Draft SP-1995
September 9, 1965

A DESCRIPTION OF THE BASIC ALGORITHM USED IN THE DETAB/65 PREPROCESSOR

By A. E. Chapman and M. Callahan

INTRODUCTION

While many papers and articles have been written on the uses and advantages of decision tables, only a few have been written on the algorithms used for converting a decision table to computer instructions.[1] Additionally, except for a few of the algorithms used in current decision table compilers or preprocessors[2], none have yet been formally described. This paper describes the conversion algorithm used in the DETAB/65 preprocessor, which converts decision tables into COBOL.

Since the preprocessor and the language associated with it were developed for COBOL users, the preprocessor was written in a modular form in required COBOL-61. Thus, any COBOL user on any computer can use the preprocessor or easily modify it for his use.

While the use of a preprocessor introduces inefficiencies due to the compile-time of the preprocessor and the run-time associated with COBOL, the advantages of this method were great enough to have warranted the effort.

---

[1] M. S. Montalbano, "Tables, Flow Charts, and Program Logic," IBM Systems Journal, September 1962.

S. L. Pollack, "Conversion of Limited-Entry Decision Tables to Computer Pfograms." RAND Corporation Memo RM-4020-PR, May 1964.

J. F. Egler, "A Procedure for Converting Logic Table Conditions into an Efficient Sequence of Test Instructions." Communications of the ACM, September 1963.

[2] FORTAB by the RAND Corporation, TABSOL and LOGTAB by the General Electric Corporation, DTS by the IBM Corporation.

PREPROCESSOR DESIGN CRITERIA

The preprocessor accepts a decision table within COBOL language called DETAB/65[3], which was itself developed from a language called DETAB-X. [4] It operates prior to the COBOL compilers and, in operation, does not process regular COBOL statements, but merely passes them along to the compiler. However, any decision tables found within the COBOL program are converted by the preprocessor into COBOL statements that are then passed along to the compiler. Each decision table will be considered a COBOL Section, thus making any basic change to the COBOL compiler itself unnecesary.

Although the generation of efficient code from decision tables may require careful selection of the sequence for testing conditions, and may even involve intermixing of decision and actions, it was felt that this type of optimization would require the preprocessor to be too complex, especially since the preprocessor was basically built to allow a great variety of users to experiment with using decision tables.

The preprocessor will accept limited-, extended-, and mixed entry decision tables (see Figure 1 below).

	R_1	R_2	R_3	ELS
A=B	-	Y	N	-
C=D	Y	N	-	-
E=F	N	-	Y	-
X	X	-	X	-
Y	-	X	X	X

Limited-Entry

	R_1	R_2	R_3	ELS
A=	-	B	C	-
C=	D	-	-	-
E≠	F	-	F	-
X	X	-	X	-
Y	-	X	X	X

Extended-Entry

	R_1	R_2	R_3	ELS
A=	-	B	C	-
C=D	Y	N	-	-
E≠	F	-	F	-
X	X	-	X	-
Y	-	X	X	X

Mixed-Entry

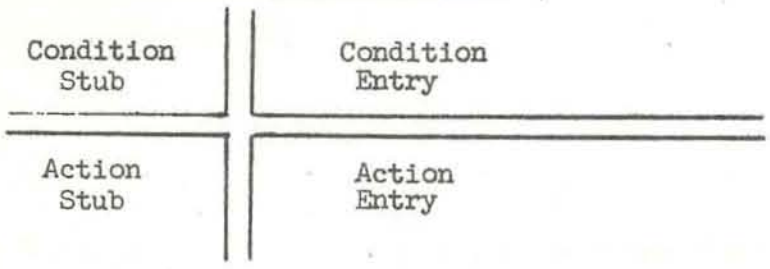
DECISION TABLES
Figure 1

[3] Developed by SIGPLAN Working Group 2 of the Los Angeles Chapter of the ACM, 1965.

[4] Developed by the Systems Group of CODASYL, September 1962.

Since the algorithm only handles limited-entry decision tables, both extended- and mixed-entry tables are transformed into limited-entry decision tables, which are then converted. It is not the purpose of this paper to explain this transformation; suffice it to say that both extended-and mixed-entry tables can be described as limited-entry decision tables.

DECISION TABLE CHARACTERISTICS



A decision table (hereafter known as a DT) can be logically divided into four sections (See Figure 2 above). The upper two sections (Condition Stub and Condition Entry) describes the set or string of conditions that is to be tested. The lower two sections (Action Stub and Action Entry) describe the set or string of actions that is to be taken upon satisfaction of a set of conditions. A rule consists of a set of conditions plus a set of actions, and a DT typically consists of several rules.

Every DETAB/65 DT must contain a special rule called the ELSE-RULE. The ELSE-RULE must not have any entries in the condition-entry section of the DT, but may or may not have entries in the action entry of the table [5]. It is generally considered the error exit of the DT. Some decision-table definitions are defined below:

1. A redundancy exists in a decision table if two or more rules do not have

---

[5] If the table contains 2^n rules (where n = no. of conditions), there will not be any actions specified as the table will be complete. However, it must be present.

at least one Y,N pair in any of the rows and the actions specified are identical.

	R_1	R_2
C_1	-	N
C_2	Y	Y
A_1	X	X
A_2	-	-

2. A <u>contradiction or logic error</u> exists in a decision table if two or more rules do not have at least one Y,N pair in any of the rows and the actions specified are not identical.

	R_1	R_2
C_1	-	N
C_2	Y	Y
A_1	X	-
A_2	-	X

3. Any <u>rules</u> not specified or implied in the table are assumed to be part of the ELSE-RULE. Only the action part of the ELSE-RULE is pertinent.
4. An <u>incomplete decision table</u> is one where there are less than 2^n non-redundant rules (n equals the number of conditions in the table). The ELSE-RULE, however, does not count as one of the 2^n rules.
5. Each <u>rule number</u>, except for the ELSE-RULE, will be denoted by R_1, R_2, R_3,.... where R_i is the rule number, i = 1, 2, 3, - - - 2^n.
6. Each <u>condition</u> will be denoted by C_1, C_2, C_3, ... where C_i is any string of conditional COBOL statements; i = 1, 2, 3, ..., n.
7. Each <u>action</u> will be noted by A_1, A_2, A_3, ... where A_k is any string of non-conditional COBOL statements; K = 1, 2, 3, ...
8. Any <u>entry</u> in the condition and/or action-entry section of a rule will be referred to as an element of the DT; thus, a rule can be considered a column of elements of the DT.

A rule may consist of any one of the five following entries:

	Entry	Definition
1.	"Y" meaning YES	This condition is to be tested to see if it is true.
2.	"N" meaning NO	This condition is to be tested to see if it is false.
3.	" " meaning BLANK	This condition does not apply, or this action is not to be taken when this rule is satisfied.
4.	"-" meaning DASH	Same as 3.
5.	"X" meaning X	This action is to be taken when all conditions for this rule are satisfied.

INPUTS

The inputs to the algorithmic generator (hereafter called the generator) consists of:

1. A matrix (which will be called the DT) composed of Y's, N's, and blanks (dashes are converted to blanks). There are the elements of the DT.
2. A vector, C_1, C_2, C_3, consisting of the n conditions to be satisfied.
3. A vector, Z_1, Z_2, Z_3, ... Z_n consisting of the series (or string) of actions to be performed depending upon the conditions to be satisfied.
4. An ELSE-RULE (which is, however, not considered an integral part of the DT matrix).

OUTPUT

All output from the generator consists of simple COBOL conditional statements of the following type:

$$\text{IF } C_n \text{ GO TO } \begin{bmatrix} \text{DXN} \\ \text{AZM} \\ \text{ELOO1} \end{bmatrix} \begin{bmatrix} . \end{bmatrix} \begin{bmatrix} \text{ELSE GO TO } \begin{bmatrix} \text{AZP} \\ \text{ELOO1} \end{bmatrix} \quad . \end{bmatrix}$$

where:

1. C_n is the nth condition being tested.
2. DXN is the statement label designating the paragraph name where further checks are to be made by the conditions (N is a 3-digit number).

3. ATM is the statement label that precedes user-defined actions for specified rules (ATP is another action label; the two are not the same). M and P are 3-digit no.'s which are based upon the no. of the rule for which the solution was generated.
4. ELOOl is the error or condition not covered exit. The brackets are used to group the possible combinations.

THE ALGORITHM

With the above table characteristics, definitions, and easily deducible corollaries, we can now describe the steps involved in converting a typical decision table (Figure 3) to a series of test instructions. Except for elimination of obviously unnecessary tests, no test optimization is attempted.

GENERAL

Generally, several "passes" are made through the DT, with each pass generating COBOL code, that makes tests on the various conditions and leads to actions that constitute a solution for one or more of the rules. These rules are then deleted from the DT and the process continues until no more rules exist in the DT. The DT is then considered to be solved. The number of passes will never be more than 2^{n-1}, (where n = 's number of conditions in the DT) and may in fact be less. Each condition-entry section will contain one of the following classes of Y's, N's and blanks. It is their satisfaction which will provide information for the generation of output code.

Class 1 - contains one or more Y's, N's and blanks in any combination.

Class 2 - contains one or more Y's and N's in any combination.

Class 3 - contains one or more Y's and blanks in any combination.

Class 4 - contains one or more N's and blanks in any combination.

Class 5 - contains all Y's.

Class 6 - contains all N's.

Class 7 - contains all blanks.

For each pass through the DT, we start with the nth condition in the DT. The "n" is initially set to one and the following steps are done:

Step 1 - Examining the Condition Entry Section

The nth condition entry section elements are examined for one of the above 7 classes. The following rules are then applied to determine what COBOL code to generate.

1. If at least one Y and/or at least one N and any, but not all blanks are found in any combination the generated code will be:

   IF C_n GO TO DXN.

   (where DXN is a generated label which is saved in all cases by placing it in a push-down list from which it can be "popped" up to provide a COBOL statement label when needed. This provides for a "last in-first out" requirement).

2. If only Y's are found the generated code is:

   IF C_n GO TO DXN ELSE GO TO ELOO1.

   and a statement label is "popped" from the push-down list.

3. If only N's are found the generated code is:

   IF C_n GO TO ELOO1.

4. If only blanks and/or 'B's[6] are found, no COBOL code is generated. The elements above the blanks (or B) are then examined and the following tests performed:

   a. If there are no blanks above this condition, that condition's element is changed to a 'Y'.

---

[6] B is a delimiter whose use is explained in Step 6.

b. If there are any blanks above this condition, a 'B' is placed in that condition's element.

This continues until all elements of that condition have been examined. Then the next step is performed.

Step 2 - Preventing Unnecessary Testing of Rules

Before the above mentioned code is implemented a check is made after a Y or an N is found in a rule to determine if the remaining elements in that rule contain all blanks. If they are not all blanks we go to Step 3; otherwise, since one solution for this rule has been found we generate either an AZM or AZP action label for inclusion within the previously generated code (if a Y-generate AZM, or if an N-generate AZP). The number of the rule determines what M or P will be; i.e., Rule 1 - Generate AZ001; Rule 2 - AZ002, etc.).

Step 3 - Last Condition Analysis

If this is not the last condition go to Step 4; otherwise, the following analysis is made:

1. If the DT contains only one rule whose last condition equals Y generate the following code:

   IF C_n GO TO ATM ELSE GO TO EL001

2. If the DT contains one rule whose last condition equals N generate the following code:

   IF C_n GO TO EL001 ELSE GO TO ATP.

3. If the DT contains two rules whose last conditions contain a Y and an N generate the following code:

   IF C_n GO TO ATM ELSE GO TO ATP (M='Y' rule no., P='N' rule no.)

4. If anything else is indicated there exists an inconsistency or redundancy in the logic of the DT. Error messages are generated but conversion is continued.

After this step is performed control goes to Step 6.

Step 4 - Modifying the DT

Each time a condition-entry section has been examined a modified DT is formed (the original DT is saved). This is done by deleting from the DT all rules whose nth condition-entry section have a Y in it. However, if only Y's are found in this condition entry section, no deletions are done and the modified DT remains the same as the original (unmodified) DT.

Step 5 - Iteration

Next increase n by 1 and go back to Step 1.

Step 6 - Handling the Solutions

When the last condition-entry section has been examined, one or more solutions to the original DT have been found. These solutions are then deleted by examining the rules which correspond to these solutions and applying the following criteria:

1. If a SOLUTION rule contains only 'Y's' and 'N's', or if the only blanks found are contained in the last consecutive elements (i.e., n, n-1, etc., contains blanks) of the rule, it is deleted from the original DT.

Any other case indicates that there is more than one solution to the rule. The solution just found is deleted by introducing a delimiter, which we shall call 'B'. By setting blanks equal to 'B', the generator is able to step backwards up the rule (once each pass) and generate code for all possible combinations of conditions in a given rule. The stepping is done in STEP 7, where the solution vector is compared with the original DT and a modified DT formed containing the next solution for this rule and possibly solutions to other rules. This is done in the following manner.

2. The rule is examined for 'B's' and:

   a. If a 'B' is found and there are no blanks in the elements above 'B' in the rule, the element containing the 'B' is changed to 'Y' and Step 6 is repeated. If there are one or more blanks above the 'B', then the blank nearest the 'B' is changed to a 'B' and all other 'B's' are replaced with blanks.

   b. If no 'B's' but any blanks are found, the lowest blank in the rule is replaced with a 'B'.

Step 7 - Setting Up for Next Pass

If the original DT now does not contain any rules, all of the code necessary for testing the conditions has been generated and a complete solution of the DT has been found. The generators work is complete. If one or more rules still exist within the original DT conversion of the DT continues.

First, a statement label is "popped" from the push-down list. Next, one of the solutions (Rules) just found is saved, n is set to one, and all blanks in the rule (hereafter called the solution vector) are set to 'N'. Following that, the nth element in the solution vector is compared with each element in the nth condition entry section of the original DT. If there are any elements in the nth condition-entry section equal to either blanks or the nth element in the solution vector, a special modified DT is found by deleting all rules (from the original DT) whose nth element is not equal to either blanks or the nth element in the solution vector. The n is set equal to n + 1 and the above process repeated until a DT is found which does not satisfy the above conditions.[7] The n is set to n + 1 and the next "pass" started at Step 1 using the special modified DT.

---

[7] Note: If n equals the last condition, redundancy exists within the DT and is so flagged.

ERROR CHECKING

COBOL code is always generated. Errors in the logic of the DT may or may not produce errors in the generated code. Redundancies and inconsistencies are checked for and all errors detected are flagged. Some of these will prevent completion of the conversion, others serve only as a warning.

CONVERSION EXAMPLE

The following is a step by step example of the conversion of a DT:

	R_1	R_2	R_3	R_4	ELS
C_1	Y	Y	N	N	-
C_2	Y	Y	Y	N	-
C_3	Y	N	-	N	-
A_1	X	-	X	-	-
A_2	X	X	-	-	-
A_3	-	X	-	X	-
A_4	-	-	-	-	X
	Z_1	Z_2	Z_3	Z_4	Z_6

R_5 = ELSE-RULE

Z_n = action string to be performed.

Set n = 1 and perform Step 1 for C_1. Since there is both a Y and an N in C_1, the following code is generated:

IF C_1 GO TO DX001.

Since DX001 will be used later as a statement label, it is saved in the push-down list.

Next, Step 4 is performed and the DT becomes:

	R_3	R_4
C_1	N	N
C_2	Y	N
C_3	-	N
	Z_3	Z_4

N is set equal to 2 and Step 1 is performed for C_2. There is both a Y and an N in C_2, but C_3 in R_3 is blank and Step 2 is applied resulting in the following

generated code:

IF C_2 GO TO AZ003

Step 4 is again applied and the DT becomes:

	R_4
C_1	N
C_2	N
C_3	N
	Z_4

N is set to 3 and Step 1 is performed for C_3. Since this is the last condition Step 3 is performed and the resulting code is

IF C_3 GO TO EL001 ELSE GO TO AZ004.

Now since one pass has been made through the DT, Step 6 is performed and the original DT becomes

	R_1	R_2
C_1	Y	Y
C_2	Y	Y
C_3	N	Y
	Z_1	Z_2

Since there are 2 rules left in the DT, Step 7 is performed resulting in n being set to 2.

First, C_2 is examined, which generates the following code:

IF C_2 GO TO DX002 ELSE GO TO EL001

The label DX002 is saved in the push-down list and a modified DT is formed; however, since C_2 contains only Y's the DT remains the same.

Since there are no N's in the C_2 condition-entry section, the statement label DX002 is set up from the "push-down" list. Then n is set to 3 and Step 1 is performed for C_3. Since C_3 contains both a Y and an N, and is also the last

condition, Step 3 states that the output code will be:

IF C_3 GO TO AZ002 ELSE GO TO AZ001.

Steps 6 and 7 are then performed resulting in an original DT with no rules in it. This signifies that all rules have been solved and a complete solution of the DT exists. The generator is now finished with its job.

The generated COBOL code is thus:

(PASS #1) IF C_1 GO TO DX001. (Step 1.1)

IF C_2 GO TO AZ003. (Step 1 and Step 2)

IF C_3 GO TO EL001 ELSE GO TO AZ004. (Step 3.2)

(PASS #2) DX001. (Step 7)

IF C_2 GO TO DX002 ELSE GO TO EL001. (Step 1.2)

DX002.

IF C_3 GO TO AZ002 ELSE GO TO AZ001. (Step 3.3)

ACTION PROCESSING

Upon the generator's completion, each of the series of actions specified for each of the rules (including the ELSE-RULE, which is labeled EL001) are set up as individual COBOL paragraphs, to be performed in the occurring sequence (i.e., A_1, A_2, A_3, ...) Each rule thus has its own sequence of actions (some of which may be redundant with another rule). As an example, the following would be generated for the example above:

AZ001.

A_1

A_2.

GO TO DEXIT.

AZ002.

A_2.

A_3

GO TO DEXIT

```
AZ003.
    A1.
    GO TO DEXIT.
AZ004
    A3
    GO TO DEXIT.
EL001.
    A4
    DEXIT. EXIT.
```

GO TO DEXIT is almost always generated since a branch may not exist to allow normal exit from the DT. DEXIT is the normal return from the decision table if the DT is operated as a closed COBOL subroutine. If it is not, the last action performed by any rules action string must be an unconditional GO TO. If it is, then the "GO TO DEXIT" will not be generated.

DETAB-65 USER'S MANUAL

1. INTRODUCTION

Decision Tables endow a user with the ability to provide a graphical representation of a complex procedure in such a way that one individual is able to readily understand a program written by another.

DETAB-65 is the decision table language which the preprocessor converts to COBOL statements for subsequent processing by an appropriate COBOL compiler.

This manual's purpose is to describe to a user how a DETAB-65 decision table should be written for inclusion within a COBOL program. It also describes the necessary linkages, formats, and restrictions used in construction of the decision table.

Since not all definitions will be defined in this manual, it is recommended that the user first study the DETAB-65 documents accompanying this manual.

In addition, a test program is documented in Appendix B.

2. STRUCTURE OF A DECISION TABLE

A decision table can be logically divided into four sections (See Figure 1 below). The upper two sections (Condition Stub and Condition Entry) describe the set or string of conditions that is to be tested. The lower two sections (Action Stub and Action Entry) describe the set or string of actions that is to be taken upon satisfaction of a set of conditions. A rule consists of a set of conditions plus a set of actions, and a decision table typically consists of several rules.

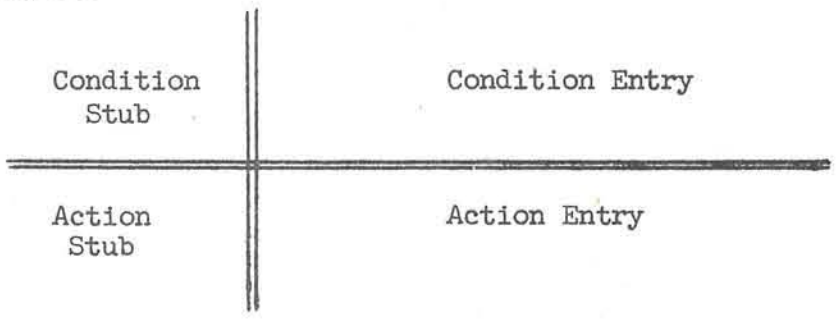


Figure 1

The three types of decision tables in current use today are the limited-, extended-, and mixed-entry types (see Figure 2 below). Eventually it will be possible to convert all three types of tables via the preprocessor; however, at the present time the preprocessor is restricted to limited-entry tables.

	R_1	R_2
C_1	N	Y
C_2	Y	N
A_1	X	-
A_2	-	X

LIMITED ENTRY

	R_1	R_2
C_1	=58	=25
C_2	≠ J	= K
A_1	X	-
A_2	-	X

EXTENDED-ENTRY

	R_1	R_2
C_1	=58	Y
C_2	≠ J	N
A_1	X	-
A_2	-	X

MIXED-ENTRY

Figure 2

3. PROGRAM FORMAT

The format for a COBOL program containing DETAB-65 decision tables must conform to the requirements for any COBOL program, except that a decision table is inserted in the COBOL PROCEDURE DIVISION as a SECTION, and is referred to by an appropriate COBOL statement (see Table Linkage).

The DATA DIVISION of a COBOL program incorporating a DETAB-65 decision table is treated as is any other COBOL DATA DIVISION. Any symbolic data reference, data structure, constant, or working storage used in a decision table must be declared in the DATA DIVISION.

The decision table(s) are placed at the end of the PROCEDURE DIVISION since they are to be treated as COBOL subroutines. This is the only difference between a COBOL program and a COBOL with decision table(s) program.

4. TABLE LINKAGE

In compilation a decision table will be treated as a closed COBOL subroutine. Thus, a decision table should not be entered via the normal operating sequence,

but only by using COBOL GO TO or PERFORM verbs. [1]

Since a GO TO results in an uncondition transfer, a return or transfer point must be specified by the user in the decision tables action stub or the processing sequence will be lost. It is recommended that the GO TO verb not be used when referring to a table from the main sequence of the program.

When transferring control to a decision table by the use of a COBOL PERFORM verb, a normal return to the processing sequence will be made by the compiler unless the user specifies otherwise in his actions. Specifically, the preprocessor will generate a "GO TO DEXIT.", for every rule whose last action does not end in a "GO TO ---.", no matter how the table was entered.

Tables may be chained together by placing GO TO's and PERFORM's in the Action Stub of one or more tables. However, it is advisable to keep very, very close track of this as it is possible to generate errors due to the way various COBOL compilers set up their procedure sections.

5. DEFINITIONS

The following is a series of decision table definitions to be used in describing a DETAB-65 decision table:

5.1 TABLE-ID

Identifies to the preprocessor that a DETAB-65 decision table has been encountered. The ID is always 4 numeric characters consisting of 4 zeros (0000).

5.2 RULE-ID

Identifies to the preprocessor that the rule-card is present (for error checking purposes). The ID is always 4 numeric characters in length and consists of (0001).

5.3 TABLE-NAME

This is a 30 alphanumeric character or less name which is then used to identify the COBOL section generated by the preprocessor.

---

[1] It is possible to enter a table from the main sequence but the trouble this can entail does not make it worthwhile.

5.4 FORM

Designates the kind of table is present (i.e., limited-, extended-, and mixed-entry tables) and is always an alphabetic character (L, E, M, left-justified).

5.5 COND ROWS

Designates the number of conditions in the Condition Stub of the table. This is 3 numeric characters (right-justified).

5.6 ACTION ROWS

Designates the number of actions in the Action Stub of the table. This is 3 numeric characters (right-justified).

5.7 RULES

Designates the number of rules in a table. This includes the ELSE-RULE and is 3 numeric characters (right-justified).

5.8 RULE NUMBERS

These are 3 numeric characters each used to identify each rule. The ELSE-RULE is the only exception and is designated by the 3 alphabetic characters ELS.

5.9 CONDITION STUB

Contains logical, arithmetic, or relational conditions answerable by a yes (Y) or a no (N).

5.10 CONDITION ENTRIES

These indicate which condition must be met to satisfy a rule. This can be a Y, N, BLANK ( ), or DASH (-). A blank or dash means that the user does not care if the rule is Y or N as it makes no difference. Also known as the elements of a task.

5.11 ACTION STUB

Contains imperative statements to be performed as indicated by the ACTION ENTRIES of a rule when the rule is satisfied.

5.12 ACTION ENTRIES

These indicate which actions must be performed if a rule is satisfied. The character used to signify this is an X.

6. CONVENTIONS AND RESTRICTIONS

When writing DETAB-65 decision tables the following words cannot be used (this is in addition to COBOL restricted words) 6.1 - 6.6.

6.1 DXN (where N is a 3 digit number)
6.2 AZM (where M is a 3 digit number)
6.3 AZP (where P is a 3 digit number)
6.4 ELOOl
6.5 DEXIT
6.6 ELS
6.7 Maximum of 50 entries in Condition Stub.
6.8 Maximum of 50 entries in Action Stub.
6.9 Maximum of 50 rules (includes ELSE-RULE).
6.10 Maximum of 12 and minimum of 3 columns in a rule.
6.11 Maximum of 58 and minimum of 12 columns in the Condition & Action Stubs.
6.12 ELSE-RULE (ELS) must always be present.
6.13 TABLE END card ($ in Column 7) must always be present.
6.14 EOF card (999X in Columns 4-7 must always be present.

7. DETAB-65 DECISION TABLE

The table itself is written in a mixture of fixed and free formats. The following lists the various sections showing theproper way to set up a DETAB-65 decision table.

7.1 Header

The header contains information used by the preprocessor to initate the conversion of the table and allow it to check for errors.

COLUMN(S)	DEFINITION
4 - 7	TABLE ID
9 - 38	TABLE NAME
39 - 40	FORM
41 - 43	COND-ROWS
44 - 46	ACTION-ROWS
47 - 49	RULES

All other columns in the header are blank. [4]

7.2 Rule

The rule part of the DETAB-65 table contains information which allows the preprocessor to determine the size of the stub area. The size of the largest entry in either the condition or action stub determines the size of the stub area.

COLUMN(S)	DEFINITION
4 - 7	RULE-ID
9 - XX	Blank. This is the size of the stub area.

The rules (RULE-NUMBERS) start immediately after the last column in the stub area and are terminated by a dollar sign ($). The only exception is if the last rule occupies column 80, thus putting the dollar sign in column 81.

7.3 Body

The body part of the table contains the Condition and Action Stubs and Entries. All actions are performed in the order of their occurrence; if it is desired to do series of action in a different order, it is necessary to repeat them in that order in the action stub.

7.4 End

The end part notifies the preprocessor when the input is over and conversion is to begin. It consists of a dollar sign ($) in column 7.

Preprocessor Output

The preprocessor will convert a DETAB-65 decision table into COBOL statements of the following format:

IF C_n GO TO { DXN / AZM / ELOO1 } . ELSE GO TO { AZP / ELOO1 } .

Where C_n is the condition to be tested, and the brackets output formats.

Errors

All errors detected during the preprocessors operation will be listed upon the list tape. For a full description of all errors see Appendix A.

---

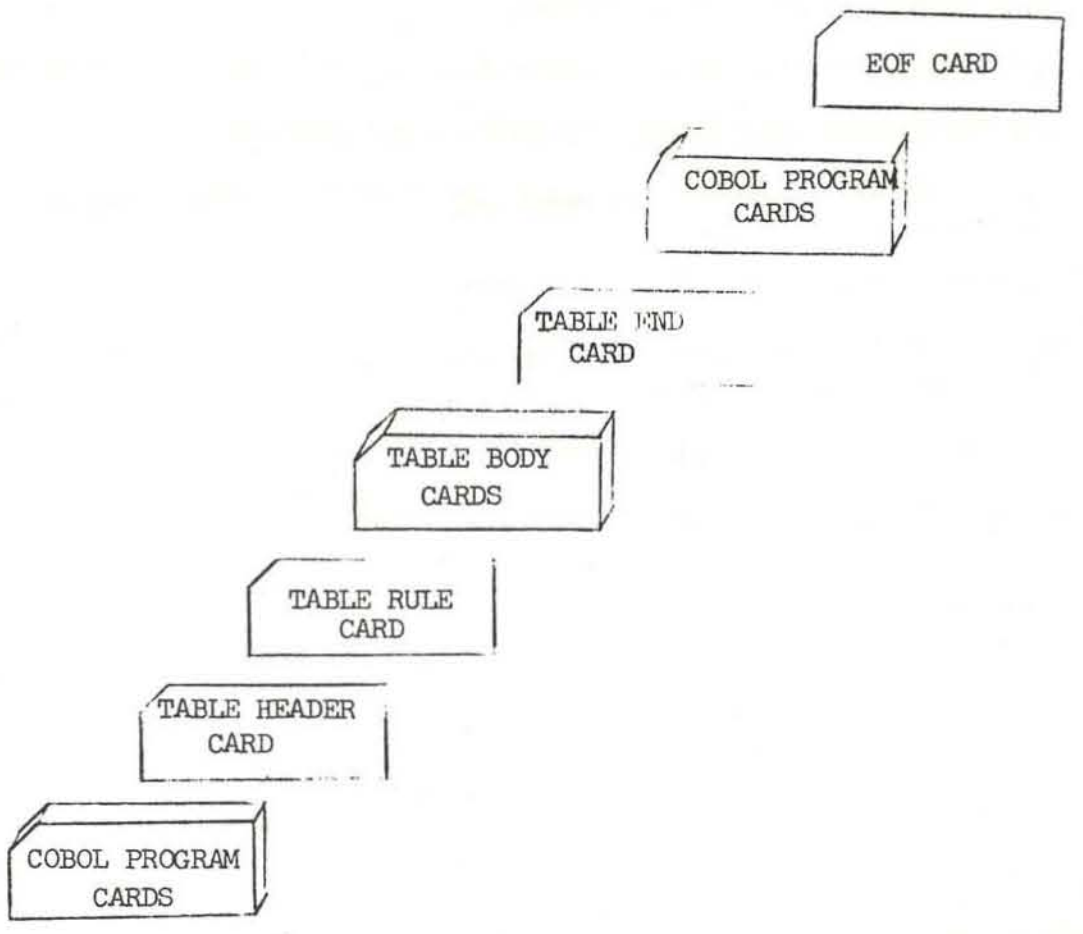
[4] NOTE: Columns 1-3, 4-6 may contain a sequence number (except for header and rule cards).

8. EOF Card

To tell the preprocessor when to terminate operations a special card is used called the EOF card. This consists of a 999X in columns 4 - 7, and is the last card processed. This means that if more than one program is to be processed the EOF card is placed after the last program.

Deck Outline

The following shows the deck structure of a COBOL program containing a DETAB-65 decision table.



Sample COBOL, DETAB-65 Program Deck

1. PRESENTLY TABLE RESTRICTED TO LIMITED ENTRIES

   FORM does not contain an 'L'.

2. TABLE-NAME MISSING FROM HEADER CARD

   Processing will be halted and table skipped.

3. RULES CARD MISSING

   Processing will be halted and table skipped.

4. LESS THAN 3 RULE COLUMNS SPECIFIED

   A rule contains 2 or less columns for processing, conversion will be halted and table skipped.

5. PRESENTLY, CONTINUED RULES NOT IMPLEMENTED

   Column 8 has an entry.

6. CONDITION STUB EXCEEDS 58 COLUMNS

   Condition stub exceeds limit processing halted and table skipped.

7. NUMBER OF RULES ENCOUNTERED DISAGREES WITH RULE CARD

   Number of rules entered in header card (RULES) differs from that specified by the user.

8. MORE THAN 50 ACTION OR CONDITION ENTRIES

   Action or Condition stub contains more than 50 entries, processing halted and table skipped.

9. DECISION TABLE LOGIC ERROR. PROCESSING HALTED.

   Check over rules for either redundancy, inconsistency or both.

```
*BEGIN JOB 296  08/10/65
*COOP,90101,DETAB65,S/1S,56,55555,4,
*COBOL,X.
       IDENTIFICATION DIVISION.
       PROGRAM-ID.  PLAYBOY.
       AUTHOR.  CHARLES CREE.
       DATE-COMPILED.  08/10/65
       REMARKS.  THIS IS A SIMPLE DATA RETRIEVAL PROGRAM TO
           ILLUSTRATE THE USE OF DETAB/65.
       ENVIRONMENT DIVISION.
       CONFIGURATION SECTION.
       SOURCE-COMPUTER.  CONTROL DATA 1604-A.
       OBJECT-COMPUTER.  CONTROL DATA 1604-A.
       INPUT-OUTPUT SECTION.
       FILE-CONTROL.
           SELECT CANDIDATES ASSIGN TO SYSTEM-INPUT-TAPE.
           SELECT RATED-FILE ASSIGN TO SYSTEM-OUTPUT-TAPE.
       DATA DIVISION.
       FILE SECTION.
       FD CANDIDATES
           LABEL RECORDS ARE OMITTED
           DATA RECORD IS INREC.
       01  INREC.
           02 FILLER   PICTURE X(80).
       FD  RATED-FILE
           LABEL RECORDS ARE OMITTED
           DATA RECORD IS EVAL.
       01  EVAL.
           02 FILLER   PICTURE X(48).
       WORKING-STORAGE SECTION.
       77  TOP-CTR          PICTURE 9(7)  COMPUTATIONAL,
             SYNCHRONIZED RIGHT,  VALUE ZERO.
       77  NEXT-BEST-CTR    PICTURE 9(7)  COMPUTATIONAL,
             SYNCHRONIZED RIGHT,  VALUE ZERO.
       77  LAST-RESORT-CTR  PICTURE 9(7)  COMPUTATIONAL,
             SYNCHRONIZED RIGHT,  VALUE ZERO.
       77  TOTAL-NO PICTURE 9(5) COMPUTATIONAL,
             SYNCHRONIZED RIGHT, VALUE ZERO.
       01  WSR.
        02  SKIP-CTL PICTURE X,
        02  RATING   PICTURE X(12),
        02  WSR1.
```

```
  03  IDNO     PICTURE 9(7).
  03  SEX      PICTURE X.
    88 FEMALE    VALUE "F".
  03  AGE      PICTURE 999.
  03  HEIGHT   PICTURE 999.
  03  WEIGHT   PICTURE 999.
  03  HAIR     PICTURE X(9).
    88 BLOND     VALUE "BLOND".
  03  EYES     PICTURE X(5).
    88 BLUE-EYES VALUE "BLUE".
  03  I-Q      PICTURE 999
  03  FILLER   PICTURE X.



PROCEDURE DIVISION.
PN01.
    OPEN INPUT CANDIDATES.
    OPEN OUTPUT RATED-FILE.
PN02.
    READ CANDIDATES INTO WSR1, AT END GO TO PNEOJ.
    PERFORM CHOICE-PICK.
    GO TO PN02.
PNEOJ.
    CLOSE CANDIDATES WITH LOCK, RATED-FILE WITH LOCK.
    DISPLAY "TOPS COUNT = " TOP-CTR.
    DISPLAY "NEXT COUNT = " NEXT-BEST-CTR.
    DISPLAY "LAST COUNT = " LAST-RESORT-CTR.
    DISPLAY "TOTAL = " TOTAL-NO.
    STOP RUN.
```

```
0010000 CHOICE-PICK                    L 010008008
 010001                               -001002003004005006007ELS$
 010101 FEMALE                          Y  Y  Y  Y  Y  Y  Y
 010102 AGE GREATER THAN 18             Y  Y  Y  Y  Y  Y  Y
 010103 AGE LESS THAN 38                Y  N  Y  Y  Y  N  Y
 010104 BLOND                           Y     N        N
 010105 BLUE-EYES                       Y     N     N  N
 010106 WEIGHT GREATER THAN 89          Y  Y  Y  Y  Y  Y  Y
 010107 WEIGHT LESS THAN 133            Y  Y  Y  N  Y  N  Y
 010108 HEIGHT GREATER THAN 59          Y  Y  Y  N  Y  N  Y
 010109 HEIGHT LESS THAN 68             Y  Y  Y     Y     N
 010110 I-Q GREATER THAN 99                Y  N  Y  Y  N  N

 010112 MOVE #TOPS# TO RATING           X              X  X
 010202 ADD 1 TO TOP-CTR                X              X  X
 010203 MOVE #NEXT BEST# TO RATING            X              X
 010204 ADD 1 TO NEXT-BEST-CTR                X              X
 010204 MOVE #LAST RESORT# TO RATING       X     X
 010206 ADD 1 TO LAST-RESORT-CTR           X     X
 010201 WRITE EVAL FROM WSR             X  X  X  X  X  X  X
 010207 ADD 1 TO TOTAL-NO               X  X  X  X  X  X  X  X
      $
```

```
-----------------------------------------------------------*

       CHOICE-PICK SECTION.
       DX000.
           IF FEMALE
            GO TO DX001  ELSE  GO TO EL001.
       DX001.
           IF AGE GREATER THAN 18
            GO TO DX002  ELSE  GO TO EL001.
       DX002.
           IF AGE LESS THAN 38
            GO TO DX003.
           IF BLOND
            GO TO DX004.
           IF BLUE-EYES
            GO TO DX005.
           IF WEIGHT GREATER THAN 89
            GO TO DX006  ELSE  GO TO EL001.
       DX006.
           IF WEIGHT LESS THAN 133
```


*Generated output follows

```
     GO TO DX007.
    IF HEIGHT GREATER THAN 59
     GO TO EL001.
    IF I-Q GREATER THAN 99
     GO TO EL001  ELSE  GO TO AZ006.
DX007.
    IF HEIGHT GREATER THAN 59
     GO TO DX008  ELSE  GO TO EL001.
DX008.
    IF HEIGHT LESS THAN 68
     GO TO DX009  ELSE  GO TO EL001.
DX009.
    IF I-Q GREATER THAN 99
     GO TO AZ002  ELSE  GO TO EL001.
DX005.
    IF WEIGHT GREATER THAN 89
     GO TO DX010  ELSE  GO TO EL001.
DX010.
    IF WEIGHT LESS THAN 133
     GO TO DX011  ELSE  GO TO EL001.
DX011.
    IF HEIGHT GREATER THAN 59
     GO TO DX012  ELSE  GO TO EL001.
DX012.
    IF HEIGHT LESS THAN 68
     GO TO DX013  ELSE  GO TO EL001.
DX013.
    IF I-Q GREATER THAN 99
     GO TO AZ002  ELSE  GO TO EL001.
DX004.
    IF WEIGHT GREATER THAN 89
     GO TO DX014  ELSE  GO TO EL001.
DX014.
    IF WEIGHT LESS THAN 133
     GO TO DX015  ELSE  GO TO EL001.
DX015.
    IF HEIGHT GREATER THAN 59
     GO TO DX016  ELSE  GO TO EL001.
DX016.
    IF HEIGHT LESS THAN 68
     GO TO DX017  ELSE  GO TO EL001.
DX017.
```

```
    IF I-Q GREATER THAN 99
     GO TO AZ002  ELSE  GO TO EL001.
DX003.
    IF BLOND
     GO TO DX018.
    IF BLUE-EYES
     GO TO DX019.
    IF WEIGHT GREATER THAN 89
     GO TO DX020  ELSE  GO TO EL001.
DX020.
    IF WEIGHT LESS THAN 133
     GO TO DX021.
    IF HEIGHT GREATER THAN 59
     GO TO EL001.
    IF I-Q GREATER THAN 99
     GO TO AZ004  ELSE  GO TO EL001.
DX021.
    IF HEIGHT GREATER THAN 59
     GO TO DX022  ELSE  GO TO EL001.
DX022.
    IF HEIGHT LESS THAN 68
     GO TO DX023.
    IF I-Q GREATER THAN 99
     GO TO EL001  ELSE  GO TO AZ007.
DX023.
    IF I-Q GREATER THAN 99
     GO TO AZ005  ELSE  GO TO AZ003.
DX019.
    IF WEIGHT GREATER THAN 89
     GO TO DX024  ELSE  GO TO EL001.
DX024.
    IF WEIGHT LESS THAN 133
     GO TO DX025.
    IF HEIGHT GREATER THAN 59
     GO TO EL001.
    IF I-Q GREATER THAN 99
     GO TO AZ004  ELSE  GO TO EL001.
DX025.
    IF HEIGHT GREATER THAN 59
     GO TO DX026  ELSE  GO TO EL001.
DX026.
    IF HEIGHT LESS THAN 68
```

```
     GO TO EL001.
    IF I-Q GREATER THAN 99
     GO TO EL001  ELSE  GO TO AZ007.
DX018.
    IF BLUE-EYES
     GO TO DX027.
    IF WEIGHT GREATER THAN 89
     GO TO DX028  ELSE  GO TO EL001.
DX028.
    IF WEIGHT LESS THAN 133
     GO TO DX029.
    IF HEIGHT GREATER THAN 59
     GO TO EL001.
    IF I-Q GREATER THAN 99
     GO TO AZ004  ELSE  GO TO EL001.
DX029.
    IF HEIGHT GREATER THAN 59
     GO TO DX030  ELSE  GO TO EL001.
DX030.
    IF HEIGHT LESS THAN 68
     GO TO DX031.
    IF I-Q GREATER THAN 99
     GO TO EL001  ELSE  GO TO AZ007.
DX031.
    IF I-Q GREATER THAN 99
     GO TO AZ005  ELSE  GO TO EL001.
DX027.
    IF BLUE-EYES
     GO TO DX032.
    IF WEIGHT GREATER THAN 89
     GO TO DX033  ELSE  GO TO EL001.
DX033.
    IF WEIGHT LESS THAN 133
     GO TO DX034.
    IF HEIGHT GREATER THAN 59
     GO TO EL001.
    IF I-Q GREATER THAN 99
     GO TO AZ004  ELSE  GO TO EL001.
DX034.
    IF HEIGHT GREATER THAN 59
     GO TO DX035  ELSE  GO TO EL001.
DX035.
```

```
    IF HEIGHT LESS THAN 68
     GO TO EL001.
    IF I-Q GREATER THAN 99
     GO TO EL001  ELSE  GO TO AZ007.
DX032.
    IF WEIGHT GREATER THAN 89
     GO TO DX036  ELSE  GO TO EL001.
DX036.
    IF WEIGHT LESS THAN 133
     GO TO DX037  ELSE  GO TO EL001.
DX037.
    IF HEIGHT GREATER THAN 59
     GO TO DX038  ELSE  GO TO EL001.
DX038.
    IF HEIGHT LESS THAN 68
     GO TO AZ001  ELSE  GO TO EL001.
AZ001.
    MOVE #TOPS# TO RATING.
    ADD 1 TO TOP-CTR.
    WRITE EVAL FROM WSR.
    ADD 1 TO TOTAL-NO.
    GO TO DEXIT.
AZ002.
    MOVE #LAST RESORT# TO RATING.
    ADD 1 TO LAST-RESORT-CTR.
    WRITE EVAL FROM WSR.
    ADD 1 TO TOTAL-NO.
    GO TO DEXIT.
AZ003.
    MOVE #NEXT BEST# TO RATING.
    ADD 1 TO NEXT-BEST-CTR.
    WRITE EVAL FROM WSR.
    ADD 1 TO TOTAL-NO.
    GO TO DEXIT.
AZ004.
    MOVE #LAST RESORT# TO RATING.
    ADD 1 TO LAST-RESORT-CTR.
    WRITE EVAL FROM WSR.
    ADD 1 TO TOTAL-NO.
    GO TO DEXIT.
AZ005.
    MOVE #TOPS# TO RATING.
```

```
           ADD 1 TO TOP-CTR.
           WRITE EVAL FROM WSR.
           ADD 1 TO TOTAL-NO.
           GO TO DEXIT.
       AZ006.
           MOVE #TOPS# TO RATING.
           ADD 1 TO TOP-CTR.
           WRITE EVAL FROM WSR.
           ADD 1 TO TOTAL-NO.
           GO TO DEXIT.
       AZ007.
           MOVE #NEXT BEST# TO RATING.
           ADD 1 TO NEXT-BEST-CTR.
           WRITE EVAL FROM WSR.
           ADD 1 TO TOTAL-NO.
           GO TO DEXIT.
       EL001.
           ADD 1 TO TOTAL-NO.
       DEXIT. EXIT.

               END PROGRAM.
           999X

       END DETAB/65 PREPROCESSOR RUN.
```

```
0010000 CHOICE-PICK                L 010008008
 010001                            00100200300400500600700ELS$
 010101 FEMALE                      Y  Y  Y  Y  Y  Y  Y
 010102 AGE GREATER THAN 18         Y  Y  Y  Y  Y  Y  Y
 010103 AGE LESS THAN 38            Y  N  Y  Y  Y  N  Y
 010104 BLOND                       Y     N        N
 010105 BLUE-EYES                   Y     N     N  N
 010106 WEIGHT GREATER THAN 89      Y  Y  Y  Y  Y  Y  Y
 010107 WEIGHT LESS THAN 133        Y  Y  Y  N  Y  N  Y
 010108 HEIGHT GREATER THAN 59      Y  Y  Y  N  Y  N  Y
 010109 HEIGHT LESS THAN 68         Y  Y  Y     Y     N
 010110 I-Q GREATER THAN 99            Y  N  Y  Y  N  N

 010112 MOVE ≠TOPS≠ TO RATING       X              X  X
 010202 ADD 1 TO TOP-CTR            X              X  X
 010203 MOVE ≠NEXT BEST≠ TO RATING        X              X
 010204 ADD 1 TO NEXT-BEST-CTR            X              X
 010204 MOVE ≠LAST RESORT≠ TO RATING   X     X
 010206 ADD 1 TO LAST-RESORT-CTR       X     X
 010201 WRITE EVAL FROM WSR         X  X  X  X  X  X  X  X
 010207 ADD 1 TO TOTAL-NO
      $
        END PROGRAM.
  999X
```

INPUT TO GENERATED PROGRAM

```
3278215F031067115BLACK     GREEN106
0567982F025065119BLOND     BLUE 103
1425893F021062110RED       GREY 101
0053247050055145BALD       BROWN98
```

OUTPUT FROM GENERATED PROGRAM

```
TOPS        3278215F031067115BLACK    GREEN106
TOPS        1425893F021062110RED      GREY 101
TOPS        0567982F025065119BLOND    BLUE 103

TOPS COUNT = 0000003
NEXT COUNT = 0000000
LAST COUNT = 0000000
TOTAL = 00004
```

TM-2288/000/00

AUTHOR C. J. Shaw

TECHNICAL E. H. Jacobs

RELEASE E. H. Jacobs for D. Drukey

DATE 4-12-65 PAGE 1 OF 14 PAGES

4/15

TECH MEMO

a working paper

System Development Corporation / 2500 Colorado Ave. / Santa Monica, California

DECISION TABLES--AN ANNOTATED BIBLIOGRAPHY

ABSTRACT

Decision tables allow complex decision rules to be represented in easily grasped, tabular form, making it easy to see what actions are to be taken for each possible combination of conditions. This bibliography contains a brief introduction and 42 references, mostly annotated, to the subject.

INTRODUCTION

A decision table is a way of representing complex decision rules in an easily grasped, tabular form, which makes it easy to see what actions are to be taken for each possible combination of conditions. Let us look at a typical decision rule.

> if CONDITION-A and CONDITION-B and CONDITION-C hold, them do ACTION-1 and ACTION-2, and ACTION-3.

This same rule could be expressed as a column in a decision table.

	rule
CONDITION-A	Y
CONDITION-B	Y
CONDITION-C	Y
ACTION-1	X
ACTION-2	X
ACTION-3	X

In such a table, the top rows represent conditions, the bottom rows represent actions. The entries in a column corresponding to a rule are chosen from the following symbols:

symbol	meaning
Y	Yes, this condition must hold for the rule to apply
N	No, this condition must not hold for the rule to apply
-	don't care if this condition holds
X	do this action if the rule applies
(blank)	do not do this action if the rule applies

With these symbols, we could add other rules to the decision table.

CONDITION-A	Y	Y	N	-
CONDITION-B	Y	-	N	-
CONDITION-C	Y	N	-	-
ACTION-1	X		X	
ACTION-2	X	X		
ACTION-3	X			X

Conventionally, the rules are examined one at a time from left to right, only the first applicable rule is applied, and the actions are taken one at a time in the order they are listed. Other conventions are feasible, though.

Decision tables where the entries are limited to the symbols used in the table above are called limited entry tables. Extended entry tables are possible wherein part of the condition or action is entered in the rule columns. The following taxonomy is an example of an extended entry decision table.

number of legs	=4	=4	=4	>4	-
length of nose	long	short	long	-	-
length of neck	short	long	long	-	-
then animal is	elephant	giraffe	hallucination	centipede	unknown
go to	zookeeper	zookeeper	psychiatrist	exterminator	biologist

Extended entry tables can be much more compact than limited entry decision tables, as you would see by expanding the table above into limited entry form.

BIBLIOGRAPHY

Armerding, G. FORTAB: A DECISION TABLE LANGUAGE FOR SCIENTIFIC COMPUTING APPLICATIONS. in Proceedings of the Decision Tables Symposium, pages 81-87. 20-21 September 1962. Also RAND Corp., RM-3306-PR, 37 pages, September 1962.

Scientific computer programs, like business programs, often involve programmed decision logic. Decision tables, which have seen use in business and commercial computer applications, can also be applied to scientific and engineering problems. FORTAB is a decision table language based on the FORTRAN scientific computing language. Programs written in the combined FORTAB and FORTRAN languages can be compiled for a FORTAB pre-processor program which has been constructed for the IBM 7090 computer. Initial experiments conducted using the FORTAB language indicate that a decision table language added to a scientific computing language results in a powerful combination of programming tools.

Brown, L.M. DECISION TABLE EXPERIENCE ON A FILE MAINTENANCE SYSTEM. in Proceedings of the Decision Tables Symposium, pages 75-80. 20-21 September 1962.

A decision table language and computer program pre-compiler were developed at the Insurance Company of North America to facilitate design, implementation and maintenance of a large file maintenance program. The results of this effort indicate that decision tables can have application over the entire systems design-programming area. Decision tables also force a disciplined modularity in the design of a program which can enable a compiler to accomplish some of the program organization function.

Clakins, L.W. PLACE OF DECISION TABLES AND DETAB-X in Proceedings of the Decision Tables Symposium, pages 9-12. 20-21 September 1962.

Cantrell, H.N., J. King and F.E.H. King. LOGIC-STRUCTURE TABLES. in Communications of the ACM, Vol. 4, No. 6, pages 272-275. June 1961.

Logic tables are an excellant way of developing and expressing the logic required in procedures, operations, systems and circuits. A set of rules for writing and using logic tables is explained by means of some simple examples. Then the logic structure of a vending machine is given in which two logic tables are used. Logic tables are two-dimensional in nature, enabling us to fully express and consider both the sequential and parallel aspects of logic. They can be compiled directly into a computer program and so eliminate the need for flow charting and hand coding.

Cantrell, H.N. COMMERCIAL AND ENGINEERING APPLICATIONS OF DECISION TABLES. in Proceedings of the Decision Tables Symposium, pages 55-61. 20-21 September 1962.

This paper covers our experience with decision tables, from the time we first heard about them through experiments in different application areas, to our present rather widespread use of tables in systems design and programming. We will discuss some of the difficulties we have had in using decision tables and some of the advantages we think we have gained from them.

DECISION TABLES-A SYSTEMS ANALYSIS AND DOCUMENTATION TECHNIQUE. IBM Corp., F20-8102, 21 pages. 1962.

Describes the basic concepts of decision tables and a minimum set of conventions for their use in systems analysis, procedure design, and documentation. Such tables provide information in a concise format that is easy to read and understand. The tabular approach is used to express complex decision logic in a manner that encourages the analyst to reduce a problem to its simplest form by arranging and presenting logical alternatives under various conditions. While the concepts in the text are presented on a level for comprehension by students in basic computer courses, the techniques are applicable at all levels of sophistication by everyone in a data processing environment.

DECISION TABLES, PRACTICE PROBLEMS AND SOLUTIONS. IBM Corp., R25-1685-1, 11 pages. 1963.

These four practice problems, with solutions, are designed to aid the student in learning how to use and prepare limited entry Decision Tables.

DETAB-X, PRELIMINARY SPECIFICATIONS FOR A DECISION TABLE STRUCTURED LANGUAGE; CODASYL Systems Group, 1962.

The Systems Group of the Development Committee of the Conference on Data Systems Languages (CODASYL), as a first step in creating a data-processing language based on decision tables, has developed DETAB-X, a decision table language based on COBOL-61. Because decision tables are structured differently from the free-form procedure statements of COBOL-61, some modifications to COBOL-61 are required; however, these are held to a minimum and are of such a nature as to enable a relatively simple preprocessor to convert the decision tables statements to COBOL-61 statements which can then, in turn, be processed by a COBOL-61 compiler (or processor).

The benefits to be derived from a tabular format are many. First, it is most important that, by the very nature of the table format, omissions in problem logic are easily spotted. Second, the analysis inherent in listing the conditions upon which a given action is based tends to clarify complicated parts of a problem. Third, the format simplifies a total systems organization through modularity. Fourth, this format is easy to use and for others to understand. In addition, the Group believes that the tabular format can be a significant tool in the building of compiling systems themselves.

This DETAB-X manual has been prepared as a language specification reference publication, supplementary to the official COBOL-61 manual published by the United States Government Printing Office. It provides sufficient information to permit experimentation by many COBOL users.

Dixon, P. SPECIAL REPORT, DECISION TABLES SYMPOSIUM. in Standard EDP Reports, Vol. 1, pages 23:030.100-23:030.601. December 1962.

Dixon, P. DECISION TABLES AND THEIR APPLICATION. in Computers and Automation, Vol. 13, No. 4, pages 14-19. April 1964.

Describes the fundamental principles of decision table design, with examples. Indicates the power and applicability of the technique to increase the efficiency of systems analysis and programming. Includes directions for further development and an eleven-point summary of the advantages.

Egler, J.F. A PROCEDURE FOR CONVERTING LOGIC TABLE CONDITIONS INTO AN EFFICIENT SEQUENCE OF TEST INSTRUCTIONS. in Communications of the ACM, Vol. 6, No. 8, pages 510-514. September 1963.

Evans, O.Y. REFERENCE MANUAL FOR DECISION TABLES. IBM Corp. September 1961.

This manual is written to provide a base language (point of departure) for using decision tables. The language is not all-inclusive and in many instances has been arbitrary in the interest of simplicity. It provides a language that interested persons can use to experiment with decision tables in documenting problem definitions. This language is very rigorous in order that it may be used at the detail level of documentation. People experimenting at higher levels of man to man communication can adjust this rigor to their needs.

Evans, O.Y. A METHOD FOR SYSTEMATIC DOCUMENTATION--KEY TO IMPROVED DATA PROCESSING ANALYSIS. In Computer Applications--1961, The Macmillan Co., New York, pages 14-34. 1962.

Evans, O.Y. GENERAL INFORMATION MANUAL, ADVANCED ANALYSIS METHOD FOR INTEGRATED ELECTRONIC DATA PROCESSING. IBM Corp., F20-8047, 21 pages. 1960.

The analysis method presented here can be best described as a systematic method for collecting, recording and maintaining all pertinent information regarding a complete data processing system. The method is particularly useful in that it requires a complete explanation of the characteristics and utilization of each piece of data involved in the system. In addition, it introduces a tabular format for the definition of procedures.

Among the advantages gained through the use of such a documenting analysis are:

1. A definite and orderly method of documenting analysis data is achieved.
2. The analysis is virtually independent of the processing media (i.e., manual, unit record, or high speed computer).
3. The tabular approach to procedure definition aids the analyst in visualizing the numerous relationships and alternatives.
4. The documented analysis and the cross-reference listings permit the data rules to be readily reviewed for omissions and inconsistencies before they are buried in detailed flow charts, control panel wiring and technical machine instructions.
5. It provides flexibility in changing any portion of the analysis.
6. By requiring the frequency of execution of a process, the best processing medium, process organization, programming and equipment requirements can be more readily determined.
7. The analysis provides material for auditing procedures.

This method of analysis was conceived and experimented with in the fall of 1958. The experiment was to restate an analysis of a computer process which contained 200 typewritten pages of narrative, tables, flow charts and block diagrams. The analysis was characterized by omissions, inconsistencies and errors. The restated analysis was depicted in tabular format on five 3' x 5' sheets. From this experiment the above and other advantages were gained. Many other concepts with great potential in the design of automatic programming systems have resulted.

Grad, B. TABULAR FORM IN DECISION LOGIC. in Datamation, Vol. 7, No. 7, pages 22-26. July 1961.

Tabular form has shown promise of being an effective way to organize and present decision logic for systems analysis and computer programming. Experience

to date clearly indicates the need to determine its range of application and assess its future potential. This report has the dual purpose of sketching the historical background on the development of tabular form, and indicating its possible advantages.

Grad, B. STRUCTURE AND CONCEPT OF DECISION TABLES. in Proceedings of the Decision Tables Symposium, pages 19-28. 20-21 September 1962.

Decision Tables, a recent development, provide a means of presenting complex decision logic in a way that is relatively easy to prepare and understand. A decision table shows the specific alternative courses of action to be taken under various combinations of conditions. This permits an analyst or programmer to concisely and completely record logical decision rules for analysis, documentation and programming.

Hawes, M.K. THE NEED FOR PRECISE PROBLEM DEFINITION. in Proceedings of the Decision Tables Symposium, pages 13-18. 20-21 September 1962.

The need for precise problem definition is one of the greatest facing the users of electronic computer systems today. Experience indicates over 65% of the costs associated with programming data processing problems can be attributed to this need. Looking ahead to real-time information processing systems, the need becomes even greater and, furthermore, must be handled at the systems level.

Hawes, M.K., et. al. DECISION TABLE TUTORIAL USING DETAB-X. CODASYL Systems Development Group, 49 pages. 1962.

Holstein, D. DECISION TABLES, A TECHNIQUE FOR MINIMIZING ROUTINE REPETITIVE DESIGN. in Machine Design, Vol. 34, No. 18, pages 76 -79. 2 August 1962.

IBM 1401 DECISION LOGIC TRANSLATOR (1401-SE-05X), APPLICATION DESCRIPTION. IBM Corp., H20-0063-0, 2 pages.

This program accepts decision tables written in a FORTAN-oriented language and automatically translates them into a FORTRAN II source program, giving

pertinent diagnostics in the process.

IBM 1401 DECISION LOGIC TRANSLATOR (1401-SE-05X), PROGRAM REFERENCE MANUAL. IBM Corp., H20-0068-0, 54 pages.

Design logic is captured using a FORTAN-oriented decision table language. The logical statements of this language are the input to the Decision Logic Translator system. After decoding the statements of a table, the system sorts them according to commonalities in rows and columns in order to produce an efficient output program. The sorted rules are then translated into FORTRAN statements. This process is continued table by table until all tables of any single run are translated into FORTRAN statements.

Kavanagh, T.F. TABSOL, A FUNDAMENTAL CONCEPT FOR SYSTEMS-ORIENTED LANGUAGES. in Proceedings of the Eastern Joint Computer Conference, Vol. 18, pages 117-136. 13-15 December 1960.

Lack of efficient methods for thinking-through and recording the logic of complex information systems has been a major obstacle to the effective use of computers in manufacturing businesses. To supply this need, this paper introduces and describes decision structure tables, the essential element in TABSOL, a tabular systems-oriented language developed by the General Electric Company. Decision structure tables can be used to describe complicated, multi-variable, multi-result decision systems. Various approaches to the automatic computer solution of decision structure tables are presented. Some benefits which have been observed in applying this language concept are also discussed. Decision structure tables appear broadly applicable in information systems design. In addition, they are of interest because they revise many earlier notions on problem formulation and systems analysis techniques. Decision structure tables will be an available feature in GECOM, General Electric's new General Compiler, which will be first implemented on the GE 225.

Kavanagh, T.F. TABSOL--THE LANGUAGE OF DECISION MAKING. in Computers and Automation, Vol. 10, No. 9, pages 15, 18-22. September 1961.

Kavanagh, T.F. MANUFACTURING APPLICATIONS OF DECISION STRUCTURE TABLES. in Proceedings of the Decision Tables Symposium, pages 89-97. 20-21 September 1962.

Kirk, H.W. USE OF DECISION TABLES IN COMPUTER PROGRAMMING. in Communications of the ACM, Vol. 8, No. 1, pages 41-43. January 1965.

A decision table is a tabular form for displaying decision logic. Decision tables have many inherent advantages from the programming point of view: (1) amount of computer memory used is drastically reduced, (2) programming is simplified, and (3) documentation is brief and clear.

The technique to be illustrated puts these advantages to use in that it enables one to program directly from a decision table. The technique is based on the creation of a binary image of a limited entry decision table in computer memory. A binary image of a given set of input conditions can also be created. This data image is used to scan the decision table image to arrive at the proper course of action.

Klick, D.C. TABSOL: A DECISION TABLE LANGUAGE FOR THE GE 225. in Preprints of Summaries of Papers Presented at the 16th National Meeting, Association for Computing Machinery, paper 10B-2. 5-8 September 1961.

Lombardi, L.A. A GENERAL BUSINESS-ORIENTED LANGUAGE BASED ON DECISION EXPRESSIONS. in Communications of the ACM, Vol. 7, pages 104-111. February 1964.

Montalbano, M. TABLES, FLOW CHARTS, AND PROGRAM LOGIC. in IBM Systems Journal, Vol. 1, pages 51-63. September 1962.

Decision tables are introduced with reference to business data processing. A method of verifying both the completeness and consistency of a problem description is given. The conversion of tables to computer programs is considered and a technique of obtaining a computer program which minimizes the branching requirements with respect to both memory and execute time is included. Program debugging and program modification are also discussed.

Naramore, F. APPLICATION OF DECISION TABLES TO MANAGEMENT INFORMATION SYSTEMS. in Proceedings of the Decision Tables Symposium, pages 63-74. 20-21 September 1962.

Since 1958, Sutherland Company has been employing decision tables for documenting management information systems. Major advantages realized through

these techniques may be enumerated as follows: (1) The ability to clearly and concisely state system requirements totally independent of procedures and processing media; (2) A uniformly high quality in the statement of system requirements; (3) The ability to associate defined decisions with responsible organizational entities; (4) An effective method for man-to-man communications; (5) The ability to establish an information repository for system specifications. The composite result may be summarized as the capability for complete and accurate definition of the "what" of a system, independent of, but relatable to, the myriad of procedural details constituting the "how."

Nickerson, R.C. AN ENGINEERING APPLICATION OF LOGIC-STRUCTURE TABLES. in Communications of the ACM, Vol. 4, No. 11, pages 516-520. November 1961.

Pollack, S.L. and K.R. Wright. DATA DESCRIPTION FOR DETAB-X (DECISION TABLE, EXPERIMENTAL). RAND Corp., RM-3010-PR, 46 pages. March 1962.

Pollack, S.L. DETAB-X: AN IMPROVED BUSINESS-ORIENTED COMPUTER LANGUAGE. RAND Corp., RM-3273-PR, 18 pages. August 1962.

This Memorandum describes DETAB-X (Decision-Tables, Experimental). In an effort to illustrate some of the features of DETAB-X, it is compared with COBOL-61 (Common Business-Oriented Language), using examples of data and procedures written in both languages.

Pollack, S.L. WHAT IS DETAB-X? in Proceedings of the Decision Tables Symposium, pages 29-39. 20-21 September 1962.

DETAB-X is an experimental language that combines COBOL-61 and decision tables. It is a proposed supplement to, not a replacement of, COBOL-61.

Pollack, S.L. ANALYSIS OF THE DECISION RULES IN DECISION TABLES. RAND Corp., RM-3669-PR, 69 pages. May 1963.

This memorandum develops a theoretical structure for decision tables. The theorems developed in this paper provide a basis for system analysts and programmers to verify the logic of their analysis. Rules are established that enable them to insure the following: (1) that all possible combinations of

conditions for the problem have been considered; (2) that the system does not prescribe different actions for the same situation; and (3) that the system describes each situation and its action once only.

The immediate effect of achieving the above is an improvement in computer programming by reducing the number of computer instructions, shortening computer running time, and decreasing programming and debugging time. In the future, we can expect computers to take over the task of checking decision tables for completeness, redundancies, and inconsistencies, using the rules developed here. The text also presents an extension of decision table theory. Most current decision tables consist of decision rules for which every condition in a set of conditions must be satisfied before a series of actions can be taken. This memorandum provides a basis for having additional decision rules in which a series of actions can be taken if any one of a set of specified conditions is satisified. This type of decision rule can be extremely useful in editing and information retrieval. This extension should prove valuable in many data processing areas.

Pollack, S.L. HOW TO BUILD AND ANALYZE DECISION TABLES. RAND Corp., P-2829, 17 pages. 12 November 1963.

Describes the conversion of system applications to decision tables, a process which entails making decisions on how large the individual tables should be and what system parameters should be included. A technique for reducing the number of written decision rules is also described.

Once decision tables are written, they should be checked for completeness and consistency. This paper will describe and illustrate the rules that enable system analysts to insure the following: (1) that all possible combinations of conditions for the problem have been considered; (2) that the system does not prescribe different actions for the same situation; and (3) that the system describes each situation and its actions once only.

Pollack, S.L. CONVERSION OF LIMITED-ENTRY DECISION TABLES TO COMPUTER PROGRAM. RAND Corp., RM-4020-PR, 15 pages. May 1964.

Decision tables are useful for describing complex decision rules based on given sets of conditions. Algorithms that can efficiently convert the tables into computer programs will extend the usefulness of decision tables to computer users. This Memorandum describes two such algorithms, based on work done by M.S. Montalbano and extended here to handle dashes and ELSE-decision rules. The first algorithm minimizes the computer storage space required for the resultant program, the second minimizes computer running time. During the conversion process, both pinpoint any contradictions or redundancies among the rules in a table.

A necessary adjunct to minimizing computer storage or running time is the allowable reduction of the number of rules in a decision table. This Memorandum describes a technique to effect this reduction for pairs, triplets and quadruplets of rules. The system analyst will find this method most helpful for pairs, and generally unprofitable for n-tuplets greater than three. The technique can be done manually or accomplished by the computer as a prelude to executing one of the two algorithms.

Pomeroy, L.K. Jr. ROAD MAPS TO DECISIONS. in Navy Management Review, Vol. 10, No. 1, pages 4-5. January 1965.

PROCEEDINGS OF THE DECISION TABLES SYMPOSIUM. Sponsored by the CODASYL Systems Group and the Joint Users Group of ACM, 116 pages. 20-21 September 1962.

This document contains the proceedings of a Symposium on Decision Tables presented September 20-21, 1962 in New York City. The Symposium was co-sponsored by the Systems Group of CODASYL, and by the Joint Users Group.

Schmidt, D.T. and T.F. Kavanagh. USING DECISION STRUCTURE TABLES. in Datamation, Vol. 10, Nos. 2 and 3, pages 42-49 and 48-54. February and March 1964.

These articles emphasize manufacturing applications because most of our experience is in this area. Decision structure tables coupled with computers are paying off because they allow you to: define and think through manufacturing problems, often providing new insights and understanding which have led to improved performance; formulate and record decision systems for subsequent use and communication; simplify computer implementation where mechnization is desirable; get manufacturing to using computers.

There is a wealth of potential computer applications in manufacturing. They offer great opportunity. Without structure tables, application costs would be exorbitantly high. It is easy to learn how to use decision structure tables, and, further, the user requires minimum computer knowledge and background. Later in these articles a structure table application using computers is described--PRONTO.

TABSOL APPLICATION MANUAL, INTRODUCTION TO TABSOL. GE Computer Dept., CPB-147A, 23 pages. June 1961.

TIME TO CONSIDER DECISION STRUCTURE TABLES AND EDP DESIGN SESSIONS. in EDP Analyzer, Vol. 1, No. 4, Canning Publications, Inc., 10 pages. May 1963.

Decision structure tables provide a powerful tool for systems analysis, for prescribing clerical procedures, and for programming. Design sessions can help develop the vitally necessary support of middle management for your EDP program. What's more, both are easy to use.

Wright, K.R. APPROACHES TO DECISION TABLE PROCESSORS. in Proceedings of the Decision Tables Symposium, pages 41-44. 20-21 September 1962.

Discusses the four basic types of processors or methods of converting decision tables to a machine language. These are (1) the manual processor, (2) the interpretive processor, (3) the translator, and (4) the compiler.

ABSTRACT

The DETAB/65 preprocessor converts limited-entry decision tables contained within COBOL programs into a form acceptable by a COBOL compiler.

The preprocessor was designed to facilitate easy modification for various COBOL implementations. It can be used either alone (as a preprocessor) or incorporated into a COBOL compiler.

The preprocessor has been successfully compiled and executed upon the CDC 1604-A, 3400, 3600 and on the IBM 7040, 7044 and 7090/94 computers.

The Preprocessor Package consists of the following:

1. ABSTRACT
2. The DETAB/65 Language
3. A Description of the Basic Algorithm Used In the DETAB/65 Preprocessor
4. DETAB/65 User's Manual
5. Decision Table Bibliography
6. Preprocessor Card Deck and Listing
7. Test Deck

THE DETAB/65 LANGUAGE

PREFACE

In June of 1963, Work Group 2 of the Special Interest Group on Programming Languages (SIGPLAN) of the Los Angeles Chapter of the Association for Computing Machinery was formed to develop a preprocessor for DETAB-X, a COBOL-oriented decision table language. The results of that effort are partially reflected in this manual, a revised set of specifications that, since they differ significantly from those originally set forth for DETAB-X, is denoted as DETAB/65.

The source document for this manual is "Preliminary Specifications for a Decision Table Structured Language-DETAB-X," issued by the CODASYL Systems Group at a symposium on decision tables co-sponsored by CODASYL Systems Group and JUG (Joint Users Group) in New York in September 1962. Upon these specifications the SIGPLAN work group has based further efforts to develop detailed program specifications. In doing so, these basic deviations from DETAB-X have resulted:

1. DETAB-X prescribed extensive revisions to the structure of the COBOL Data Division, most notably a fixed format for data declaration. In view of the efforts of other groups in developing fixed formats for COBOL, this portion of the DETAB-X language specifications was deemed an unnecessary and redundant effort.

2. In the interests of conserving space, DETAB-X specified several short forms and substitute expressions (DO rather than PERFORM, SET rather than COMPUTE, etc.) and placed restrictions upon allowable COBOL expressions within decision tables. In the interests of maintaining maximum compatibility with the COBOL language, placing as few restrictions on the programmer as possible, and keeping the decision table processor as simple as possible, these short forms and limitations have been removed. Any COBOL expression (legal or illegal) is permissible and may be used at the programmers' discretion.

3. The expression NOT has been added to Extended Entry Condition Entries, meaning all conditions not otherwise specified.

4. A special section of table-specific formulae were specified in DETAB-X. In DETAB/65, formulae too long to be part of the Condition or Action Stubs or Entries are relegated to COBOL code.

5. In DETAB/65, a decision table input to the preprocessor results in a COBOL Section being generated. These sections are set up as closed subroutines and must be treated as such by the accompanying COBOL code.

6. Several other changes of a more minor nature are treated in the text that follows.

SIGPLAN (DETAB) Working Group 2 was chaired by Wim Boerdam of Richfield Oil The principal participants were:

Mike Callahan	System Development Corporation
Anson Chapman	System Development Corporation
Charles M. N. Cree	International Business Machines
Robert L. Dover	Control Data Corporation
Stanley Naftaly	Lockheed Aircraft
Soloman L. Pollock	North American Aircraft
Wylie Robertson	International Business Machines (originally Northrop)
Richard W. Senseman	UNIVAC
Ralph Shoffner	Informatics
Barry Smith	Control Data Corporation
N. E. Willmorth	System Development Corporation

Other participants have been George Amerding of RAND Corporation, who advised the group on FORTAB, R. T. Fife (now of UNIVAC), Leonard Longo (then of Douglas), C. J. Shaw of SDC (and presently Chairman of SIGPLAN), Charles Powell of Richfield, and Ed Manderfield of North American Aviation and previously Chairman of SIGPLAN.

These specifications and the DETAB/65 processor are being distributed through JUG. Requests for copies of the specifications or for the DETAB/65 processor should be addressed to:

Miss Joan Van Horn
Secretary, JUG
MITRE Corporation
Bedford, Massachusetts

or an affiliate of the JUG organization. Correspondence on technical error, comments, criticisms and suggestions may be directed to:

Mr. Wim Boerdam
Richfield Oil Corporation
645 South Mariposa
Los Angeles, California

CHAPTER I
THE DETAB/65 LANGUAGE

PURPOSE

The purpose of DETAB/65 is to provide a practical foundation for experimenting with a decision-table based language. The language is designed to be convenient for preparing a preprocessor to go from DETAB/65 to COBOL-61. As such, many restraints and limitations have been placed upon the language to make it readily compatible with COBOL-61.

The DETAB/65 language is designed to fit within the framework of the COBOL language. Decision tables input to the DETAB/65 processor will be output as closed subroutines and treated as COBOL sections. Normal COBOL program formats are used. Symbolic data references made inside decision tables must be declared in the DATA DIVISION just as for non-decision table sections. Formulae whose names are given in a table must be expressed in the PROCEDURES DIVISION prior to entering the DETAB/65 section that gives the formulae values. Within a decision table all normal COBOL expressions may be used, plus a few minor language extensions and symbology necessary to the direction of the DETAB preprocessor and construction of DETAB expressions.

DECISION TABLES

Of the various activities that go into setting up a data-processing procedure for a computer, one of the more difficult is the development of a definition of exactly what is to be done under all combinations of circumstances of the data processing problem. Every problem step must be specified. The conditions to normal processing must be identified. Necessary sequences of operations must be precisely indicated.

Determining what is required of the computer system is called analysis; deciding how to go about meeting these requirements is the area of system design; and communicating the chosen procedure to the computer is called programming. In each of these areas a language is needed for defining the data-processing procedures. Ideally, a language form or structure should be suitable for man-to-man and man-to-machine communication.

Many languages are used for these purposes. Procedures are often communicated to the machine in a form closely resembling the language of the machine. Symbolic logic and equations are sometimes used, but this imposes a heavy and unnecessary burden on the person writing the procedures. This condition occurs because human language, used for man-to-man communication, and machine language are quite different. Flow charts are widely used for man-to-man communication about data-processing procedures. However, such charts have several drawbacks: Flow charts can become confusing in complex situations; it is relatively difficult to check all possible paths; and the flow chart form is not particularly suitable for direct communications with the machine. Flow charts sometimes present logical equations, but they do not display relationships in as graphical a form as one might wish. Furthermore, they are not a comfortable form of expression for most system designers, except to the person who designed the program.

Decision tables offer the promise of nullifying and correcting many of these language objections. Decision tables provide a graphical representation of complex procedures in a way that is easy to visualize and understand. They show alternatives and exceptions much more explicitly than other language forms. They present relationships among variables clearly. They show the necessary sequences of conditions and actions in an unambiguous manner. Decision-table form can be used with equal effectiveness

for system analysis, procedure design, and computer coding. Thus, a computer procedure written as a set of decision tables is, to a large extent, its own documentation.

	RULE 1	RULE 2	RULE 30
AGE GREATER THAN	25	25	65
AGE LESS THAN	35	35	
HEALTH EQUALS	"EXCELLENT"	"EXCELLENT"	"POOR"
SECTION-OF-COUNTRY EQUALS	"EAST"	"WEST"	"WEST"
SET RATE-PER-1000 EQUAL TO	1.57	1.72	5.92
SET POLICY-LIMIT EQUAL TO	200000	200000	20000

FIGURE 1. AN EXAMPLE OF A DECISION TABLE

There is a growing body of evidence to indicate that these claims are justified. Those who have used decision tables for man-to-machine work say that:

1. programming is much faster;
2. program checkout time is significantly reduced;
3. the use of tables leads to greater accuracy and completeness in problem formulation;
4. program maintenance is simpler; and
5. a program written in tabular form is indeed a powerful communication and documentation device.

STRUCTURE OF A DECISION TABLE

Figure 1 is an example of a simple decision table. The use of such a table is illustrated in the following statements about Rule 1:

Rule 1 says: If age is greater than or equal to 25 and age is less than 35, and health is excellent, and section of country is east, then rate per thousand is 1.57 and policy limit is 200,000. The underlined words are implied by the table layout. The quote marks in the table are used to differentiate non-numeric values from names (as in COBOL-61). Each rule of a decision table is an alternative to each other rule. Therefore, logically it does not matter which rule is examined first; at most, one rule can be satisfied by a single set of conditions.

To more clearly indicate the parts of a table and the terms that are used to describe them, the information in Figure 1 is shown in an exploded view in Figure 2. The double lines serve as demarcation: The condition stub is shown in the upper left corner; the action stub below; the condition entry is in the upper right portion; and the action entry is in the lower right. Each vertical combination of condition and action entries is called a decision rule. The essential nomenclature is completed by adding at the top of the table a title section, called a table header, and by adding a rule header over the entries.

A more detailed description of decision table structure showing the actual location of the various segments of this sample table on a coding form may be found in Chapter III.

As shown in Figure 2, tables may be used in a slightly different way to state decision logic.

	RULE NO.			
TABLE-CREDIT-APPROVAL	1	2	3	4
CREDIT-LIMIT-OK	Y	N	N	N
PAY-EXPERIENCE-FAVORABLE		Y	N	N
SPECIAL-CLEARANCE-OBTAINED			Y	N
PERFORM APPROVE-ORDER	X	X	X	
PERFORM RETURN-ORDER-TO-SALES				X

FIGURE 2. A LIMITED ENTRY TABLE

Note that the form of the individual conditions and actions is somewhat different between Figure 1 and Figure 2. In a limited entry table the entire condition or action is written in the stub; the condition entry is limited to 'Y,' 'N,' '-' or blank. That is, asserting (Y), reversing (N), or ignoring (- or blank) a condition. An action entry is limited to 'X' or '-' or blank. That is, executing (X) or skipping (- or blank) an action. In contrast, an extended entry form (as in Figure 1) has part of the condition or action extended directly into the condition or action entry area. Both forms may be used within one table, but any one horizontal row (condition or action) must be entirely limited or entirely extended.

This example points out that the basic concept of a single rule in a table is quite straightforward, being used on the "if...then..." relationship. If A = B, and C is greater than 5, and...then assign the value 7 to X, and GO TO Table 10. The interpretation is: If all the conditions in rule 1 are not met, then try rule 2, etc. Continue for rules 2 and 3, 3 and 4, etc., until a rule is satisfield. The program must still be told what to do if all rules have been considered and the set of conditions that exist do not satisfy any of them. Therefore, the last rule in every decision table is the ELS-rule in which we tell the program what "else" to do if no rule is satisfied. An ELS is written in the rule header entry as the last rule in the table. If no ELS is given, the program will enter an automatic error routine. The flow chart in Figure 2 shows schematically the way in which a table is executed. In practice, the actual solution technique may vary, but the logical result remains the same.

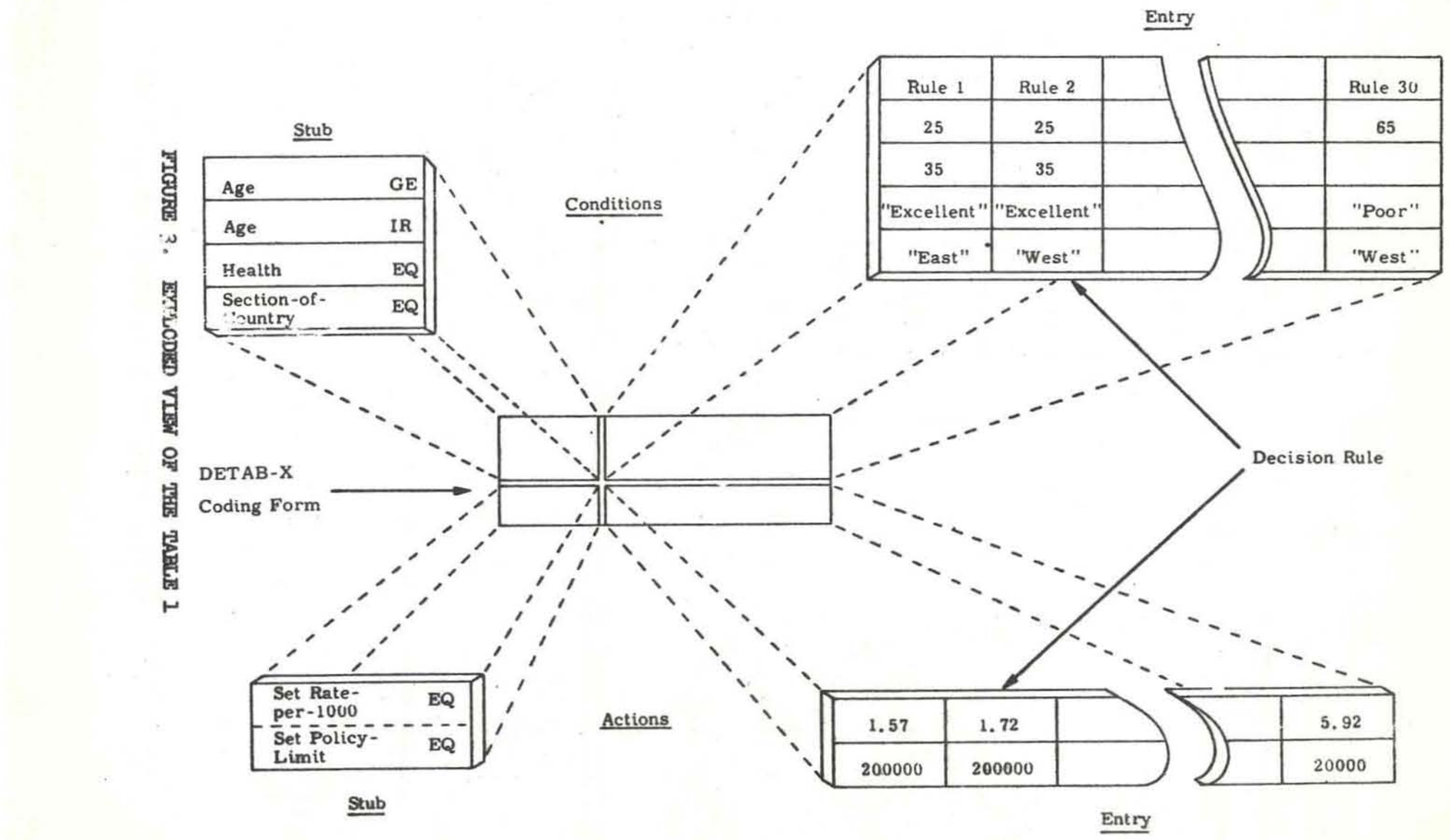


FIGURE 3. EXPLODED VIEW OF THE TABLE 1

DECISION TABLE

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

FLOW CHART OF ABOVE DECISION TABLE

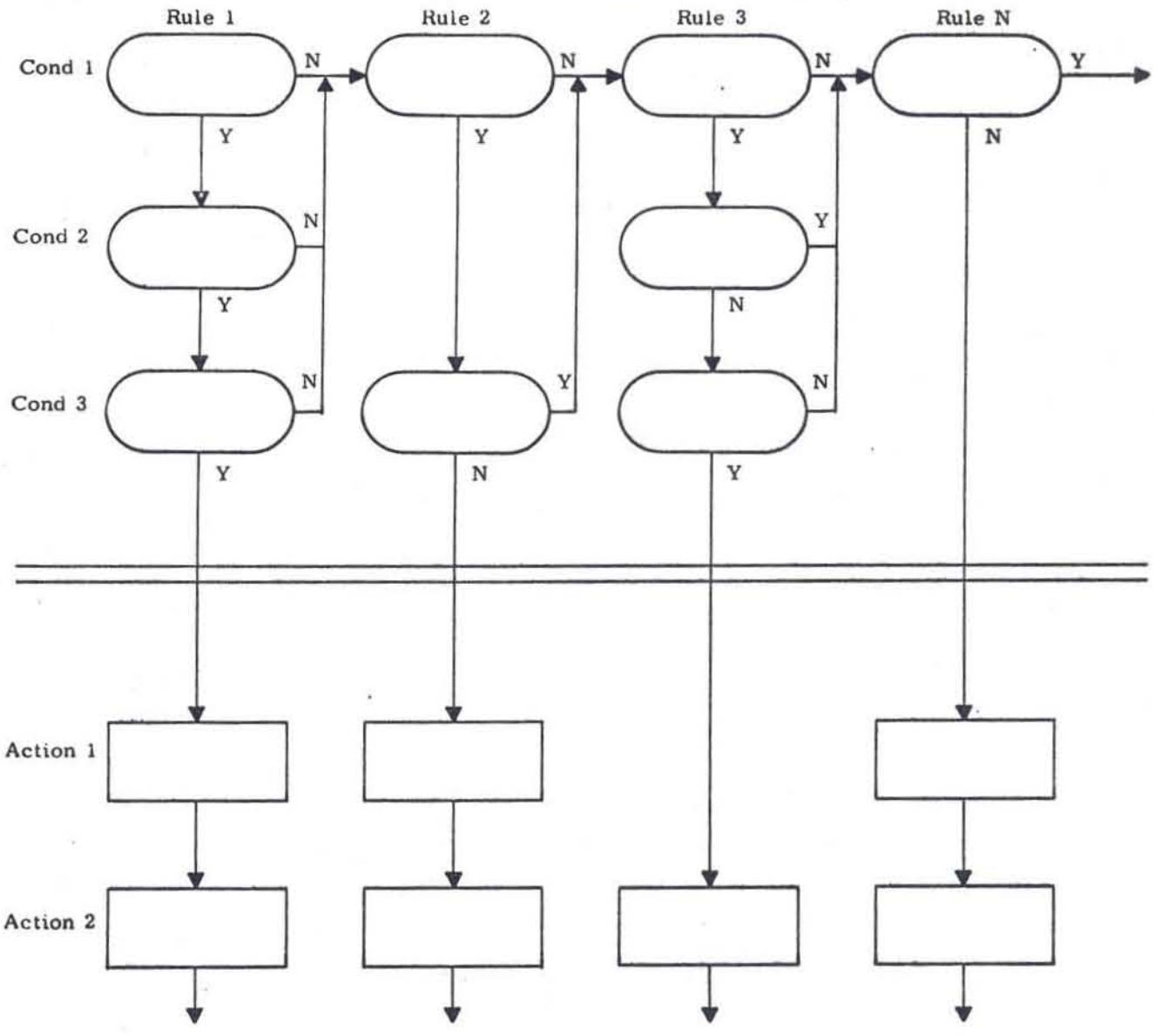


FIGURE 4. SCHEMATIC REPRESENTATION OF THE SEQUENCE OF TESTS AND ACTIONS IN EXECUTING A DECISION TABLE

ORGANIZATION OF THE MANUAL

The balance of this language specification manual covers the individual areas of the language, indicating the characteristics and restrictions. In general, the rules of COBOL-61 are followed explicitly except in matters of format. Where there are differences, these are noted or are self-evident through the text itself.

CHAPTER II
SOURCE PROGRAM FORMAT

The format for a COBOL program containing DETAB/65 decision table expressions must conform to the requirements for any COBOL program, except that a decision table may be inserted in the PROCEDURES DIVISION as a SECTION. In compilation a decision table will be treated as a closed subroutine; that is, as a closed COBOL PROCEDURE. Within the table, of course, transfers to other than the normal return point may be specified.

IDENTIFICATION DIVISION

A normal COBOL division name followed optionally by a PROGRAM-ID and other identifying information must be given. No special requirements are levied by DETAB/65.

ENVIRONMENT DIVISION

The Environment Division must be filled out as required by the particular implementation of COBOL-61. Minimum required entries are CONFIGURATION SECTION with SOURCE-COMPUTER, OBJECT COMPUTER and SPECIAL NAMES, INPUT-OUTPUT SECTION with FILE CONTROL and I-O-CONTROL.

DATA DIVISION

The Data Division for a program incorporating DETAB/65 sections is treated as is any other COBOL Data Division. Any symbolic data references used in a decision table must be declared in the Data Division as for any other COBOL procedures section. Any data structures, working storage and constants used by the program must be described in the Data Division. Symbolic references used within a decision table must conform to the requirements of COBOL data references. Any COBOL data description forms that have been implemented in a particular COBOL processor may be used.

The normal COBOL character set, as implemented in a particular computer, may be used to form NAMES.* One such name will be a DECISION-TABLE-NAME, which is a name given to the procedure table that describes a series of conditions and actions, and which is equivalent to a PROCEDURE-NAME in COBOL-61. A DECISION-TABLE-NAME is a SECTION-NAME and may be composed of alphabetic, numeric, alphanumeric, or combinations of these characters joined by one or more hyphens (-). The DECISION-TABLE-NAME must be used to call a table from the main sequence of COBOL instructions.**

---

*NOTE: Special symbols suggested by the DETAB-X Manual have been rejected in favor of normal COBOL forms, i.e., /= (not equal), <= (less than or equal), >= (greater than or equal to) are rejected.

**NOTE: The so-called "short form" of decision table names specified in DETAB-X will not be implemented in DETAB/65. Neither will the capability of calling the TABLE-ID (e.g., GO TO TAB XXX) rather than the table name.

PROCEDURES DIVISION

The Procedures Division of a COBOL source program is used to specify the logical decisions and actions that provide the desired processing. Procedures are normally written as COBOL SENTENCES that are combined to form COBOL PARAGRAPHS, one or more of which may be combined to form a COBOL SECTION.

However, within a section that is a decision table, the normal sentence structure of the COBOL language is abandoned in favor of the more formal structure of the decision table. The syntactic content of the decision table structure may be interpreted as a complex set of conditional statements, plus the information necessary to initialize the closed subroutine that the table represents. A decision table may not be entered via the normal operating sequence and may be referenced by a GO TO, or a PERFORM, but not by an ENTER. A GO TO results in an unconditional or conditional transfer to the specified decision table. For a GO TO from the main processing sequence from another table, a return or transfer point must be specified by the programmer in the decision table actions or the processing sequence will be lost.

If the transfer to the table is accomplished by a PERFORM, a normal return to the processing sequence will be made unless the programmer specifies otherwise. Specifically, the DETAB/65 processor will generate a GO TO DEXIT for every rule that does not end in a GO TO, however the table was entered.

CHAPTER III
DECISION TABLES

This chapter describes the decision table format expected by the DETAB/65 processor. The processor will accept both extended and limited entry tables.

Each of the sections of a decision table will be discussed and each of the entries permissible in a section will be described and the reasons for the various requirements and rules governing the entries are given. The description assumes that punched card or card images are used as the input mode. If punched tape, typewriter keyboard or other continuous input mode is adopted considerable revision of the DETAB/65 processor Data Division would have to be respecified.

A decision table consists of 6 parts or sections: a Table Header, a Rule Header, a Condition Stub, a Condition Entry Area, an Action Stub, and an Action Entry Area. The functions of these sections are demonstrated briefly in Chapter I. A sample DETAB/65 specification form is shown in Figure 5 to provide a ready reference for the reader as the decision table sections and entries are discussed.

FORM HEADER

The Form Header serves to identify system, program, and author of completed Decision Table Input Forms, and to specify the data of completion and enumerate the pages. None of this information is part of the decision table proper.

TABLE HEADER

Each table has two header lines: 1) a Table Header that serves to identify the table and provide information that covers the table as a whole and 2) a Rule Header that is used to indicate rule numbers.

A table may require more than one page, either because of a large number of rules, or because of a large number of conditions and actions. The table header card for the subsequent pages should not be filled in, but, in the case of row continuation (i.e., due to more rules than will fit on a page), a continuation flag - the number "1" - should be set in the RSET rules set entry area of the initial header card. The flag should not be set for multiple pages due to condition and action overflow (i.e., overflow of the entries for a rule or other column onto another page).

The entries in the Table Header are:

TABLE ID

Each table must be identified by a three-digit number (e.g., 001, 074, 694). This number is nominative only and table ID's need not be sequential nor ordered in any way, but each table ID must be unique within a program.* The Table ID is repeated for every row of a table.

---

*NOTE: DETAB-X specifications permitted the TABLE ID to be substituted for the Decision Table Name in calls to the table. This has not been implemented in DETAB/65. TABLE ID serves only to distinguish one table from another in sequence checking or EAM processing of cards.

DETAB / 65 PROCEDURAL DIVISION

SYSTEM _____ SECTION _____ PREPARED BY _____ DATE _____ PAGE NO. ___

TABLE I.D.	ROW NO.	LINE	DSET	TABLE NAME	FORM	PARAMETERS: COND ROWS	PARAMETERS: ACTION ROWS	PARAMETERS: RULES
	000	0						
				RULE NUMBERS				
	000	1						

Row No.

The Row Number is a three-digit number that for the Table Header is always 000. This designation indicates to the processor that this is a Header Card.

Line

The Line entry is one alphanumeric character that for the Table Header is always "0". This designation indicates to the processor that this is a Table Header Card.

RSET

The Rules Set (RSET) entry is left blank unless more rules than can be specified on one page (i.e., card) are required. If more than one page is used, a numeral "1" is entered in the RSET entry of the Table Header Card. If an entry is made in the RSET entry of the Table Header, a single numeric digit must appear in the RSET entry of all other rows of the decision table.

Table Name

A table may be given any name that conforms to the specifications of COBOL for a procedure-name, usually indicating the function or content of the table. That is, a procedure-name may be composed of alphabetic, numeric or alphanumeric characters (at least one but not more than thirty) or sets of such characters separated by hyphens (-). However, a hyphen may not begin or end a name. Names must be unique. The Table Name may be referenced by GO TO and PERFORM operators. The DETAB/65 processor will use this name in generating a COBOL section-name for the table. A dummy paragraph-name must also be generated and inserted after the section-name.

Form

Three basic formats are permitted for decision tables: limited entry (L), extended entry (E), and mixed entry (M). The format of the decision table being specified is indicated by placing one of these three values (L, E, or M) in the Form entry. That is, one alphabetic character.

Cond Rows

A three-digit number containing the number of conditions rows (not lines) in the table. That is, this entry specifies the number of conditions that are contained in the table. Leading zeros must be given.

Action Rows

A three-digit number containing the number of action rows (not lines) in the table. That is, this entry specifies the number of actions that are contained in the table. Leading zeros must be given.

Rules

A three-digit number specifying the total number of rules in the table. The ELS rule should be included in the count. Leading zeros must be given.

RULE HEADER

The second header card is the Rule Header used to specify the rule numbers for the table. Besides rule numbers, it will contain the identifying information given for every line of a decision table. As many Rule Header cards will be used as are required to specify all rules.

Table ID

Always the same as the Table Header.

Row No.

The Row Number (ROW-NO.) for a Rule Header Card is always 000, indicating that this is a Header card.

Line

The Line (LINE) entry for a Rule Header is always "1" indicating that this is a Rule Header.

RSET

The Rules Set (RSET) entry will be blank except when the number of rules specified requires more than one page. If more than one page is required, the RSET entry will be "1" for the first page (or card) and following pages will be numbered sequentially.

Rule No.

A Rule Number must be a three-digit number (e.g., 001, 002, 142, etc.) or ELS. (Every table must have an ELS-rule as the last rule to be given.) The first Rule Number is entered in the Rule Header Card in the first space available beyond the longest entry in the Condition Stub, although blanks may be left between the last character of the longest condition and the first Rule Number. The column that the first digit of the first Rule Number occupies defines the point at which the DETAB/65 processor construes the Condition Entry Area to begin. Spaces occupied by Rule Numbers may vary from three through twelve, but the Rule Number must begin in the leftmost column of the area to be reserved ror the rule. The spaces beginning with the first digit of one Rule Number and ending at the last digit before the next is interpreted by the DETAB/65 processor as the number of spaces that are reserved for the condition entries subsumed under that Rule: The number of spaces thus reserved must be equal to or greater than the longest Condition Entry in that rule. The number of spaces reserved may vary from rule to rule, but must not be less than three nor more than twelve. The end of the last rule, and the end of the rules, is indicated by placing a "$" in the first space beyond the last space required by the last rule (i.e., the ELS-rule).* A rule entry may not be split between pages.

*NOTE: An exception is made in the case where the last (ELS) rule includes the 80th column of the card. That is, if RSET = 0, the card column count = 80, the rule header entry is ELS, and the spaces reserved less than or equal to 12, then this is the last rule. Otherwise, an error has occurred.

If, in an extended entry table, there are not enough columns left on a page to contain all of a particular rule, a new page must be started and an intermediate end-of-rule marker (a "$") set to define the end of the preceding rule and to inform the processor to go on to the next card. When the processor encounters a $, it will check the preceding rule for an ELS-rule. If the preceding rule is an ELS, search for further rules is stopped and the processor proceeds to the next card. NOTE: Although Rule Numbers should be sequential and entered in ascending order (e.g., 001, 002, 003, etc.), this does not imply that the rules will be executed in that order. The DETAB/65 processor evaluates the matrix of condition entries and optimizes the decision tree for effieient processing.

CONDITION AREA

The cards following the Rule Header Card are used to specify sets of conditions. Each row specifies the states, specific values, or ranges of values that a particular piece of data may assume, or relationships to other data or combinations of these states that the data may assume, and upon which decisions are to be based.

Structurally, a condition consists of two parts: (1) a Condition Stub and (2) a Condition Entry. The Condition Stub area consists of the entries for Table ID, Row No., Line, RSET, and a conditional statement, or portion thereof. The Condition Entry area consists of entries specifying the values of the data or condition specified in the stub that will satisfy the decision requirements of the rules.

The presence of a Condition Area in a table is not required; however, if it is absent, the table can have only one rule in the action area.

TABLE ID

As above, a three-digit entry uniquely identifying a particular decision table.

Row

Row number is a three-digit number used to identify particular conditions, actions, or notes. Condition and action row numbers may vary from 001 to 899; 900 to 999 will designate Notes. Conditions must have lower row numbers than do actions. Row numbers should be, but need not be, assigned in ascending order. Sequential numbering improves legibility of tables for later reference, but the processor may reorder the conditions in sorting to minimize the decision tree. Leading zeros must be filled in and duplicate row numbers are not permitted.* A row may cover as many lines on a specification form as are required to write out the operators and operands of an expression, up to the maximum permissible value of LINE (i.e., 9).

*NOTE: Presently, the number of conditions is limited to 50, although row numbers may run much higher.

Line

Line is a one-digit entry used to specify continuations of a row. A Line entry may be blank or range from 1 to 9. Lines, if specified, must be specified in exact sequence because following lines are considered as a continuous description of the condition, action, or comment being given. A blank entry signifies that only one line will be used; if a digit is given, the DETAB/65 processor will look for continuations of the line until the Row Number changes. For limited entry tables, line numbers greater than 1 may be dropped on continuation pages. For extended entry tables, lines not required by the condition entries may be dropped.

RSET

The Rules Set entry may be either blank or contain a digit from 1 to 9 or the symbol "$." As specified above, RSET is used to number the sub-tables required to specify all the rules of a table. If more than one sub-table is required, they are numbered sequentially; that is, the RSET entry will be the same for each condition, action or comment of any sub-table. RSET applies to row continuation horizontally and not vertically. The sheets containing the Condition and Action stubs are considered sub-table 1 no matter how many physical sheets are used. If more rules are required than can be contained in nine sub-tables, a new table must be created, using the same conditions and actions, and to which the ELS-rule may transfer if none of the rules in the first table are satisfied. The $ is used to indicate the end of the table. The entry for the end card will contain 9999$.

Deck Sequence

The entries for row, line and continuation are used in sequencing the deck for presentation to the processor. The expected deck order will be a sort on (1) row, (2) RSET, and (3) Line. The deck make-up is illustrated in Figure 6.

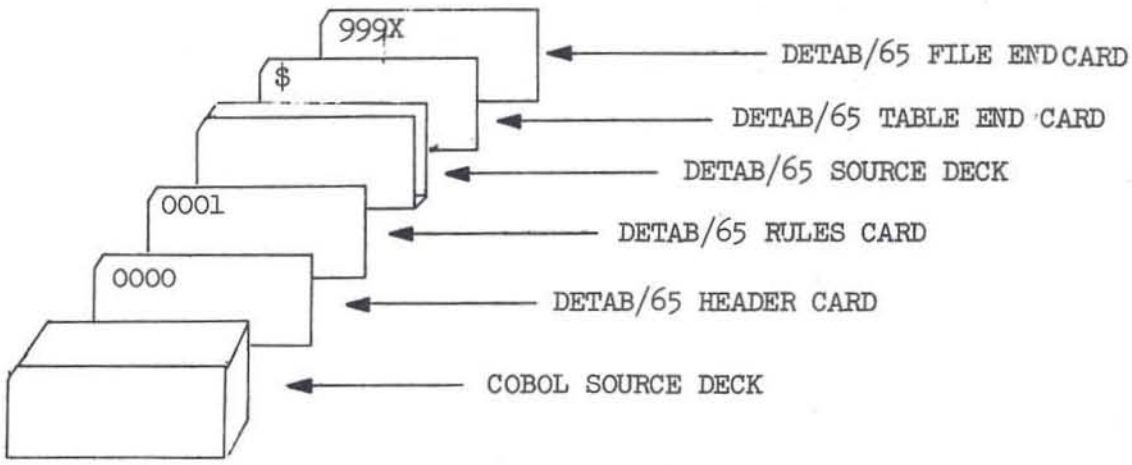


FIGURE 6. DECK FORMAT FOR A DETAB/65 PROGRAM

Condition Stub

Beginning in column 9 of the specification sheet and continuing for as many lines of a row (up to 9) as are necessary to contain it, any condition that constitutes a legitimate COBOL conditional statement (with the IF implied) may be specified. A condition stub must contain at least one operand. A condition stub entry is bounded by column 9 of the input card format and the first column of the first rule, but may be continued on several lines. A condition stub entry must be contained entirely upon one page (i.e., the first logical sub-table) in row-line length, but additional lines may be given on subsequent physical pages. Continuation of lines do not require hyphenation since continuations are treated as part of a single entry. Blanks other than those required because of language specifications are ignored at the beginning and end of lines, permitting the user to indent or organize operands in various ways to increase legibility.

Despite this flexibility in specifying conditions, it is recommended that condition stub entries be kept as short and concise as possible. Any lengthy calculations should be relegated to a separate expression and assigned a name that may be referenced in a condition stub entry.

Entry

A condition entry is specified within the bounds of a rule. That is, it begins in the column containing the left-most digit of its rule number and ends where the rule immediately to its right begins. A rule must not be less than three nor more than twelve columns in width, but an entry may be continued on the subsequent lines of a row.

Three kinds of condition entries are permitted: Limited, Extended, and Mixed. In a limited entry form, each rule or entry is three columns wide. Permissible entries are (a) Y (i.e., Yes), signifying that the stated condition must be true to satisfy the rule, (b) N (i.e., No), signifying that the condition must be false to satisfy, (c) - (i.e., a hyphen or blank), signifying that the condition may be either true or false, that the programmer doesn't care which it is. The entry must be made in the second column of the rule and have a blank space both to the right and to the left within the rule.

In extended entry form, leading and following blanks are permissible, but not required. Permissible entries are (a) and operator and an operand, (b) an operand, (c) -, or (d) a blank. In mixed entry format, the entries for any one condition may be either ext nded or limited, but not both.

In extended and mixed entries, a condition entry too long to be contained within the allotted 12 spaces may be continued on successive lines of a row. Limited entries in a mixed table must still contain a leading and trailing blank and appear in the second column of the rule.

The last rule will be the ELS-rule. The ELS-rule must be given, but all condition entries for the ELS-rule must be blank (or "-").

Neither names nor values may be split between stub and entry. At least one operand or operand and verb must appear in the stub, and at least one condition entry must be given for each condition. (Except in the case of the 'empty' table in which case no entries are made.)

A condition entry may not be split between pages or "sub-tables." (See Rule No., Rule Header section.) If there is less than 12 spaces left on a page and the entry exceeds the available space, the entry must either be made on a new page or sub-table, or multiple lines may be used.

Formula references must not duplicate any data references made in the data division.

On sub-tables beyond the first (i.e., RSET <1), condition entries must begin in column 9 of the DETAB specification sheet; that is, the first rule number on continuation pages should begin in column 9. Limited entries must still be in the middle column of the entry and extended entries may or may not have leading and trailing blanks.

In all sub-tables, condition entries must start in the first line of a row. For limited entry tables, only Line 1 of a row need be given in sub-tables beyond the first since all condition entries on other lines would be blank. The programmer may wish to retain the vertical alignment, however, to help avoid mistakes. In constructing a condition matrix for a limited entry table all lines except Line 1 will be rejected (i.e., not read into the matrix) and entries so misplaced will be lost.

For extended entries, the programmer may include as many lines as are necessary (up to nine) to express the longest entry; however, this number of lines may vary as required from sub-table to sub-table of the specifications. In writing extended entries, no continuation marks, such as hyphens, need be given since each line is picked up as a continuation of the row. Blanks should therefore be inserted where appropriate to avoid two words being read as one. In setting up the condition matrix, the input editor will continue reading cards into the matrix until a new row number is encountered, creating an image of the input condition matrix area.

In an extended entry table, the programmer may desire to specify a condition such that the value specified is <u>not</u> any of the values specified in any other condition entry in that row. He may do this by specifying "NOT." In the processor this expression will be expanded into an "N" entry and row in combination with every other condition in the row for which a value is given. Don't care entries ("-" or blank) will be ignored. In illustration, consider this example:

CONDITION STUB RULE NO.

COND.		001	002	003	004	005	006	007	008	009	010	011	012	ELS
001	A = 12	Y	Y	Y	Y	Y	Y	N	N	N	N	N	N	
002	B =	5	10	NØT	5	10	NØT	5	10	NØT	5	10	NØT	
003	C = OPEN	Y	Y	Y	N	N	N							
004	D = SUNDAY							Y	Y	Y	N	N	N	

This expands to:

		001	002	003	004	005	006	007	008	009	010	011	012	ELS
001	A = 12	Y	Y	Y	Y	Y	Y	N	N	N	N	N	N	
002	B = 5	Y	N	N	Y	N	N	Y	N	N	Y	N	N	
003	B = 10		Y	N		Y	N		Y	N		Y	N	
004	C = OPEN	Y	Y	Y	N	N	N							
005	D = SUNDAY							Y	Y	Y	N	N	N	

After optimization this table will read:

		001	004	002	005	003	006	007	010	008	011	009	012	ELS
001	A = 12	Y	Y	Y	Y	Y	Y	N	N	N	N	N	N	
002	B =	Y	Y	N	N	N	N	Y	Y	N	N	N	N	
003	B = 10			Y	Y	N	N			Y	Y	N	N	
004	C = OPEN	Y	N	Y	N	Y	N							
005	D = SUNDAY							Y	N	Y	N	Y	N	

FIGURE 7. EXAMPLE OF THE EXPANSION OF "NOT" AND OF OPTIMIZATION

Action Area

The action area is used to specify the actions the program is to take when the conditions satisfying the various rules are met. The actions represent the "THEN" part of the conditional statements for which the conditions are the "IF" portion. All the rules and principles stated for conditions apply equally to the action area. The action area consists of an Action Stub and an Action Entry area just as the Condition Area consisted of a Condition Stub and a Condition Entry area.

Table ID

The Table ID entry must be the same unique identifier as for the rest of the table.

Row

Row Number entries are a continuation of Row Numbers for the Conditions. However, while the rows need not be numbered in sequential order, it is recommended that they be so ordered, for they will be compiled and executed in the numbered sequence. That is, while condition rows are subject to reordering in the optimization process, actions are not, thus insuring that the program will take a sequence of actions in the order specified by the programmer. That is, the processor will not create any logical errors by reordering actions. (The programmer, however, is free to create his own.)

Line

The Line entry should be left blank if an action can be specified on a single line; otherwise the digits 1, 2, 3, etc., must be used in precise sequence so that the proper action statement is picked up.

RSET

The RSET entry is left blank if only one sub-table is required. If more than one sub-table is required to contain the additonal rules, RSET will be numbered 1, 2, 3, as required, up to 9.

Action Stub

The Action Stub must contain at least an operator. In a limited entry table it will contain the entire action in a single "row" using as many lines as re required to specify the entire action to be taken. In extended entry tables an expression may be split between the Action Stub and the Action Entries. Although names and values may be broken and continued from line to line, they may not be split between the Stub and an Entry. This does not prevent subscript values of a name being used as entry values to specify different table destinations resulting from different rules being satisfied.

While no restriction is placed on the kind or length of actions taken (with a procedure or another conditional, if you wish), it is recommended that all lengthy Action Stub and Action Entry entries be defined outside the table and referenced by an appropriate name.

Although, again, no restrictions are imposed, it is recommended that all calls to system (COBOL) subroutines be made in a general fashion. E. G., an I/O order should be expressed PERFORM READ, PERFORM WRITE, etc. This convention will enable the programmer to write a program that is compatible with the manner in which various COBOL I/O modifiers (AT END, EOF, ON SIZE, ETC.) have been implemented in different COBOL processors.

Action Entry

For limited entry tables, any action that is associated with a given rule will be indicated by placing an "X" in the central column of the three column entry space of line 1 (the first line) of the action row. The "X" must be followed and preceded by a blank.

Extended action entries will be treated in the same manner as extended condition entries. Care will be taken in generating COBOL statements not to disturb the basic operating sequence specified by the programmer.

NOTES

A note is a descriptive statement that has no functional significance and that carries the same uses and restrictions as notes in COBOL-61. A note has the same Table ID as the table that it is applied to. Notes are given a row number of 900 to 999 and note lines are numbered 1, 2, 3, etc., as for other rows and lines to maintain their sequential order. Notes may be written anywhere in the table except between the Table Header and Rule Header Cards. When the processor encounters a note row designator, it will transfer the card into the COBOL code area as a note without further processing.

Note, however, that as presently constituted all notes would be sorted to the end of the table by preprocessing to put the cards in order. If it is desired that notes be inserted in place, the cards may be hand-ordered if more than one page or sub-table is used, or not sorted if only one sub-table (the first) is used.

RESTRICTIONS AND USAGES

Certain restrictions and peculiarities of usage become apparent as decision table specifications are used in combination with the COBOL languages. These restrictions are, in general, not serious, but minor logical errors may be avoided if the programmer is aware of these.

LINKAGES

A decision table may be entered in a variety of ways. The preferred mode of entry is through a PERFORM verb, in which case a linkage back to the main control sequence may be generated. If the table is to be used iteratively on a succession of inputs or cases (as in a scan routine), then PERFORM is the logical choice. If tables are nested, the PERFORM verb should be used to step from one table to the next and back. If a table may be entered from any one of a set of points, the PERFORM verb and RETURN entry are most convenient.

A table may be entered through a GO TO, but in this case the processor has no means of knowing what location to return to in the main control sequence unless this location is explicitly stated in conjunction with the separate rules. Of course, if one of the functions of the table is to switch control to various program regions depending upon the conditions encountered, this may be the preferred mode of specification. A sequence of tables may be stepped through with a series of GO TO's, but it is felt that this mode is inferior to the use of PERFORM's.

Note that a table .may not be entered via an ENTER verb, nor may an ENTER verb be used within a table without creating difficulties. All information necessary for the processing of the table and its COBOL statements are contained in the header cards.

Note also that tables cannot be entered at any point except at the beginning. If entry is desired at intermediate points in a set of conditions, this effect may be achieved by creating a sequence of tables and chaining them together. If tables are nested, however, returns to points in the Action Area of previous tables may be made through the operation of the PERFORM verb and normal exits, or may be specified directly by a GO TO.

Note that if a GO TO table-name-1 is given within the range of a COBOL PERFORM... THRU that the sequence of control may be lost unless the programmer has established instructions that will get the sequence back within ghe loop. However, no such trouble should be encountered when a PERFORM table-name-1 is given unless the table contains only explicit GO TO's that pick up the sequence elsewhere.

ACKNOWLEDGMENT

The DETAB/65 Preprocessor was prepared by Working Group 2 on Decision Tables of the Special Interest Group on Programming Languages (SIGPLAN) of the Los Angeles Chapter of the ACM.

The initial distribution is being made by ACM Headquarters on behalf of the Joint Users Group.

```
000000 IDENTIFICATION DIVISION.                                          DETAB-65
000005 PROGRAM-ID. PREPROCESSOR FOR DETAB-65.                            DETAB-65
000010 AUTHOR. ANSON CHAPMAN.                                            DETAB-65
000015 DATE-WRITTEN. 12/30/64.                                           DETAB-65
000020 DATE-COMPILED.                                                    DETAB-65
000025 REMARKS.                                                          DETAB-65
000030     THE GENERATOR PORTION OF THE PREPROCESSOR ANALIZES A          DETAB-65
000035     DECISION TABLE AND GENERATES SIMPLE CONDITIONAL STATEMENTS    DETAB-65
000040     FOR Y'S, N'S AND BLANKS AND WILL GENERATE IF STATEMENTS FOR   DETAB-65
000045     ONE PATH THRU THE TREE THE ACTION CORRESPONDING TO THE PATH   DETAB-65
000050     IS GENERATED IN STMTS DX014 THRU DX032 THIS PATH IS DELETED   DETAB-65
000055     FROM THE TREE IN DX016 THRU DX020  DX301 THRU DX061           DETAB-65
000060     REINITIALIZES THE TREE, FINDS THE LAST NODE CONNECTED TO      DETAB-65
000065     THIS PATH AND COMES BACK TO DX003 FOR ANOTHER PASS THRU THE   DETAB-65
000070     NEXT PATH  THIS PROCESS IS REPEATED UNTIL IF STATEMENTS       DETAB-65
000075     HAVE BEEN GENERATED FOR ALL PATHS THRU THE DECISION TABLE     DETAB-65
000080     TREE STRUCTURE.                                               DETAB-65
000085 ENVIRONMENT DIVISION.                                             DETAB-65
000090 CONFIGURATION SECTION.                                            DETAB-65
000095 SOURCE-COMPUTER.  CONTROL DATA 1604A.                             DETAB-65
000100 OBJECT-COMPUTER.  CONTROL DATA 1604A.                             DETAB-65
000105 SPECIAL-NAMES.                                                    DETAB-65
000110     SYSTEM-INPUT-TAPE IS SIT.                                     DETAB-65
000115 INPUT-OUTPUT SECTION.                                             DETAB-65
000120 FILE-CONTROL.                                                     DETAB-65
000125     SELECT CARD-INPUT, ASSIGN TO SYSTEM-INPUT-TAPE MULTIPLE REEL. DETAB-65
000130     SELECT CARD-OUTPUT, ASSIGN TO SYSTEM-PUNCH-TAPE.              DETAB-65
000135     SELECT LIST-OUTPUT, ASSIGN TO SYSTEM-OUTPUT-TAPE.             DETAB-65
000140 DATA DIVISION.                                                    DETAB-65
000145 FILE SECTION.                                                     DETAB-65
000150 FD  CARD-INPUT                                                    DETAB-65
000155     LABEL RECORDS ARE OMITTED,                                    DETAB-65
000160     DATA RECORDS ARE TEST-CARD.                                   DETAB-65
000165 01  TEST-CARD.                                                    DETAB-65
000170     02 FILLER   PICTURE X(80).                                    DETAB-65
000175 FD  CARD-OUTPUT                                                   DETAB-65
000180     LABEL RECORDS ARE OMITTED,                                    DETAB-65
000185     DATA RECORDS ARE CRD-OUT, DETAB-CRD, DUM-1.                   DETAB-65
000190 01 CRD-OUT.                                                      DETAB-65
000195     02 FILLER   PICTURE X(7).                                     DETAB-65
000200     02 BODY.                                                      DETAB-65
000205       03 FILLER PICTURE X(4).                                     DETAB-65
000210       03 B-MARG PICTURE X(61).                                    DETAB-65
000215     02 IDFLD    PICTURE X(8).                                     DETAB-65
000220 01  DETAB-CRD.                                                    DETAB-65
000225     02 FILLER   PICTURE XXX.                                      DETAB-65
000230     02 IDENT.                                                     DETAB-65
000235       03 ROW-NO PICTURE 999.                                      DETAB-65
000240       03 LINE-ID PICTURE X.                                       DETAB-65
000245     02 FILLER   PICTURE X(73).                                    DETAB-65
000250 01  DUM-1.                                                        DETAB-65
000255     02 CRD-COL  PICTURE X       OCCURS 80 TIMES.                  DETAB-65
```

```
000260 FD  LIST-OUTPUT                                                          DETAB-65
000265     LABEL RECORDS ARE OMITTED.                                           DETAB-65
000270     DATA RECORD IS TAPE-LIST.                                            DETAB-65
000275 01  TAPE-LIST.                                                          DETAB-65
000280     02 FILLER    PICTURE X(11).                                        DETAB-65
000285     02 CARDX     PICTURE 999.                                          DETAB-65
000290     02 FILLER    PICTURE X(66).                                        DETAB-65
000295 WORKING-STORAGE SECTION.                                                 DETAB-65
000300 77  AZ          PICTURE XX       VALUE 'AZ'.                          DETAB-65
000305 77  CARDCNT     PICTURE 999      COMPUTATIONAL SYNCHRONIZED RIGHT.DETAB-65
000310 77  COLIX       PICTURE 999      COMPUTATIONAL SYNCHRONIZED RIGHT.DETAB-65
000315 77  COLUM       PICTURE 999      COMPUTATIONAL SYNCHRONIZED RIGHT.DETAB-65
000320 77  DUMIX       PICTURE 999      COMPUTATIONAL SYNCHRONIZED RIGHT.DETAB-65
000325 77  ELMCT       PICTURE 999      COMPUTATIONAL SYNCHRONIZED RIGHT.DETAB-65
000330 77  ELMCX       PICTURE 999      COMPUTATIONAL SYNCHRONIZED RIGHT.DETAB-65
000335 77  ELMRX       PICTURE 999      COMPUTATIONAL SYNCHRONIZED RIGHT.DETAB-65
000340 77  EXIX        PICTURE 999      COMPUTATIONAL SYNCHRONIZED RIGHT.DETAB-65
000345 77  KEY-1       PICTURE 999      COMPUTATIONAL SYNCHRONIZED RIGHT.DETAB-65
000350 77  KEY-2       PICTURE 999      COMPUTATIONAL SYNCHRONIZED RIGHT.DETAB-65
000355 77  KEY-3       PICTURE 999      COMPUTATIONAL SYNCHRONIZED RIGHT.DETAB-65
000360 77  LABIX       PICTURE 999      COMPUTATIONAL SYNCHRONIZED RIGHT.DETAB-65
000365 77  LABNO       PICTURE 999      COMPUTATIONAL SYNCHRONIZED RIGHT.DETAB-65
000370 77  NACTS       PICTURE 999      COMPUTATIONAL SYNCHRONIZED RIGHT.DETAB-65
000375 77  NCOLS       PICTURE 999      COMPUTATIONAL SYNCHRONIZED RIGHT.DETAB-65
000380 77  NORLS       PICTURE 999      COMPUTATIONAL SYNCHRONIZED RIGHT.DETAB-65
000385 77  NOCON       PICTURE 999      COMPUTATIONAL SYNCHRONIZED RIGHT.DETAB-65
000390 77  NRLS        PICTURE 999      COMPUTATIONAL SYNCHRONIZED RIGHT.DETAB-65
000395 77  NROWS       PICTURE 999      COMPUTATIONAL SYNCHRONIZED RIGHT.DETAB-65
000400 77  ROWIX       PICTURE 999      COMPUTATIONAL SYNCHRONIZED RIGHT.DETAB-65
000405 01  DUM-2.                                                              DETAB-65
000410     02 FILLER          OCCURS 50 TIMES.                                 DETAB-65
000415       03 STRTCOL PICTURE 99      COMPUTATIONAL SYNCHRONIZED RIGHT.DETAB-65
000420       03 NMCOLS PICTURE 99       COMPUTATIONAL SYNCHRONIZED RIGHT.DETAB-65
000425 01  DUM-3.                                                              DETAB-65
000430     02 COLS      PICTURE X      OCCURS 12 TIMES.                      DETAB-65
000435 01  DUM-4.                                                              DETAB-65
000440     02 EGOTO     PICTURE X      OCCURS 5 TIMES.                       DETAB-65
000445 01  DUM-5.                                                              DETAB-65
000450     02 TEMP      PICTURE X      OCCURS 58 TIMES.                      DETAB-65
000455 01  DUM-10  PICTURE X(8)  VALUE 'SECTION.'.                             DETAB-65
000460 01 DUM-12  REDEFINES DUM-10.                                           DETAB-65
000465     02 NMSEC     PICTURE X      OCCURS 8 TIMES.                       DETAB-65
000470 01  HEADER.                                                             DETAB-65
000475     02 FILLER    PICTURE X(8).                                         DETAB-65
000480     02 TBLNME    PICTURE X(30).                                        DETAB-65
000485     02 FORMID    PICTURE XX.                                           DETAB-65
000490     02 NCOND     PICTURE 9(3).                                         DETAB-65
000495     02 ACTNS     PICTURE 9(3).                                         DETAB-65
000500     02 NORULS    PICTURE 9(3).                                         DETAB-65
000505     02 FILLER    PICTURE X(51).                                        DETAB-65
000510 01  DPRINT.                                                             DETAB-65
000515     02 DLABEL.                                                         DETAB-65
000520       03 FILLER PICTURE X(7)      VALUE SPACES.                       DETAB-65
000525       03 DUM-6.                                                        DETAB-65
000530         04 LABNM PICTURE XX.                                           DETAB-65
000535         04 LABVL PICTURE 9(3).                                         DETAB-65
000540       03 FILLER PICTURE X         VALUE '.'.                          DETAB-65
000545     02 DGOTO.                                                          DETAB-65
000550       03 FILLER PICTURE A(7)      VALUE ' GO TO '.                    DETAB-65
```

```
000555          03 DGOLN.                                                DETAB-65
000560            04 DGOLB PICTURE XX.                                   DETAB-65
000565            04 DGONO PICTURE 999.                                  DETAB-65
000570        02 HOUSTON.                                                DETAB-65
000575          03 CNDI    PICTURE X(58) OCCURS 50 TIMES.                DETAB-65
000580          03 ATBL    PICTURE X(58) OCCURS 50 TIMES.                DETAB-65
000585 01   LINE1.                                                       DETAB-65
000590        02 FILLER    PICTURE X(14)    VALUE '           IF '.      DETAB-65
000595        02 COND      PICTURE X(58).                                DETAB-65
000600 01   TEXAS.                                                       DETAB-65
000605        02 LINE2.                                                  DETAB-65
000610          03 FILLER PICTURE A(11).                                 DETAB-65
000615          03 CDOPR  PICTURE X(12).                                 DETAB-65
000620          03 PIF    PICTURE X.                                     DETAB-65
000625          03 DELSE  PICTURE X(6).                                  DETAB-65
000630          03 ELOPR  PICTURE X(12).                                 DETAB-65
000635          03 PELSE  PICTURE X.                                     DETAB-65
000640          03 FILLER PICTURE A(29).                                 DETAB-65
000645        02 LINE3  REDEFINES  LINE2.                                DETAB-65
000650          03 FILLER PICTURE X(7).                                  DETAB-65
000655          03 DNAME.                                                DETAB-65
000660            04 TCOLS PICTURE X    OCCURS 58 TIMES.                 DETAB-65
000665          03 FILLER PICTURE X(7).                                  DETAB-65
000670        02 FILLER   REDEFINES  LINE2.                              DETAB-65
000675          03 FILLER PICTURE X(11).                                 DETAB-65
000680          03 BNAME  PICTURE X(58).                                 DETAB-65
000685          03 FILLER PICTURE XXX.                                   DETAB-65
000690        02 DECISION-TABLE.                                         DETAB-65
000695          03 ROW                  OCCURS 50 TIMES.                 DETAB-65
000700            04 COLMN PICTURE X    OCCURS 100 TIMES.                DETAB-65
000705 01   ELIMT.                                                       DETAB-65
000710        02 ELIMC     PICTURE 999     OCCURS 25 TIMES.              DETAB-65
000715 01   MATIT.                                                       DETAB-65
000720        02 MATIX     PICTURE 999     OCCURS 25 TIMES.              DETAB-65
000725 01   MICDESCR.                                                    DETAB-65
000730        02 PDPUL     PICTURE 999   OCCURS 128 TIMES.               DETAB-65
000735        02 SAVCL     PICTURE X       OCCURS 25 TIMES.              DETAB-65
000740 01   WRNING-PRINT.                                                DETAB-65
000745        02 FILLER    PICTURE X(17)    VALUE                        DETAB-65
000750           ' ****** WARNING. '.                                     DETAB-65
000755        02 WRNING-IMAGE PICTURE X(52).                             DETAB-65
000760 01   WARNING-MESSAGES.                                            DETAB-65
000765        02 WRNING-1 PICTURE X(52)    VALUE                        DETAB-65
000770           'NO ELSE RULE CARD. LAST RULE PROCESSED AS ELSE RULE.'.  DETAB-65
000775        02 WRNING-2 PICTURE X(31)    VALUE                        DETAB-65
000780           'REDUNDANCY. CHECK THESE RULES -'.                       DETAB-65
000785 01   ERR-PRNT.                                                    DETAB-65
000790        02 FILLER    PICTURE X(30)    VALUE                        DETAB-65
000795           ' ****** ERROR. TABLE SKIPPED. '.                        DETAB-65
000800        02 ERR-IMAGE PICTURE X(53).                                DETAB-65
000805 01   ERROR-MESSAGES.                                              DETAB-65
000810        02 ERR-1     PICTURE X(48)    VALUE                        DETAB-65
000815           'PRESENTLY, TABLES RESTRICTED TO LIMITED ENTRIES.'.      DETAB-65
000820        02 ERR-2     PICTURE X(42)    VALUE                        DETAB-65
000825           'TABLE-NAME MISSING FROM TABLE HEADER CARD.'.            DETAB-65
000830        02 ERR-3     PICTURE X(19)    VALUE                        DETAB-65
000835           'RULES CARD MISSING.'.                                   DETAB-65
000840        02 ERR-4     PICTURE X(39)    VALUE                        DETAB-65
000845           'LESS THAN THREE RULE COLUMNS SPECIFIED.'.               DETAB-65
```

```
000850      02 ERR-5     PICTURE X(43)     VALUE                          DETAB-65
000855         'PRESENTLY, CONTINJED RULES NOT IMPLEMENTED.'.              DETAB-65
000860      02 ERR-6     PICTURE X(40)     VALUE                          DETAB-65
000865         'CONDITION STUB ENTRY EXCEEDS 58 COLUMNS.'.                 DETAB-65
000870      02 ERR-7     PICTURE X(26)     VALUE                          DETAB-65
000875         'MORE THAN 12 RULE COLUMNS.'.                               DETAB-65
000880      02 ERR-8     PICTURE X(53)     VALUE                          DETAB-65
000885         'NUMBER OF RULES ENCOUNTERED DISAGREES WITH RULE CARD.'.    DETAB-65
000890      02 ERR-9     PICTURE X(41)     VALUE                          DETAB-65
000895         'MORE THAN 50 ACTION OR CONDITION ENTRIES.'.                DETAB-65
000900      02 ERR-10    PICTURE X(46)     VALUE                          DETAB-65
000905         'DECISION TABLE LOGIC ERROR. PROCESSING HALTED.'.           DETAB-65
000910 PROCEDURE DIVISION.                                                 DETAB-65
000915 DETAB65.                                                            DETAB-65
000920      OPEN INPUT CARD-INPJT, OUTPUT CARD-OUTPUT, LIST-OUTPUT.        DETAB-65
000925 DT001.                                                              DETAB-65
000930      PERFORM READ-1.                                                DETAB-65
000935      IF '0000' = IDENT OF DETAB-CRD GO TO MONITOR.                  DETAB-65
000940      WRITE DETAB-CRD.                                               DETAB-65
000945      GO TO DT001.                                                   DETAB-65
000950 MONITOR.                                                            DETAB-65
000955      MOVE DETAB-CRD TO HEADER.                                      DETAB-65
000960      IF TBLNME = SPACES GO TO EM02.                                 DETAB-65
000965      IF FORMID OF HEADER NOT = 'L' GO TO EM01.                      DETAB-65
000970      MOVE SPACES TO HOUSTON, TEXAS.                                 DETAB-65
000975      MOVE ZEROES TO DUM-2.                                          DETAB-65
000980      MOVE TBLNME TO DUM-5, DNAME.                                   DETAB-65
000985      PERFORM RSCAN.                                                 DETAB-65
000990      PERFORM DT005 VARYING EXIX FROM 1 BY 1 UNTIL EXIX = 9.         DETAB-65
000995      PERFORM READ-1.                                                DETAB-65
001000      IF IDENT OF DETAB-CRD NOT = '0001' GO TO EM03.                 DETAB-65
001005                                                                     DETAB-65
001010      NOTE RULES CONVERSION SECTION.                                 DETAB-65
001015                                                                     DETAB-65
001020      MOVE 0 TO CARDCNT.                                             DETAB-65
001025      MOVE 1 TO NRLS.                                                DETAB-65
001030      MOVE 9 TO COLUM, STRTCOL (NRLS).                               DETAB-65
001035 DT050.                                                              DETAB-65
001040      IF CRD-COL (COLUM) = SPACE GO TO DT053.                        DETAB-65
001045      IF CARDCNT IS LESS THAN 3 GO TO EM04.                          DETAB-65
001050      MOVE CARDCNT TO NMCOLS (NRLS).                                 DETAB-65
001055      IF CRD-COL (COLUM) = '$' GO TO DT055.                          DETAB-65
001060      ADD 1 TO NRLS.                                                 DETAB-65
001065      MOVE COLUM TO STRTCOL (NRLS).                                  DETAB-65
001070      MOVE 3 TO CARDCNT.                                             DETAB-65
001075      ADD 3 TO COLUM.                                                DETAB-65
001080      IF COLUM IS GREATER THAN 80 GO TO EM05.                        DETAB-65
001085      GO TO DT050.                                                   DETAB-65
001090 DT005.                                                              DETAB-65
001095      MOVE NMSEC (EXIX) TO TCOLS (DUMIX).                            DETAB-65
001100      ADD 1 TO DUMIX.                                                DETAB-65
001105 DT053.                                                              DETAB-65
001110      ADD 1 TO CARDCNT, ADD 1 TO COLUM.                              DETAB-65
001115      IF CARDCNT IS NOT GREATER THAN 12 GO TO DT050.                 DETAB-65
001120      IF CARDCNT IS GREATER THAN 58 GO TO EM06.                      DETAB-65
001125      IF NRLS = 1 GO TO DT050 ELSE GO TO EM07.                       DETAB-65
001130 DT055.                                                              DETAB-65
001135      SUBTRACT 1 FROM NMCOLS (NRLS), SUBTRACT 1 FROM NRLS.           DETAB-65
001140      IF NRLS NOT = NORJLS GO TO EM08.                               DETAB-65
```

```
001145                                                                  DETAB-65
001150     NOTE DETAB CARD SECTION.                                     DETAB-65
001155                                                                  DETAB-65
001160     ADD 1 TO NRLS.                                               DETAB-65
001165     MOVE STRTCOL (NRLS) TO COLUM.                                DETAB-65
001170     IF CRD-COL (COLUM) = 'E' GO TO DT056.                        DETAB-65
001175     MOVE WRNING-1 TO WRNING-IMAGE.                               DETAB-65
001180     WRITE TAPE-LIST FROM WRNING-PRINT.                           DETAB-65
001185 DT056.                                                           DETAB-65
001190     MOVE 1 TO KEY-2, KEY-3, ROWIX.                               DETAB-65
001195 DT057.                                                           DETAB-65
001200     PERFORM READ-1.                                              DETAB-65
001205     IF ROW-NO OF DETAB-CRD =  999  GO TO DT057.                  DETAB-65
001210     MOVE 1 TO KEY-1, COLIX.                                      DETAB-65
001215     IF LINE-ID OF DETAB-CRD = '$' GO TO TBLPROC.                 DETAB-65
001220     MOVE STRTCOL (KEY-1) TO COLUM.                               DETAB-65
001225                                                                  DETAB-65
001230     NOTE CONDACT SECTION.                                        DETAB-65
001235                                                                  DETAB-65
001240     MOVE SPACES TO DUM-5.                                        DETAB-65
001245     MOVE 1 TO EXIX.                                              DETAB-65
001250 CONDACT.                                                         DETAB-65
001255     MOVE CRD-COL (COLUM) TO TEMP (EXIX).                         DETAB-65
001260     IF EXIX GREATER NMCOLS (KEY-1) GO TO DT057-1.                DETAB-65
001265     ADD 1 TO EXIX, ADD 1 TO COLUM, GO TO CONDACT.                DETAB-65
001270 DT057-1.                                                         DETAB-65
001275     IF KEY-2 IS GREATER THAN 50 GO TO EM09.                      DETAB-65
001280     IF KEY-2 IS GREATER THAN NCOND GO TO DT058.                  DETAB-65
001285     MOVE DUM-5 TO CNDI (KEY-2).                                  DETAB-65
001290     ADD 1 TO KEY-2.                                              DETAB-65
001295     GO TO DT059.                                                 DETAB-65
001300 DT058.                                                           DETAB-65
001305     IF KEY-3 IS GREATER THAN 50 GO TO EM09.                      DETAB-65
001310     MOVE DUM-5 TO ATBL (KEY-3).                                  DETAB-65
001315     ADD 1 TO KEY-3.                                              DETAB-65
001320 DT059.                                                           DETAB-65
001325     PERFORM DT060 THRU DT061 VARYING KEY-1 FROM 2 BY 1 UNTIL     DETAB-65
001330       KEY-1 IS GREATER THAN NRLS.                                DETAB-65
001335     ADD 1 TO ROWIX.                                              DETAB-65
001340     GO TO DT057.                                                 DETAB-65
001345 DT060.                                                           DETAB-65
001350     MOVE STRTCOL (KEY-1) TO COLUM.                               DETAB-65
001355                                                                  DETAB-65
001360     NOTE VARAMOVE SECTION.                                       DETAB-65
001365                                                                  DETAB-65
001370     MOVE SPACES TO DUM-3.                                        DETAB-65
001375     MOVE 1 TO EXIX.                                              DETAB-65
001380 VARAMVE.                                                         DETAB-65
001385     MOVE CRD-COL (COLUM) TO COLS (EXIX).                         DETAB-65
001390     IF EXIX GREATER NMCOLS (KEY-1) GO TO DT060-1.                DETAB-65
001395     ADD 1 TO EXIX, ADD 1 TO COLUM, GO TO VARAMVE.                DETAB-65
001400 DT060-1.                                                         DETAB-65
001405     EXAMINE DUM-3 REPLACING ALL '-' BY SPACES.                   DETAB-65
001410     IF DUM-3 = SPACES GO TO DT061.                               DETAB-65
001415     EXAMINE DUM-3 TALLYING UNTIL FIRST 'N'.                      DETAB-65
001420     IF TALLY = 12 MOVE 'Y' TO COLMN (ROWIX, COLIX) ELSE          DETAB-65
001425       MOVE 'N' TO COLMN (ROWIX, COLIX).                          DETAB-65
001430 DT061.                                                           DETAB-65
001435     ADD 1 TO COLIX.                                              DETAB-65
```

```
00.:4  TBLPROC.                                                          DETAB-65
001445      PERFORM L2OUT THRJ RITAB.                                    DETAB-65
001450      MOVE 'DX000' TO DJM-6.                                       DETAB-65
001455      PERFORM DLOUT THRU RITAB.                                    DETAB-65
001460                                                                   DETAB-65
001465      NOTE DECISION SECTION.                                       DETAB-65
001470                                                                   DETAB-65
001475      MOVE ZERO TO LABIX, LABNO.                                   DETAB-65
001480      MOVE ACTNS TO NACTS.                                         DETAB-65
001485      COMPUTE NORLS = NORULS - 1.                                  DETAB-65
001490      MOVE NCOND TO NOCON.                                         DETAB-65
001495      PERFORM DX042 VARYING COLIX FROM 1 BY 1 UNTIL COLIX = NORLS. DETAB-65
001500 DX042.                                                            DETAB-65
001505      MOVE COLIX TO MATIX (COLIX).                                 DETAB-65
001510 DX001.                                                            DETAB-65
001515      PERFORM DX002 VARYING COLIX FROM 1 BY 1 UNTIL COLIX = NORLS. DETAB-65
001520 DX002.                                                            DETAB-65
001525      MOVE COLIX TO ELIMC (COLIX).                                 DETAB-65
001530 DX050.                                                            DETAB-65
001535      MOVE NOCON TO NROWS.                                         DETAB-65
001540      MOVE NORLS TO NCOLS.                                         DETAB-65
001545      MOVE 0 TO ROWIX.                                             DETAB-65
001550      GO TO DX004.                                                 DETAB-65
001555 DX003.                                                            DETAB-65
001560      PERFORM L1OUT THRJ RITAB.                                    DETAB-65
001565      PERFORM L2OUT THRU RITAB.                                    DETAB-65
001570 DX004.                                                            DETAB-65
001575      MOVE SPACES TO LINE2.                                        DETAB-65
001580 DX005.                                                            DETAB-65
001585      ADD 1 TO ROWIX.                                              DETAB-65
001590      MOVE ZERO TO DUMIX.                                          DETAB-65
001595      IF ROWIX = NOCON GO TO DX014.                                DETAB-65
001600      MOVE 1 TO COLIX.                                             DETAB-65
001605                                                                   DETAB-65
001610      NOTE  ARE THERE ALL BLANKS IN THIS ROW.                      DETAB-65
001615                                                                   DETAB-65
001620 DX005-1.                                                          DETAB-65
001625      IF COLIX GREATER NCOLS GO TO DX005-2.                        DETAB-65
001630      MOVE ELIMC (COLIX) TO ELMCX.                                 DETAB-65
001635      IF COLMN (ROWIX, ELMCX) = ' ' OR 'B'                         DETAB-65
001640          NEXT SENTENCE ELSE GO TO DX051.                          DETAB-65
001645      ADD 1 TO COLIX.                                              DETAB-65
001650      GO TO DX005-1.                                               DETAB-65
001655 DX005-2.                                                          DETAB-65
001660      PERFORM DX400 THRU DX402 VARYING COLIX FROM 1 BY 1           DETAB-65
001665        UNTIL COLIX IS GREATER THAN NCOLS.                         DETAB-65
001670      GO TO DX005.                                                 DETAB-65
001675 DX400.                                                            DETAB-65
001680      MOVE ELIMC (COLIX) TO ELMCT.                                 DETAB-65
001685      MOVE 1 TO ELMRX.                                             DETAB-65
001690 DX400-1.                                                          DETAB-65
001695      IF ELMRX = ROWIX GO TO DX400-2.                              DETAB-65
001700      IF COLMN (ELMRX, ELMCT) = ' '                                DETAB-65
001705         MOVE 'B' TO COLMN (ROWIX, ELMCT)                          DETAB-65
001710         GO TO DX402.                                              DETAB-65
001715      ADD 1 TO ELMRX.                                              DETAB-65
001720      GO TO DX400-1.                                               DETAB-65
001725 DX400-2.                                                          DETAB-65
001730      MOVE 'Y' TO COLMN (ROWIX, ELMCT).                            DETAB-65
```

```
001735 DX402.                                                          DETAB-65
001740     EXIT.                                                       DETAB-65
001745 DX051.                                                          DETAB-65
001750     MOVE CNDI (ROWIX) TO COND.                                  DETAB-65
001755                                                                 DETAB-65
001760     NOTE  IS THERE A Y OR N IN THIS ROW.                        DETAB-65
001765                                                                 DETAB-65
001770     MOVE 1 TO COLIX.                                            DETAB-65
001775 DX051-1.                                                        DETAB-65
001780     IF COLIX GREATER NCOLS GO TO DX051-2.                       DETAB-65
001785     MOVE ELIMC (COLIX) TO ELMCX.                                DETAB-65
001790     IF COLMN (ROWIX, ELMCX) NOT = "N" GO TO DX052.              DETAB-65
001795     ADD 1 TO COLIX.                                             DETAB-65
001800     GO TO DX051-1.                                              DETAB-65
001805 DX051-2.                                                        DETAB-65
001810     MOVE "EL001" TO DGOLN.                                      DETAB-65
001815     MOVE DGOTO TO CDOPR.                                        DETAB-65
001820     GO TO DX202.                                                DETAB-65
001825 DX052.                                                          DETAB-65
001830     MOVE ROWIX TO ELMRX.                                        DETAB-65
001835                                                                 DETAB-65
001840     NOTE  ARE THE REST OF THE ELEMENTS IN THIS COLUMN BLANK.    DETAB-65
001845                                                                 DETAB-65
001850 DX052-1.                                                        DETAB-65
001855     IF ELMRX = NOCON GO TO DX052-2.                             DETAB-65
001860     COMPUTE ELMCT = ELMRX + 1.                                  DETAB-65
001865     IF COLMN (ELMCT, ELMCX) NOT = " " GO TO DX201.              DETAB-65
001870     ADD 1 TO ELMRX.                                             DETAB-65
001875     GO TO DX052-1.                                              DETAB-65
001880 DX052-2.                                                        DETAB-65
001885     IF NCOLS = 1 THEN MOVE ROWIX TO NOCON GO TO DX014.          DETAB-65
001890     MOVE COLIX TO DUMIX.                                        DETAB-65
001895     GO TO DX202.                                                DETAB-65
001900                                                                 DETAB-65
001905     NOTE  PUSH LAST-IN-FIRST-OUT LIST.                          DETAB-65
001910                                                                 DETAB-65
001915 DX201.                                                          DETAB-65
001920     MOVE "DX" TO DGOLB.                                         DETAB-65
001925     ADD 1 TO LABNO, ADD 1 TO LABIX.                             DETAB-65
001930     MOVE LABNO TO DGONO, PDPUL (LABIX).                         DETAB-65
001935     MOVE DGOTO TO CDOPR.                                        DETAB-65
001940 DX202.                                                          DETAB-65
001945     MOVE 1 TO COLIX.                                            DETAB-65
001950                                                                 DETAB-65
001955     NOTE  IS THERE A N OR A BLANK IN THIS ROW.                  DETAB-65
001960                                                                 DETAB-65
001965 DX202-1.                                                        DETAB-65
001970     IF COLIX GREATER NCOLS GO TO DX202-2.                       DETAB-65
001975     MOVE ELIMC (COLIX) TO ELMCX.                                DETAB-65
001980     IF COLMN (ROWIX, ELMCX) NOT = "Y" GO TO DX053.              DETAB-65
001985     ADD 1 TO COLIX.                                             DETAB-65
001990     GO TO DX202-1.                                              DETAB-65
001995 DX202-2.                                                        DETAB-65
002000     MOVE "EL001" TO DGOLN.                                      DETAB-65
002005     MOVE " ELSE " TO DELSE.                                     DETAB-65
002010     MOVE DGOTO TO ELOPR.                                        DETAB-65
002015     PERFORM DX204 THRU DX205.                                   DETAB-65
002020     GO TO DX300.                                                DETAB-65
002025 DX053.                                                          DETAB-65
```

```
002030      MOVE ROWIX TO ELMRX.                                          DETAB-65
002035                                                                    DETAB-65
002040      NOTE  ARE THE REST OF THE ELEMENTS IN THIS COLUMN BLANK.      DETAB-65
002045                                                                    DETAB-65
002050 DX053-1.                                                           DETAB-65
002055      IF ELMRX = NOCON GO TO DX053-2.                               DETAB-65
002060      COMPUTE ELMCT = 1 + ELMRX.                                    DETAB-65
002065      IF COLMN (ELMCT, ELMCX) NOT = ' '                             DETAB-65
002070          MOVE '.' TO PIF, GO TO DX204.                             DETAB-65
002075      ADD 1 TO ELMRX.                                               DETAB-65
002080      GO TO DX053-1.                                                DETAB-65
002085 DX053-2.                                                           DETAB-65
002090      MOVE ROWIX TO NOCON.                                          DETAB-65
002095      IF DUMIX NOT = ZERO OR NCOLS = 1 THEN GO TO DX014.            DETAB-65
002100      MOVE COLIX TO ELMRX.                                          DETAB-65
002105      MOVE AZ TO DGOLB.                                             DETAB-65
002110      MOVE ELMCX TO DGONO.                                          DETAB-65
002115      MOVE ' ELSE ' TO DELSE.                                       DETAB-65
002120      MOVE DGOTO TO ELOPR.                                          DETAB-65
002125      PERFORM DX016 THRU DX020.                                     DETAB-65
002130      PERFORM DX011 THRU DX055.                                     DETAB-65
002135      MOVE NOCON TO ROWIX.                                          DETAB-65
002140      MOVE NROWS TO NOCON.                                          DETAB-65
002145 DX300.                                                             DETAB-65
002150      MOVE '.' TO PELSE.                                            DETAB-65
002155      PERFORM L1OUT THRU RITAB.                                     DETAB-65
002160      PERFORM L2OUT THRU RITAB.                                     DETAB-65
002165      IF NORLS = ZERO GO TO DX038.                                  DETAB-65
002170      MOVE 'DX' TO LABNM.                                           DETAB-65
002175      MOVE PDPUL (LABIX) TO LABVL.                                  DETAB-65
002180      SUBTRACT 1 FROM LABIX.                                        DETAB-65
002185      PERFORM DLOUT THRU RITAB.                                     DETAB-65
002190      GO TO DX004.                                                  DETAB-65
002195 DX204.                                                             DETAB-65
002200      IF DUMIX = ZERO GO TO DX205.                                  DETAB-65
002205      MOVE ROWIX TO NOCON.                                          DETAB-65
002210      MOVE AZ TO DGOLB.                                             DETAB-65
002215      MOVE ELIMC (DUMIX) TO DGONO.                                  DETAB-65
002220      MOVE DGOTO TO CDOPR.                                          DETAB-65
002225      MOVE DUMIX TO COLIX.                                          DETAB-65
002230      PERFORM DX016 THRU DX020.                                     DETAB-65
002235      MOVE NOCON TO ROWIX.                                          DETAB-65
002240      MOVE NROWS TO NOCON.                                          DETAB-65
002245 DX205.                                                             DETAB-65
002250      EXIT.                                                         DETAB-65
002255 DX009.                                                             DETAB-65
002260      PERFORM DX010 THRU DX055 VARYING ELMRX FROM 1 BY 1 UNTIL      DETAB-65
002265        ELMRX IS GREATER THAN NCOLS.                                DETAB-65
002270      GO TO DX003.                                                  DETAB-65
002275                                                                    DETAB-65
002280      NOTE  DELETE FROM PATH INDEX ALL COLUMNS THAT HAVE A Y        DETAB-65
002285      IN THIS ROW.                                                  DETAB-65
002290                                                                    DETAB-65
002295 DX010.                                                             DETAB-65
002300      MOVE ELIMC (ELMRX) TO COLIX.                                  DETAB-65
002305      IF COLMN (ROWIX, COLIX) NOT = 'Y' GO TO DX055.                DETAB-65
002310 DX011.                                                             DETAB-65
002315      SUBTRACT 1 FROM NCOLS.                                        DETAB-65
002320      PERFORM DX012 VARYING ELMCX FROM ELMRX BY 1 UNTIL ELMCX       DETAB-65
```

```
002325       GREATER THAN NCOLS.                                          DETAB-65
002330     SUBTRACT 1 FROM ELMRX, SUBTRACT 1 FROM COLIX.                  DETAB-65
002335 DX012.                                                             DETAB-65
002340     COMPUTE ELMCT = 1 + ELMCX.                                     DETAB-65
002345     MOVE ELIMC (ELMCT) TO ELIMC (ELMCX).                           DETAB-65
002350 DX055.                                                             DETAB-65
002355     EXIT.                                                          DETAB-65
002360 DX014.                                                             DETAB-65
002365     MOVE ELIMC (1) TO COLIX.                                       DETAB-65
002370     PERFORM DX015 VARYING ROWIX FROM 1 BY 1 UNTIL ROWIX = NROWS.   DETAB-65
002375 DX015.                                                             DETAB-65
002380     MOVE COLMN (ROWIX, COLIX) TO SAVCL (ROWIX).                    DETAB-65
002385 DX056.                                                             DETAB-65
002390     MOVE 4 TO DUMIX.                                               DETAB-65
002395     PERFORM DX022 THRU DX031 VARYING COLIX FROM 1 BY 1 UNTIL       DETAB-65
002400       COLIX IS GREATER THAN NCOLS.                                 DETAB-65
002405     GO TO DX032.                                                   DETAB-65
002410                                                                    DETAB-65
002415     NOTE  DETERMINE ACTION LABELS AND CHECK FOR REDUNDENCY.        DETAB-65
002420                                                                    DETAB-65
002425 DX022.                                                             DETAB-65
002430     MOVE ELIMC (COLIX) TO ELMCX.                                   DETAB-65
002435     IF COLMN (NOCON, ELMCX) NOT = 'Y' GO TO DX029.                 DETAB-65
002440     IF DUMIX = 3 OR DUMIX = 1 THEN GO TO DX059.                    DETAB-65
002445     IF DUMIX = 2 MOVE 3 TO DUMIX ELSE MOVE 1 TO DUMIX.             DETAB-65
002450     MOVE AZ TO DGOLB.                                              DETAB-65
002455     MOVE ' ELSE ' TO DELSE.                                        DETAB-65
002460     MOVE ELMCX TO DGONO.                                           DETAB-65
002465     MOVE  DGOTO TO CDOPR.                                          DETAB-65
002470     GO TO DX031.                                                   DETAB-65
002475 DX059.                                                             DETAB-65
002480     MOVE WRNING-2 TO WRNING-IMAGE.                                 DETAB-65
002485     WRITE TAPE-LIST FROM WRNING-PRINT.                             DETAB-65
002490     PERFORM DX028 VARYING ELMRX FROM 1 BY 1 UNTIL ELMRX = NCOLS.   DETAB-65
002495 DX028.                                                             DETAB-65
002500     MOVE '       RULE' TO TAPE-LIST.                               DETAB-65
002505     MOVE ELIMC (ELMRX) TO CARDX.                                   DETAB-65
002510     WRITE TAPE-LIST.                                               DETAB-65
002515 DX013.                                                             DETAB-65
002520     EXIT.                                                          DETAB-65
002525 DX029.                                                             DETAB-65
002530     IF COLMN (NOCON, ELMCX) NOT = 'N' GO TO DX031.                 DETAB-65
002535     IF DUMIX = 3 OR DUMIX = 2 PERFORM DX059 THRU DX013,            DETAB-65
002540       GO TO DX031.                                                 DETAB-65
002545     IF DUMIX = 1 MOVE 3 TO DUMIX ELSE MOVE 2 TO DUMIX.             DETAB-65
002550     MOVE AZ TO DGOLB.                                              DETAB-65
002555     MOVE ' ELSE ' TO DELSE.                                        DETAB-65
002560     MOVE ELMCX TO DGONO.                                           DETAB-65
002565     MOVE DGOTO TO ELOPR.                                           DETAB-65
002570 DX031.                                                             DETAB-65
002575     EXIT.                                                          DETAB-65
002580 DX032.                                                             DETAB-65
002585     MOVE 'EL001' TO DGOLN.                                         DETAB-65
002590     MOVE '.' TO PELSE.                                             DETAB-65
002595     IF DUMIX = 2 MOVE DGOTO TO CDOPR ELSE                          DETAB-65
002600       IF DUMIX = 1 MOVE DGOTO TO ELOPR.                            DETAB-65
002605     MOVE CNDI (NOCON) TO COND.                                     DETAB-65
002610     PERFORM DX016 THRU DX020 VARYING COLIX FROM 1 BY 1 UNTIL       DETAB-65
002615       COLIX IS GREATER THAN NCOLS.                                 DETAB-65
```

```
002620      GO TO DX301.                                                    DETAB-65
002625  DX016.                                                              DETAB-65
002630      MOVE ELIMC (COLIX) TO DUMIX.                                    DETAB-65
002635      MOVE 1 TO ROWIX.                                                DETAB-65
002640  DX016-1.                                                            DETAB-65
002645      IF ROWIX GREATER NOCON GO TO DX016-2.                           DETAB-65
002650      IF COLMN (ROWIX, DUMIX) = 'B' GO TO DX504.                      DETAB-65
002655      ADD 1 TO ROWIX.                                                 DETAB-65
002660      GO TO DX016-1.                                                  DETAB-65
002665  DX016-2.                                                            DETAB-65
002670      MOVE 0 TO ROWIX.                                                DETAB-65
002675  DX016-3.                                                            DETAB-65
002680      IF ROWIX = NOCON GO TO DX016-4.                                 DETAB-65
002685      COMPUTE ELMCX = NOCON - ROWIX.                                  DETAB-65
002690      IF COLMN (ELMCX, DUMIX) = ' ' THEN                              DETAB-65
002695          MOVE 'B' TO COLMN (ELMCX, DUMIX), GO TO DX020.              DETAB-65
002700      ADD 1 TO ROWIX.                                                 DETAB-65
002705      GO TO DX016-3.                                                  DETAB-65
002710  DX016-4.                                                            DETAB-65
002715      SUBTRACT 1 FROM NORLS.                                          DETAB-65
002720      PERFORM DX100 VARYING ELMCX FROM 1 BY 1                         DETAB-65
002725      UNTIL ELMCX IS GREATER THAN NORLS.                              DETAB-65
002730      GO TO DX020.                                                    DETAB-65
002735  DX100.                                                              DETAB-65
002740      COMPUTE ELMCT = ELMCX + 1                                       DETAB-65
002745      IF MATIX (ELMCX) IS NOT LESS THAN DUMIX                         DETAB-65
002750          MOVE MATIX (ELMCT) TO MATIX (ELMCX).                        DETAB-65
002755  DX504.                                                              DETAB-65
002760      MOVE 1 TO ELMCT.                                                DETAB-65
002765  DX504-1.                                                            DETAB-65
002770      IF ELMCT = ROWIX GO TO DX504-2.                                 DETAB-65
002775      COMPUTE ELMCX = ROWIX - ELMCT.                                  DETAB-65
002780      IF COLMN (ELMCX, DUMIX) = ' ' GO TO DX507.                      DETAB-65
002785      ADD 1 TO ELMCT.                                                 DETAB-65
002790      GO TO DX504-1.                                                  DETAB-65
002795  DX504-2                                                             DETAB-65
002800      MOVE 'Y' TO COLMN (ROWIX, DUMIX).                               DETAB-65
002805      GO TO DX016.                                                    DETAB-65
002810  DX507.                                                              DETAB-65
002815      MOVE 'B' TO COLMN (ELMCX, DUMIX).                               DETAB-65
002820      PERFORM DX508 VARYING ELMCX FROM ROWIX BY 1                     DETAB-65
002825         UNTIL ELMCX = NOCON.                                         DETAB-65
002830      GO TO DX020.                                                    DETAB-65
002835  DX508.                                                              DETAB-65
002840      IF COLMN (ELMCX, DUMIX) = 'B'                                   DETAB-65
002845         MOVE ' ' TO COLMN (ELMCX, DUMIX).                            DETAB-65
002850  DX020.                                                              DETAB-65
002855      EXIT.                                                           DETAB-65
002860                                                                      DETAB-65
002865      NOTE  POP  LAST-IN-FIRST-OUT LIST.                              DETAB-65
002870                                                                      DETAB-65
002875  DX301.                                                              DETAB-65
002880      PERFORM L1OUT THRU RITAB.                                       DETAB-65
002885      PERFORM L2OUT THRU RITAB.                                       DETAB-65
002890      IF NORLS = ZEROES GO TO DX038.                                  DETAB-65
002895      MOVE 'DX' TO LABNM.                                             DETAB-65
002900      MOVE POPUL (LABIX) TO LABVL.                                    DETAB-65
002905      SUBTRACT 1 FROM LABIX.                                          DETAB-65
002910      PERFORM DLOUT THRU RITAB.                                       DETAB-65
```

```
002915                                                                  DETAB-65
002920     NOTE  SETUP INDEXES FOR NEXT PASS.                           DETAB-65
002925                                                                  DETAB-65
002930 DX302.                                                           DETAB-65
002935     MOVE NORLS TO NCOLS.                                         DETAB-65
002940     MOVE NROWS TO NOCON.                                         DETAB-65
002945     MOVE MATIT TO ELIMT.                                         DETAB-65
002950     MOVE 1 TO ROWIX.                                             DETAB-65
002955 DX302-1.                                                         DETAB-65
002960     IF ROWIX = NOCON                                             DETAB-65
002965       MOVE ERR-10 TO ERR-IMAGE                                   DETAB-65
002970       WRITE TAPE-LIST FROM ERR-PRNT                              DETAB-65
002975       GO TO DT001.                                               DETAB-65
002980                                                                  DETAB-65
002985     NOTE  DELETE THAT PATH GENERATED ON THE LAST PASS AND        DETAB-65
002990     FIND THE NEXT HIGHER NODE ON THE TREE.                       DETAB-65
002995                                                                  DETAB-65
003000     MOVE 1 TO COLIX.                                             DETAB-65
003005 DX034-1.                                                         DETAB-65
003010     IF SAVCL (ROWIX) = ' ' MOVE 'N' TO SAVCL (ROWIX).            DETAB-65
003015     IF COLIX GREATER NCOLS GO TO DX004.                          DETAB-65
003020     MOVE ELIMC (COLIX) TO ELMCX.                                 DETAB-65
003025     IF COLMN (ROWIX, ELMCX) = ' ' OR COLMN (ROWIX, ELMCX)        DETAB-65
003030         = SAVCL (ROWIX) GO TO DX034-2.                           DETAB-65
003035     ADD 1 TO COLIX.                                              DETAB-65
003040     GO TO DX034-1.                                               DETAB-65
003045 DX034-2.                                                         DETAB-65
003050     PERFORM DX037 VARYING COLIX FROM 1 BY 1 UNTIL COLIX = NCOLS. DETAB-65
003055 DX037.                                                           DETAB-65
003060     MOVE ELIMC (COLIX) TO ELMCX.                                 DETAB-65
003065     MOVE COLIX TO ELMRX.                                         DETAB-65
003070     IF COLMN (ROWIX, ELMCX) NOT = ' ' AND COLMN (ROWIX, ELMCX)   DETAB-65
003075         NOT = SAVCL (ROWIX) PERFORM DX011 THRU DX055.            DETAB-65
003080 DX061.                                                           DETAB-65
003085     ADD 1 TO ROWIX.                                              DETAB-65
003090     GO TO DX302-1.                                               DETAB-65
003095 DX038.                                                           DETAB-65
003100     MOVE SPACES TO LINE3.                                        DETAB-65
003105     COMPUTE KEY-2 = NORULS - 1.                                  DETAB-65
003110     PERFORM DX039 THRU DX039B VARYING COLIX FROM 1 BY 1           DETAB-65
003115       UNTIL COLIX = KEY-2.                                       DETAB-65
003120 DX039.                                                           DETAB-65
003125     MOVE AZ TO LABNM.                                            DETAB-65
003130     MOVE COLIX TO LABVL.                                         DETAB-65
003135     PERFORM DLOUT THRU RITAB.                                    DETAB-65
003140     ADD 1 NCOND GIVING KEY-1.                                    DETAB-65
003145     PERFORM DXA01 THRU DXA04 VARYING EXIX FROM 1 BY 1 UNTIL      DETAB-65
003150       EXIX IS GREATER THAN NACTS.                                DETAB-65
003155     MOVE SPACES TO CRD-OUT.                                      DETAB-65
003160     EXAMINE DUM-5 TALLYING UNTIL FIRST 'G'.                      DETAB-65
003165     IF TALLY = 58 GO TO DX039H.                                  DETAB-65
003170     IF TALLY NOT = ZERO, THEN                                    DETAB-65
003175         IF TEMP (TALLY) NOT = SPACE GO TO DX039H.                DETAB-65
003180     COMPUTE DUMIX = TALLY + 1.                                   DETAB-65
003185     PERFORM DX039F VARYING TALLY FROM 1 BY 1 UNTIL TALLY = 6.    DETAB-65
003190     GO TO DX039G.                                                DETAB-65
003195 DX039F.                                                          DETAB-65
003200     MOVE TEMP (DUMIX)  TO EGOTO (TALLY).                         DETAB-65
003205     ADD 1 TO DUMIX.                                              DETAB-65
```

```
003210 DXA01.                                                          DETAB-65
003215     IF COLMN (KEY-1, COLIX) = ' ' GO TO DXA04.                  DETAB-65
003220     MOVE ATBL (EXIX) TO DUM-5, BNAME.                           DETAB-65
003225     PERFORM RSCAN.                                              DETAB-65
003230     ADD 3 TO DUMIX.                                             DETAB-65
003235     MOVE '.' TO TCOLS (DUMIX).                                  DETAB-65
003240     PERFORM L2OUT THRU RITAB.                                   DETAB-65
003245 DXA04.                                                          DETAB-65
003250     ADD 1 TO KEY-1.                                             DETAB-65
003255 DX039G.                                                         DETAB-65
003260     IF DUM-4 = 'GO TO' GO TO DX039B.                            DETAB-65
003265 DX039H.                                                         DETAB-65
003270     MOVE 'GO TO DEXIT.' TO B-MARG OF CRD-OUT.                   DETAB-65
003275     MOVE CRD-OUT TO TAPE-LIST.                                  DETAB-65
003280     PERFORM RITAB.                                              DETAB-65
003285 DX039B.                                                         DETAB-65
003290     EXIT.                                                       DETAB-65
003295 DX040.                                                          DETAB-65
003300     MOVE SPACES TO LINE3.                                       DETAB-65
003305     COMPUTE KEY-1 = NCOND + 1.                                  DETAB-65
003310     MOVE NORULS TO COLIX.                                       DETAB-65
003315     MOVE 1 TO EXIX.                                             DETAB-65
003320     MOVE KEY-1 TO TALLY.                                        DETAB-65
003325     MOVE 0 TO NRLS.                                             DETAB-65
003330 DX040-2.                                                        DETAB-65
003335     IF EXIX GREATER NACTS GO TO DX040-3.                        DETAB-65
003340     IF COLMN (TALLY, COLIX) NOT = ' ' ADD 1 TO NRLS.            DETAB-65
003345     ADD 1 TO TALLY, ADD 1 TO EXIX.                              DETAB-65
003350     GO TO DX040-2.                                              DETAB-65
003355 DX040-3.                                                        DETAB-65
003360     IF NRLS = ZEROES GO TO DX040-1.                             DETAB-65
003365     MOVE 'EL001' TO DUM-6.                                      DETAB-65
003370     PERFORM DLOUT THRU RITAB.                                   DETAB-65
003375 DX040-1.                                                        DETAB-65
003380     PERFORM DXA01 THRU DXA04 VARYING EXIX FROM 1 BY 1 UNTIL     DETAB-65
003385       EXIX IS GREATER THAN NACTS.                               DETAB-65
003390     MOVE SPACES TO CRD-OUT.                                     DETAB-65
003395     MOVE 'DEXIT. EXIT.' TO BODY OF CRD-OUT.                     DETAB-65
003400     MOVE CRD-OUT TO TAPE-LIST.                                  DETAB-65
003405     PERFORM RITAB.                                              DETAB-65
003410     GO TO DT001.                                                DETAB-65
003415 L1OUT.                                                          DETAB-65
003420     MOVE LINE1  TO CRD-OUT, TAPE-LIST.  GO TO RITAB.            DETAB-65
003425 L2OUT.                                                          DETAB-65
003430     MOVE LINE2  TO CRD-OUT, TAPE-LIST.  GO TO RITAB.            DETAB-65
003435 DLOUT.                                                          DETAB-65
003440     MOVE DLABEL TO CRD-OUT, TAPE-LIST.                          DETAB-65
003445 RITAB.                                                          DETAB-65
003450     WRITE TAPE-LIST.                                            DETAB-65
003455     WRITE CRD-OUT.                                              DETAB-65
003460 RSCAN.                                                          DETAB-65
003465     MOVE 58 TO DUMIX.                                           DETAB-65
003470     PERFORM RS001 THRU RS003.                                   DETAB-65
003475 RS001.                                                          DETAB-65
003480     IF TEMP (DUMIX) = SPACE GO TO RS002.                        DETAB-65
003485     ADD 2 TO DUMIX.                                             DETAB-65
003490     GO TO RS003.                                                DETAB-65
003495 RS002.                                                          DETAB-65
003500     IF DUMIX = 1 GO TO RS003.                                   DETAB-65
```

```
003505      SUBTRACT 1 FROM DUMIX.                                          DETAB-65
003510      GO TO RS001.                                                    DETAB-65
003515 RS003.                                                               DETAB-65
003520      EXIT.                                                           DETAB-65
003525                                                                      DETAB-65
003530      NOTE DIAGNOSTIC SECTION.                                        DETAB-65
003535                                                                      DETAB-65
003540 EM01.                                                                DETAB-65
003545      MOVE ERR-1 TO ERR-IMAGE.                                        DETAB-65
003550      GO TO EM99.                                                     DETAB-65
003555 EM02.                                                                DETAB-65
003560      MOVE ERR-2 TO ERR-IMAGE.                                        DETAB-65
003565      GO TO EM99.                                                     DETAB-65
003570 EM03.                                                                DETAB-65
003575      MOVE ERR-3 TO ERR-IMAGE.                                        DETAB-65
003580      GO TO EM99.                                                     DETAB-65
003585 EM04.                                                                DETAB-65
003590      MOVE ERR-4 TO ERR-IMAGE.                                        DETAB-65
003595      GO TO EM99.                                                     DETAB-65
003600 EM05.                                                                DETAB-65
003605      MOVE ERR-5 TO ERR-IMAGE.                                        DETAB-65
003610      GO TO EM99.                                                     DETAB-65
003615 EM06.                                                                DETAB-65
003620      MOVE ERR-6 TO ERR-IMAGE.                                        DETAB-65
003625      GO TO EM99.                                                     DETAB-65
003630 EM07.                                                                DETAB-65
003635      MOVE ERR-7 TO ERR-IMAGE.                                        DETAB-65
003640      GO TO EM99.                                                     DETAB-65
003645 EM08.                                                                DETAB-65
003650      MOVE ERR-8 TO ERR-IMAGE.                                        DETAB-65
003655      GO TO EM99.                                                     DETAB-65
003660 EM09.                                                                DETAB-65
003665      MOVE ERR-9 TO ERR-IMAGE.                                        DETAB-65
003670 EM99.                                                                DETAB-65
003675      WRITE TAPE-LIST FROM ERR-PRNT.                                  DETAB-65
003680 READ-1.                                                              DETAB-65
003685      READ CARD-INPUT INTO DETAB-CRD, AT END GO TO EOF.               DETAB-65
003690      MOVE SPACES TO IDFLD.                                           DETAB-65
003695      IF IDENT OF DETAB-CRD = '0000'                                  DETAB-65
003700        MOVE '0' TO TAPE-LIST,                                        DETAB-65
003705        WRITE TAPE-LIST.                                              DETAB-65
003710      WRITE TAPE-LIST FROM DETAB-CRD.                                 DETAB-65
003715      IF IDENT OF DETAB-CRD = '999X' GO TO EOF.                       DETAB-65
003720 SKIP01.                                                              DETAB-65
003725      IF LINE-ID OF DETAB-CRD NOT = '$' GO TO READ-1.                 DETAB-65
003730      GO TO DT001.                                                    DETAB-65
003735 EOF.                                                                 DETAB-65
003740      MOVE '0END DETAB/65 PREPROCESSOR RUN.' TO TAPE-LIST.            DETAB-65
003745      WRITE TAPE-LIST.                                                DETAB-65
003750      CLOSE CARD-INPUT WITH LOCK.                                     DETAB-65
003755      CLOSE CARD-OUTPUT WITH LOCK, LIST-OUTPUT WITH LOCK.             DETAB-65
003760      STOP RUN.                                                       DETAB-65

$STOP
```

```
1  0000 TABLEXX                                L 004001003
   0001    001002ELS$
        C1  N  N
        C2     N
        C3     N
        C4  Y  N
        A1  X
     $
1  0000 TABLEXXX                               L 004001004
   0001    001002003ELS$
        C1  Y  Y  Y
        C2     Y  N
        C3     N  N
        C4  Y  N  N
        A1  X
     $
1  0000 TABLEXXXX                              L 006001004
   0001    001002003ELS$
        C1  Y  Y  Y
        C2  Y     N
        C3  N  N  N
        C4  N     N
        C5  N     N
        C6  N  Y  N
        A1  X  X  X
     $
1  0000   TEST-001                             L 003001004
   0001            001002003ELS$
             C-1     Y  Y  N
             C-2     Y  Y  Y
             C-3     Y  N  Y
           ACTION-1  X  X  X  X
     $
1  0000   TEST-002                             L 002001005
   0001            001002003004ELS$
             C-1     Y  N  Y  N
             C-2     N  Y  Y  N
           ACTION-1  X  X  X  X  X
     $
1  0000   TEST-003                             L 003001009
   0001            001002003004005006007008ELS$
             C-1     Y  Y  Y  N  Y  N  N  N
             C-2     Y  Y  N  Y  N  N  Y  N
             C-3     Y  N  Y  Y  N  Y  N  N
           ACTION-1  X  X  X  X  X  X  X  X  X
     $
1  0000   TEST-004                             L 004001017
   0001            001002003004005006007008009010011012013014015016ELS$
             C-1     Y  Y  Y  Y  N  Y  Y  N  Y  N  N  Y  N  N  N  N
             C-2     Y  Y  Y  N  Y  Y  N  N  N  Y  Y  N  N  N  Y  N
             C-3     Y  Y  N  Y  Y  N  N  Y  Y  N  Y  N  N  Y  N  N
             C-4     Y  N  Y  Y  Y  N  Y  Y  N  Y  N  N  Y  N  N  N
           ACTION-1  X  X  X  X  X  X  X  X  X  X  X  X  X  X  X  X  X
     $
1  0000   TEST-005                             L 004001003
   0001            001002ELS$
             C-1     Y  Y
             C-2     Y  Y
```

```
          C-3      Y  Y
          C-4      Y  N
        ACTION-1   X  X  X
    $
1   0000   TEST-006                              L 004001004
    0001              001002003ELS$
          C-1      Y  Y  Y
          C-2      Y  Y  Y
          C-3      Y  Y  N
          C-4      Y  N  Y
        ACTION-1   X  X  X  X
    $
1   0000   TEST-007                              L 004001005
    0001              001002003004ELS$
          C-1      Y  Y  N  -
          C-2      Y  Y  Y  Y
          C-3      Y  Y  Y  N
          C-4      Y  N  Y  Y
        ACTION-1   X  X  X  X  X
    $
    0000   TEST-009                              L 003001009
    0001              001002003004005006007008ELS$
          C-1      Y  Y  N  N  Y  N  N  Y
          C-2      Y  Y  Y  N  N  N  Y  N
          C-3      Y  N  Y  Y  Y  N  N  N
        ACTION-1   X  X  X  X  X  X  X  X  X  X
    $
1   0000   TEST-010                              L 007001009
    0001              001002003004005006007008ELS$
          C-1      Y  Y  Y  Y  N  N  N  N
          C-2      Y  Y  N  N  Y  N  Y  N
          C-3      Y  N  Y  N  N  Y  Y  N
        ACTION-1   X  X  X  X  X  X  X  X  X
    $
1   0000   TEST-011                              L 005001005
    0001              001002003003ELS$
          C-1      Y  Y  Y  N
          C-2      Y  Y  Y  Y
          C-3      Y  Y  Y  Y
          C-4      Y  Y  N  Y
          C-5      Y  N  Y  Y
        ACTION-1   X  X  X  X  X
    $
1   0000   TEST-012                              L 003001009
    0001              001002003004005006007008ELS$
          C-1      N  N  N  N  Y  Y  Y  Y
          C-2      N  N  Y  N  Y  N  N  Y
          C-3      N  Y  N  Y  N  Y  N  Y
        ACTION-1   X  X  X  X  X  X  X  X  X
    $
1   0000   TEST-013                              L 005001005
    0001              001002003004ELS$
          C-1      Y  Y  Y  Y
          C-2      Y  Y  Y  N
          C-3      Y  Y  Y  Y
          C-4      Y  Y  N  Y
          C-5      Y  N  Y  Y
        ACTION-1   X  X  X  X  X
    $
```

```
1  0000   TEST-014                           L 010001010
   0001             001002003004005006007008009ELS$
             C-1    Y  N  Y  Y  N  Y  Y  N  N
             C-2    Y  Y  Y  Y  Y  -  Y  -  N
             C-3    Y  Y  Y  Y  Y  N  Y  N  Y
             C-4    Y  Y  Y  Y  Y  N  Y  N  Y
             C-5    Y  Y  Y  Y  Y  Y  Y  Y  Y
             C-6    Y  Y  -  Y  N  -  N  N  -
             C-7    Y  Y  -  Y  N  -  -  N  -
             C-8    Y  Y  N  Y  Y  Y  Y  N  Y
             C-9    Y  Y  Y  Y  Y  Y  Y  Y  Y
             C-10   Y  Y  Y  N  Y  Y  Y  Y  Y
          ACTION-1  X  X  X  X  X  X  X  X  X  X
      $
1  0000   TEST-015                           L 010001010
   0001             001002003004005006007008009ELS$
             C-1    Y  Y  Y  N  Y  Y  Y  Y  Y
             C-2    Y  Y  Y  Y  Y  Y  Y  Y  Y
             C-3    Y  Y  N  Y  Y  Y  Y  N  Y
             C-4    Y  Y  -  Y  N  -  -  N  -
             C-5    Y  Y  -  Y  N  -  N  N  -
             C-6    Y  Y  Y  Y  Y  Y  Y  Y  Y
             C-7    Y  Y  Y  Y  Y  N  Y  N  Y
             C-8    Y  Y  Y  Y  Y  N  Y  N  Y
             C-9    Y  Y  Y  Y  Y  -  Y  -  N
             C-10   Y  N  Y  Y  N  Y  Y  N  N
          ACTION-1  X  X  X  X  X  X  X  X  X  X
      $
1  0000   TEST-016                           L 005001005
   0001             001002003004ELS$
             C-1    Y  Y  Y  Y
             C-2    N  Y  Y  Y
             C-3    -  N  Y  Y
             C-4    -  -  N  Y
             C-5    -  -  -  N
          ACTION-1  X  X  X  X  X
      $
1  0000   TEST-017                           L 005001005
   0001             001002003004ELS$
             C-1    Y  Y  Y  Y
             C-2    Y  Y  Y  N
             C-3    Y  Y  N  -
             C-4    Y  N  -  -
             C-5    N  -  -  -
          ACTION-1  X  X  X  X  X
      $
1  0000 TEST-018 (CHOISE-PICK)               L 010011010
   0001                                      001002003004005006007008009ELS$
        C1                                   Y  Y  Y  N  Y  Y  Y  Y  Y
        C2                                   Y  Y  Y  Y  Y  Y  Y  Y  Y
        C3                                   Y  Y  N  Y  Y  Y  Y  N  Y
        C4                                   Y  Y     Y  N        N
        C5                                   Y  Y     Y  N        N
        C6                                   Y  Y  Y  Y  Y  Y  Y  Y  Y
        C7                                   Y  Y  Y  Y  Y  N  Y  N  Y
        C8                                   Y  Y  Y  Y  Y  N  Y  N  Y
        C9                                   Y  Y  Y  Y  Y     Y     N
        C10                                  Y  N  Y  Y  N  Y  Y  N  N
        A1                                   X  X  X  X  X  X  X  X  X  X
```

A2
A3
A4
A5
A6
A7
A8
A9
A10
A11
$
X656