DECISION TABLES--A TOOL FOR DOCUMENTING LOGICAL CONDITION RELATIONSHIPS

Thomas G. LaFleur, Montgomery County Government, Rockville, Maryland 20850

GENERAL FORMAT The decision table basically takes the form of a matrix, which defines a set of IF-THEN relationships. There are three basic components of a decision table: (1) condition stub, the IF portion; (2) action stub, the THEN portion, (3) rules, which define the relationships of the IFs and THENs. Each rule has two parts: (1) condition entries, which define the condition relations, and (2) action entries, which define the action relationships. Each rule or column can be translated into a narrative IF-THEN statement.

GENERAL TYPES There are basically three types of decision tables: (1) limited entry, (2) extended entry, and (3) mixed entry.

Limited entry tables are restricted to the following: (1) Condition entries limited to Y, N, or -, where Y indicates that the condition in the stub must be met, N indicates that the condition in the stub must not be met, and - indicates IGNORE, it doesn't matter whether or not the condition is met. (2) Action entries limited to X or blank, where X indicates that the action in the stub is to be performed, and blank indicates that the action in the stub is not to be performed. Note that action statements must be listed in order of execution.

The limited entry example in Figure 1 shows the logic of a vending machine that sells a single product for 15¢. The machine accepts nickels, dimes, and quaters, and returns change. The table assumes unlimited product and change; the initial condition of the counters is zero.

In the extended-entry form, part of the condition or action statements are extended into the entry portion of the table. Note that actions must be listed in the order of their execution, unless the entry value indicates the sequence of execution.

In the vending machine example, above, the extended-entry form would have two independent conditions: (1) Coin is ..., with rule entries, nickel, dime, quarter, and other; (2) nickel count = ..., with rule entries 0, 1, or 2.

Mixed-entry tables are simply a combination of limited- and extended-entry types within a single table.

Figure 1

conditions	1	2	3	4	5	6	7	8	9
1. Coin is nickel	Y	Y	N	N	N	N	N	N	N
2. Coin is dime	N	N	Y	Y	Y	N	N	N	N
3. Coin is quarter	N	N	N	N	N	Y	Y	Y	N
4. Nickel count = 1	-	N	Y	N	N	Y	N	N	-
5. Nickel count = 2	N	Y	N	Y	N	N	Y	N	-
actions									
1. Reject coin									X
2. Increment nickel counter by 1	X								
3. Increment nickel counter by 2					X				
4. Deliver product		X	X	X		X	X	X	
5. Deliver 5¢				X					
6. Deliver 10¢								X	
7. Deliver 15¢						X			
8. Deliver 20¢							X		
9. Nickel count := 0		X	X	X		X	X		
10. Accept next coin & re-enter table	X	X	X	X	X	X	X	X	X

GENERAL RULES Listed below are the general rules that govern the structure and composition of decision tables:

a. The decision rules must provide all possible logical combinations of relationships that the stated conditions can reach.

b. Only one rule may satisfy a given set of condition relationships. (Each rule must be unique and independent).

c. No actions may appear in the condition area.

d. No conditions may appear in the action area.

e. No values may be changed by condition statements.

f. The table must have a single entry point--the top of the condition stub.

g. The table must have at least one exit point.

h. Exit points may only appear in the action area.

i. A table re-entry may be either to the original exit point (in the action area) or to the beginning of the table.

j. The vocabulary and grammar of the language used should be understandable to those who are to use the table.

STEPS IN BUILDING A DECISION TABLE The general steps for building a decision table are listed below. The first several steps may be repeated until a workable problem definition is reached.

a. Define the problem and its limits.

b. List the conditions and actions associated with the problem (table stub).

c. Determine the number of rules to be accounted for: (1) in the limited-entry case, 2^n, where n is the number of independent conditions; (2) in the extended-entry case, first determine the number of possible entries for each independent condition, then multiply this by 2^n.

d. Layout all of the possible condition entries: (1) First condition--Layout N/M_1 rules for each possible value assumed by this condition, where N = total number of rules determined in step c, and M = total number of possible values for this condition. Example. For a limited-entry table with five conditions, we would layout 16 Y's in the first 16 rule entry blocks for the first condition (row 1) followed by 16 N's (2^5=32, $N/M_1 = 32/2 = 16$). (2) For each subsequent condition--Layout $N/(M_1*M_2* \ldots *M_c)$ rules in succession for each possible value of condition (c) in row (c), where (c) is the condition number. Repeat this layout in row (c) until all N rules have an entry for condition (c). Example. Following the previous example, the second condition would have 8 Y's in the first 8 decision rules, 8 N's in the the next eight rules ($N/M_1*M_2 = 32/2*2 = 32/4 = 8$). This format is repeated until all 32 (N) rules have an entry, so that we end up with 8 Y's in rules 17-24 and 8 N's in rules 25-32.

e. For each column or decision rule, specify the actions that must be taken.

f. Combine and simplify: (1) Where 2 decision rules have the same series of actions and the condition entries are the same except for a single row, the two decision rules can be combined into one. In the combined rule, the entry for the row containing the difference is changed to a "-", meaning to ignore the action. This process is repeated until no further simplification can be achieved. (2) The ELSE rule--In all decision tables, rules may be combined to form an ELSE rule, one per table. This rule is the only dependent rule allowed. If none of the other rules match a given set of condition relationships, then the actions indicated in the ELSE rule. Typically, this rule is used to combine all error conditions into a common decision rule.

VARIATIONS TO CONSTRUCTING DECISION TABLES There are several practical methods of developing and simplifying the decision rules without listing all possibilities beforehand. These techniques are useful where the number of possibilities is very large.

1. Method I. Reduce the number of conditions by forming assumptions. With this method, we simplify the problem to a reasonable size. We develop and simplify, then expand the table by eliminating assumptions, adding a condition or two at a time. For each condition added, we double the number of decision rules (in a limited-entry table). We then simplify, and continue to add conditions until we have developed the table in full.

2. Method II. Set the decision rule entries for the first X conditions to one value. Then insert the entries for the remaining conditions. For a limited-entry table, we should have 2^{n-x} decision rules, with the first X conditions of each having the same value. Simplify these decision rules. Now change one value for one of the first X conditions and repeat the process. This must occur for each of the entry combinations taken by the first X conditions (2^x for the limited-entry case). As each iteration is developed, simplification can occur, with all of the decision rules developed during previous iterations.

APPLICATIONS OF DECISION TABLES IN DATA PROCESSING During detailed analysis and documentation of an existing system (manual or otherwise), the use of decision tables can assist in defining policies, regulations, interpretations, and actual methods of operation. Inconsistencies may be uncovered in addition to achieving a thorough understanding of a policy or method of operation. Decision tables can assist in redefining user procedures, including input preparation, report usage, and reconciliation. Procedures, as well as corrective actions for problems encountered, can be effectively defined in table format. Finally, documentation of program specifications and detailed program logic can be accomplished in all or in part in table format (see Figure 2).

DECISION TABLES AND FLOW CHARTS Listed below are some comparisons of decision tables and flowcharts:

1. In a flowchart, it is difficult to identify each decision rule.

2. In a flowchart, it is difficult to determine whether all combinations of conditions have been accounted for.

3. In a decision table, the entire problem and solution is in full view for examination. A flowchart normally takes many pages, and is difficult to follow.

4. When changes occur, it is more difficult to update a flowchart than a decision table.

5. Decision tables and flowcharts are not mutually exclusive; they can be used in combination.

DECISION TABLES AND PROGRAM CODING

Optimum coding can be arrived at by analysis of a decision table describing the logic to be coded. General rules for analyzing a table prior to coding are listed below:

1. Examine all condition rows. Select the condition with the fewest "-" entries. (If more than one, select one of the conditions with this minimum number of "-" entries). This will be the first condition tested. This test splits the table into two sets of decision rules.

2. Now examine one of the rule-sets or legs determined from step (1). Select as the next condition to be tested in this leg the row with the fewest "-" entries.

3. Continue breaking down the table until no further condition tests are required to identify uniquely each rule to be followed.

4. Examine the action entry portion of the rules for uniqueness. Identify those action-rules that are identical.

5. Additional analysis may be made on the action portion: (a) the relative frequency, if known, of processing each rule, (b) action groupings and identification of those rules requiring these groupings.

The above analysis will assist the programmer in determining how best to code the program, i.e., in identifying main line code and subroutines. Figure 3 shows the flowchart obtained by applying these rules to the decision table in Figure 2.

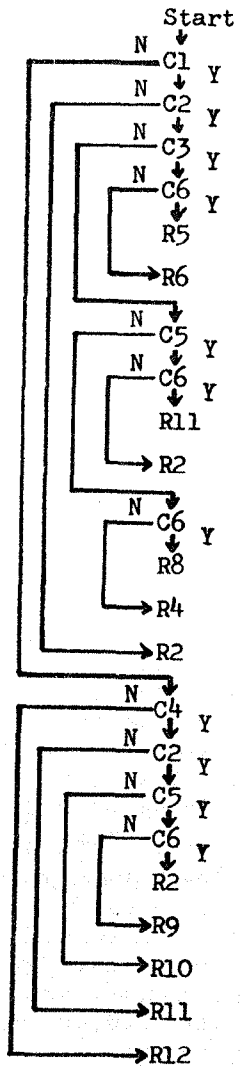
DECISION TABLE TRANSLATORS

Computer software exists, generally called decision table preprocessors, which accept decision tables as input. These preprocessors generate Cobol or other high-level language programs. The better ones perform optimizing analysis, as mentioned above, while generating the output program.

Figure 2

	R1	R2	R3	R4	R5	R6	R7	R8	R9	R10	R11	R12
C1	Y	Y	Y	Y	Y	Y	Y	N	N	N	N	N
C2	Y	Y	Y	Y	Y	Y	N	Y	Y	Y	N	-
C3	N	N	N	N	Y	Y	-	-	-	-	-	-
C4	-	-	-	-	-	-	-	Y	Y	Y	Y	N
C5	Y	Y	N	N	-	-	-	Y	Y	N	-	-
C6	Y	N	Y	N	Y	N	-	Y	N	-	-	-
A1	X		X		X		X			X	X	
A2		X	X	X	X			X	X		X	
A3		X						X	X			
A4				X		X	X					
A5	X		X			X	X		X	X		X

Figure 3



Note: The Rn entries, above, refer only to the Action portions of the rules, since the Cn entries exhaust the condition entries. Thus, we can substitute rules where the actions sequences are the same for more than one rule.

Some of the advantages and disadvantages of decision table programming via preprocessor are listed below.

1. Decision table language is designed to make simple, clear, unambiguous statements. Its notation, borrowed from symbolic logic, assures greater accuracy and completeness in problem definition. Systems maintenance due to overlooked situations or inconsistencies should be reduced.

2. Decision tables are an excellent tool for analysis and programming. They can therefore simplify communication between the systems analyst and the programmer.

3. Programming time is considerably reduced. According to users' experience, decision table processors require less time in problem definition (flow charting vs. decision table) less time in coding, and less time debugging.

4. Although total computer time utilized for implementing a program should decrease, individual program compile time (due to table preprocessing) will increase.

5. Adequate education on the use of decision tables is required. First attempts at using decision tables can be slow and painful, as with any new tool.

NOTE: Montgomery County, Maryland (Data Processing) has been using decision tables for approximately one year, and can attest to the pain of learning, but have become enthusiastic users of decision tables for problem definition, analysis, and programming.

REFERENCES McDaniel, H. (editor) Applications of Decision Tables, Brandon Systems Press, Inc. 1970. This book is a series of extracts by many authors, and includes many references to the use of decision table processors.

McDaniel, H. Decision Table Software, Brandon Systems Press, Inc. 1970.

StClair, Jr., Paul R. Decision Tables Clear the Way for Sharp Selection, Computer Decisions, February 1970, pages 14-18.

UTILIZATION OF A DECISION TABLE TRANSLATOR FOR BASIC PROGRAM CREATION

Henry O. Arnold, Jr, Management Systems Development Office

I. DESCRIPTION This decision table translator is part of a package of programs written in Basic to generate and execute Basic programs. This package allows the programmer to use symbolic addresses rather than program line-numbers, and multiple conditions in IF-statements. The package generates line numbers incremented by 100, and provides several levels of statement indentation, for i/o, program transfers, paragraph names, etc.

The decision table is generated in-line with the remainder of the program.

II. ADVANTAGES The coding generated resembles the input table to the point that individual conditions or actions can be changed without regenerating the program; also, condition and action entries are stored in the program as DATA-statements, and can easily be changed in the generated program.

The action entries are sequenced for each rule, so that actions can be executed in different sequences in each rule, as opposed to normal sequence from top to bottom.

Table coding is not dependent on columns, but is controlled by delimiters.

If need be, additional rules may be added, by modifying DATA-statements, without regenerating the program.

III. LIMITATIONS Only one table can be generated within a program.

Other DATA-statements used by the programmer must be entered after the table.

Certain variables are used by the table, and should be used elsewhere only with discretion.

The program generated is probably larger than that created by other translators.

Maximum table size, at present, is: 9 conditions and actions, and 10 rules.

IV. PROGRAMMING CONSIDERATIONS Programming language is essentially Basic, as implemented by Alcom's timesharing system, with certain modifications:

1. No line numbers are used.
2. © is used in place of ".