PRELIMINARY MANUAL

7090 FORTRAN

DECISION TABLE SYSTEM

July 15, 1962

7090 FORTRAN DECISION TABLE SYSTEM

Preface

Of the various activities that go into setting up a procedure for a computer, often the hardest is obtaining a precise definition of exactly what is to be done under all combinations of circumstance. The conditions under which each action is to be performed must be enumerated and every step specified in detail. The sequence of execution of operations must be explicit and the exceptions to normal processing must be identified.

Complex procedures have generally been described with block diagrams, a form of man-to-man communication and documentation. Unfortunately, block diagrams may become confusing in complex situations. Also, they are not a man-to-machine communication.

Many languages have been developed for communication with machines, some using terminology which closely resembles that of English or mathematics. Most of these languages are oriented toward some particular area of computing and are powerful within those areas. To use such a language with equal power in some other area of computing, the language must be either modified or expanded.

The 7090 FORTRAN Decision Table System is the result of such an expansion. FORTRAN, by itself, is powerful in specifying arithmetic

computations and iterative processes. It is less powerful in describing complex decision logic. Decision tables offer a means of describing such logic, as they provide a descriptive representation of complex decision procedures in a manner that is easy to visualize and simple to follow. They show alternatives and exceptions in an explicit manner, while presenting the relationships among the variables clearly. They show the sequence of conditions and actions in an unambiguous manner. Additionally, a computer procedure written with decision tables, is, to a large extent, its own documentation.

As decision tables were specifically developed to aid in defining decision logic, it would seem that the combining of decision tables and FORTRAN would result in an extremely powerful tool for developing scientific and engineering solutions in a machine processable form.

With this aim in mind, the IBM 7090 FORTRAN Decision Table System has been developed. Initial studies indicate that the following advantages are to be gained:

- Significantly easier handling of complex conditions and their corresponding actions.
- Faster program preparation.
- Greater accuracy in program preparation.
- Simpler and faster debugging.
- Continuously accurate, readable documentation.

A preprocessor has been written to convert programs written in decision table form into standard FORTRAN programs. These programs are then submitted to the FORTRAN compiler for conversion to machine code. Experimentation with the specified language is being undertaken. If it is successful, suitable modifications and extensions will be made to further improve the system.

Chapter 1 of this manual introduces the concepts of decision tables and discusses the preparation of decision tables for machine processing.

Chapter 2 shows how these concepts are used in the 7090 Decision Table System and presents the essentials of the system and language. This chapter utilizes a graded series of examples - each example becoming progressively more sophisticated as further concepts are introduced and tied into the system. Each example has review questions and answers for self-testing.

Chapter 3 is a reference section, describing the details of the language elements and their relationships to FORTRAN and the system.

TABLE OF CONTENTS

CHAPTER 1

An Introduction to the 7090 FORTRAN Decision Table System

Some orderly arrangement of information is necessary for the logical solution of any problem. Consider, for example, the problem given below:

Problem: Deduce the names of the player in each position of a baseball team, given the following facts:

> Andy disliked the catcher. Ed's sister was engaged to the second baseman. The center fielder was taller than the right fielder. Harry and the third baseman lived in the same building. Paul and Allen each won $20 from the pitcher at pinochle. Ed and the outfielders played poker during their free time. The pitcher's wife was the third baseman's sister. All the battery and infield, except Allen, Harry, and Andy, are shorter than Sam. Paul, Andy, and the shortstop each lost $50 at the racetrack. Paul, Harry, Bill, and the catcher took a trouncing from the second baseman at pool. Sam was undergoing a divorce suit. The catcher and the third baseman each had two children. Ed, Paul, Jerry, and the right and center fielders were bachelors. The others were married. The shortstop, the third baseman, and Bill each cleaned up $100 betting on the fight. One of the outfielders was Mike or Andy. Jerry was taller than Bill. Mike was shorter than Bill. Each of them was heavier than the third baseman.

At first glance, this problem does not seem to lend itself to an easy solution. However, when the solution of the problem is considered in a series of orderly steps, the way to the solution is much less complex.

One method is to construct a table to record decisions as they are made about each player and his position on the team.

	C	P	1st	2nd	3rd	SS	LF	RF	CF
Andy	N								
Ed				N			N	N	N
Harry									
Paul		N							
Allen		N							
Bill									
Sam									
Jerry									
Mike									

It is now necessary to examine the statements of the problem, one at a time, and record the decisions which can be inferred from them. For example, the first sentence of the problem implies that Andy is not the catcher, while the second statement leads to the decision that Ed is not the second baseman.

When the entire problem is analyzed in this manner, the process of elimination will remove any doubt as to which player is in which position.

	P	C	1st	2nd	3rd	SS	LF	RF	CF
Andy	N	N	N	N	(Y)	N	N	N	N
Ed	N	N	N	N	N	(Y)	N	N	N
Harry	(Y)	N	N	N	N	N	N	N	N
Paul	N	N	(Y)	N	N	N	N	N	N
Allen	N	(Y)	N	N	N	N	N	N	N
Bill	N	N	N	N	N	N	N	N	(Y)
Sam	N	N	N	N	N	N	(Y)	N	N
Jerry	N	N	N	(Y)	N	N	N	N	N
Mike	N	N	N	N	N	N	N	(Y)	N

At the heart of most data processing is a set of decisions which must be made by the computer regarding data and the sequence of actions which must be taken as a result of these decisions. The statements may be regarded as a set of conditions which, when satisfied, cause a certain action or set of actions to be taken.

A decision table allows the defining of all conditions and separating them from resulting actions. Further, a decision table relates a given condition or set of conditions to the actions associated with them. A series of conditions that must result in a given series of actions constitutes a rule. Alternate conditions that result in other actions constitute other rules.

Basically then, a decision table does two things.

(1) It relates a series of conditions and actions.

(2) It aligns alternate rules in a parallel manner.

Decision Table Structure

To begin to see what decision tables are all about, look at the example of a decision table shown in Figure 1-1.

	Rule 1	Rule 2	Rule 3	Rule 4
X	GE 0.	GE 10.9	GE 21.6	
X	LT 10.9	LT 21.6	LT 50.	
Y=	8.72	16.19	24.07	
PRINT 3,	X, Y	X, Y	X, Y	
GO TO	10	20	30	99

Figure 1-1. An example of a Decision Table

The decision table contains four rules. It is assumed that a value of X is read prior to entry into the table. This value of X is then compared to the ranges of numbers specified in the section above the double horizontal line. For every value of X within a certain range, the value of Y is a constant.

The first rule of this table (consisting of the general column to the left of the double vertical line and the rule 1 column) is expressed as follows:

If the value of X is greater than or equal to zero, and if the value of X is less than 10.9, then assign the value 8.72 to Y, and print the values of X and Y, and go to table number 10.

The underlined words of this rule are implied by table layout. The condition area above the double line consists of one or more IF statements that are answerable by either yes or no. If more than one statement is present, then all statements are connected by logical AND's.

Due to this logical AND connection, it may be seen that all portions of the conditions of a rule must be satisfied in order for the rule to be invoked.

Similar requirements of connection are also made for the action area below the double line. That is, every action is performed. In addition, every action is performed in the order given.

Figure 1-2 is an exploded view of the components of the decision table of Figure 1-1, to show more clearly the parts of the table and the terms used to describe them.

The horizontal and vertical double lines serve as lines of demarcation, to divide the table into logical portions. The entire area above the horizontal double line is the condition area. The area below this line is the action area. For each of the areas, the portion to the left of the double vertical line is the stub. The portion to the right of this line is the entry area.

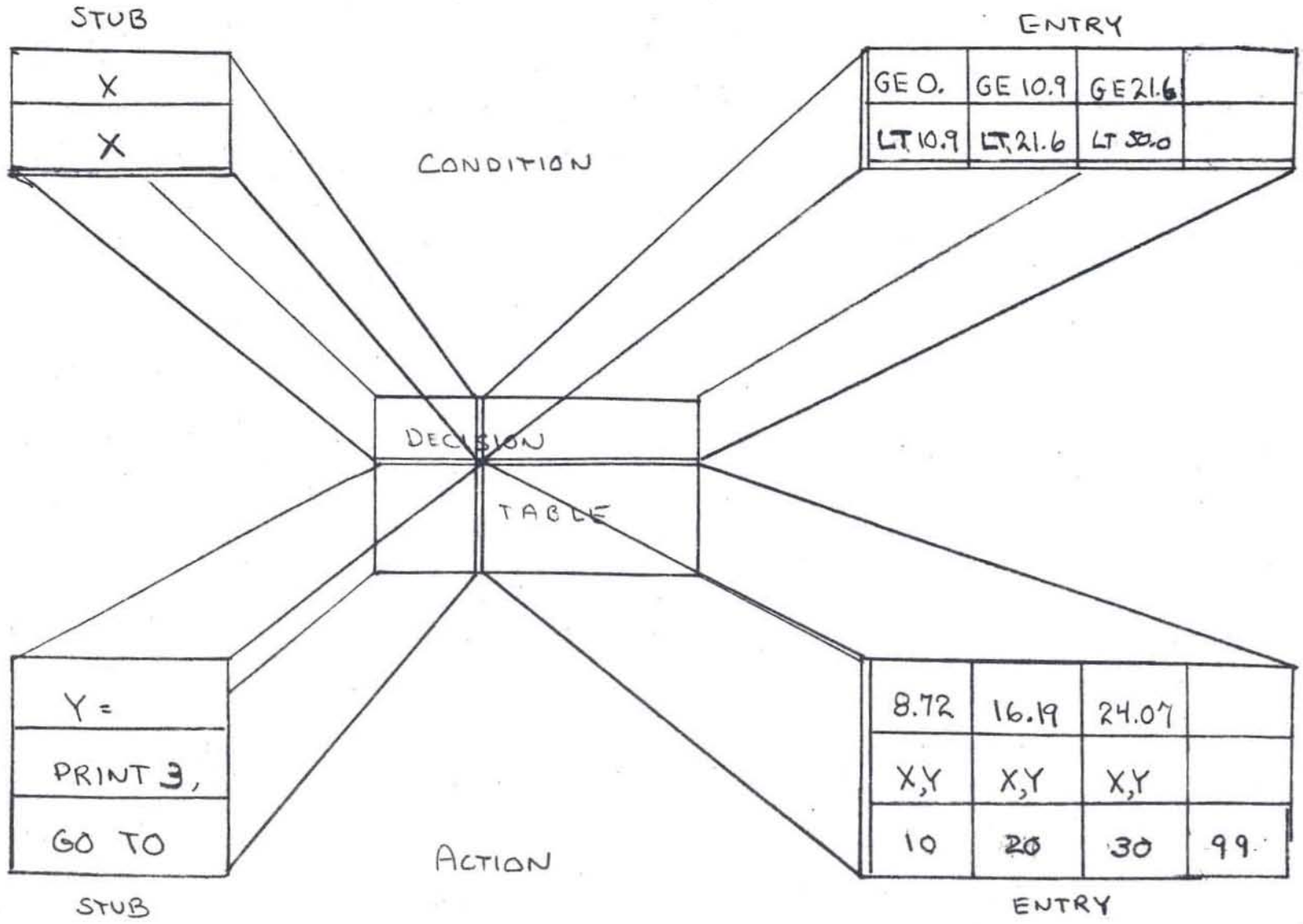


Figure 1-2. Components of a Decision Table

Each table then consists of four separate areas -- condition stub, condition entries, action stub, and action entries.

The entry area, both condition and action is further divided into columns. Each column constitutes a rule. Any decision table may have as many rules as required, limited only by the format of the system in use.

A rule is specifically defined by its conditions. The stub area is a portion of each rule and is assumed to be contiguous to each of the rules. Therefore, a decision table rule consists of the vertical column of condition and action stubs in conjunction with a single vertical column of condition and action entries. The general form of the rule is:

rule n = 1st condition stub, 1st condition entry of column n,

... ,

last condition stub, last condition entry of column n;

1st action stub, 1st action entry of column n,

... ,

last action stub, last action entry of column, n.

For example,

rule 2 = If X is greater than or equal to 10.9, and if X is less than 21.6, then assign the value 16.19 to Y, and print the values of X and Y, and go to table 20.

Each continuous action row, a stub and entry of a rule, is a complete

FORTRAN statement in the 7090 FORTRAN DTS. For instance, the first action row, using rule 2 is:

Y = 16.19

with the same connotations as the exact FORTRAN statement. The format shown in Figures 1-1 and 1-2 is known as the extended entry format. In this type of format, the statement is begun in the stub and extended into the entry column.. The statement may be split wherever desired since the stub and the rule column are the components of the entire statement.

It is important to note that the conditions of any set of rules of a decision table must take care of all possible conditions. It is seen that the first three rules of Figure 1-2 do cover all circumstances of the value of X in the range of X equal to 0 to X less than 50. The fourth rule has been designed to take care of everything else. It would be assumed that the value of X that has been read in is in the required range of values. But to cover the situation where this is not the case, rule 4 reads:

If the value of X is not in the range of 0 to 50, then go to table 99.

It is not necessary that a decision table contain the else rule, so long as the values to be tested cause one rule of the table to be executed. Conditions should not be written such that more than one rule satisfied for any one pass through the table.

An alternate method of composition for the decision table is shown in Figure 1-3. Again, the components of the table are identical and in the same place. Only the method of writing the statements, and the indication of performance is different.

	Rule 1	Rule 2	Rule 3	Rule 4	Rule 5
X GE 0.	N	Y			
X GE 10.9		N	Y		
X GE 21.6			N	Y	
X GE 50.				N	Y
Y = 8.72		X			
Y = 16.19			X		
Y = 24.07				X	
PRINT 3, X, Y		X	X	X	
GO TO 99	X				X
GO TO 10		X			
GO TO 20			X		
GO TO 30				X	

Figure 1-3 - Limited Entry Decision Table

The same general definition of a rule is still true. For example, rule 3 of this table reads:

If the value of X is greater than or equal to 10.9, and the value of X is not greater than or equal to 21.6, then assign to Y the value 16.19, print the values of X and Y, and go to 20.

Again the underlined words are implied by the format of the decision table. By comparing rule 3 of this table, with rule 2 of the previous table, it is seen that they are indeed exactly alike.

In fact, the entire table is identical, with only the form of the conditions and actions being different. This format is known as limited entry. In a limited entry condition or action, the entire condition or action is written in the stub and the entry is limited to asserting, negating, or ignoring the existence of a condition or the execution of an action.

Either the extended entry format or the limited entry format or both may be used in any single decision table. It is strictly a matter of convenience. However, any single row of the conditions or actions must be of only one form.

The skeletonized table in Figure 1-4 is directly related to the block diagram in Figure 1-5, which shows schematically the way in which a table is conceptually "executed." In actuality, the processor will eliminate much of the apparent redundency when translating to FORTRAN.

Cond 1	Y	Y	Y		
Cond 2	Y		N		
Cond 3	Y	N	Y		
Action 1	X	X			X
Action 2	X	X	X		X

Figure 1-4. A decision table of the logic of Figure 1-5.

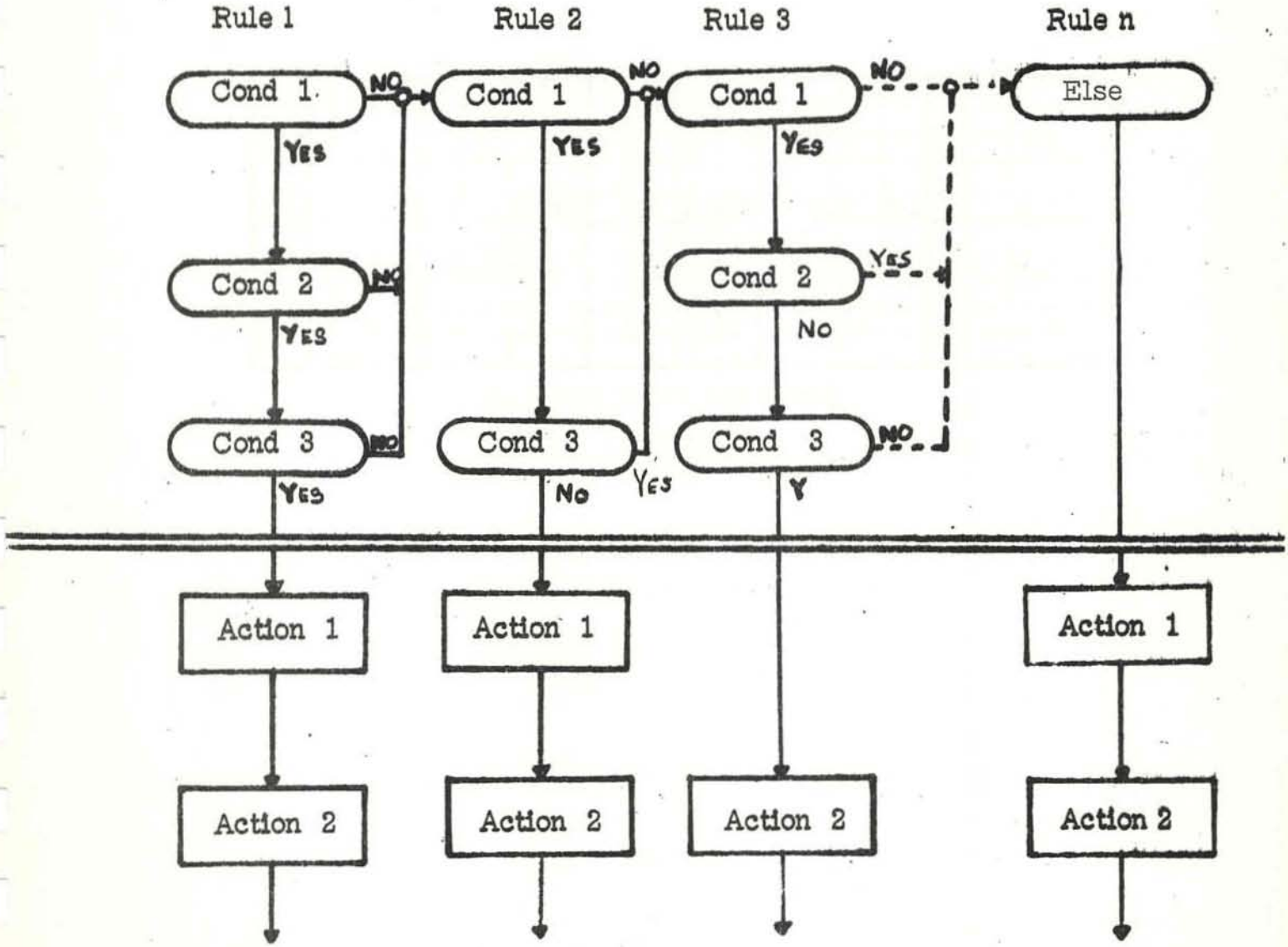


Figure 1-5. Schematic representation of the sequence of tests and actions in executing a table.

At this point the basic ideas and use of decision tables have been presented in rather broad outline. The next step is to learn how these ideas are implemented in the 7090 FORTRAN Decision Table System.

X	LT 1.0	EQ 1.0	GT 1.0
Y =	X * X	1.0	-X*X+4.*X-2.

Situation 1 - Decision Table

CHAPTER 2

Using the 7090 FORTRAN Decision Table System

The basic ideas and concepts of the 7090 FORTRAN Decision Table System are relatively easy to learn. This chapter presents those ideas in the framework of a series of problem situations, each chosen to illustrate some new concepts. At the end of the discussion relating to each situation, there are review questions with answers on the back of the same page. This self-test is included to permit the reader to evaluate his understanding of the subject material before proceeding to new ideas.

This chapter does not fully discuss all the features of the language. For example, all of the statement types are not shown. Complete information on the 7090 FORTRAN DTS will be found in Chapter 4, the reference section of this manual.

For a first example, consider the situation where the value of Y is computed differently depending upon the value of X. The decision table on the opposite page is easily interpreted. In English, the table reads:

> If the value of X is less than 1, then the value of Y is computed by squaring X.
>
> If X is equal to 1, then the value of Y is set to 1.
>
> If X is greater than 1, then Y is set equal to the expression $-X^2 + 4X - 2$.

X	LT 1.0	EQ 1.0	GT 1.0
Y =	X * X	1.0	-X*X+4.*X-2.

Situation 1 - Decision Table

The format of the table allows this easy translation into logical thought. In this table, there is one condition row. This is the row above the horizontal double line, which establishes the desired relationships between X and 1.0, i.e., less than, equal to, or greater than. There is also one action row, below the double horizontal line. The action row states what is to be done for each condition, i.e., set Y equal to X^2, or 1, or $-X^2 + 4X - 2$. The evaluation and assignment performed by the action depends upon the value of X which satisfies one of the conditions previously noted.

This illustrates the basic function of a table. That is, that conditions and actions are always related. If a specific condition, or more generally a set of conditions, is satisfied; then the corresponding action or set of actions is executed. The conditions which are satisfied and the corresponding actions make up a decision rule. A vertical column containing one or more conditions above the horizontal double line and one or more actions below that line is a rule. Thus, rule 1 of this table reads:

If X is less than 1, then Y is set to the value of X * X, where X is a standard FORTRAN variable. Similarly, Y is the variable which will contain the actual computed value of the expression to the right of the =.

X	LT 1.0	EQ 1.0	GT 1.0	
Y =	XXX	1.0	-XXX+4.*XX-2.	

Situation 1 - Decision Table

The action row of the second rule is:

$$Y = 1.0$$

This is readily seen to be a standard FORTRAN statement. However, note the method of writing the statement in the decision table format. The Y is written in the area to the left of the double vertical line. This area is known as the stub area. The area to the right, where the rules are delineated, is the entry area. Every statement of a rule consists of the information for that statement in the stub plus the information for that statement in the entry. Thus, the rule 3 action statement reads:

$$Y = -X * X + 4. * X - 2.$$

This table also introduces three of the condition operators, LT, EQ, and GT. The available condition operators and their abbreviations are:

EQ	EQuals
UN or NE	UNequal or Not Equal
GR or GT	GreateR or Greater Than
LR or LT	LesseR or Less Than
GE	Greater than or Equal to
LE	Lesser than or Equal to
VS	Versus (compared to)

These operators (except VS) are to be understood in the sense of a question to which the answer can only be yes or no. The first condition of Rule 1

X	LT 1.0	EQ 1.0	GT 1.0
Y =	X * X	1.0	-X*X+4.*X-2.

Situation 1 - Decision Table

reads, "Is the value of X less than one?" If the answer is yes, then the action of the first rule is carried out. If the answer is no, then the conditions of the next rule are investigated.

If the reader can correctly answer the following questions, he is ready to proceed; otherwise, some review will help to understand the later material more quickly. Answers are given on the back of this page.

QUESTIONS - Situation 1

1. What is the meaning of rule 2?

2. If the value of X, as previously read, is equal to 1.64, is any rule satisfied? If so, why?

3. Is there any possibility that two of the rules might be executed at a single pass?

Situation 2 is a subroutine defined in the 7090 FORTRAN DTS. This particular table handles the writing of output information for different conditions of value of the variables.

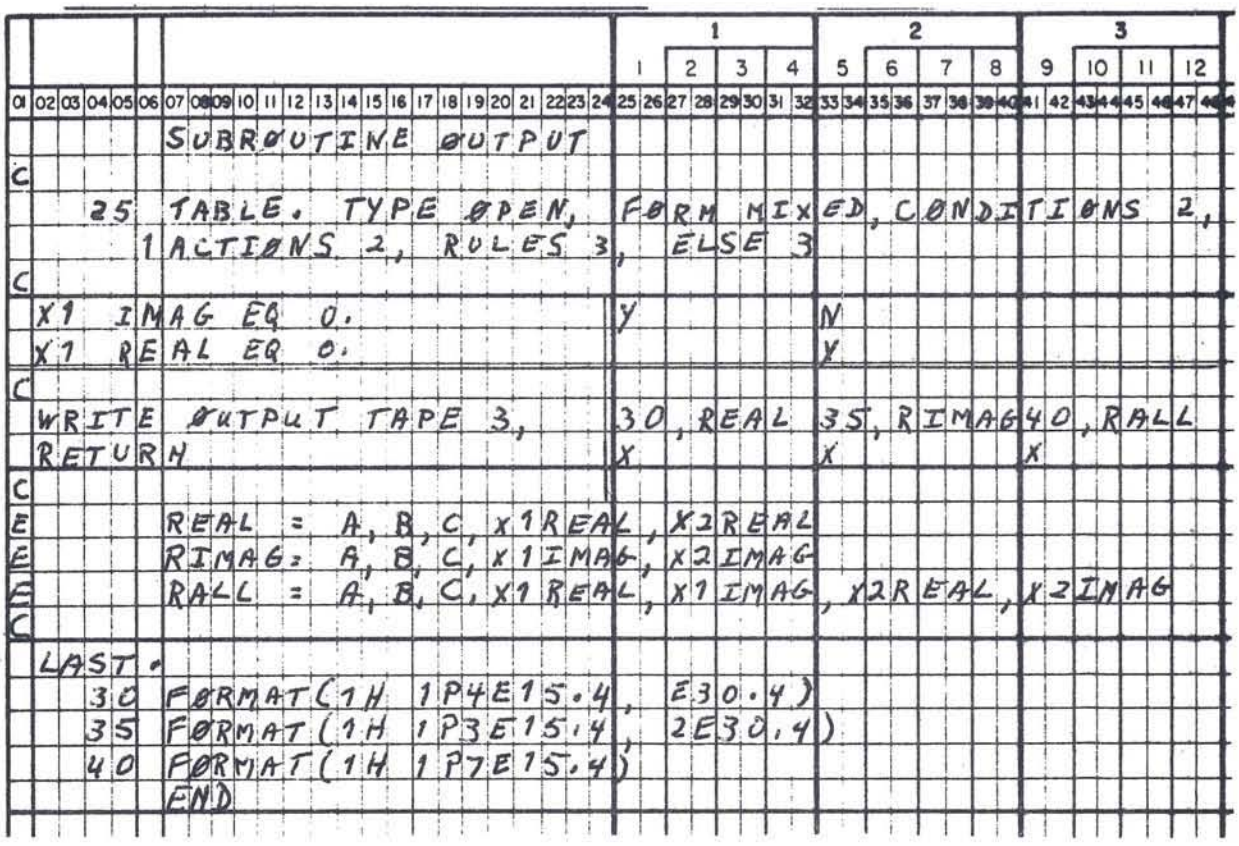
In this situation, the decision table calls upon regular FORTRAN statements, i.e., the FORMAT statements. A complete program may consist of a combination of decision tables and FORTRAN statements. With the use of decision tables, there are some limitations. For example, in rule 1, the action called for is the writing of an output tape. This is readily recognizable as an output statement in FORTRAN.

ANSWERS - Situation 1

1. If X is equal to 1., then assign the value of 1. to Y.

2. Yes, rule 3 is satisfied. The condition of rule 3 asks explicitly if the value of X is greater than 1., and that is all.

3. No. Rules are mutually exclusive. That is, the value of X cannot be both greater than one and one at the same time.



Situation 2 - Decision Table

FORTRAN statement 30, the FORMAT statement, is not allowed within the decision table. Other than in the table itself, it may be placed in any position in the program. In this example, the FORMAT statement follows the table. This exclusion also applies to other non-executable statements such as COMMON, EQUIVALENCE, and DIMENSION.

In the previous situation, it was noted that the FORTRAN statements within the body of the decision table were initiated in the stub and completed in the entry portion of the rows. This type of format is called extended entry, as the statement is begun in the stub and extended into the entry.

In the condition area of this table, a different format is used. In this style, the entire condition is written in the stub and the entry is limited to asserting, ignoring, or negating the condition. Such format is called limited entry. For example, in rule 2, the conditions are read:

If X1 IMAG is not equal to zero, and X1 REAL is equal to zero, then write an output record and return from the subroutine. The Y (yes) and N (no) are used to indicate whether a condition must be true or false to satisfy a rule. The blank means "not pertinent for the rule."

Note the appearance of the entries in the limited form when used in the action rows. In limited entry form, the entire action statement is

```
      SUBRØUTINE ØUTPUT
C
   25 TABLE. TYPE ØPEN, FØRM MIXED, CØNDITIØNS 2,
     1ACTIØNS 2, RULES 3, ELSE 3
C
 X1 IMAG EQ 0.            Y       N
 X1 REAL EQ 0.                    Y
C
 WRITE ØUTPUT TAPE 3,     30,REAL 35,RIMAG40,RALL
 RETURN                   X       X       X
C
E     REAL  = A,B,C,X1REAL,X2REAL
E     RIMAG = A,B,C,X1IMAG,X2IMAG
E     RALL  = A,B,C,X1REAL,X1IMAG,X2REAL,X2IMAG
C
 LAST.
   30 FØRMAT(1H 1P4E15.4,  E30.4)
   35 FØRMAT(1H 1P3E15.4,  2E30.4)
   40 FØRMAT(1H 1P7E15.4)
      END
```

Situation 2 - Decision Table

entered in the stub area, and the rule entries are then marked with an X, to signify the execution of that action for that rule, or a blank (a decimal point in the entry is equivalent to the blank) would indicate that this action is not required in the execution of that rule.

Decision tables may be written in a completely extended format, a completely limited format, or any combination of the two. However, when the combination of forms is used, any single row must be written in a consistent format. This is illustrated in the action rows of this table. The first action row is in the extended form, and the second in limited form.

A single coding form allows for either the extended or limited entry format. Although a table may be as long as required vertically (extending on other pages), it may not use more than 72 positions horizontally. A standard limited entry table consists of a one-character row indicator, a 23-character stub, and twenty-four 2-character entry columns. A standard extended entry table consists of a 1-character row indicator, a 23-character stub, and six 8-character entry columns. In this example, the format is extended entry. The double lines of demarcation which separate the components of the table may be drawn on the coding sheet, but are ignored by the keypunch operator.

This decision table also introduces the concept of the table header. The table header contains information about the decision table which must immediately follow it. Several fields are required in every table header.

```
      SUBRØUTINE ØUTPUT
C
   25 TABLE. TYPE ØPEN, FØRM MIXED,CØNDITIØNS 2,
     1ACTIØNS 2, RULES 3, ELSE 3
C
 X1 IMAG EQ 0.          Y       N
 X1 REAL EQ 0.                  Y
C
 WRITE ØUTPUT TAPE 3,   30,REAL 35,RIMAG40,RALL
 RETURN                 X       X       X
C
E     REAL  = A,B,C,X1REAL,X2REAL
E     RIMAG = A,B,C,X1IMAG,X2IMAG
E     RALL  = A,B,C,X1REAL,X1IMAG,X2REAL,X2IMAG
C
 LAST.
   30 FØRMAT(1H 1P4E15.4, E30.4)
   35 FØRMAT(1H 1P3E15.4, 2E30.4)
   40 FØRMAT(1H 1P7E15.4)
      END
```

Situation 2 - Decision Table

The first required field is the number which is assigned to the decision table within the program. This number is written in the form, as shown:

25 TABLE.

Each table number must be unique and always below 20,000. The table number is used when the table is referenced in the program, either by a decision table or a FORTRAN statement. Note that although the FORTRAN statements external to the table are numbered, those within the table are not numbered by the programmer. This is a matter of reference and control. The statements within the various rules of the table cannot be executed independently of the table. Therefore, the whole table is referenced, rather than any single portion thereof.

The manner of entering and referencing a decision table is the subject of the second field of the table header, Table Type. There are two types of tables, closed and open. The closed table will be discussed in a later illustration. An open table is entered by a GO TO from either a FORTRAN section or another decision table. Control then resides with the open table. If TYPE is not specified in the header, it is assumed to be OPEN.

```
      SUBRØUTINE ØUTPUT
C
   25 TABLE. TYPE ØPEN, FØRM MIXED, CØNDITIØNS 2,
     1ACTIØNS 2, RULES 3, ELSE 3
C
                                   1          2          3
                                   1  2  3  4 5  6  7  8 9  10 11 12
 X1 IMAG EQ 0.                     Y          N
 X1 REAL EQ 0.                                Y
C
 WRITE ØUTPUT TAPE 3,              30,REAL    35,RIMAG   40,RALL
 RETURN                            X          X          X
C
E     REAL  = A,B,C,X1REAL,X2REAL
E     RIMAG = A,B,C,X1IMAG,X2IMAG
E     RALL  = A,B,C,X1REAL,X1IMAG,X2REAL,X2IMAG
C
 LAST.
   30 FØRMAT(1H 1P4E15.4, E30.4)
   35 FØRMAT(1H 1P3E15.4, 2E30.4)
   40 FØRMAT(1H 1P7E15.4)
      END
```

Situation 2 - Decision Table

The third field of the table header is the field, FORM. This alludes to the limited or mixed format as previously explained. In this situation, the table header:

25 TABLE, TYPE OPEN, FORM MIXED, etc.

indiates that open table number 25 has its information contained in the 1-position row indicator, the 23-position stub, and the six 8-position rule columns. Mixed form uses the extended entry format but in fact may also contain individual statements in limited entry format. If FORM is not specified in the header, it is assumed to be MIXED.

The first two rules of this table do not exhaust all the combinations of the variables in the conditions. In this table, provisions for such possibilities are made with a special designation called ELSE. This rule specifies the actions to be performed if none of the other rules can be executed. In this example, it is reasonable to expect that occasionally neither of the first two rules will be satisfied, so that the else rule is more or less a normal occurrence. In other tables, the failure to satisfy any of the rules might represent an error in the logic of the table or in the data. In such a case, an error table would be indicated rather than an ELSE rule. The ERROR entry in the header would specify another table or FORTRAN statement to be considered when none of the rules are satisfied. The table header must specify either ELSE or ERROR.

This table header contains the specification of a rule number for ELSE, i.e., rule 3. In this case, because the ELSE rule takes care

```
           SUBRØUTINE ØUTPUT
C
     25    TABLE. TYPE ØPEN,     FØRM MIXED, CØNDITIØNS 2,
          1ACTIØNS 2, RULES 3,   ELSE 3
C
 X1 IMAG EQ 0.           Y       N
 X1 REAL EQ 0.                   Y
C
 WRITE ØUTPUT TAPE 3,    30,REAL 35,RIMAG40,RALL
 RETURN                  X       X       X
C
E          REAL  = A,B,C,X1REAL,X2REAL
E          RIMAG = A,B,C,X1IMAG,X2IMAG
E          RALL  = A,B,C,X1REAL,X1IMAG,X2REAL,X2IMAG
C
 LAST.
     30    FØRMAT(1H 1P4E15.4,   E30.4)
     35    FØRMAT(1H 1P3E15.4,   2E30.4)
     40    FØRMAT(1H 1P7E15.4)
           END
```

Situation 2 - Decision Table

of all other possibilities, the entry allowed in the table header for the number of an error table would not be used. The explicit table header details are given in Chapter 4 of the manual.

There are three header fields which are always required. Those specifying the dimensions of the table, i.e., number of rules, number of condition rows, and number of action rows. In an unconditional table (one rule), only the number of actions is required. In addition to indicating the start of a decision table with a header, the end must be specified by placing the notation (LAST.) following the table starting in or beyond Column 2.

In the 7090 FORTRAN DTS, subroutines, such as is illustrated in this situation, may be written in either FORTRAN or the decision table format. Since this is a subroutine, it would be expected that the subroutine is called by some FORTRAN statement or decision table action statement elsewhere in the program. The subroutine can consist of one or more tables and FORTRAN statements. Likewise, it is expected that control will return to the calling table or statement when any rule of this table is executed. For example, the last action of every rule is RETURN. This is a decision table statement which calls for return of control to the calling statement.

This illustrates one of the basic concepts of a decision table. That is, that every rule of every table should transfer control to some other place in the program. The RETURN in this table is the control transfer operator for a subroutine. Other control operators will be discussed in later examples.

```
      SUBRØUTINE ØUTPUT
C
   25 TABLE. TYPE ØPEN, FØRM MIXED, CØNDITIØNS 2,
     1ACTIØNS 2, RULES 3, ELSE 3
C
 X1 IMAG EQ 0.          Y       N
 X1 REAL EQ 0.                  Y
C
 WRITE ØUTPUT TAPE 3,   30, REAL 35, RIMAG 40, RALL
 RETURN                 X       X       X
C
E     REAL  =  A, B, C, X1REAL, X2REAL
E     RIMAG =  A, B, C, X1IMAG, X2IMAG
E     RALL  =  A, B, C, X1REAL, X1IMAG, X2REAL, X2IMAG
C
 LAST.
   30 FØRMAT(1H 1P4E15.4, E30.4)
   35 FØRMAT(1H 1P3E15.4, 2E30.4)
   40 FØRMAT(1H 1P7E15.4)
      END
```

Situation 2 - Decision Table

A final concept introduced by this example is the use of the defined expression. The statements following the table show examples of the use of this tool. Rule 1, action 1, reads:

WRITE OUTPUT TAPE 3, 30, REAL

where REAL is defined to be the list, by the following statement:

REAL = A, B, C, X1 REAL, X2 REAL.

A defined expression can be any legal FORTRAN statement or string of characters which make up part of a statement. It is a method of providing a synonym for part of a statement. An expression name may be used only in the entry part of the table. As the table is being processed, the expression name will be replaced by the characters in its definition. The expression is only defined locally in the table where used. A different definition can be used with the same name in another decision table within the system.

Only one other point should be noted at this time. That is, that comments may be included anywhere in either the FORTRAN statements or in the decision tables. They are identified in the usual manner by placing a C in column 1 of the coding form.

QUESTIONS - Situation 2

1. Is it possible to use an EQUIVALENCE statement within a decision table to equate two defined expressions?
2. Are comments allowed in the table?
3. What is the difference between extended and limited entry?

JOB _____ SECT. _____ PAGE ____ OF ____

NAME _____ DATE _____

```
      20 TABLE. TYPE CLØSED, FØRM SPC(12,32,52,72), RØRDER(2/50,1/30,3/20),
       1 RULES 3, CØNDITIØNS 1, ACTIØNS 5, ERRØR 999
C
          DISC    LT 0.          GT 0.          EQ 0.
C
     RØØT =                     SQRTF(DJSC)
     X1 REAL =    -B/(2.*A      (-B+RØØT)/(2.*A)      -B/(2.*A)
     X2 REAL =    X1REAL        (-B-RØØT)/(2.*A)      X1REAL
     X1 IMAG =    SQRTF(-DISC)/(2.*A)  0.0            0.0
     X2 TMAG =    -X1IMAG       0.0                   0.0
C
     LAST.
```

4. Could the first action row be written in limited form so that the whole table is in limited form?

5. What fields are required in the table header?

Situation 3 involves a decision table used in the evaluation of the roots of a quadratic equation. This example introduces several new concepts of table structure, format, and usage.

The first rule of this table reads:

If the discriminant (DISC) is less than zero, then X1 REAL is assigned the value of -B/2A, and X2 REAL is assigned the value of X1 REAL, and X1 IMAG is assigned the value of the square root of minus the discriminant divided by 2 times A, and X2 IMAG is assigned the value of minus X1 IMAG.

The underlined words are understood in the table format. Of course, the conditional statement, If....., requires the affirmative answer for the execution of the rule, as previously discussed. The action statements are all connected by the AND. In addition, they will be executed in the order given.

ANSWERS - Situation 2

1. No, for two reasons. First, the EQUIVALENCE statement is not allowed within the decision table. Second, a defined expression is only defined locally and must be redefined in each table in which it is used.

2. Yes, comments are allowed anywhere in 7090 FORTRAN DTS. They are indicated by the usual C in column 1 of of the coding form.

3. In extended entry, the statement is begun in the stub and extended into the entry. In limited entry, the statement is written in the stub, and the entries are limited to Y, N, X or blank.

4. Yes. This would involve the writing of three statements in the stub and placing an X in the proper rule entries. The header ACTIONS field must be changed to ACTIONS 4.

5. The number of rules, number of conditions, number of actions, and either the ELSE or the ERROR designation.

In the previous example, an open table was discussed. A closed table is shown in this example.

As a FORTRAN subroutine is always entered by the action operator CALL, the closed table is always entered by the action operator PERFORM, followed by the table number. In this example, it may be assumed that in some previous table of the program, there is the action statement:

PERFORM 20

Upon execution of this statement, control is relinquished to the closed table. One of the rules of the table is executed and then control returns to the action statement following the calling statement.

For example, the table is entered, as previously noted, by a PERFORM 20 and the value of the DISC is equal to zero, X1 REAL is assigned the value of -B/2A, X2 REAL is assigned the same value, and X1 IMAG and X2 IMAG are set to zero. Program control is then transferred back to the action statement which succeeds the original PERFORM 20. PERFORM is a legal statement only within decision tables.

In the FORM field, in the table header, a different notation is illustrated. In previous examples, the limited format and the extended format have been discussed. Here is an example of the third format of the 7090 FORTRAN DTS. In this format, the special format, a decision table

JOB _____ SECT. _____ PAGE ____ OF ____

NAME _____ DATE _____

```
      20 TABLE. TYPE CLOSED, FORM SPC(12,32,52,72), RORDER(2/50,1/30,3/20),
       1   RULES 3, CONDITIONS 1, ACTIONS 5, ERROR 999
C
                DISC    LT 0.                   GT 0.                   EQ 0.
C
      ROOT      =                               SQRTF(DISC)
      X1 REAL   =       -B/(2.*A                (-B+ROOT)/(2.*A)        -B/(2.*A)
      X2 REAL   =       X1REAL                  (-B-ROOT)/(2.*A)        X1REAL
      X1 IMAG   =       SQRTF(-DISC)/(2.*A)     0.0                     0.0
      X2 IMAG   =       -X1IMAG                 0.0                     0.0
C
      LAST.
```

may be written in such a manner that it is designed to fit other requirements of the programmer. He can essentially design his own table format. When this format is used, it must be so noted in the table header. In addition, the field requires the insertion of special information. In this example,

FORM SPC (12, 32, 52, 72)

indicates that the special format is used, and the rule boundaries are defined. The number 12 indicates that the last column of the stub ends in column 12. The numbers 32, 52, and 72 allude to the last columns of the three rules. The special format allows the use of any size fields and any number of rules, with the following restrictions:

a. There must be a 1-character row indicator.

b. There must be a stub of at least 2 characters.

c. Each entry must be at least 2 characters wide.

The table header also contains one of the optional fields, that is,

R ORDER (2/50, 1/30, 3/20)

R ORDER alludes to the order in which the programmer wishes the rules in the object program to be considered. The numbers to the left of the slash indicate positionally the order of testing the conditions, in this case, the preferred order is 2, 1, 3 and rules would be arranged in that order for processing. This technique allows the programmer to arrange his rules in any order as they occur to him, and later to have them reordered in the light of more efficient processing.

JOB _____ SECT. _____ PAGE ____ OF ____

NAME _____ DATE _____

	1	2	3	4	5	6
	1 2 3 4	5 6 7 8	9 10 11 12	13 14 15 16	17 18 19 20	21 22 23 24
20	TABLE, TYPE CLOSED, FORM SPC(12,32,52,72), RORDER(2/50,1/30,3/20),					
	1 RULES 3, CONDITIONS 1, ACTIONS 5, ERROR 999					
	DISC LT 0.	GT 0.		EQ 0.		
	ROOT =	SQRTF(DISC)				
	X1 REAL = -B/(2.*A	(-B+ROOT)/(2.*A)		-B/(2.*A)		
	X2 REAL = X1REAL	(-B-ROOT)/(2.*A)		X1REAL		
	X1 IMAG = SQRTF(-DISC)/(2.*A)	0.0		0.0		
	X2 IMAG = -X1IMAG	0.0		0.0		
C						
	LAST.					

The number below the slash, within the parenthesis, indicates the frequency with which the rules of the table are expected to have their conditions satisfied. For example, rule 1 is executed 30% of the time and rule 2 is executed about 50% of the time, while rule 3 will only be executed about 20% of the time. If the frequency of success is not known, or not desired, they may be omitted (together with the /) and the rule numbers alone entered for ordering.

QUESTIONS - Situation 3

1. What is the difference between a closed table and an open table?
2. What is the special format?
3. What follows the special format in the table header?
4. What is R ORDER?
5. Must the rule frequency always be included in the R ORDER statement?

Situation 4 is an example of a complete program written in the 7090 FORTRAN DTS. The object of the program is the reading of the coefficients of quadratic equations, the solution of the quadratic equations, and the writing of an output report with the coefficients and the roots. This program incorporates both the subroutine discussed in situation 2, and the closed table discussed in situation 3.

ANSWERS - Situation 3

1. An open table is entered by a GO TO type statement and in turn indicates where to go next. A closed table is entered by a PERFORM statement with control reverting to the original table when the actions for a rule are completed.

2. A free form, up to the programmer's discretion. However, it must have a 1-character row indicator, and every entry must be at least two columns wide.

3. The indicator of the last column of the stub and of each rule of the table, all enclosed within parentheses.

4. An optional entry in the table header which cites the preferred order of rules, and the relative frequency of satisfying the rules.

5. No.

This example serves to illustrate more clearly how FORTRAN and decision tables complement each other in the 7090 FORTRAN DTS.

Most of the concepts and ideas illustrated in this example relate to information previously given. One new concept is the use of the action statement GO TO, shown in the last action of table 15. For the first and third rules, the transfer of control is right back to the beginning of the table, in rule 2 it is to the FORTRAN input statement, 12. In the last rule, the transfer is to 45. This is the statement number of the first FORTRAN statement for the end of job procedure.

Table 15 also indicates the action statements used to enter closed tables and subroutines. The closed table illustrated in situation 3, table 20, is entered by means of the statement:

PERFORM 20 (action 3, rule 2)

The subroutine of situation 2, table 25, is entered by means of the statement:

CALL OUTPUT (also in rule 2)

The complete transfer of control in this program is more evident when rule 2 is read in English:

If A is not equal to zero, and NPAGE is not equal to 1, and if the number of lines (on the output report) is less than 40, then set DISC = B^2 -4AC, PERFORM 20 (execute closed table number 20 satisfy one of the conditions of that table, assume rule 3, which sets

JOB ______ SECT. ______ PAGE ____ OF ____

NAME ______ DATE ______

```
C     PRØGRAM FØR THE SØLUTIØN ØF QUADRATIC EQUATIØNS
   50 FØRMAT(115H1                A                   B                   C                X1
     1REAL          X1 IMAG                X2 REAL             X2 IMAG             PAGE I4//)
   10 FØRMAT(3F10.5)
C
      NPAGE = 1
      LINES = 0
   12 READ INPUT TAPE 2,10,A,B,C
C
   15 TABLE. TYPE ØPEN, FØRM SPECIAL(32,42,52,62,72), RULES 4,
     1CØNDITIØNS 3, ACTIØNS 7, ELSE 4
C
      A EQ 0.                         N                   N                   N         Y
      NPAGE EQ 1                      Y                   N                   N
      LINES LT 40                                         Y                   N
C
 WRITE ØUTPUT TAPE 3, 5, NPAGE        X                                       X
 DISC = B*B - 4. *A*C                                     X
 PERFØRM 20                                               X
 CALL ØUTPUT                                              X
 LINES =                                                  LINES + 1           0
 NPAGE =                              NPAGE +1                                NPAGE + 1
 GØ TØ                                15                  12                  15        45
 LAST.
   20 TABLE.TYPE CLØSED, FØRM SPC(12,32,52,72),R ØRDER(2/50,1/30,3/20)
     1 RULES 3, CØND 1, ACT 5, ERRØR 47
      DISC        LT 0.                GT 0.                                  EQ 0.
C
 RØØT =                               SQRT(DISC)
 X1 REAL =        -B/(2.*A)           (-B+RØØT)/(2.*A)                        -B/(2.*A)
 X2 REAL =        X1REAL              (-B-RØØT)/(2.*A)                        X1REAL
 X1 IMAG =        SQRTF(-DISC)/(2.*A) 0.0                                     0.0
 X2 IMAG =        -X1IMAG             0.0                                     0.0
 LAST.
```

X1 REAL = -B/2A

X2 REAL - X1 REAL

X1 IMAG = O.

X2 IMAG = O.

and then return to rule 2 of table 15); CALL the output subroutine (enter the output subroutine, under the conditions shown, rule 1 is satisfied, the report line is written for the listing, and control is returned to rule 2 of table 15); set LINES = LINES + 1, and go back to the READ statement (12).

Table 15 also illustrates a second example of the special format as evidenced in the table header. The last stub column is 32, rule 1 finishes at column 42, rule 2 at column 52, etc. The remainder of the program is fairly self-evident.

JOB ______ SECT. ______ PAGE ____ OF ____

NAME ______ DATE ______

```
C
C
C
      COMMØN A,B,C,X1REAL,X2REAL,X1IMAG,X2IMAG
C
   45 END FILE 3
      REWIND 2
      REWIND 3
      PRINT 50
   47 STØP
C
   50 FØRMAT(14H  JØB FINISHED)
C
      END
      SUBRØUTINE ØUTPUT
      CØMMØN A,B,C,X1REAL,X2REAL,X1IMAG,X2IMAG
   25 TABLE. TYPE ØPEN, FØRM MIXED,CØNDITIØNS 2,
     1ACTIØNS 2, RULES 3, ELSE 3
C
 X1 IMAG EQ 0.                              Y         N
 X1 REAL EQ 0.                                        Y
C
 WRITE ØUTPUT TAPE 3,                       30, REAL  35, RIMAG40, RALL
 RETURN                                     X         X         X
C
E     REAL  = A,B,C,X1REAL,X2REAL
E     RIMAG = A,B,C,X1IMAG,X2IMAG
E     RALL  = A,B,C,X1REAL,X1IMAG,X2REAL,X2IMAG
C
 LAST.
   30 FØRMAT(1H 1P4E15.4,  E30.4)
   35 FØRMAT(1H 1P3E15.4,  2E30.4)
   40 FØRMAT(1H 1P7E15.4)
      END
```

Situation 4 (continued)

CHAPTER 3

7090 FORTRAN Decision Table System Reference Section

The 7090 FORTRAN Decision Table System has been developed to evaluate the use of decision tables in conjunction with FORTRAN in the solution of scientific and engineering problems. The language used with the decision tables coincides very closely with FORTRAN except that decision logic statements are written to take full advantage of the decision table format. The 7090 FORTRAN DTS provides the means to replace all IF statements with decision tables.

The initial version of 7090 FORTRAN DTS has been designed for the 709/90 family of FORTRAN dialects. Little consideration has been given to the restrictions or differences imposed by other FORTRAN dialects. The system has been designed to make use of the current FORTRAN monitor system for the 709/90 (version II) and requires the same machine configuration as that system.

The character set for the 7090 FORTRAN DTS is the FORTRAN character set denoted by IBM as set "H." This set includes the apostrophe in place of the redundant 8-4 minus.

When a program is written in 7090 FORTRAN DTS, the programmer may assume either FORTRAN or 7090 FORTRAN DTS to be dominant. The combined language operates either entirely as FORTRAN, entirely

as 7090 FORTRAN DTS, or as any combination of the two at the discretion of the programmer.

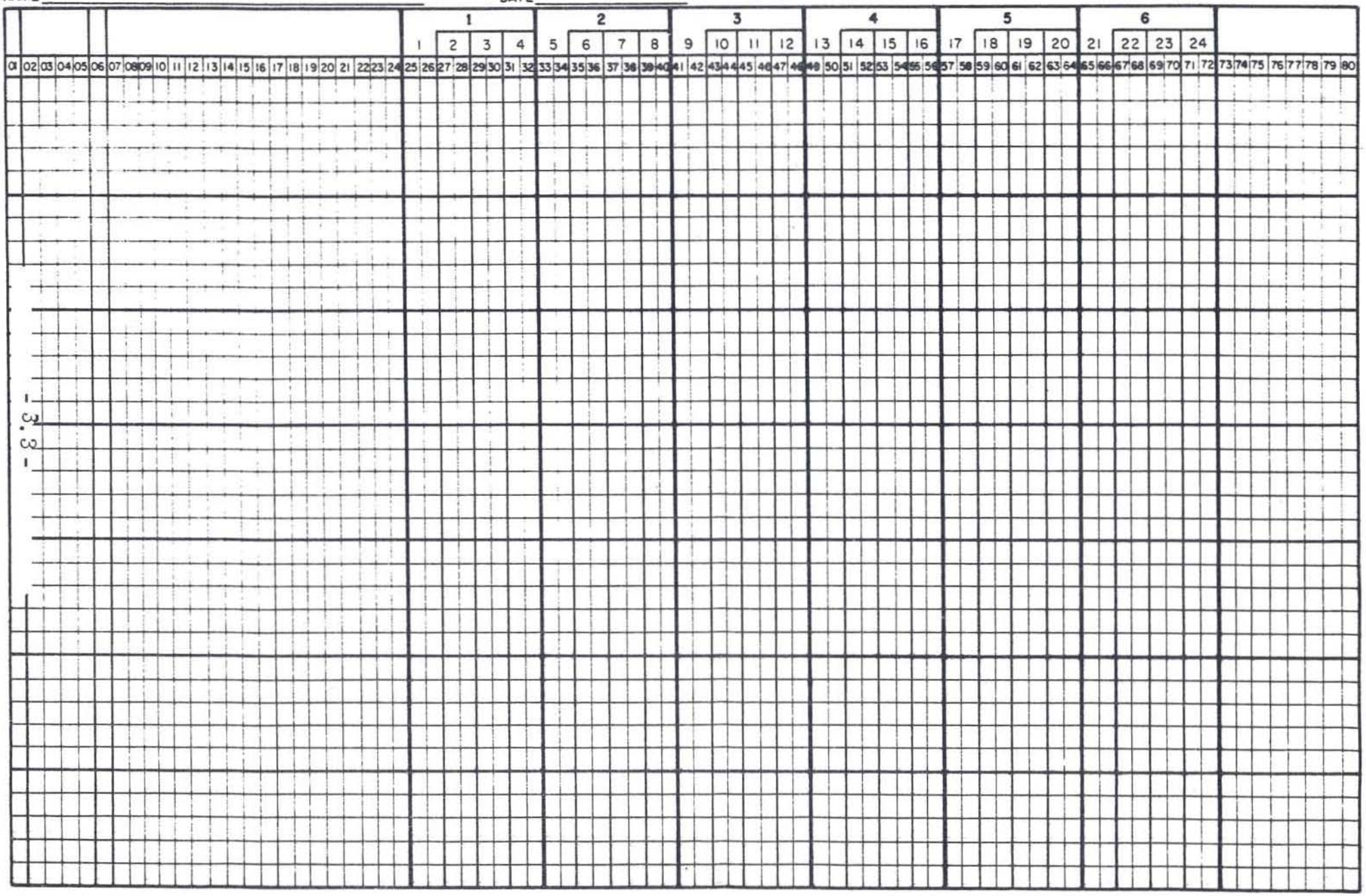
Table Characteristics

The following information pertains to decision tables in the 7090 FORTRAN DTS:

- The table structure is vertical with conditions and actions listed on successive lines. Decision rules are read from top to bottom.
- Decision tables are always entered at their top left. There is no other access. Tables may be open (in line) or closed (callable).
- There is never more than one successful rule per pass through a table. Therefore, each rule has an exclusive set of conditions.
- Since each rule has an exclusive set of conditions the analyst or processor may rearrange rules without affecting the table logic.
- Decision tables without conditions are allowed. These unconditional tables signify the execution of a stipulated single set of actions whenever the table is entered.
- If the conditions of no rule are satisfied and an ELSE rule is not specified, an ERROR table must be specified.
- An open table may be entered by a GO TO statement, a closed table must be entered by a PERFORM statement.
- An open table may be exited from by means of a GO TO statement or by the specification of a next table in the Table Header. A closed table automatically returns to the table which called it.
- A TALLY facility is included in the language to allow automatic tracing of the path of control through the program tables.
- The decision table preprocessor provides a listing so that the programmer is encouraged to work with the latest compiled version of his program.
- All table actions are performed in the listed order.

JOB _____ SECT. _____ PAGE ____ OF ____

NAME _____ DATE _____



133-1

Figure 3 - 1

Table Format

A 7090 FORTRAN DTS program is made up of a series of FORTRAN statements intermixed with decision tables. The statements of the decision tables are also FORTRAN statements with modifications that will be discussed later in this chapter.

A sample of the coding form used for the insertion of information is shown in figure 3-1. All information for decision tables must appear in columns 1 through 72. As usual, columns 73 through 80 are not processed.

Comment cards may be used freely within the tables. A C in column 1 of the coding form indicates that the statement is a comment. The card is not processed by the table processor, except for listing.

FORTRAN statements that are separate from the decision tables within the 7090 FORTRAN DTS system are written in the usual manner.

Each decision table within the 7090 FORTRAN DTS has the following general form:

Condition stub	Condition entries
Action stub	Action entries

The double lines indicate logical divisions of the table. They may be drawn on the coding form, but are ignored by the keypunch operator.

The input format of the programmed tables may be free form; a standard limited entry form and extended entry form is also defined. For both of these forms, the table may continue vertically on additional pages; however, it may not be wider than 72 horizontal positions.

The standard limited entry table consists of a 1-character row indicator (column 1), a 23-character stub (columns 2 through 24) and twenty-four 2-character entry columns (25 through 72). The rules are numbered 1 through 24 just above the card column numbers (see Figure 3. 1). A standard extended entry table consists of a 1-character row indicator, a 23-character stub, and six 8-character entry columns, numbers 1 through 6 in Figure 3. 1.

The programmer uses either of these standard forms, or a special form of his own choosing. A special form table must contain the 1-character row indicator (column 1) but the rest of the table format is divided as the programmer wishes. The only definite requirement is that the stub and each entry must be at least 2 characters in width. For example, a programmer might require a 1-character row indicator, a 27-character stub, and five entry columns consisting of 2 characters, 2 characters, 10 characters, 10 characters, and 20 characters. For clarity, the programmer should draw in with heavy lines the vertical rule dividers he has chosen.

Row indicator

The 1-character row indicator within the decision tables is used for three purposes:

(1) A C in this column indicates that the card is a comment.

(2) A number in this column indicates that this is a continuation of the stub to another card. The continuation is indicated by placing a 1 in column 1 of the first continuation card, a 2 in column 1 of the second continuation card, ... , up to a maximum of nine, indicating a total of ten cards. The continuation numbers must start with 1, and must be consecutive integers.

(3) The letter E in this column indicates that this card is an expression definition. The formation of an expression is described later in this chapter.

Except for these three uses, the row indicator column is left blank within the decision table. The stubs for all tables begin in column 2.

Table Header

Each table must be preceded by a table header statement. This statement provides information to the table processor. The general form of this statement is:

nnnnn TABLE. field 1, field 2, ... , field m.

The nnnnn is the table number. It is always unique and less than 20,000.

Following this table number will be a number of other fields. Several are required; the remainder are optional depending upon program needs.

Fields (*indicates required fields)

TYPE t where t is either OPEN or CLOSED. (If not specified, OPEN is assumed.)

FORM f where f is LIM or LIMITED if the table is the standard limited entry form.

where f is MIX or MIXED if the table is the standard extended entry form.

where f is SPC or SPECIAL if an optional format is defined. If the special format is indicated, the FORM entry must be followed by a set of parenthesis enclosing a description of the desired format of the input card:

FORM SPC (c_1, c_2, c_3, ... , c_n)

Each c indicates the last column of a field on the card. c_1 indicates the last column of the stub, c_2 indicates the last column of rule 1, etc. For example,

22 TABLE. TYPE OPEN, FORM SPC (28, 30, 32, 42, 52, 72)

(If not specified, MIXED is assumed.)

NEXT N — where n is the number of the next table or FORTRAN statement. This field can be used in place of a GO TO statement in the various rules.

* RULES R — where r is the number of the rules in the table. (Not required for unconditional tables.)

TALLY — which indicates the TALLY feature is to be exercised whenever the table is entered. The TALLY feature has been included to provide a trace through the tables during debugging. The table number and rule satisfied will be printed each time the table is executed. If TALLY is punched on a continuation card, it may be easily removed after the program has been debugged.

* COND c or CONDITIONS c

where c is the number of conditions in the table (Not required for unconditional tables.)

* ACT a or ACTIONS a

where a is the number of actions in the table.

* ELSE r — where r is the number of the rule whose actions are to be exercised in the event that the conditions for no other rule are satisfied.

* or

ERROR n — where n is the number of the table or statement to which transfer is made in the event that the conditions of no rule are satisfied and an ELSE rule is not specified. The ERROR destination should not be specified if there is an ELSE rule.

R ORDER (r_1, r_2, ... , r_n)

where r_1 through r_n are the numbers of the rules in the table arranged in the order in which they are to be considered. If this information is not specified in the table header, the rules are considered in the order given in the table. Rules not mentioned in this field will be arranged in their given order after the mentioned rules.

R ORDER (r_1/f_1, r_2/f_2, ... , r_n/f_n)

where f_1 through f_n are the relative frequencies of success for each rule. Each frequency must be specified as an integer. This field also specifies rule order, as above.

Some examples of the uses of these optional fields are given below:

```
98 TABLE, TYPE CLOSED, FORM MIX, R ORDER (3/40, 1/40, 4/10, 2/10),
   RULES 4, CONDITIONS 3, ACTIONS 10
```

(This table is #98, a closed table of mixed form, with four rules that are to be rearranged in the order, 3, 1, 4, 2. Rules 3 and 1 have a relative

frequency of success of 40% each, while rules 4 and 2 have a relative frequency of success of 10% each. There are 3 condition rows and 10 actions.

18799 TABLE, TYPE OPEN, FORM MIX, RULES 4, COND 3, ACTIONS 7

(This table is #18799, an open table of mixed form, with four rules. There are three conditions and seven actions in the table.)

7090 FORTRAN DTS OPERANDS

The operands in decision tables within the 7090 FORTRAN DTS are FORTRAN expressions. One other operand of the 7090 FORTRAN DTS is the defined expression name. The general form of the definition of this operand is:

E name = expression,

where the name is any legal FORTRAN variable name and the expression is any legal FORTRAN expression or character string.

```
E OUTLST = 3, K, S(N), P(N)
E ROOT = SQRTF (-B + B**2/2*A**3)
```

Whenever any such name is used in the entry portion of the table, it will be replaced automatically by the expression so defined. The definition is local and holds only for the table in which it is used.

On the coding form, expression definitions are indicated by placing E in the row indicator (column 1) as previously noted.

7090 FORTRAN DTS Statements

The formation of statements for 7090 FORTRAN DTS is similar to the formation of statements in FORTRAN. The following exceptions are noted:

- All FORTRAN statement numbers and decision table numbers must be less than 20,000. All higher numbers are reserved by the system.
- No fixed point variables of the form NXXXXX or floating point variables of the form OXXXXX, where XXXXX is five numeric characters, may be used. These are reserved by the system.
- Statements within a decision table cannot be numbered.
- Certain standard FORTRAN statements are excluded from action statements of the decision tables. These exceptions are enumerated in the section of this chapter titled ACTIONS.

Table Conditions

Condition statements are analogous to IF statements of FORTRAN. Each condition statement consists of a stub and an entry. Each condition statement is a two operand test. That is:

operand 1 operator operand 2.

The first operand is always in the stub. The remainder of the statement may be split anywhere and placed in the stub and entry, or the entire statement placed in the stub. The latter condition is the limited entry format, where the entire statement is written in the stub and the entry is limited to asserting the condition, negating the condition, or ignoring the condition.

For example:

where operand 1 is B

operand 2 is A*C*LOGF(A*C**2)

and the operator is equals (EQ)

The statement is

B EQ A*C*LOGF(A*C**2),

and can be written in the condition area as:

stub	entry	
B EQ A*C*LOGF(A*C**2)	Y	(limited entry)
B EQ A*C*	LOGF (A*C**2)	(mixed entry)
B	EQ A*C*LOGF(A*C**2)	(mixed entry)
B EQ	A*C*LOGF(A*C**2)	(mixed entry)

Condition Operands

Each condition operand is an expression in the FORTRAN sense of the word. That is, an operand is a variable, a subscripted variable, a constant, a function usage, an expression name, or any combination of these linked by the appropriate FORTRAN operators. For example:

ROOT	(variable)
31.675	(constant)
SQRTF (B**2)	(function usage)

Conditional Operators

The condition operators in the 7090 FORTRAN DTS are shown below:

Operator	Meaning
EQ	EQuals
UN or NE	UNequal or Not Equal
LR or LT	LesseR or Less Than
LE	Lesser than or Equal to
GR or GT	GreateR or Greater Than
GE	Greater than or Equal to
VS	VersuS (compared to)

stub	entry	
ROOT EQ 0.0	Y	(limited entry)
ROOT	GR 50.0	(mixed entry)
A LE	B	(mixed entry)
A VS B	GR	(limited entry)

Table Actions

Action statements provide for all value assignment, sequence control, and input/output operations in decision tables. 7090 FORTRAN DTS action statements include all FORTRAN statements except as noted below:

a. No IF statements

b. No non-executable statements: DIMENSION, FORMAT, COMMON, FREQUENCY, or EQUIVALENCE.

c. No function definition

d. No subroutine definition

Certain additional operators will be introduced later in this section.

The programmer may use either the limited entry format or the extended entry format for all actions. In the limited format the entire action is placed in the action stub, and the notation of X, blank, or . is placed in the action entry to indicate the requirement or nonrequirement of that action for a particular rule.

In the extended format, the action is begun in the stub and completed in the entry. The statement may be split at any location desired. The two areas are assumed to be contiguous for each individual rule.

Arithmetic Statements

The nominal value assignment statement is identical to the FORTRAN arithmetic statement:

operand 1 = operand 2,

where operand 1 is a variable or subscripted variable and operand 2 is an expression in the FORTRAN sense, or a defined expression as previously explained. For example:

P = 7.109

R = A*B/(A-B)

Q = DISC (where DISC is a defined expression)

MOVE

An additional value assignment operator is MOVE... TO .... The form is:

MOVE operand 2 TO operand 1

This operator is identical to the FORTRAN arithmetic statement:

operand 1 = operand 2

The allowable operands for the MOVE... TO ... action operator are the same as the FORTRAN arithmetic statement.

MOVE 7.109 TO	B	C	D
MOVE A*B/(A-B) TO	R1	R2	R2
MOVE VAL TO TEMP1	X		X

ASSIGN

The third value assignment statement is the FORTRAN

ASSIGN i to n

and is used in an identical manner.

Action Sequence Control

DO

The first action operator of the sequence control operators is the FORTRAN DO, expressed in the following form:

DO $i = m_1, m_2, m_3$

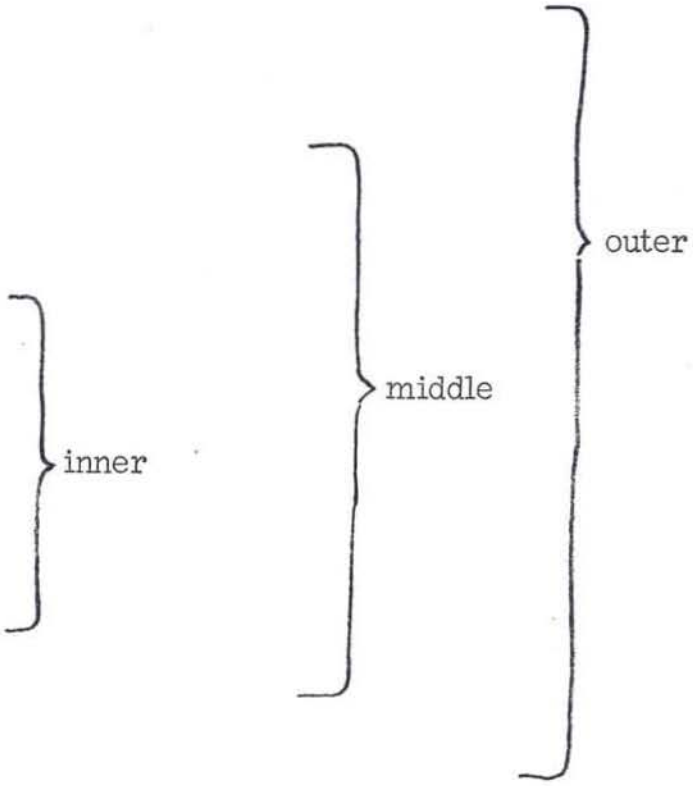
This action operator must always have in its sequence of actions, the action operator CONTINUE. It is the last statement in the range of the DO and delimits the DO.

with the usual FORTRAN rules. Unlike FORTRAN no statement number is used with the DO operator used within tables.

```
DO J = 1, 20, 5
-----------
-----------
CONTINUE
```

Nested DO's are permissable provided each DO has its own separate CONTINUE

```
DO J = 1, 200
--------------
DO I = 1, KK, 2
---------------
DO K = 90, 100, 1
----------------
----------------
----------------
CONTINUE
CONTINUE
CONTINUE
```



DO and CONTINUE are always paired within a table.

The DO statement used as an action may be in either limited or extended entry format. If the DO statement action entry is blank, neither the DO statement nor any statement within its range is executed. If the DO statement action entry is not blank, then the statement is performed but the statements within the range of the DO are performed or not dependent upon the appearance of their action entries.

This is illustrated in Figure 3 - 2. In this table, (of which only the action statements are shown) rule 1 does not allow for the performance of any of the DO statements as the entry is blank for that rule. In rule 2, the first DO (the outer DO) is executed, but only two of eight statements within its range are performed. None of these are DO statements (no nested DO's).

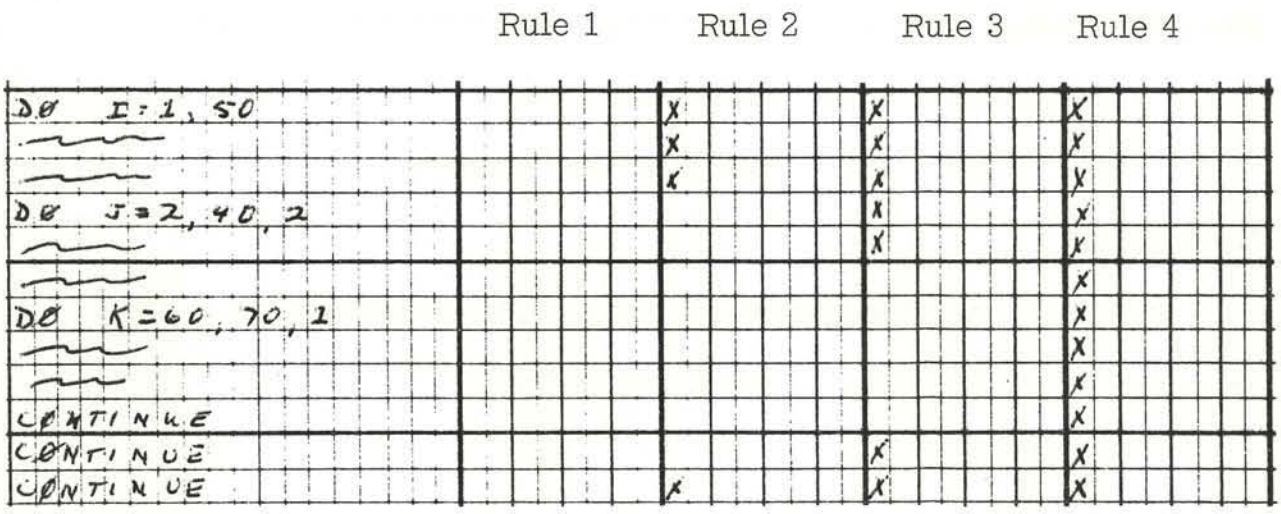


Figure 3-2. DO Sequence Control

The same general rules apply for nested DO's and are illustrated in rules 3 and 4 of Figure 3 - 2. In rule 3, all the nested DO's are executed, but only one statement within the range of each is performed. In rule 4, all the DO's and the statements within their ranges are performed.

GO TO

A second sequence control action operator is the GO TO, expressed in the following form:

GO TO n

In this case, n is a decision table number (of an open table only, see PERFORM) or a FORTRAN statement number. This action statement can be used when automatic control is not specified in the table header information, to supply the exit from the open decision table. The GO TO takes precedence if there is a "next table" specified in the header.

GO TO 612 (table number)

GO TO 500 (FORTRAN statement number)

PERFORM

The action operator used to execute a closed decision table is the operator PERFORM, used in the following form:

PERFORM n

where n is the number of the closed decision table. The return from a closed table is automatic, to the action subsequent to the PERFORM in the rule of the table where called. Since this return is automatic, the

use of the action operator GO TO in a closed table should be used only with caution since the use of the GO TO would destroy the automatic return to the calling table.

In summary, an open table has no automatic return and for every action list there must be an indication of where to go next. A closed table has automatic return and normally does not contain GO TO statements.

A closed table which is executed by the PERFORM statement is written in all other aspects as an open table. Each closed table must physically appear after all other tables which make use of it.

OTHER

Other sequence control operators are summarized below:

GO TO (n_1, n_2, n_3, .. , n_n), i

which is the FORTRAN computed GO TO.

GO TO n (n_1, n_2, ... n_m)

which is the FORTRAN assigned GO TO.

CALL name (arguments)

RETURN

which is used in a decision table within 7090 FORTRAN DTS in the definition of subroutines to indicate return to the calling table or statement. It is the logically last statement of each rule of the table which terminates the subroutine.

PAUSE n

STOP n

where n is an integer constant, as is normal FORTRAN.

Input/Output Action Operators

The action operators for input/output commands are used in the manner identical to FORTRAN. A list of the allowable input/output operators is given below:

BACKSPACE i

END FILE i

PRINT n, list

PUNCH n, list

READ n, list

READ DRUM i, j, list

READ INPUT TAPE i, n, list

READ TAPE i, list

REWIND i

WRITE DRUM i, j, list

WRITE OUTPUT TAPE i, n, list

WRITE TAPE i, list

ERRATA

7090 FORTRAN Decision Table System

Page	Location	Correction
1.8	L 2nd from bottom	... more than one rule is ...
2.5	Left page--L 2 of table and subsequent left pages	~~bbbbbb~~ COMMON A, B, C, X1REAL, X2REAL, X1IMAG, X2IMAG
2.8	L 4	25 TABLE.
2.11	Left page L 7, Rule 2	SQRTF(DISC)
2.16	Between L 1 and 2	Same as 2.5 above.
3.1	Left page L 4	Delete
3.8	L 4th from bottom	98 TABLE.
3.9	L 4	18799 TABLE.
3.12	L 13	A VS B GR (mixed entry)
3.18	L 14	GO TO n, $(n_1, n_2, \ldots n_m)$

September 25, 1962