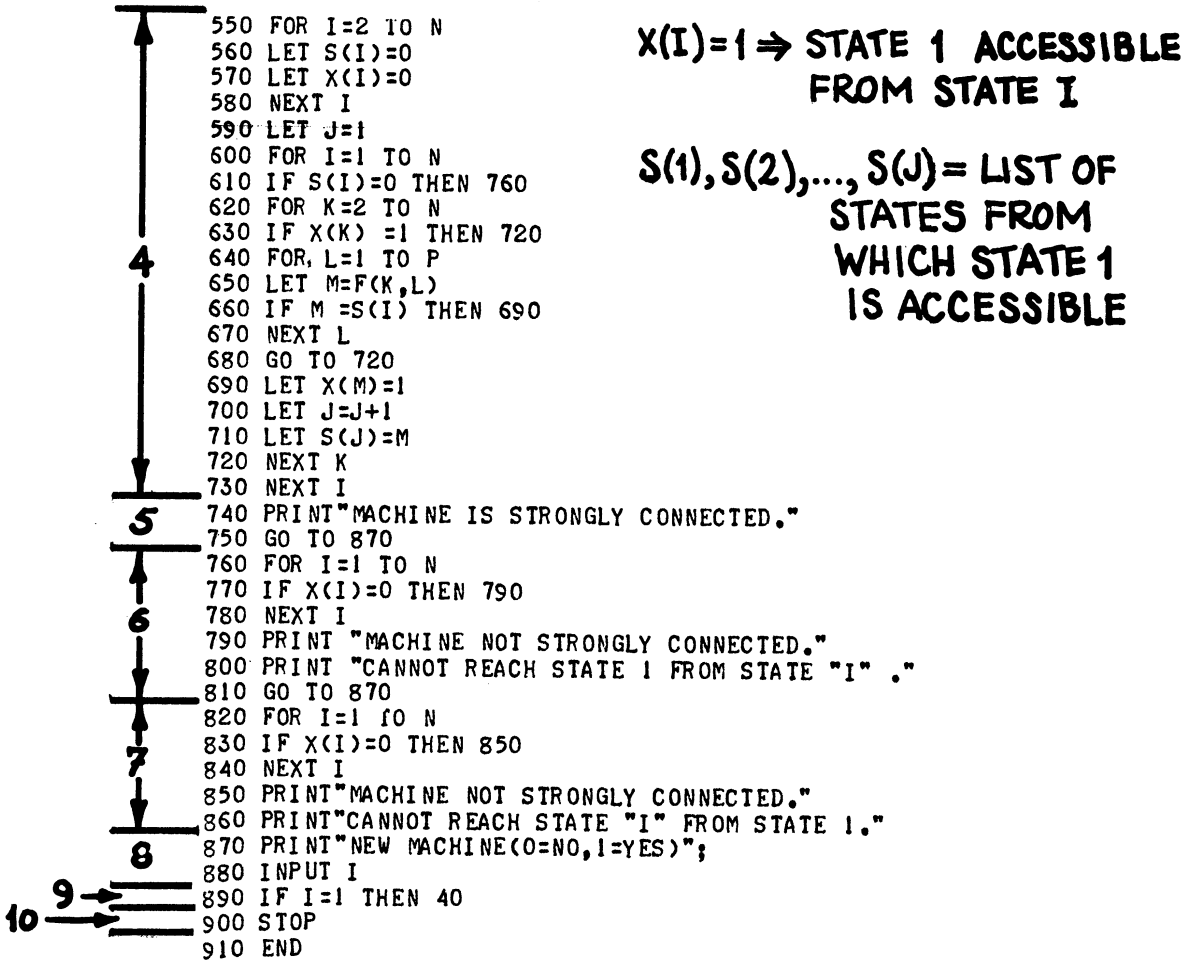
	values and are needed only to satisfy the inflexible format requirements of BASIC.
3) The program will retype the machine table and the results of the analysis; if the machine is not strongly connected, an unconnected state pair will be displayed.	
4) "NEW MACHINE(0=NO, 1=YES)?"	Zero or one as desired, RETURN.

E. Flow Chart


START
INPUT N,P
CHECK THEIR RANGES
INPUT FS-TABLE
PRINT MACHINE TABLE
1
ARE ALL STATES ACCESSIBLE FROM STATE 1 ?
2
F
T
PRINT "ALL STATES ACCESSIBLE FROM STATE 1"
3
IS STATE 1 ACCESSIBLE FROM ALL STATES?
4
F
T
PRINT "MACHINE IS STRONGLY CONNECTED"
5
FIND STATE I FROM WHICH STATE 1 IS NOT ACCESSIBLE.
PRINT "NOT STRONGLY CONNECTED, CAN'T REACH STATE 1 FROM STATE I."
6
FIND STATE I NOT ACCESSIBLE FROM STATE 1.
PRINT "NOT STRONGLY CONNECTED, CAN'T REACH STATE I FROM STATE 1."
7
INPUT I (SET SWITCH)
8
I=0 ?
9
F
T
STOP
10

F. Annotated Program Listing




X(I)=1 ⇒ STATE 1 ACCESSIBLE FROM STATE I
S(1), S(2),..., S(J) = LIST OF STATES FROM WHICH STATE 1 IS ACCESSIBLE
4
5
6
7
8
9
10
550 FOR I=2 TO N
560 LET S(I)=0
570 LET X(I)=0
580 NEXT I
590 LET J=1
600 FOR I=1 TO N
610 IF S(I)=0 THEN 760
620 FOR K=2 TO N
630 IF X(K) =1 THEN 720
640 FOR, L=1 TO P
650 LET M=F(K,L)
660 IF M =S(I) THEN 690
670 NEXT L
680 GO TO 720
690 LET X(M)=1
700 LET J=J+1
710 LET S(J)=M
720 NEXT K
730 NEXT I
740 PRINT"MACHINE IS STRONGLY CONNECTED."
750 GO TO 870
760 FOR I=1 TO N
770 IF X(I)=0 THEN 790
780 NEXT I
790 PRINT "MACHINE NOT STRONGLY CONNECTED."
800 PRINT "CANNOT REACH STATE 1 FROM STATE "I" ."
810 GO TO 870
820 FOR I=1 TO N
830 IF X(I)=0 THEN 850
840 NEXT I
850 PRINT"MACHINE NOT STRONGLY CONNECTED."
860 PRINT"CANNOT REACH STATE "I" FROM STATE 1."
870 PRINT"NEW MACHINE(0=NO,1=YES)";
880 INPUT I
890 IF I=1 THEN 40
900 STOP
910 END

G. Sample Run

```
STR        21:05    MON.---04/24/67

N,P?4,2

FS-TABLE
STATE        ENTRIES
 1              ?3,4,0,0,0
 2              ?1,2,0,0,0
 3              ?4,4,0,0,0
 4              ?2,3,0,0,0


MACHINE TABLE FS
                INPUTS
STATE            1              2

 1               3              4
 2               1              2
 3               4              4
 4               2              3

ALL STATES ACCESSIBLE FROM STATE 1.
MACHINE IS STRONGLY CONNECTED.
NEW MACHINE[0=NO,1=YES]?1
N,P?7,2

FS-TABLE
STATE        ENTRIES
 1              ?3,2,0,0,0
 2              ?3,2,0,0,0
 3              ?1,1,0 ←,0,0
 4              ?6,7,0,0,0
 5              ?4,5,0,0,0
 6              ?7,7,0,0,0
 7              ?5,6,0,0,0


MACHINE TABLE FS
                INPUTS
STATE            1              2

 1               3              2
 2               3              2
 3               1              1
 4               6              7
 5               4              5
 6               7              7
 7               5              6

MACHINE NOT STRONGLY CONNECTED.
CANNOT REACH STATE  4     FROM STATE 1.
NEW MACHINE[0=NO,1=YES]?1
```

```
N,P?7,1

FS-TABLE
STATE        ENTRIES
 1              ?2,0,0,0,0
 2              ?3,0,0,0,0
 3              ?4,0,0,0,0
 4              ?5,0,0,0,0
 5              ?6,0,0,0,0
 6              ?7,0,0,0,0
 7              ?7,0,0,0,0


MACHINE TABLE FS
                INPUTS
STATE            1

 1               2
 2               3
 3               4
 4               5
 5               6
 6               7
 7               7

ALL STATES ACCESSIBLE FROM STATE 1.
MACHINE NOT STRONGLY CONNECTED.
CANNOT REACH STATE 1 FROM STATE  2
NEW MACHINE[0=NO,1=YES]?1
N,P?7,9

P ILLEGAL
N,P?397,5

N ILLEGAL
N,P?STOP

RAN  29 SEC.
```

V. Machine Minimization

A. Program Name: MIN

B. Purpose: To find and display the minimal form of a specified machine.

C. Method: The k-equivalence method is used wherein for any $k \geq 1$, P_k is a partition on S such that for I, J ϵ S

$I \equiv J(P_k)$ iff I is k-equivalent to J.

P_1 is constructed from the FZ table according to

$I \equiv J(P_1)$ iff $(\forall X \epsilon A_I)[FZ(I,X) = FZ(J,X)]$.

P_k is constructed from P_{k-1} according to

$I \equiv J(P_k)$ iff $I \equiv J(P_{k-1})$ and

$(\forall X \epsilon A_I)[FS(I,X) \equiv FS(J,X)(P_{k-1})]$.

For some $K \leq N$ $P_{K-1} = P_K$ and the minimal form of M will have $|P_K|$ states. If $|P_K| = |S|$ M is minimal; if $|P_K| < |S|$ then the minimal form of M is obtained by associating its states with the equivalence classes of P_K.

D. Operation: All user input is supplied at program request in the following sequence:

Program Types	Machine Types
1) "N, P?"	Two integers separated by a comma and followed by a RETURN and corresponding to

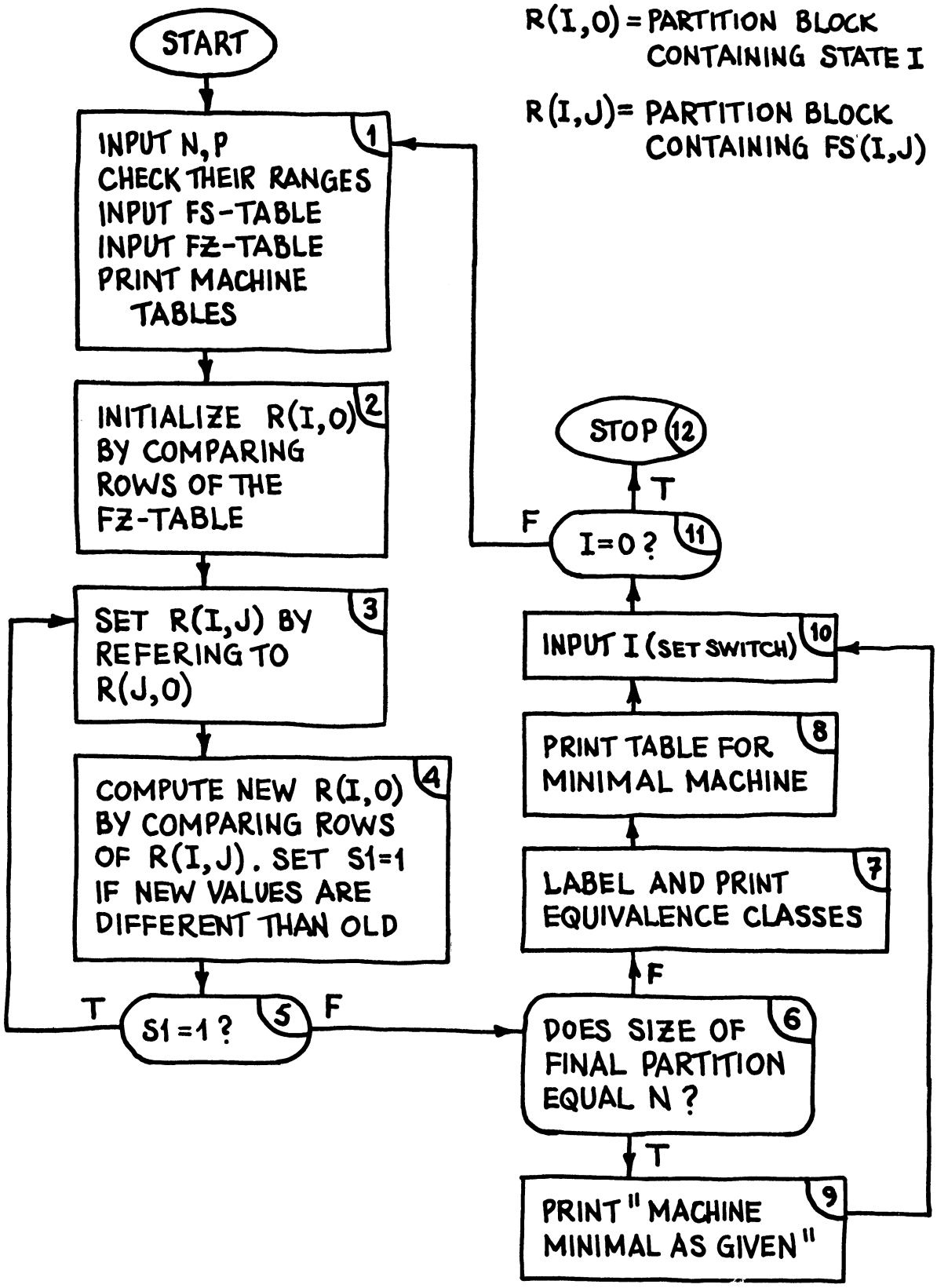
	N and P for the machine to be minimized; $1 \leq N \leq 100$; $1 \leq P \leq 3$. If N or P is out of range, the program will ask for N, P again.
2) After typing two labels for the FS-table, the program will type N of the following form: "I?"	After each "I?" type three integers separated by commas and followed by RETURN. The first P integers correspond to FS(I, 1), FS(I, 2), . . . , FS(I, P); the remaining integers are ignored but must be typed. They may be given any values and are needed only to satisfy the inflexible format requirements of BASIC.
3) After typing two labels for the FZ table, the program will type N lines of the following form: "I?"	After each "I?" type three integers separated by commas and followed by a RETURN. The first P integers correspond to FZ(I, 1), FZ(I, 2), . . . , FZ(I, P); the remaining integers are ignored (see 2 above).
4) Program types full machine table in N lines of form:	

I, FS(I, 1), FZ(I, 1),
FS(I, 2), . . . , FZ(I, N)

5) If the machine is minimal as given the program types "MACHINE MINIMAL AS GIVEN."; if not, the equivalence classes of M are displayed and labelled and the full table of the minimal machine is displayed using the equivalence class labels as state names.

6) "NEW MACHINE(0=NO, 1=YES)?" — Zero or one as desired, RETURN.

E. Flow Chart


R(I,0) = PARTITION BLOCK CONTAINING STATE I
R(I,J) = PARTITION BLOCK CONTAINING FS(I,J)
START
1
INPUT N, P
CHECK THEIR RANGES
INPUT FS-TABLE
INPUT FZ-TABLE
PRINT MACHINE TABLES
2
INITIALIZE R(I,0) BY COMPARING ROWS OF THE FZ-TABLE
3
SET R(I,J) BY REFERING TO R(J,0)
4
COMPUTE NEW R(I,0) BY COMPARING ROWS OF R(I,J). SET S1=1 IF NEW VALUES ARE DIFFERENT THAN OLD
5
S1 = 1 ?
T
F
6
DOES SIZE OF FINAL PARTITION EQUAL N ?
7
LABEL AND PRINT EQUIVALENCE CLASSES
8
PRINT TABLE FOR MINIMAL MACHINE
9
PRINT " MACHINE MINIMAL AS GIVEN "
10
INPUT I (SET SWITCH)
11
I = 0 ?
12
STOP

F. Annotated Program Listing

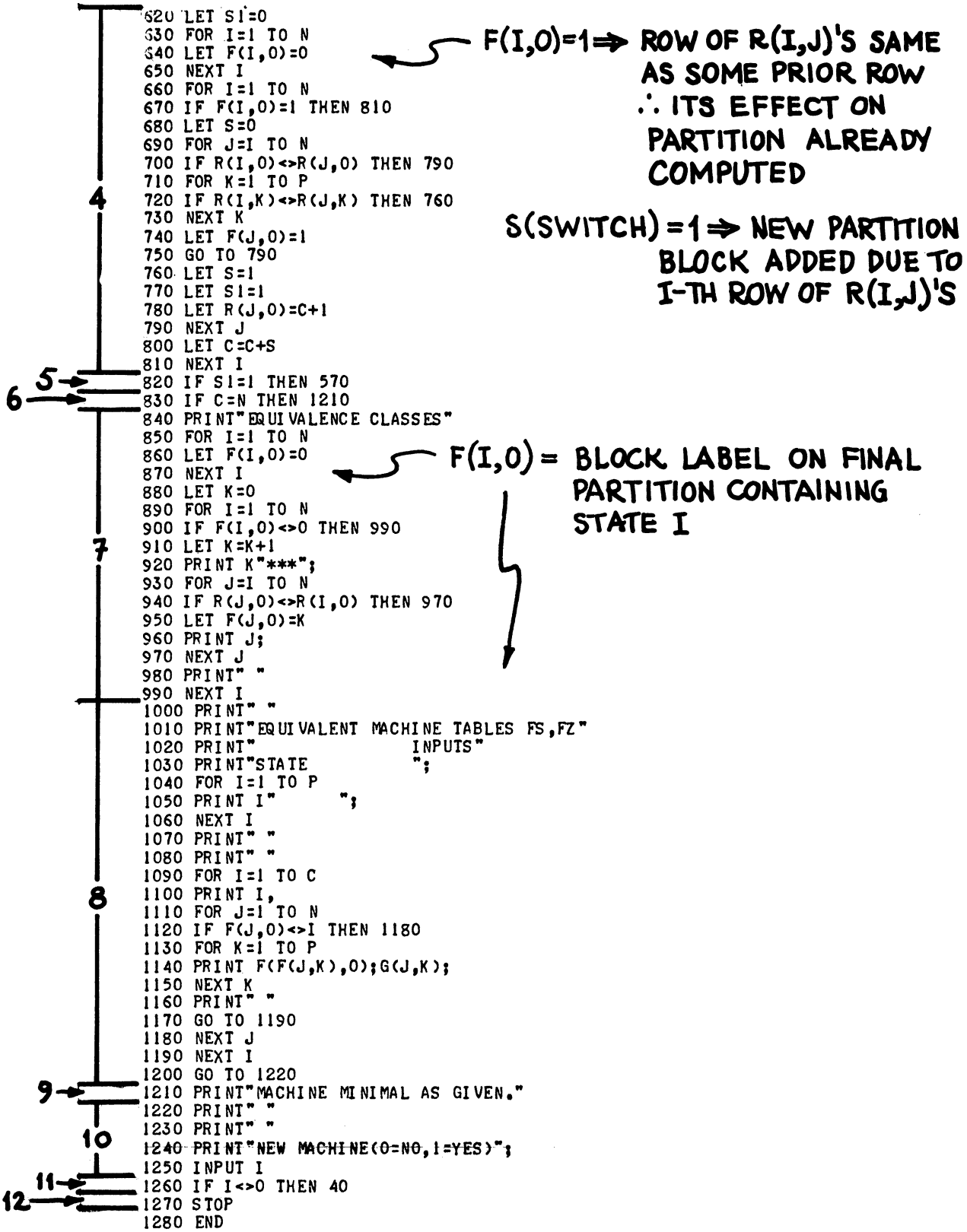
```
MIN       22:33    MON.---04/24/67

10 DIM F(100,3)
20 DIM G(100,3)
30 DIM R(100,3)
40 PRINT"N,P";
50 INPUT N,P
60 PRINT" "
70 IF N*(101-N)>0 THEN 100
80 PRINT "N ILLEGAL"
90 GO TO 40
100 IF P*(4-P)>0 THEN 130
110 PRINT "P ILLEGAL"
120 GO TO 40
130 PRINT "FS-TABLE"
140 PRINT "STATE        ENTRIES"
150 FOR I=1 TO N
160 PRINT I,
170 INPUT F(I,1),F(I,2),F(I,3)
180 NEXT I
190 PRINT" "
200 PRINT "FZ-TABLE"
210 PRINT "STATE        ENTRIES"
220 FOR I=1 TO N
230 PRINT I,
240 INPUT G(I,1),G(I,2),G(I,3)
250 NEXT I
260 PRINT" "
270 PRINT"MACHINE TABLES FS,FZ"
280 PRINT"                INPUTS"
290 PRINT"STATE           ";
300 FOR I=1 TO P
310 PRINT I"      ";
320 NEXT I
330 PRINT" "
340 PRINT" "
350 FOR I=1 TO N
360 PRINT I,
370 FOR J=1 TO P
380 PRINT F(I,J);G(I,J);
390 NEXT J
400 PRINT" "
410 NEXT I
420 PRINT" "
430 FOR I=1 TO N
440 LET R(I,0)=0
450 NEXT I
460 LET C=0
470 FOR I=1 TO N
480 IF R(I,0)<>0 THEN 560
490 LET C=C+1
500 FOR J=I TO N
510 FOR K=1 TO P
520 IF G(I,K)<>G(J,K) THEN 550
530 NEXT K
540 LET R(J,0)=C
550 NEXT J
560 NEXT I
570 FOR I=1 TO N
580 FOR J=1 TO P
590 LET R(I,J)=R(F(I,J),0)
600 NEXT J
610 NEXT I
```

1

2

3

F(I,J) = FS(I,J)
G(I,J) = FZ(I,J)
C = NUMBER OF BLOCKS IN PARTITION


620 LET S1=0
630 FOR I=1 TO N
640 LET F(I,0)=0
650 NEXT I
660 FOR I=1 TO N
670 IF F(I,0)=1 THEN 810
680 LET S=0
690 FOR J=I TO N
700 IF R(I,0)<>R(J,0) THEN 790
710 FOR K=1 TO P
720 IF R(I,K)<>R(J,K) THEN 760
730 NEXT K
740 LET F(J,0)=1
750 GO TO 790
760 LET S=1
770 LET S1=1
780 LET R(J,0)=C+1
790 NEXT J
800 LET C=C+S
810 NEXT I
820 IF S1=1 THEN 570
830 IF C=N THEN 1210
840 PRINT"EQUIVALENCE CLASSES"
850 FOR I=1 TO N
860 LET F(I,0)=0
870 NEXT I
880 LET K=0
890 FOR I=1 TO N
900 IF F(I,0)<>0 THEN 990
910 LET K=K+1
920 PRINT K"***";
930 FOR J=I TO N
940 IF R(J,0)<>R(I,0) THEN 970
950 LET F(J,0)=K
960 PRINT J;
970 NEXT J
980 PRINT" "
990 NEXT I
1000 PRINT" "
1010 PRINT"EQUIVALENT MACHINE TABLES FS,FZ"
1020 PRINT" INPUTS"
1030 PRINT"STATE ";
1040 FOR I=1 TO P
1050 PRINT I" ";
1060 NEXT I
1070 PRINT" "
1080 PRINT" "
1090 FOR I=1 TO C
1100 PRINT I,
1110 FOR J=1 TO N
1120 IF F(J,0)<>I THEN 1180
1130 FOR K=1 TO P
1140 PRINT F(F(J,K),0);G(J,K);
1150 NEXT K
1160 PRINT" "
1170 GO TO 1190
1180 NEXT J
1190 NEXT I
1200 GO TO 1220
1210 PRINT"MACHINE MINIMAL AS GIVEN."
1220 PRINT" "
1230 PRINT" "
1240 PRINT"NEW MACHINE(0=NO,1=YES)";
1250 INPUT I
1260 IF I<>0 THEN 40
1270 STOP
1280 END
4
5
6
7
8
9
10
11
12
F(I,0)=1 ⇒ ROW OF R(I,J)'S SAME AS SOME PRIOR ROW ∴ ITS EFFECT ON PARTITION ALREADY COMPUTED
S(SWITCH)=1 ⇒ NEW PARTITION BLOCK ADDED DUE TO I-TH ROW OF R(I,J)'S
F(I,0)= BLOCK LABEL ON FINAL PARTITION CONTAINING STATE I

G. Sample Run

```
MIN        22:24    MON.---04/24/67

N,P?9,3

FS-TABLE
STATE        ENTRIES
 1              ?2,2,5
 2              ?1,4,4
 3              ?2,2,5
 4              ?3,2,2
 5              ?6,4,3
 6              ?8,9,6
 7              ?6,2,8
 8              ?4,4,7
 9              ?7,9,7

FZ-TABLE
STATE        ENTRIES
 1              ?1,0,0
 2              ?0,1,1
 3              ?1,0,0
 4              ?0,1,1
 5              ?1,0,0
 6              ?0,1,1
 7              ?1,0,0
 8              ?1,0,0
 9              ?0,1,1

MACHINE TABLES FS,FZ
               INPUTS
STATE           1          2          3

 1              2   1      2   0      5   0
 2              1   0      4   1      4   1
 3              2   1      2   0      5   0
 4              3   0      2   1      2   1
 5              6   1      4   0      3   0
 6              8   0      9   1      6   1
 7              6   1      2   0      8   0
 8              4   1      4   0      7   0
 9              7   0      9   1      7   1

EQUIVALENCE CLASSES
 1    *** 1     3     8
 2    *** 2     4
 3    *** 5     7
 4    *** 6
 5    *** 9

EQUIVALENT MACHINE TABLES FS,FZ
               INPUTS
STATE           1          2          3

 1              2   1      2   0      3   0
 2              1   0      2   1      2   1
 3              4   1      2   0      1   0
 4              1   0      5   1      4   1
 5              3   0      5   1      3   1

NEW MACHINE[0=NO,1=YES]?1
```

```
N,P?3,2

FS-TABLE
STATE        ENTRIES
 1              ?2,3,0
 2              ?1,3,0
 3              ?3,2,0

FZ-TABLE
STATE        ENTRIES
 1              ?1,5,0          1
 2              ?2,3,0
 3              ?2,10,0

MACHINE TABLES  FS,FZ
                INPUTS
STATE            1              2

 1               2      1       3       5
 2               1      2       3       3
 3               3      2       2       10

MACHINE MINIMAL AS GIVEN.


NEW MACHINE[0=NO,1=YES]?1
N,P?4,4

P ILLEGAL
N,P?1,5

P ILLEGAL
N,P?107,2

N ILLEGAL
N,P?STOP

RAN  41 SEC.
```

VI. Displaying a Machine's Automorphism Group

A. Program Name: AUTO

B. Purpose: To compute and display the state automorphisms and their group for a specified machine. We (1) will limit consideration to machines in which the full state set S is reachable from State 1 and (2) due to format constraints will display the group table for the automorphisms only when $N \leq 10$. If $N > 10$ the user can construct the group table himself by examining the automorphisms directly.

C. Method: If all states of M are reachable from state 1 then

1) there are at most N automorphisms of M
2) each corresponds to a permutation on S and
3) each can be generated from the single point mapping $h(1) \in S$ on state 1 and the recursion equation $(\forall I \in S, X \in A_I)[h(FS(I,X)) = FS(h(I), X)]$.

Conversely we can test if a mapping $h(1) \in S$ on state 1 generates an automorphism on M by applying the above recursion equation. One of three possibilities results:

1) a complete permutation is induced on S
2) a conflict occurs in using the recursion equation

thus indicating no automorphism exists for the assumed h(1.), or

3) a permutation is induced on a proper subset of S—indicating some states are not reachable from state 1.

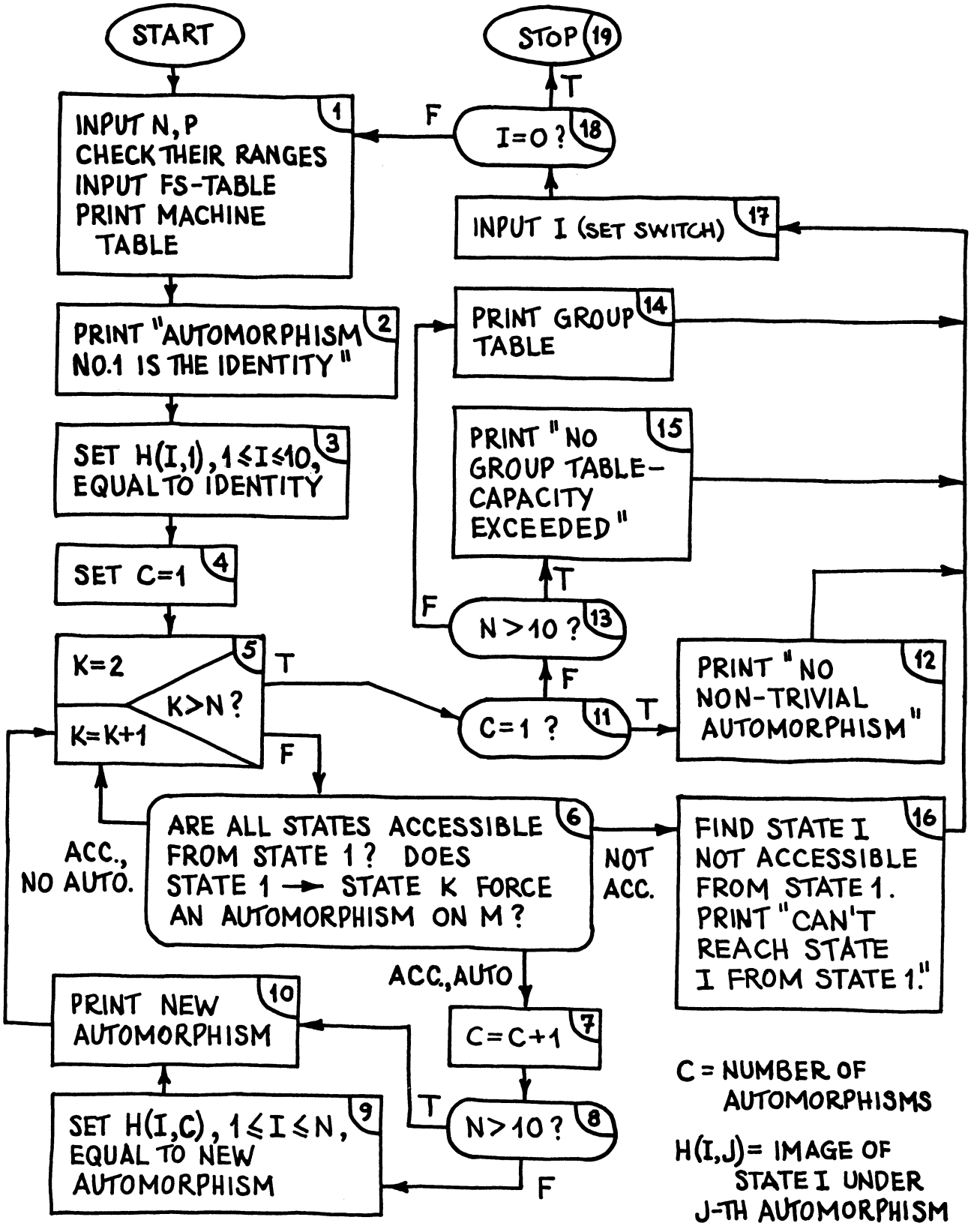
We will generate the full automorphism group by testing iff h(1) = k generates an automorphism for each $1 \leq k \leq N$ (i.e. $k \in S$).

D. Operation: All user input is supplied at program request in the following sequence:

Program Types	User Types
1) "N, P?"	Two integers separated by a comma and followed by a RETURN and corresponding to N and P for the machine to be simulated; $1 \leq N \leq 100$; $1 \leq P \leq 5$. If N or P is out of range the program will ask for N, P again.
2) After typing two lines of labels for the FS-table, the program will type N lines of the following form: "I?"	After each "I?" type five integers separated by commas and followed by RETURN. The first P integers correspond to FS(I, 1), FS(I, 2), . . . , FS(I, P); the remaining

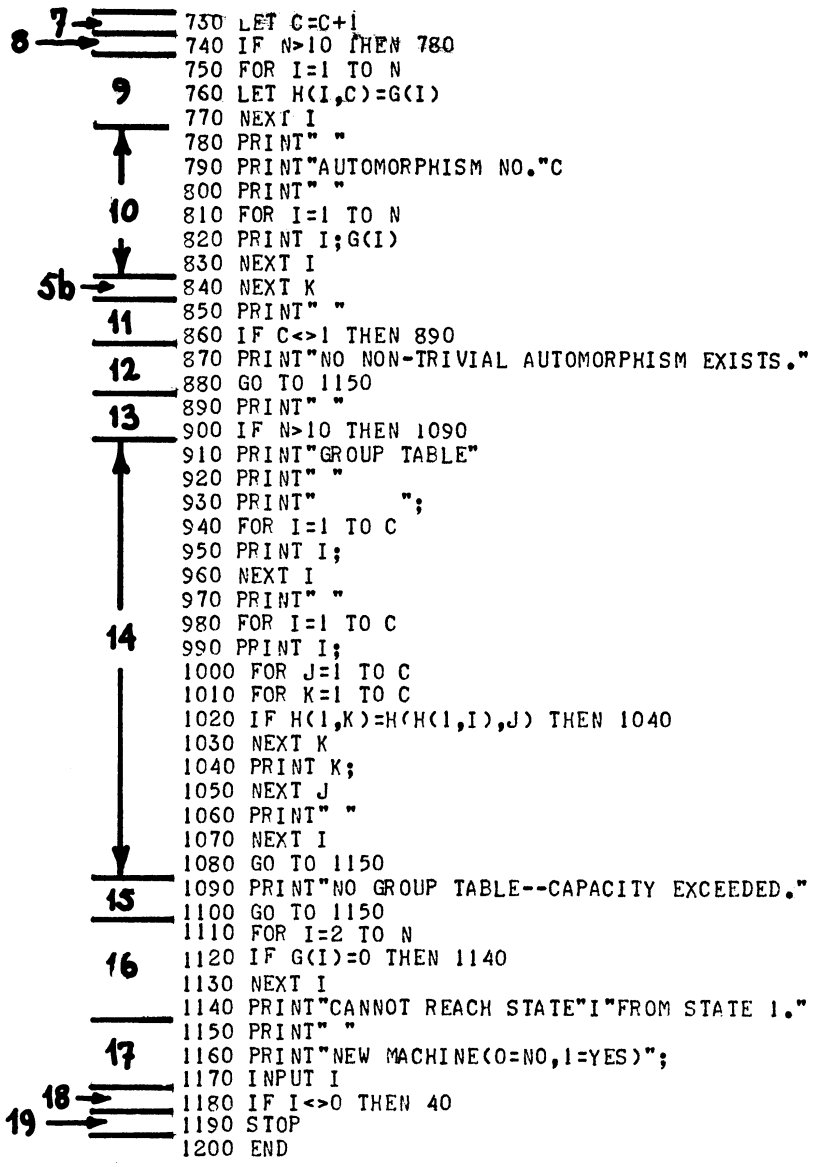
	integers are ignored but must be typed. They may be given any values and are needed only to satisfy the inflexible format of BASIC.
3) The program will re-type the machine table and the results of the analysis. Every non-trivial automorphism will be displayed and if $N \leq 10$, the full group table will be typed.	
4) "NEW MACHINE(0=NO, 1=YES)?"	Zero or one as desired, RETURN.

E. Flow Chart


START
STOP 19
INPUT N, P
CHECK THEIR RANGES
INPUT FS-TABLE
PRINT MACHINE TABLE
1
F
I=0 ? 18
T
INPUT I (SET SWITCH) 17
PRINT "AUTOMORPHISM NO.1 IS THE IDENTITY" 2
PRINT GROUP TABLE 14
SET H(I,1), 1≤I≤10, EQUAL TO IDENTITY 3
PRINT "NO GROUP TABLE-CAPACITY EXCEEDED" 15
SET C=1 4
T
F
N>10 ? 13
K=2
K=K+1
K>N ? 5
T
F
C=1 ? 11
T
PRINT "NO NON-TRIVIAL AUTOMORPHISM" 12
ARE ALL STATES ACCESSIBLE FROM STATE 1? DOES STATE 1 → STATE K FORCE AN AUTOMORPHISM ON M? 6
ACC., NO AUTO.
NOT ACC.
FIND STATE I NOT ACCESSIBLE FROM STATE 1. PRINT "CAN'T REACH STATE I FROM STATE 1." 16
ACC., AUTO
PRINT NEW AUTOMORPHISM 10
C=C+1 7
SET H(I,C), 1≤I≤N, EQUAL TO NEW AUTOMORPHISM 9
T
N>10 ? 8
F
C = NUMBER OF AUTOMORPHISMS
H(I,J)= IMAGE OF STATE I UNDER J-TH AUTOMORPHISM

F. Annotated Program Listing

```
AUTO      22:38   TUES.--04/25/67

10 DIM F(100,5)
20 DIM G(100)
30 DIM H(10,10)
40 PRINT"N,P";                                  F(I,J)= FS(I,J)
50 INPUT N,P
60 PRINT" "
70 IF N*(101-N)>0 THEN 100
80 PRINT "N ILLEGAL"
90 GO TO 40
100 IF P*(6-P)>0 THEN 130
110 PRINT "P ILLEGAL"
120 GO TO 40
130 PRINT "FS-TABLE"
140 PRINT "STATE       ENTRIES"
150 FOR I=1 TO N
160 PRINT I,
170 INPUT F(I,1),F(I,2),F(I,3),F(I,4),F(I,5)
180 NEXT I
190 PRINT" "
200 PRINT" "
210 PRINT"MACHINE TABLE FS"
220 PRINT"                INPUTS"
230 PRINT"STATE           ";
240 FOR I=1 TO P
250 PRINT I"       ";
260 NEXT I
270 PRINT" "
280 PRINT" "
290 FOR I=1 TO N
300 PRINT I,
310 FOR J=1 TO P
320 PRINT F(I,J)"       ";
330 NEXT J
340 PRINT" "
350 NEXT I
360 PRINT" "
370 PRINT"AUTOMORPHISM NO.1 IS THE IDENTITY."
380 FOR I=1 TO 10
390 LET H(I,1)=I
400 NEXT I
410 LET C=1
420 FOR K=2 TO N
430 FOR I=1 TO N                    G(I)= IMAGE OF STATE I UNDER
440 LET G(I)=0                            AUTOMORPHISM CURRENTLY
450 LET F(I,0)=0                          DEVELOPING
460 NEXT I
470 LET G(1)=K                      F(I,0) [SWITCH] = 1 ⇒ EFFECT
480 LET S=1                               OF G(I)'S SUCCESSORS ON
490 FOR I=S TO N                          CURRENTLY DEVELOPING
500 IF G(I)=0 THEN 590                    AUTOMORPHISM HAS
510 IF F(I,0)<>0 THEN 590                 BEEN NOTED
520 LET F(I,0)=1
530 FOR J=1 TO P                    S[SWITCH] = 0 ⇒ SOME STATE NOT
540 IF G(F(I,J))=0 THEN 570               ACCESSIBLE FROM STATE 1,
550 IF G(F(I,J))=F(G(I),J) THEN 580         = 1 ⇒ EFFECT OF G(I)'S
560 GO TO 840                             SUCCESSORS NOTED FOR
570 LET G(F(I,J))=F(G(I),J)               ALL I
580 NEXT J                                  ≥ 1 ⇒ G(S) IS KNOWN
590 NEXT I                                BUT F(S,0) = 0
600 LET S=1
610 FOR I=1 TO N
620 IF F(I,0)<>0 THEN 670
630 IF G(I)=0 THEN 660
640 LET S=I
650 GO TO 490
660 LET S=0
670 NEXT I
680 IF S=0 THEN 1110
690 FOR I=2 TO N
700 IF G(I)=1 THEN 730
710 NEXT I
720 GO TO 840
```


7
8
9
10
5b
11
12
13
14
15
16
17
18
19
730 LET C=C+1
740 IF N>10 THEN 780
750 FOR I=1 TO N
760 LET H(I,C)=G(I)
770 NEXT I
780 PRINT" "
790 PRINT"AUTOMORPHISM NO."C
800 PRINT" "
810 FOR I=1 TO N
820 PRINT I;G(I)
830 NEXT I
840 NEXT K
850 PRINT" "
860 IF C<>1 THEN 890
870 PRINT"NO NON-TRIVIAL AUTOMORPHISM EXISTS."
880 GO TO 1150
890 PRINT" "
900 IF N>10 THEN 1090
910 PRINT"GROUP TABLE"
920 PRINT" "
930 PRINT"   ";
940 FOR I=1 TO C
950 PRINT I;
960 NEXT I
970 PRINT" "
980 FOR I=1 TO C
990 PRINT I;
1000 FOR J=1 TO C
1010 FOR K=1 TO C
1020 IF H(1,K)=H(H(1,I),J) THEN 1040
1030 NEXT K
1040 PRINT K;
1050 NEXT J
1060 PRINT" "
1070 NEXT I
1080 GO TO 1150
1090 PRINT"NO GROUP TABLE--CAPACITY EXCEEDED."
1100 GO TO 1150
1110 FOR I=2 TO N
1120 IF G(I)=0 THEN 1140
1130 NEXT I
1140 PRINT"CANNOT REACH STATE"I"FROM STATE 1."
1150 PRINT" "
1160 PRINT"NEW MACHINE(0=NO,1=YES)";
1170 INPUT I
1180 IF I<>0 THEN 40
1190 STOP
1200 END

G. Sample Run

```
AUTO      22:23   TUES.--04/25/67

N,P?10,3

FS-TABLE
STATE       ENTRIES
 1            ?1,2,3,0,0
 2            ?2,3,4,0,0
 3            ?3,4,5,0,0
 4            ?4,5,6,0,0
 5            ?5,6,7,0,0
 6            ?6,7,8,0,0
 7            ?7,8,9,0,0
 8            ?8,9,10,0,0
 9            ?9,10,1,0,0
 10           ?10,1,2,0,0


MACHINE TABLE FS
                INPUTS
STATE            1          2          3

 1               1          2          3
 2               2          3          4
 3               3          4          5
 4               4          5          6
 5               5          6          7
 6               6          7          8
 7               7          8          9
 8               8          9          10
 9               9          10         1
 10              10         1          2

AUTOMORPHISM NO.1 IS THE IDENTITY.

AUTOMORPHISM NO. 2

 1      2
 2      3
 3      4
 4      5
 5      6
 6      7
 7      8
 8      9
 9      10
 10     1

AUTOMORPHISM NO. 3

 1      3
 2      4
 3      5
 4      6
 5      7
 6      8
 7      9
 8      10
 9      1
 10     2

AUTOMORPHISM NO. 4

 1      4
 2      5
 3      6
 4      7
 5      8
 6      9
 7      10
 8      1
 9      2
 10     3
```

AUTOMORPHISM NO. 5

1	5
2	6
3	7
4	8
5	9
6	10
7	1
8	2
9	3
10	4

AUTOMORPHISM NO. 6

1	6
2	7
3	8
4	9
5	10
6	1
7	2
8	3
9	4
10	5

AUTOMORPHISM NO. 7

1	7
2	8
3	9
4	10
5	1
6	2
7	3
8	4
9	5
10	6

AUTOMORPHISM NO. 8

1	8
2	9
3	10
4	1
5	2
6	3
7	4
8	5
9	6
10	7

AUTOMORPHISM NO. 9

1	9
2	10
3	1
4	2
5	3
6	4
7	5
8	6
9	7
10	8

AUTOMORPHISM NO. 10

1	10
2	1
3	2
4	3
5	4
6	5
7	6
8	7
9	8
10	9

GROUP TABLE

	1	2	3	4	5	6	7	8	9	10
1	1	2	3	4	5	6	7	8	9	10
2	2	3	4	5	6	7	8	9	10	1
3	3	4	5	6	7	8	9	10	1	2
4	4	5	6	7	8	9	10	1	2	3
5	5	6	7	8	9	10	1	2	3	4
6	6	7	8	9	10	1	2	3	4	5
7	7	8	9	10	1	2	3	4	5	6
8	8	9	10	1	2	3	4	5	6	7
9	9	10	1	2	3	4	5	6	7	8
10	10	1	2	3	4	5	6	7	8	9

```
NEW MACHINE[0=NO,1=YES]?1
N,P?3,1

FS-TABLE
STATE       ENTRIES
 1             ?2,0,0,0,0
 2             ?2,0,0,0,0
 3             ?3,0,0,0,0


MACHINE TABLE FS
                INPUTS
STATE            1

 1               2
 2               2
 3               3

AUTOMORPHISM NO.1 IS THE IDENTITY.
CANNOT REACH STATE 3    FROM STATE 1.

NEW MACHINE[0=NO,1=YES]?1
N,P?3,2

FS-TABLE
STATE       ENTRIES
 1             ?3,2,0,0,0
 2             ?3,2,0,0,0
 3             ?2,3,0,0,0


MACHINE TABLE FS
                INPUTS
STATE            1           2

 1               3           2
 2               3           2
 3               2           3

AUTOMORPHISM NO.1 IS THE IDENTITY.

NO NON-TRIVIAL AUTOMORPHISM EXISTS.

NEW MACHINE[0=NO,1=YES]?1
N,P?12,6

P ILLEGAL
N,P?123,2

N, ILLEGAL
```

```
N,P?12,2

FS-TABLE
STATE       ENTRIES
 1             ?8,2,0,0,0
 2             ?3,3,0,0,0
 3             ?4,4,0,0,0
 4             ?5,5,0,0,0
 5             ?6,6,0,0,0
 6             ?1,12,0,0,0
 7             ?2,8,0,0,0
 8             ?9,9,0,0,0
 9             ?10,10,0,0,0
 10            ?11,11,0,0,0
 11            ?12,12,0,0,0
 12            ?7,6,0,0,0

MACHINE TABLE FS
                INPUTS
STATE            1          2

 1               8          2
 2               3          3
 3               4          4
 4               5          5
 5               6          6
 6               1          12
 7               2          8
 8               9          9
 9               10         10
 10              11         11
 11              12         12
 12              7          6

AUTOMORPHISM NO.1 IS THE IDENTITY.

AUTOMORPHISM NO. 2

 1     7
 2     8
 3     9
 4     10
 5     11
 6     12
 7     1
 8     2
 9     3
 10    4
 11    5
 12    6

NO GROUP TABLE--CAPACITY EXCEEDED.

NEW MACHINE[0=NO,1=YES]?0

TIME:  53 SECS.
```

VII. Bibliography

General reference on finite state machines and in particular on strong connectedness and machine minimization:

[1] Gill, A., Introduction to the Theory of Finite-State Machines, McGraw-Hill, New York (1962).

Automorphisms on automata:

[2] Bayer, R., Automorphism Groups and Quotients of Strongly Connected Automata and Monadic Algebras, IEEE Conference Record of 1966 Seventh Annual Symposium on Switching Theory, October 26-28, 1966.

References to computer programs dealing with finite state machines:

[3] Farr, E. H., Lattice Properties of Sequential Machines, Journal of the Association for Computing Machinery, July, 1963, vol. 10, no. 3, p. 365.

This paper mentions a program to produce all the partitions with S. P. for a given machine for which $N \leq 38$, $P \leq 40$.

[4] Griffiths, T. V., M-460 Program Notes: Some LISP Routines for Manipulating Automata, AFCRL-66-84, February 1967, Physical and Mathematical Sciences Research Papers, No. 192, Data Sciences Laboratory Project 4641, Air Force Cambridge Research Laboratories, L. G. Hanscom Field, Bedford, Massachusetts.

This paper details a number of LISP programs to represent machines, cascade them, minimize them, complement them, compute their behavioral union, intersection, star, and reverse, etc.

[5] Roberts, M. B., A Generalized Recognizer for Finite State Languages, Moore School of Electrical Engineering, University of Pennsylvania Report No. 66-03, August 1965.

This report discusses an IPL-V program to determine if a given symbol string is denoted by a given regular expression; programs to convert a regular expression to a state table and minimize it are also mentioned.

[6] Silverstein, M., Computer Procedures for Analysis of Finite Automata, term project, University of California, Berkeley (1962).

This paper mentioned IPL-V routines to minimize a given machine, develop distinguishing sequences for state pairs, determine multiple preset diagnosing and regular preset homing experiments (neither of these necessarily minimal).

[7] Sarma, Hota S., Design of a Computer Program for Minimization and State Identification of Automata, Masters thesis, Electrical Engineering, Tuskegee Institute, May 1966.

This thesis describes a program written for online use of an IBM 1620 and which minimizes a given machine and determines minimal simple preset homing and diagnosing experiments for a given admissible set.

[8] Sacco, W. J., A Computer Technique Useful for Some Problems in the Partitioning Theory of Sequential Machines, Memorandum Report No. 1733, March 1966, Ballistic Research Laboratories, U.S. Army Material Command, Aberdeen Proving Ground, Maryland.

This paper gives no computer program but rather a technique for determining if (1) a given partition has SP, (2) two given partitions constitute a partition pair, and (3) two set systems constitute a system pair.

In addition, W. D. Maurer uses a computer to assit in minimally decomposing group machines; a report on his work is to appear in a new journal, The Journal of Computer and Systems Science.