MEMORANDUM
RM-3669-PR
MAY 1963

ANALYSIS OF THE DECISION RULES IN DECISION TABLES

Solomon L. Pollack

PREPARED FOR:
UNITED STATES AIR FORCE PROJECT RAND

The RAND Corp.
SANTA MONICA • CALIF

MEMORANDUM
RM-3669-PR
MAY 1963

ANALYSIS OF THE DECISION RULES IN DECISION TABLES

Solomon L. Pollack

This research is sponsored by the United States Air Force under Project RAND—contract No. AF 49(638)-700 monitored by the Directorate of Development Planning, Deputy Chief of Staff, Research and Development, Hq USAF. Views or conclusions contained in this Memorandum should not be interpreted as representing the official opinion or policy of the United States Air Force.

The RAND Corporation
1700 MAIN ST. • SANTA MONICA • CALIFORNIA

PREFACE

The Systems Group of the CODASYL (Conference on Data System Languages)[1][2] Development Committee has produced an experimental language, DETAB-X,[3][4] that is structured on decision tables, a framework for describing a set of related decision rules.

Over the past year, the author has discussed with many Air Force people the possibilities and advantages of using decision tables to describe complex applications that involve numerous decision rules. The Air Force, in turn, has indicated a strong interest in the tables in order to apply decision rules in such areas as personnel and financial accounting. Their interest is predicated upon the improvement in communication and documentation that decision tables offer over previous techniques, such as flow chart and narrative form.

Decision tables offer system analysts the potential to eliminate inconsistencies and redundancies among a set of specified decision rules, and to insure completeness of problem statement. The tables may make it possible to have more efficient computer programs, by reducing needs for computer storage and reducing the length of computer running time.

To realize these and other benefits, we must understand the fundamental characteristics of decision rules and their relationships within a decision table. Toward this end, this Memorandum develops a theoretical basis for analyzing the decision rules of decision tables in general, and of a specific type of table used in DETAB-X.

This Memorandum will interest a wide reading audience -- system analysts, military and business managers, mathematicians and computer

programmers may find, in the theories advanced here, specific applications to their own work. To ease the reader's task, the text has been developed in two parts. The first (Secs. I through IV) gives background information on decision tables, discusses the implications of the basic theorems developed in Secs. V and VI, and provides examples to illustrate the relationships among decision rules in a table. The second part states the basic assumptions of the theorems and then provides proofs for each of them; those readers having a working knowledge of decision tables can move directly to Sec. V.

The comments of Stanley Naftali and Harley Robertson, both of Space Technology Laboratories, El Segundo, California, were extremely useful to the author, particularly in the early stages of development of the decision table theory.

SUMMARY

Decision tables, a framework for describing a set of related decision rules, can improve the communication and documentation achievable by previous techniques such as flow charting and narrative descriptions of data processing problems, particularly those containing many complex decision rules. In addition, decision tables offer system analysts the potential to eliminate inconsistencies and redundancies in each set of specified decision rules and to produce computer programs that are efficient in the use of computer storage and computer running time. Decision tables also enable the system analyst to determine if he has considered all of the possible decision rules that can be formed from a particular set of conditions.

This Memorandum develops for decision tables a theoretical structure that serves as the foundation for achieving these benefits. As background for this development, the author describes a basic structure of decision tables in Secs. I-IV. The theorems developed in this paper (Secs. V-VII) 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 actions 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 completness, 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 satisfied. This type of decision rule can be extremely useful in editing and information retrieval. This extension should prove valuable in many data processing areas.

CONTENTS

I. INTRODUCTION

Throughout military and business systems, a series of actions is taken only when a set of conditions is satisfied. The expression that describes the conditions and actions is called a Decision Rule. For instance, one example of such a rule is the policy governing hourly wage earners -- if an employee works more than 40 hours in one week and is not salaried, then that employee shall be paid an overtime rate.

Decision rules can be written in any language and in any form as long as they are intelligible. A popular method of expression is by means of flow charts (illustrated in Fig. 1). The flow chart technique has serious defects, however. First, the charts are difficult to draw because of the symbols and spacing. Second, they are difficult to comprehend, in that it is hard to follow the exact path of a series of conditions and actions through the charts. Third, it is difficult to determine whether the charts cover all possible cases. And fourth, it is hard to insure the specification of the same series of actions for a particular set of conditions. These same drawbacks apply in a larger degree to the free-form English used to describe the decision rules.

The need for faster and better communication and analysis has led to the development of Decision Tables, structures for describing a set of decision rules. Computer languages, adapted for decision tables, have been developed for describing and processing scientific and business problems. Examples in the scientific area are FORTAB[5] and STRUCTURE TABLE LANGUAGE,[6] while the business area has TABSOL[7] and DETAB-X.[4]

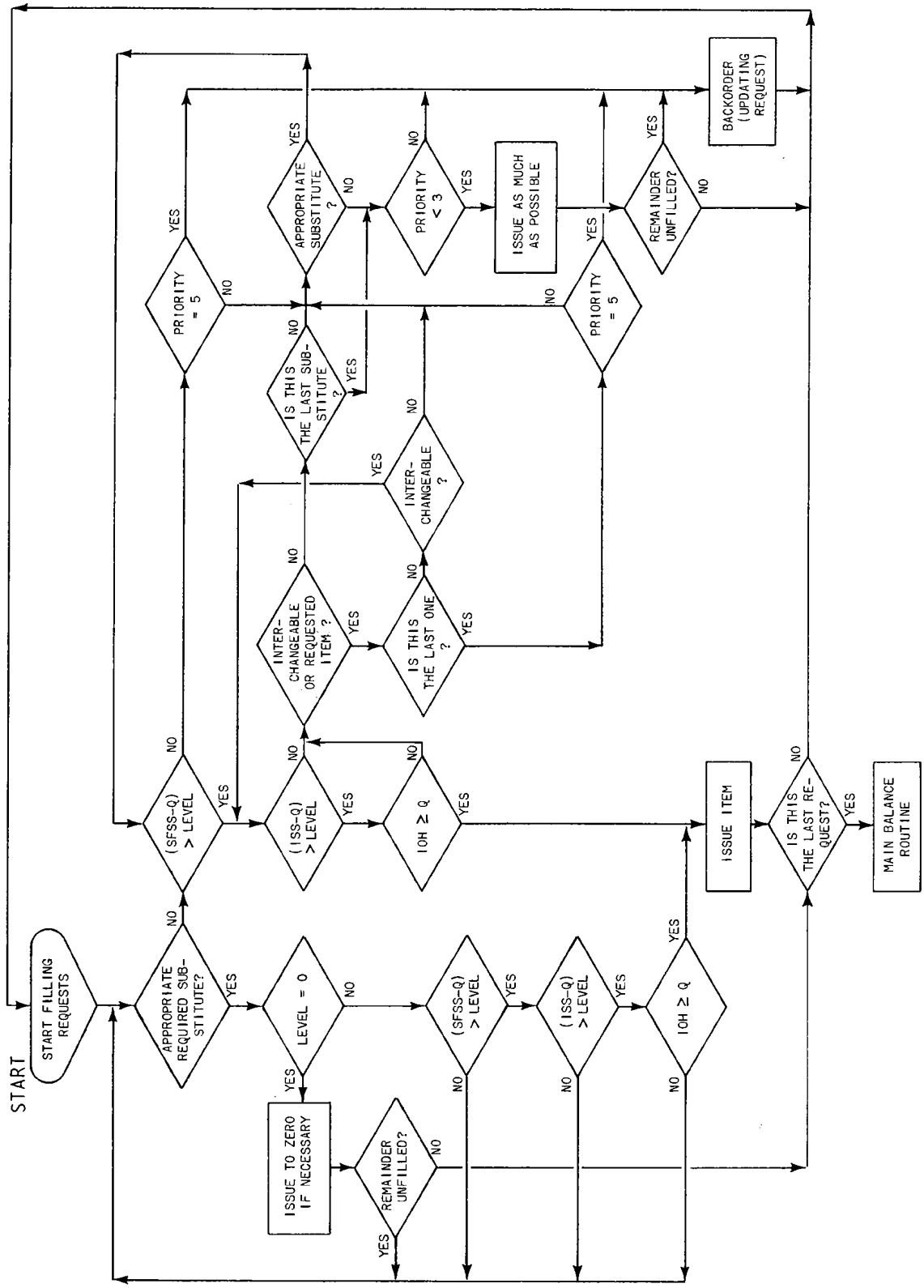


Fig. 1 -- Substitution and Allocation: General Procedure for Handling Requests

System analysts and programmers use decision tables to describe the decision rules for their business data processing systems or for their scientific problems. To assist these specialists, this Memorandum describes a set of rules for insuring that each decision table is complete, and contains no redundant or contradictory rules. The rules described here are based on a collection of decision-table theorems formulated in the latter half of this Memorandum. As background for the discussion, the author briefly explains the basic structure of decision tables,* and then shows how the theorems apply to a particular decision-table language, DETAB-X.

---

*For a fuller description of decision-table structure and development, see refs. 8 and 9.

II. BASIC STRUCTURE OF DECISION TABLES

Decision tables contain decision rules. The basic structure of the tables is universal. Their language, however, varies depending on the application being described. While the format of decision tables can be of many types, one that is in general use is described here and illustrated in Table 1.

Table 1

DECISION TABLE STRUCTURE

		Decision Rule 1	Decision Rule 2	Decision Rule 3	Decision Rule 4	Decision Rule 5	Decision Rule 6
if							
and	Condition Stubs			Condition Entries			
and							
and							
then							
and	Action Stubs			Action Entries			
and							

The vertical and horizontal double lines divide the table into four major parts. All entries above the horizontal double line are conditions; all below are actions. The boxes to the left of the vertical double line are called stubs; the boxes to the right are called entries. A condition is described by what is in a stub and entry above the double horizontal line; an action is described by what is in a stub and entry below. Single horizontal lines separate each condition and action. Single vertical lines separate each decision rule.

We interpret Table 1 in this manner. The combination of the contents of the condition stubs and entries specifies a condition that must be satisfied; the combination of the contents of the action stubs and entries specifies an action that must be executed. The top horizontal is read as "if." All other single horizontal lines are read as "and." The horizontal double line is read as "then." The decision rules are read down, combining each entry with its stub. The "Y" says the condition in the stub must be satisfied; the "N" says the condition in the stub must not be satisfied; the "I" says the condition in the stub is immaterial.

Table 2

SAMPLE DECISION TABLE

	Rule 1	Rule 2	Rule 3
SALARIED-EMPLOYEE	N	N	Y
HOURS-WORKED > 40	Y	N	I
PAY	OVERTIME-RATE	REGULAR-RATE	REGULAR-SALARY

For example, Table 2 is read this way:

> Decision Rule 1 states: If employee is not salaried and worked over 40 hours, then pay his overtime rate.
>
> Decision Rule 2 states: If employee is not salaried and worked 40 hours or less, then pay his regular rate.
>
> Decision Rule 3 states: If employee is salaried, regardless of whether he worked over 40 hours or not, then pay his regular salary.

LIMITED ENTRY VERSUS EXTENDED ENTRY

Each horizontal line of a table must be either limited entry or extended entry. A limited-entry condition line contains the whole condition in the stub, a Y (yes), N (no), or I (immaterial) in each

entry box of that line. A limited entry action line contains the whole action in the stub, an X (execute) or "-" (don't execute) in each entry box of that line. An extended-entry line contains part of the condition (or action) in the stub and part of the condition (or action) in the entry box of that line.

For example, lines 1 and 2 in Table 2 are each limited-entry lines; line 3 is an extended-entry line. It is permissible to have both limited-entry lines and extended-entry lines in the same table, but any one line is either limited-entry or extended-entry. If required, limited-entry lines can be converted to extended-entry lines and vice-versa.

For example, line 2 in Table 2 could have been written as follows:

HOURS-WORKED	>40	$\leq$ 40	$\geq$ 0

or line 3 in Table 2 could have been written

PAY OVERTIME-RATE	Y	-	-
PAY REGULAR-RATE	-	Y	-
PAY REGULAR-SALARY	-	-	Y

Note: The dash "-" signifies that the action listed in the stub is to be ignored, i.e., the action shall not be executed.

The remainder of this paper deals only with decision tables that have limited-entry lines in the condition area of the table, i.e., limited-entry lines above the horizontal double-line.*

A special class of decision tables, DETAB-X[4][11] specified by the CODASYL Systems Group, is presently under test in the

*Concurrent research on extended-entry condition lines is going on. For one effort in this area, see Ref. 10.

data processing community. The next section describes the requirements of DETAB-X decision tables and the implications on them of the theorems developed in the latter half of this paper. But first, we offer some additional notation.

DECISION RULE NOTATION

The AND-FUNCTION of a decision rule is the ordered set of Y, N, or I that appears in the condition entry boxes of that decision rule. For example, in Table 2

the AND-FUNCTION of Rule 1 = NY,

the AND-FUNCTION of Rule 2 = NN, and

the AND-FUNCTION of Rule 3 = YI.

A decision rule is satisfied by a transaction if all the conditions specified by its AND-FUNCTION are identical with the corresponding conditions specified in the transaction. Again referring to Table 2, if an employee hands in a work report on which he states that he is not salaried and he worked 45 hours, then Rule 1 is satisfied by that work report. The act of comparing a transaction against each of the rules of a decision table is called testing the decision rules.

We now define two AND-FUNCTIONS to be dependent if there exists at least one transaction such that both AND-FUNCTIONS are satisfied by that transaction. Otherwise, they are independent.

For example, referring once more to Table 2, AND-FUNCTION 1 and AND-FUNCTION 2 are independent. On a transaction of each salaried employee, either the hours worked is greater than 40 or is not. If they are greater than 40, Rule 1 is satisfied and Rule 2 is not; if

they are less than 40, Rule 2 is satisfied and Rule 1 is not. Hence Rules 1 and 2 are independent.

We define a pure AND-FUNCTION as one that contains only Y's and/or N's; i.e., it contains no I. A mixed AND-FUNCTION contains one or more I's. We define a simple decision rule as one whose AND-FUNCTION is pure; a complex decision rule is one whose AND-FUNCTION is mixed. Rule 1 and Rule 2 in Table 2 are each simple decision rules; Rule 3 is complex.

III. ANALYSIS OF DETAB-X DECISION TABLES

In this section, we use the theorems developed in Sec. V, and the special type of decision tables specified for DETAB-X to describe how table rules can be analyzed for completeness, redundancies, and inconsistencies. But first, we describe two significant requirements that the CODASYL Systems Group specified for DETAB-X decision tables.

Requirement 1: Every decision rule must specify at least one action. It makes no sense to say "If employee works overtime and he is an hourly worker," without specifying at least one action to take if these conditions are met. It is perfectly reasonable, however, to say "Give every employee an extra holiday," without specifying any conditions.

Requirement 2: Each transaction that tests the decision rules of a decision table must be able to satisfy one, and only one, of them. This requirement offers advantages in two vital areas: (1) it insures completeness of decision tables, and (2) it reduces contradictions and redundancies among decision rules. We discuss these areas subsequently.

Other computer languages have alternatives to Requirement 2. For example, FORTAB tests rules in order starting from the left and working toward the right until one rule is satisfied; the actions of the satisfied decision rule are then executed. With FORTAB then, it is possible to satisfy more than one decision rule with a single transaction, with the leftmost satisfied rule being executed. Checking FORTAB tables for completeness, redundancy and contradiction of decision rules would require more complex rules than those we now describe.

COMPLETENESS OF DECISION TABLES

As previously stated, a decision table is complete if, and only if, (1) every decision rule contains at least one action; and (2) each transaction that tests the table rules satisfies one, and only one, decision rule. This means that the table must contain all of the possible independent AND-FUNCTIONS that can be formed from the possibilities listed in the condition stubs.

A natural question is "How does one determine that a decision table is complete"? It is easy to satisfy Requirement 1 by verifying that each decision rule in the table has at least one action. But determining if Requirement 2 is satisfied (each transaction satisfies only one rule) requires a further explanation which we shall give by examining decision tables that contain:

1. Simple decision rules only,
2. One or more complex decision rules and no ELSE-Decision-Rule,
3. An ELSE-Decision-Rule.

Decision Tables Containing Simple Decision Rules Only

Theorem V states that there exist exactly 2^n independent AND-FUNCTIONS in a table based on n conditions. Hence a table must specify or imply 2^n independent decision rules in order for there to exist one, and only one, decision rule a transaction can satisfy. As an example, a decision table based on 3 conditions should contain 2^3 (or 8) independent AND-FUNCTIONS (see Table 3).

Theorem IV states that a table based on n conditions contains exactly 2^n distinct pure AND-FUNCTIONS each of which is independent

of every other. Therefore, if a decision table contains simple decision rules only, it must contain exactly 2^n distinct pure AND-FUNCTIONS. Table 3 contains simple decision rules only, i.e., its AND-FUNCTIONS contain no I's. The table is based on 3 conditions; it should contain $2^3 = 8$ decision rules. It does. Are they independent? Yes, since Theorem I states that two AND-FUNCTIONS are independent if, in at least one position, one function contains Y and the other function

Table 3

CREDIT APPROVAL DECISION TABLE

	Rule							
	1	2	3	4	5	6	7	8
CREDIT OK	Y	Y	Y	Y	N	N	N	N
PAY-EXPERIENCE FAVORABLE	Y	Y	N	N	Y	Y	N	N
SPECIAL-CLEARANCE OBTAINED	Y	N	Y	N	Y	N	Y	N
APPROVE ORDER	X	X	X	X	X	X	X	-
RETURN ORDER TO SALES	-	-	-	-	-	-	-	X

Note: "X" says execute the action in the Stub.
"-" says do not execute the action in the Stub.

contains N. In Table 3, this occurs for every pair of AND-FUNCTIONS; hence, they are independent.

In summary, a decision table that is based on n conditions and contains simple decision rules only is complete if it has 2^n distinct decision rules, and each decision rule contains at least one action.

Decision Tables Containing One or More Complex Decision Rules and No ELSE-Decision-Rule

Table 3 contains several decision rules -- Rules 1 through 7 -- which take the same action, regardless of whether the condition in

one of the stubs has a Y or N associated with it. Therefore, Table 3 can be rewritten as a table which contains simple and complex decision rules, i.e., those whose AND-FUNCTIONS contain at least one I. Rules 1 through 4 of Table 3 we rewrite as complex Decision Rule 1 in Table 4; Rules 5 and 6 we rewrite as complex Decision Rule 2.

Table 4

CREDIT APPROVAL DECISION TABLE

	Rule 1	Rule 2	Rule 3	Rule 4
CREDIT OK	Y	N	N	N
PAY-EXPERIENCE FAVORABLE	I	Y	N	N
SPECIAL-CLEARANCE OBTAINED	I	I	Y	N
APPROVE ORDER	X	X	X	-
RETURN ORDER TO SALES	-	-	-	X

Assuming Table 4 as a starting point, it is possible to reverse the process described above and expand it to Table 3 using Theorem III.* Following expansion, the procedure described earlier for showing that Table 3 is complete could be followed thereby deducing the completeness of Table 4. Another preferable procedure for testing the completeness of Table 4 exists and is presented below:

1. Check that each decision rule contains at least one action.

2. Use Theorem I to show that the AND-FUNCTIONS of Table 4 are independent. Theorem I states that two AND-FUNCTIONS are independent if in at least one position, one function contains Y and the other function contains N. Note that this is true for every pair of AND-FUNCTIONS in Table 4. Hence, the AND-FUNCTIONS are independent.

*The reader, as an exercise, can expand Table 4 to Table 3, using Theorem III. (See Appendix for the statement of all theorems.)

3. Then show that the 4 decision rules in Table 4 imply (or are equivalent to) 8 decision rules. For this, refer to Theorem VI which states that each decision rule containing an AND-FUNCTION with I in r positions is equivalent to 2^r simple decision rules. Hence, in Table 4:

   Rule 1 is equivalent to 2^2 (or 4) simple decision rules;

   Rule 2 is equivalent to 2^1 (or 2) simple decision rules;

   Rule 3 is equivalent to 1 simple decision rule; and

   Rule 4 is equivalent to 1 simple decision rule.

   Total 8

Since Rules 1, 2, 3, and 4 of Table 4 are equivalent to 8 simple independent decision rules, Table 4 is complete.

Decision Tables Containing an ELSE-Decision-Rule

In decision tables based on many conditions, and consequently containing many decision rules, it is often highly desirable to group as one those decision rules that specify the same series of actions. This decreases the amount of writing and reduces the amount of computer coding, thereby reducing errors and decreasing the amount of computer storage required for the program.

The ELSE-Decision-Rule satisfies this purpose. The AND-FUNCTION of the ELSE-Decision-Rule is equal to the disjunction of all independent AND-FUNCTIONS not specified or implied by the written decision rules in the table. Assume, for example, that Table 6 is equivalent to Table 5. Then the ELSE-Decision-Rule (abbreviated ELS) in Table 6 replaces Rules 4, 5, and 6 of Table 5; therefore the AND-FUNCTION of ELS is equal to the disjunction of (is the equivalent of) AND-FUNCTIONS 4, 5, and 6.

Table 5

DEPRECIATION EXPENSE

	Rule					
	1	2	3	4	5	6
ASSET-PURCHASED	Y	Y	N	Y	Y	Y
PROPERTY-CLASS ≥ "A"	Y	Y	I	N	Y	N
PROPERTY-CLASS ≤ "J"	Y	Y	I	Y	N	N
ASSET-NEW-WHEN-PURCHASED	Y	N	I	I	I	I
COMPUTE DEPREC-EXPENSE BY	SUM OF DIGITS	STRAIGHT LINE	-	-	-	-
GO TO ASSET-LEASED-TABLE	-	-	X	-	-	-
WRITE ERROR-MESSAGE	-	-	-	X	X	X

Table 6

DEPRECIATION EXPENSE

	Rule			
	1	2	3	ELS
ASSET-PURCHASED	Y	Y	N	-
PROPERTY-CLASS ≥ "A"	Y	Y	I	-
PROPERTY-CLASS ≤ "J"	Y	Y	I	-
ASSET-NEW-WHEN-PURCHASED	Y	N	I	-
COMPUTE DEPREC-EXPENSE BY	SUM OF DIGITS	STRAIGHT LINE	-	-
GO TO ASSET-LEASED-TABLE	-	-	X	-
WRITE ERROR-MESSAGE	-	-	-	X

In the general case, if t independent AND-FUNCTIONS are specified or implied in a table based on n conditions, the ELSE-Decision-Rule is defined as the equivalent of (2^n-t) decision rules, each of whose AND-FUNCTIONS are independent of each other and the t specified or implied AND-FUNCTIONS. Expanding the specified AND-FUNCTIONS of Table 6 results in t = 10 AND-FUNCTIONS shown below:

$$\begin{bmatrix}Y\\Y\\Y\\Y\end{bmatrix},\begin{bmatrix}Y\\Y\\Y\\N\end{bmatrix},\begin{bmatrix}N\\Y\\Y\\Y\end{bmatrix},\begin{bmatrix}N\\Y\\Y\\N\end{bmatrix},\begin{bmatrix}N\\Y\\N\\Y\end{bmatrix},\begin{bmatrix}N\\Y\\N\\N\end{bmatrix},\begin{bmatrix}N\\N\\Y\\Y\end{bmatrix},\begin{bmatrix}N\\N\\Y\\N\end{bmatrix},\begin{bmatrix}N\\N\\N\\Y\end{bmatrix},\begin{bmatrix}N\\N\\N\\N\end{bmatrix}$$

and since n = 4, $2^n = 2^4 = 16$. Therefore, $(2^n-t) = 16-10 = 6$.

The following are expressed by the ELSE AND-FUNCTION:

$$\begin{bmatrix}Y\\N\\Y\\Y\end{bmatrix},\begin{bmatrix}Y\\N\\Y\\N\end{bmatrix},\begin{bmatrix}Y\\Y\\N\\Y\end{bmatrix},\begin{bmatrix}Y\\Y\\N\\N\end{bmatrix},\begin{bmatrix}Y\\N\\N\\Y\end{bmatrix},\begin{bmatrix}Y\\N\\N\\N\end{bmatrix}$$

These are equivalent to the AND-FUNCTIONS of Rules 4, 5, and 6 of Table 5. The above 6 AND-FUNCTIONS were derived by noting that where there are 4 conditions, there are 16 AND-FUNCTIONS. Among the 16, 8 will have Y in the first row, and 8 will have N in the first row; the second, third and fourth rows are similar. No pair can be equal. The original 10 AND-FUNCTIONS had 2 Y's and 8 N's in the first row. Hence, the remaining 6 AND-FUNCTIONS must have a Y in the first row. In the second row, the original 10 AND-FUNCTIONS had 6 Y's and 4 N's. Hence, the remaining 6 AND-FUNCTIONS must have 2 Y's and 4 N's. Looking at the 1st and 2nd row only, there should be

4 " $\begin{bmatrix} Y \\ Y \end{bmatrix}$ ", 4 " $\begin{bmatrix} Y \\ N \end{bmatrix}$ ", 4 " $\begin{bmatrix} N \\ Y \end{bmatrix}$ ", and 4 " $\begin{bmatrix} N \\ N \end{bmatrix}$ ". There exist only 2 " $\begin{bmatrix} Y \\ Y \end{bmatrix}$ " and no " $\begin{bmatrix} Y \\ N \end{bmatrix}$ ". Hence, 2 " $\begin{bmatrix} Y \\ Y \end{bmatrix}$ " and 4 " $\begin{bmatrix} Y \\ N \end{bmatrix}$ " must exist in the 6 AND-FUNCTIONS. This process is completed until all the rows have been filled in.

A decision table having an ELSE-Decision-Rule is complete if each decision rule (including ELS) contains at least one action. It is interesting to note that the ELSE-Decision-Rule is automatically satisfied when all the specified or implied decision rules are not satisfied. Hence, of all the rules in a decision table, the ELSE-Decision-Rule must be tested last. As to rules other than ELS, their testing order is immaterial since any transaction can satisfy only one rule. For example, entry into Table 5 with a transaction is the same whether we test Rules 1, 2, 3, 4, 5, and 6 in order, or if we test Rules 3, 2, 5, 4, 6, and 1 in that order. In Table 6, however, Rules 1, 2, and 3 must be tested first in any order; if they all fail, ELS is automatically satisfied. This follows because each transaction must satisfy one and only one rule. Therefore, if Rules 1, 2, and 3 are not satisfied, and ELS represents the remaining decision rules, the transaction must satisfy it.

CONTRADICTIONS AND REDUNDANCIES AMONG DECISION RULES

The preceding discussion on completeness of decision tables assumes that Requirement 2 is not violated. If it is violated, the decision table contains redundant and/or contradictory rules. The following discussion assumes that Requirement 1 is not violated.

Redundant Decision Rules

If Requirement 2 is violated, there exists at least one transaction that satisfies two or more decision rules, i.e., the AND-FUNCTIONS of two or more decision rules are dependent. For ease of discussion consider Rules 1 and 2, both of which can be satisfied by one transaction, i.e., their AND-FUNCTIONS are dependent. If the two AND-FUNCTIONS are identical and the sequence of actions specified for Rules 1 and 2 are identical, then one of them is redundant; someone mistakenly repeated a decision rule.

It is not as apparent that redundancy exists if the two AND-FUNCTIONS are not identical, even though they are dependent; for AND-FUNCTION 1 and AND-FUNCTION 2 cannot both be pure AND-FUNCTIONS by virtue of Theorem II which states that each pure AND-FUNCTION is independent of every other pure AND-FUNCTION. There are then two possible explanations:

Case I - One of the AND-FUNCTIONS is pure and the other is mixed.

Case II - Both AND-FUNCTIONS are mixed.

For Case I, refer to Theorem IV, corollary 3, which states that if a pure AND-FUNCTION and a mixed AND-FUNCTION are dependent, the pure AND-FUNCTION is contained in the canonical form* of the mixed AND-FUNCTION. Since Rules 1 and 2 have the same sequence of actions, the decision rule that contains the pure AND-FUNCTION is redundant. This can be illustrated by looking at Rules 1 and 2 of Table 7.

---

*The canonical form describes a mixed AND-FUNCTION in pure terms.

Table 7

CREDIT APPROVAL

	Rule 1	Rule 2	Rule 3	Rule 4
CREDIT OK	Y	Y	N	N
PAY-EXPERIENCE FAVORABLE	I	Y	Y	N
APPROVE ORDER	X	X	X	-
RETURN ORDER TO SALES	-	-	-	X

Their AND-FUNCTIONS are dependent (by Theorem I). Rule 1 breaks down as follows:

$$\begin{bmatrix} Y \\ I \\ \underline{\underline{X}} \\ - \end{bmatrix} \longrightarrow \begin{bmatrix} Y \\ Y \\ \underline{\underline{X}} \\ - \end{bmatrix}, \begin{bmatrix} Y \\ N \\ \underline{\underline{X}} \\ - \end{bmatrix}.$$

(1) (2) (3)

Note that (2) is identical to Rule 2, Table 7. Hence, Rule 2 of Table 7 is redundant. In other words, if credit is ok, approve order.

Case II, where both AND-FUNCTIONS are mixed, refer to Theorem IV, Corollary 4, which states that if two mixed AND-FUNCTIONS are dependent there exists in their canonical form at least one pure AND-FUNCTION that is common to both. The one or more common pure AND-FUNCTIONS constitute the redundancy in this case and can be eliminated by removing each redundant common AND-FUNCTION. For example, in Table 8, by Theorem I, AND-FUNCTIONS 1 and 2 are dependent.

Rule 1 breaks down as follows: $\begin{bmatrix} Y \\ I \\ \underline{\underline{X}} \\ - \end{bmatrix} \longrightarrow \begin{bmatrix} Y \\ Y \\ \underline{\underline{X}} \\ - \end{bmatrix}, \begin{bmatrix} Y \\ N \\ \underline{\underline{X}} \\ - \end{bmatrix}.$

Rule 2 breaks down as follows: $\begin{bmatrix} I \\ Y \\ \hline X \\ - \end{bmatrix} \rightarrow \begin{bmatrix} Y \\ Y \\ \hline X \\ - \end{bmatrix}, \begin{bmatrix} N \\ Y \\ \hline X \\ - \end{bmatrix}.$

The common AND-FUNCTION is $\begin{bmatrix} Y \\ Y \\ \hline X \\ - \end{bmatrix}$

which can be eliminated from Rules 1 and 2. Eliminating it from Rule 2 produces the redundancy-free Decision Table 9. Note that there are 5 decision rules in Table 8 ($2^1 + 2^1 + 1 = 5$) which is more than the required $4 = 2^2$(2 conditions). In Table 9, by eliminating a redundant decision rule, the number of independent decision rules is $2^1 + 1 + 1 = 4$.

Table 8

CREDIT APPROVAL

	Rule 1	Rule 2	Rule 3
CREDIT OK	Y	I	N
PAY-EXPERIENCE FAVORABLE	I	Y	N
APPROVE ORDER	X	X	-
RETURN ORDER TO SALES	-	-	X

Table 9

CREDIT APPROVAL

	Rule 1	Rule 2	Rule 3
CREDIT OK	Y	N	N
PAY-EXPERIENCE FAVORABLE	I	Y	N
APPROVE ORDER	X	X	-
RETURN ORDER TO SALES	-	-	X

Summary in Decision-Table Form of Contradiction and Redundancy

All possible contradiction and redundancy situations can be summarized in decision table form by representing all pairs of decision rules in a decision table by Rules 1 and 2 whose AND-FUNCTIONS are AF1 and AF2 respectively, and whose series of actions are A1 and A2 respectively. This is shown in Table 10.

Table 10 is complete because each decision rule has at least one action (passive) and the AND-FUNCTIONS of the decision rules are independent of each other. There are $2^4 + 2^2 + 1 + 1 + 1 + 1 + 2^3 = 32$ decision rules. Since the table is based on five conditions, there should be $2^5 = 32$ independent decision rules. Table 10, then, is complete.

DECISION TABLES VIS-A-VIS FLOW CHARTS

To illustrate some advantages decision tables have over flow charts, the decision rules diagrammed in Fig. 1 are written into Tables 11A, 11B, and 11C. In the flow chart, it is difficult to recognize each decision rule; in the tables each rule is clearly defined. The reader has no way of determining whether a particular decision has been omitted from the chart, thus making the satisfaction by certain transactions impossible. The first time a transaction tests a decision table and does not satisfy one of the specified decision rules, the ELSE-Decision-Rule will reject it as an error. Any redundancies and contradictions among the decision rules in a table can be located using Theorem I. Tables 11A, 11B, and 11C contain no contradictory or redundant decision rules; this cannot be determined from the flow chart (see Fig. 1).

Table 10

CONTRADICTION AND REDUNDANCY OF DECISION RULES

	Rule 1	Rule 2	Rule 3	Rule 4	Rule 5	Rule 6	Rule 7
AF1 & AF2 are dependent	N	Y	Y	Y	Y	Y	Y
AF1 = AF2	I	Y	N	N	N	N	I
A1 = A2	I	Y	Y	Y	Y	Y	N
AF1 is pure	I	I	Y	Y	N	N	I
AF2 is pure	I	I	Y	N	Y	N	I
Rules 1 and 2 are valid	Y	Y	-	Y	Y	Y	N
Redundant rule is	-	Rule 2	-	Rule 1	Rule 2	-	-
Rules 1 and 2 contain redundant rules	N	N	-	N	N	Y	-
Rules 1 and 2 are contradictory	N	N	-	N	N	N	Y
This rule is impossible	N	N	Y	N	N	N	N

Table 11A

SUBSTITUTION AND ALLOCATION PROCEDURE FOR HANDLING REQUESTS

	Rule											
	1	2	3	4	5	6	7	8	9	10	11	ELS
APPROPRIATE REQUIRED SUBSTITUTE?	Y	Y	Y	Y	Y	Y	N	N	N	N	N	-
SFSS-Q ≥ WARNING LEVEL	Y	I	I	N	Y	Y	Y	N	N	N	Y	-
ISS-Q ≥ WARNING LEVEL	Y	I	I	I	N	Y	Y	I	I	I	N	-
INVENTORY-ON-HAND ≥ QUANTITY	Y	I	I	I	I	N	Y	I	I	I	I	-
APPROPRIATE SUBSTITUTE?	I	I	I	I	I	I	I	I	Y	N	I	-
LEVEL = 0	N	Y	Y	N	N	N	I	I	I	I	I	-
REMAINDER UNFILLED?	I	Y	N	I	I	I	I	I	I	I	I	-
PRIORITY = 5	I	I	I	I	I	I	I	Y	N	N	I	-
ISSUE QUANTITY-ORDERED	X	-	-	-	-	-	X	-	-	-	-	-
BACKORDER & UPDATE REQUEST	-	-	-	-	-	-	-	X	-	-	-	-
ISSUE TO ZERO IF NECESSARY	-	X	X	-	-	-	-	-	-	-	-	-
GET NEXT REQUEST	X	-	X	-	-	-	X	X	-	-	-	-
GO TO TABLE	A	A	A	A	A	A	A	A	A	B	C	Error

Table 11B

SUBSTITUTION AND ALLOCATION PROCEDURE FOR HANDLING REQUESTS

	Rule 1	Rule 2	Rule 3
PRIORITY < 3	Y	Y	N
REMAINDER UNFILLED?	N	Y	I
ISSUE AS MUCH AS POSSIBLE FROM > 1 ITEM	X	-	-
GET NEXT REQUEST	-	X	X
GO TO TABLE A	X	X	X

Table 11C

SUBSTITUTION AND ALLOCATION PROCEDURE FOR HANDLING REQUESTS

	Rule									
	1	2	3	4	5	6	7	8	9	ELS
APPROPRIATE SUBSTITUTE?	Y	N	I	Y	N	I	Y	N	I	-
INTERCHANGEABLE OR REQUESTED ITEM?	N	N	N	Y	Y	Y	Y	Y	Y	-
LAST ITEM?	I	I	I	N	N	N	Y	Y	Y	-
LAST SUBSTITUTE?	N	N	Y	I	I	I	I	I	I	-
INTERCHANGEABLE?	I	I	I	N	N	Y	I	I	I	-
PRIORITY = 5	I	I	I	I	I	I	N	N	Y	-
BACKORDER & UPDATE REQUEST	-	-	-	-	-	-	-	-	X	-
GET NEXT REQUEST	-	-	-	-	-	-	-	-	X	-
GO TO TABLE	A	B	B	A	B	A	A	B	A	Error

To further compare the two, the decisions in a flow chart must be tested in the order in which they appear; in a decision table, except for the ELSE-Decision-Rule, the decisions can be tested in any order. This enables programmers to consider the relative frequency

with which transactions satisfy decision rules and should lead to more efficient computer programs. And finally, when policy changes occur, it is easier to correct each of the affected decision rules than it is to correct a series of interconnected flow-chart boxes.

SUMMARY OF PROCEDURE FOR CHECKING DECISION TABLES FOR COMPLETENESS, REDUNDANCY AND CONTRADICTIONS

1. Be sure that each decision rule contains at least one action. Where no action is specified, either specify the action(s) or remove the decision rule.

2. Be sure every pair of AND-FUNCTIONS is independent by verifying that in at least one position of each pair there exists a Y in one function and an N in the other. For each pair that is dependent, refer to Table 10 to determine where redundancy or contradiction exists and take appropriate steps to correct this deficiency.

3. After all contradictions and redundancies have been deleted and all decision rules have at least one action, check to see if the decision table contains an ELSE-Decision-Rule. If it does, the table is complete and contains no contradictions or redundancies.

4. If the decision table contains no ELSE-Decision-Rule, count the I's in each decision rule. Suppose Rules 1, 2, ---, m contain r_1, r_2, ---, r_m. Then the decision table contains

$$N = 2^{r_1} + 2^{r_2} + \cdots + 2^{r_{m-1}} + 2^{r_m}$$

independent decision rules. If the table contains n independent conditions, N should equal 2^n. If N is greater than 2^n, all redundant or contradictory rules have not been eliminated, and the analyst should go back to step 2.

If N is less than 2^n, the table has not specified all possible allowable decision rules. This can be corrected by either

a. inserting an ELSE-Decision-Rule, or

b. checking which of the 2^n independent decision rules are missing from the table and inserting them (see page 15 for method of finding missing AND-FUNCTIONS, and hence, missing decision rules).

IV. THE USE OF OR-FUNCTIONS

Many data processing areas, such as editing and information retrieval, need decision rules that specify a series of actions if any one of a number of conditions is satisfied. While this can be handled by decision rules containing AND-FUNCTIONS, it generally requires a great deal of writing that could be eliminated if it were possible to connect condition requirements with an inclusive "OR" operator, instead of the "AND" operator used in AND-FUNCTIONS.

STRUCTURE FOR OR-FUNCTIONS

We can illustrate the structure and features of OR-FUNCTIONS by using an input editing example. If a particular field is not numeric and/or has more than 7 characters, and/or a decimal point is missing, then it is invalid and an error procedure must be executed; otherwise continue processing. Table 12 depicts these decisions in DETAB-X form. The decisions described in Table 12 are now shown in Table 13 with the OR-FUNCTION to the right of the rightmost double vertical line and above the double horizontal line. A double vertical line separates the decision rules that have AND-FUNCTIONS from those that have OR-FUNCTIONS, i.e., those that have conditions connected by inclusive "OR" operators.

Where I in the AND-FUNCTIONS says "immaterial" (having the same effect as Y or N), ϕ in the OR-FUNCTIONS says "immaterial" (here the effect is to ignore the conditions). For example, in Table 14, Rule 3 says that if field is not numeric or the number of characters is greater than 7, go to error procedure. This decision rule is not concerned with whether or not the 5th character is a decimal point.

Table 12

INPUT EDIT

	Rule 1	Rule 2	Rule 3	Rule 4
FIELD IS NUMERIC	N	Y	Y	Y
NO. of CHARACTERS > 7	I	Y	N	N
5th CHARACTER = DECIMAL PT.	I	I	N	Y
GO TO ERROR PROCEDURE	X	X	X	-
CONTINUE PROCESSING	-	-	-	X

Table 13

INPUT EDIT

	Rule 1	Rule 2
FIELD IS NUMERIC	Y	N
NO. OF CHARACTERS > 7	N	Y
5th CHARACTER = DECIMAL PT.	Y	N
GO TO ERROR PROCEDURE	-	X
CONTINUE PROCESSING	X	-

The ELSE-Decision-Rule still appears as the rightmost rule, and represents the remainder of the 2^n possible independent rules that have not been specified or implied in the decision table (see Table 15).

ANALYSIS OF DECISION RULES CONTAINING AND-FUNCTIONS and OR-FUNCTIONS

Decision table Requirements 1 and 2 specified earlier for AND-FUNCTIONS continue to apply to OR-FUNCTIONS. Also, the definition given for dependence and independence of AND-FUNCTIONS applies equally well to OR-FUNCTIONS. However, the criteria for determining the

dependence of a pair of OR-FUNCTIONS or the dependence of an AND-FUNCTION and an OR-FUNCTION are different from those previously described. Also, the rules for counting OR-FUNCTIONS differ. Because the completeness of decision tables in terms of AND-FUNCTIONS was described earlier, the following sections are confined to explaining the equivalence relation between OR-FUNCTIONS and AND-FUNCTIONS.

Table 14

INPUT EDIT

	Rule 1	Rule 2	Rule 3
FIELD IS NUMERIC	Y	Y	N
NO. OF CHARACTERS > 7	N	N	Y
5th CHARACTER = DECIMAL PT.	Y	N	ϕ
GO TO ERROR PROCEDURE	-	-	X
INSERT DECIMAL POINT	-	X	-
CONTINUE PROCESSING	X	X	-

Theorem VI' states that an OR-FUNCTION that contains ϕ in r positions is equivalent to $(2^n - 2^r)$ pure AND-FUNCTIONS; where $0 \leq r < n$. In Table 13, for example, the OR-FUNCTION of Rule 2 has no ϕ. Therefore, r = 0, and OR-FUNCTION 2 is equivalent to $(2^n - 2^0) = (2^n - 1)$ pure AND-FUNCTIONS. In another example, Table 14, the OR-FUNCTION of Rule 3 has one ϕ. Therefore r = 1, and OR-FUNCTION 3 is equivalent to $(2^n - 2^1) = (2^n - 2)$ pure AND-FUNCTIONS.

In order to make checks on completeness, redundancy, and contradictions described earlier for decision tables containing AND-FUNCTIONS, it is necessary to state when OR-FUNCTIONS are dependent, and when an OR-FUNCTION and an AND-FUNCTION are dependent.

Theorem I' states that two OR-FUNCTIONS are dependent if, in at least one position, both contain a Y, or both contain an N. Otherwise, they are independent.

Theorem I'' states that an AND-FUNCTION and an OR-FUNCTION are dependent if, in at least one position, there exists a Y in both, or an N in both, or there exists an I in the AND-FUNCTION and a Y or N in the OR-FUNCTION. Otherwise they are independent. For example, in Table 14, on lines 1 and 2, AND-FUNCTION 1 and OR-FUNCTION 3 do not both contain a Y; nor do they both contain an N; also there is no I in AND-FUNCTION 1. Therefore, AND-FUNCTION 1 and OR-FUNCTION 3 are independent. Similarly, AND-FUNCTION 2 and OR-FUNCTION 3 are independent.

Decision tables containing AND- and OR-FUNCTIONS can now be tested according to the procedures specified in Sec. III of this paper. Consider Table 15. Requirement 1 is satisfied; every decision rule has at least one action. In checking all pairs of AND-FUNCTIONS and OR-FUNCTIONS for dependence, we note that AND-FUNCTION 1 and OR-FUNCTION 5 are dependent because for the condition "Property-Class > 'J'", AND-FUNCTION 1 has an I and OR-FUNCTION 5 has a Y. Also, AND-FUNCTION 2 and OR-FUNCTION 5 are dependent because for the condition "Property-Class < 'A'", the former has an I, and the latter has a Y. Table 16 corrects this dependence and shows all the AND-FUNCTIONS and OR-FUNCTIONS as independent.

Counting the number of AND-FUNCTIONS specified or implied in Table 16 we derive the number of decision rules contained therein:

Table 15

DEPRECIATION EXPENSE OR LEASE EXPENSE

	Rule 1	Rule 2	Rule 3	Rule 4	Rule 5	ELS
ASSET-LEASED	Y	Y	N	N	ϕ	-
ASSET-GOVT-COST-FREE	Y	N	I	I	ϕ	-
PROPERTY-CLASS < "A"	N	I	N	N	Y	-
PROPERTY-CLASS > "J"	I	N	N	N	Y	-
ASSET-NEW-WHEN-PURCHASED	I	I	Y	N	ϕ	-
WRITE LOCATION-RECORD	X	-	-	-	-	-
DO	Table 4	Table 5	Table 6	Table 7	Table 8	Table 9
ADD CURRENT-DATE TO EXPENSE-TO-DATE	-	X	X	X	-	-
PRINT ERROR	-	-	-	-	X	-
NOTIFY MANAGER	-	-	-	-	-	X

NOTE: Table 5 computes current-lease amount.
Table 6 computes sum-of-digits-expense.
Table 7 computes straight-line-depreciation-expense.

Table 16

DEPRECIATION EXPENSE OR LEASE EXPENSE

	Rule 1	Rule 2	Rule 3	Rule 4	Rule 5	ELS
ASSET-LEASED	Y	Y	N	N	ϕ	-
ASSET-GOVT-COST-FREE	Y	N	I	I	ϕ	-
PROPERTY-CLASS < "A"	N	N	N	N	Y	-
PROPERTY-CLASS > "J"	N	N	N	N	Y	-
ASSET-NEW-WHEN-PURCHASED	I	I	Y	N	ϕ	-
WRITE LOCATION-RECORD	X	-	-	-	-	-
DO	Table 4	Table 5	Table 6	Table 7	Table 8	Table 9
ADD CURRENT-DATE TO EXPENSE-TO-DATE	-	X	X	X	-	-
PRINT ERROR	-	-	-	-	X	-
NOTIFY MANAGER	-	-	-	-	-	X

NOTE: Table 5 computes current-lease amount.
Table 6 computes sum-of-digits-expense.
Table 7 computes straight-line-depreciation-expense.

AND-FUNCTION 1 implies 2^1 = 2

AND-FUNCTION 2 implies 2^1 = 2

AND-FUNCTION 3 implies 2^1 = 2

AND-FUNCTION 4 implies 2^1 = 2

OR-FUNCTION 5 implies (2^5-2^3) = 24

TOTAL 32 decision rules.

Since there are 5 conditions, the required number of decision rules is $2^5 = 32$. Having specified the 32 decision rules (each pair containing independent AND-FUNCTION), the decision table is then complete. Consequently, the ELSE-Decision-Rule is not necessary and Table 17 illustrates the correction.

In the next sections, we develop the theorems on dependence of AND-FUNCTIONS and OR-FUNCTIONS as well as the theorems on number of functions and decision rules contained within a decision table.

Table 17

DEPRECIATION EXPENSE OR LEASE EXPENSE

	Rule 1	Rule 2	Rule 3	Rule 4	Rule 5
ASSET-LEASED	Y	Y	N	N	ϕ
ASSET-GOVT-COST-FREE	Y	N	I	I	ϕ
PROPERTY-CLASS < "A"	N	N	N	N	Y
PROPERTY-CLASS > "J"	N	N	N	N	Y
ASSET-NEW-WHEN-PURCHASED	I	I	Y	N	ϕ
WRITE LOCATION-RECORD	X	-	-	-	-
DO	Table 4	Table 5	Table 6	Table 7	Table 8
ADD CURRENT-DATE TO EXPENSE-TO-DATE	-	X	X	X	-
PRINT ERROR	-	-	-	-	X

NOTE: Table 5 computes current-lease amount.
Table 6 computes sum-of-digits-expense.
Table 7 computes straight-line-depreciation-expense.

V. AND-FUNCTION THEOREMS

ASSUMPTIONS

Every decision table is based upon a set of conditions, each of which can be either true or false. We begin our development of decision-table theory by assuming there exist n conditions, C_1, C_2, C_3, ..., C_n each of which can be either true or false at any point in time.

Let $S = (a_1, a_2, a_3, \ldots, a_n)$ where $a_i = 0$ or 1, represent the status (true or false) of the n conditions. In the k^{th} position of S, a 1 signifies that C_k is true; while a 0 in the same position signifies that C_k is false.

For example, if I have four conditions: $C_1 = (w < 3)$, $C_2 = (x = 3)$, $C_3 = (y > 4)$, $C_4 = (z \leq 0)$ and if the condition variables have the following values: $w = 2$, $x = 5$, $y = 3$, $z = -1$, then $S = (1001)$. Note that the components of S represent the truth value of the corresponding conditions for a given set of values of the condition variables. $V(C_k)$ denotes the truth value of C_k.

Independence of Conditions

Each of the n conditions, C_1, C_2, ..., C_n, consists of two operands and a relational operator ($=$, $\leq$, $\geq$, $<$, $>$, $\neq$). One operand must be a condition variable and the other must be a condition variable or a constant.

Examples of conditions

1. $C = (x \leq 5)$
2. $C = (x > y)$

3. C = (13 = x)

4. C = (Y = 12)

Conditions C_k and C_e are dependent if both contain the same condition variables, and if these variables have at least one set of values such that $V(C_k) = 1$ and $V(C_e) = 1$. Otherwise, conditions C_k and C_e are independent.

Examples

1. If $C_k = (X < 5)$
   and $C_e = (X < 7)$,
   C_k and C_e are dependent since for any $X < 5$, $V(C_k) = 1$ and $V(C_e) = 1$.

2. If $C_k = (Y < 5)$
   and $C_e = (Y \geq 5)$,
   C_k and C_e are independent since there exists no value of Y such that $V(C_k) = 1$ and $V(C_e) = 1$.

3. If $C_k = (X \leq Y)$
   and $C_e = (X = Y)$.
   C_k and C_e are dependent since for $X = Y$, $V(C_k) = 1$ and $V(C_e) = 1$.

For the remainder of this paper, we assume that the tables are based on n independent conditions.

Condition Requirements

For each C_i, we can specify one of the following requirements:

1. Y_i that signifies C_i must be satisfied.
2. N_i that signifies C_i must not be satisfied

$$\overline{Y}_i = N_i \; ; \; Y_i = \overline{N}_i$$

where "-" signifies "not."

3. I_i that signifies either C_i must or must not be satisfied. If I_i has been specified, it is immaterial what values the condition variables of C_i assume.

$$I_i = Y_i + N_i$$

where + denotes the Boolean "INCLUSIVE OR" operator.

In addition, "$\oplus$" will be the Boolean "EXCLUSIVE OR" operator; "." will be the Boolean "AND" operator.

AND-FUNCTIONS BASED ON $C_1, C_2, \ldots, C_n$

Let W_i be a variable that represents Y_i, N_i, or I_i. We now define an AND-FUNCTION

$$(1) \qquad B_j = W_{1j} \cdot W_{2j} \cdot W_{3j} \cdot \text{---} \cdot W_{n-1,j} \cdot W_{nj}$$

Since each W_{ij}, $i = 1, 2, \text{---}, n$, represents one of three requirements, $j = 3^n$. We denote the set of 3^n AND-FUNCTIONS as a table.

$$(2) \qquad T = \begin{bmatrix} B_1 \\ B_2 \\ \vdots \\ B_{3^n} \end{bmatrix}$$

Let $V(B_j)$ denote the truth value of B_j. Then $V(B_j) = 1$ or 0, depending on whether or not all of the n requirements of B_j are met. An AND-FUNCTION is satisfied if its truth value is 1. To illustrate, suppose

$$B_4 = Y_1 \cdot Y_2 \cdot N_3 \cdot Y_4 \cdot N_5$$

and S = (1 1 0 1 0) for a given set of values of the condition variables. Then $V(B_4) = 1$; S = (1 1 0 1 0) satisfies B_4. If, however,

S = (0 1 0 1 0) for another set of values of the condition variables, then $V(B_4) = 0$.

DEPENDENCY OF AND-FUNCTIONS IN A TABLE

Definition

The AND-FUNCTIONS B_k and B_t are dependent if for at least one set of values of the condition variables, both $V(B_k) = 1$ and $V(B_t) = 1$. Otherwise, B_k and B_t are independent, i.e., there exists no set of values of the condition variables such that both $V(B_k) = 1$ and $V(B_t) = 1$.

Example of Two Dependent AND-FUNCTIONS

Suppose $B_5 = Y_1 \cdot N_2 \cdot Y_3 \cdot I_4$ (Note: $I_4 = Y_4 + N_4$),

and $B_8 = Y_1 \cdot N_2 \cdot Y_3 \cdot N_4$.

Then for that set of values of the condition variables that yields S = (1 0 1 0), both $V(B_5) = 1$ and $V(B_8) = 1$. Then, B_5 and B_8 are dependent.

Example of Two Independent AND-FUNCTIONS

Suppose $B_3 = Y_1 \cdot N_2 \cdot Y_3 \cdot Y_4 \cdot N_5$,

and $B_7 = N_1 \cdot N_2 \cdot Y_3 \cdot Y_4 \cdot N_5$.

The only set of values of the condition variables for which $V(B_3) = 1$, is the one that yields S = (1 0 1 1 0). For this set of values, $V(B_7) = 0$. Therefore there exists no set of values of the condition variables such that both $V(B_3) = 1$ and $V(B_7) = 1$. Hence B_3 and B_7 are independent.

Theorem I. Within a Table, two AND-FUNCTIONS are independent if, in at least one position, one function contains Y and the other contains N. Otherwise, they are dependent.

Proof of Theorem I. Let B_r and B_s be two AND-FUNCTIONS in T, where B_r contains a Y in at least one position, say the k^{th} position, and B_s contains an N in that same position, i.e.,

$$B_r = W_{1r} \cdot W_{2r} \cdot \text{---} \cdot Y_k \cdot \text{---} \cdot W_{n-1,r} \cdot W_{nr},$$

$$B_s = W_{1s} \cdot W_{2s} \cdot \text{---} \cdot N_k \cdot \text{---} \cdot W_{n-1,s} \cdot W_{ns}.$$

The only possible sets of values of the condition variables that can enable $V(B_r)$ to equal 1 are those that result in S's that have a 1 in the k^{th} position. With every one of those sets of values, $V(B_s) = 0$. $\therefore$ B_r and B_s are independent. This proves the first part of the theorem.

To prove the second part, suppose B_p and B_q are two AND-FUNCTIONS in T, and in every position of each there does not exist a Y in one, and an N in the other. We will show that B_p and B_q are dependent. Suppose Y exists in the first d positions, N in the next e positions, I in the next (n-d-e) positions of B_p, i.e.,

$$B_p = Y_1 \cdot Y_2 \cdot \text{---} \cdot Y_d \cdot N_{d+1} \cdot \text{---} \cdot N_{d+e} \cdot I_{d+e+1} \cdot \text{---} \cdot I_n$$

then $$B_q = I_1 \cdot I_2 \cdot \text{---} \cdot I_d \cdot I_{d+1} \cdot \text{---} \cdot I_{d+e} \cdot W_{d+e+1} \cdot \text{---} \cdot W_n$$

where W = Y, N, or I

$$0 \leq d \leq n \qquad 0 \leq e \leq n$$

for $S = (1 \underbrace{1 \text{ ----- } 1}_{d} \; \underbrace{0 \; 0 \text{ ----- } 0}_{e} \; A_{d+e+1} \; A_{d+e+2} \text{ --- } A_n)$

Where $A_j = 1$ if $W_j = Y$

and $A_j = 0$ if $W_j = N$, or I

$j = d+e+1, d+e+2, --, n$

both $V(B_p) = 1$ and $V(B_q) = 1$ for all possible values of d and e. Hence, B_p and B_q are dependent. We can apply the same logic where Y appears in any d positions, N appears in any e positions, and I appears in the remaining (n-d-e) positions.

Definitions

A pure AND-FUNCTION is one that has exactly n terms, each of which is either a Y or an N. In the remainder of this paper, P will signify a pure AND-FUNCTION. For example, for n = 5,

$P = Y_1 \cdot N_2 \cdot N_3 \cdot Y_4 \cdot Y_5$ is a pure AND-FUNCTION.

A mixed AND-FUNCTION is one that is not pure. It will be signified by M in the remainder of this paper. A mixed AND-FUNCTION either 1) contains one or more I's, for example

$M = N_1 \cdot Y_2 \cdot Y_3 \cdot I_4 \cdot N_5 \cdot N_6$;

or 2) is expressed as a combination of pure AND-FUNCTIONS, all connected by "EXCLUSIVE OR" operators, for example

$M = P_1 \oplus P_2 \oplus P_3$.*

Theorem II. Within a Table T, each pure AND-FUNCTION is independent of every other pure AND-FUNCTION.

Proof of Theorem II. Let Z_{im} be a variable that represents Y_i or N_i. Let

1. $P_m = Z_{1m} \cdot Z_{2m} \cdot Z_{3m} \cdot \text{------} \cdot Z_{n-1,m} \cdot Z_{nm}$.

*We will show later how a mixed AND-FUNCTION containing one or more I's can be expanded into a mixed AND-FUNCTION containing a combination of pure AND-FUNCTIONS.

Since Z represents one of two requirements, $m = 2^n$, and there exists a sub-table of T

2. $$E = \begin{bmatrix} P_1 \\ P_2 \\ \vdots \\ P_{2^n} \end{bmatrix}$$

where P_1, P_2, ---, P_{2^n} are the only pure AND-FUNCTIONS of T.

3. Consider all the pairs (P_r, P_s) of Table E

   $r = 1, 2, ---, 2^n$; $s = 1, 2, ---, 2^n$; $r \neq s$

4. P_r and P_s differ from each other in at least one position, say the k^{th} position.

5. Either P_r contains Y_k, and P_s contains N_k or, P_r contains N_k, and P_s contains Y_k.

6. Then P_r and P_s are independent (by Theorem I).

7. This is true for all pairs of AND-FUNCTIONS of E, and Theorem II is proved.

Corollary to Theorem II. Within a Table T, there exists exactly 2^n pure AND-FUNCTIONS. Each of the remaining $(3^n - 2^n)$ AND-FUNCTIONS is mixed.

Theorem III. The form of a mixed AND-FUNCTION that contains I in r positions $(1 \leq r < n)$ can be expanded into a canonical form that consists of 2^r pure AND-FUNCTIONS each connected by an exclusive "OR" operator ($\oplus$).

Proof of Theorem III. Let Z_{mt} be a variable that represents Y_m or N_m ;

$$m = r + 1,\ r + 2,\ ---,\ n.$$

Let

1. $M_t = I_1 \cdot I_2 \cdot ---- \cdot I_{r-1} \cdot I_r \cdot Z_{r+1,t} \cdot Z_{r+2,t} \cdot ----- \cdot Z_{nt}$

2. $I = Y + N.$

3. $M_t = (Y_1 + N_1) \cdot (Y_2 + N_2) \cdot ---- \cdot (Y_{r-1} + N_{r-1}) \cdot (Y_r + N_r) \cdot Z_{r+1,t} \cdot ---- \cdot Z_{nt}$

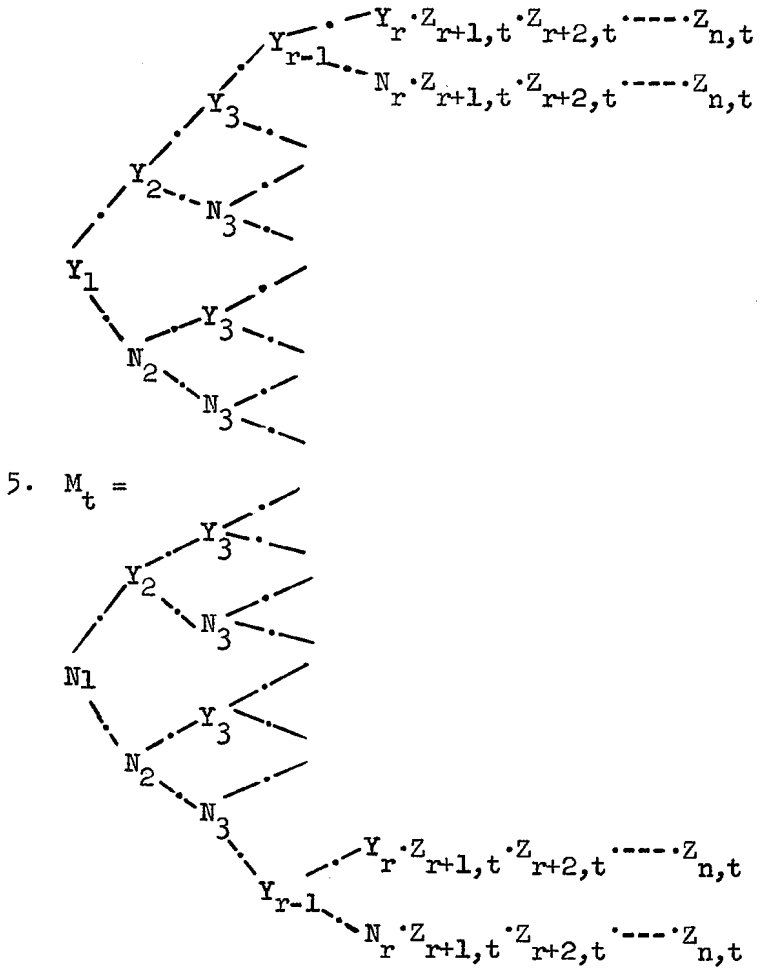
An accepted theorem in Boolean logic is

$$(R + S) \cdot T = R \cdot T + S \cdot T.$$

Applying this theorem to the 1st term of 3,

4. $M_t = \left[Y_1 \cdot (Y_2 + N_2) \cdot ---- \cdot (Y_r + N_r) \cdot Z_{r+1,t} \cdot ---- \cdot Z_{nt} \right] + \left[N_1 \cdot (Y_2 + N_2) \cdot ---- \cdot (Y_r + N_r) \cdot Z_{r+1,t} \cdot ---- \cdot Z_{nt} \right] .$

Repeating this expansion $(r - 1)$ times, we get a tree effect where an "AND" operator connects each element in a line to the next and an inclusive "OR" operator connects each line across to the next.


$Y_r \cdot Z_{r+1,t} \cdot Z_{r+2,t} \cdots Z_{n,t}$
Y_{r-1}
$N_r \cdot Z_{r+1,t} \cdot Z_{r+2,t} \cdots Z_{n,t}$
Y_3
Y_2
N_3
Y_1
Y_3
N_2
N_3

5. M_t =

Y_3
Y_2
N_3
N_1
Y_3
N_2
N_3
$Y_r \cdot Z_{r+1,t} \cdot Z_{r+2,t} \cdots Z_{n,t}$
Y_{r-1}
$N_r \cdot Z_{r+1,t} \cdot Z_{r+2,t} \cdots Z_{n,t}$

6. Since Y_i and N_i appear in r positions, M_t consists of 2^r pure AND-FUNCTIONS connected by "+".

   $M_t = P_1 + P_2 + P_3 + \text{---} + P_{2^r}$

   where P_1, P_2, ---, P_{2^r} are all contained in T.

7. P_1, P_2, ---, P_{2^r} are each independent of each other. (by Theorem II).

8. For every pair of (P_p, P_q) there exists no set of values of the condition variables, such that both $V(P_p) = 1$ and $V(P_q) = 1$.

Hence,

9. $M_t = P_1 \oplus P_2 \oplus \text{---} \oplus P_{2^r}$.

10. We showed this theorem to be true when I appears in the first r positions. The logic used in this proof can be applied for I appearing in any r positions.

11. The theorem is therefore proved.

Corollary 1 of Theorem III. The canonical form of a mixed AND-FUNCTION contains an even number of pure AND-FUNCTIONS.

Corollary 2 of Theorem III. The canonical form of a mixed AND-FUNCTION contains at least two pure AND-FUNCTIONS.

Theorem IV. Within a Table T, every mixed AND-FUNCTION that contains I in r positions ($1 \leq r < n$) is dependent upon each of 2^r pure AND-FUNCTIONS of T.

Proof of Theorem IV. Let M be a mixed AND-FUNCTION that contains I in r positions; $1 \leq r \leq n$. Then, we can expand M to its canonical form.

1. $M = P_1 \oplus P_2 \oplus P_3 \oplus \text{---} \oplus P_{2^r}$ where P_1, P_2, ---, P_{2^r} are all

contained in T. (by Theorem III).

2. When $V(P_1) = 1$, $V(M) = 1$, $\therefore$ M and P_1 are dependent.

   When $V(P_2) = 1$, $V(M) = 1$, $\therefore$ M and P_2 are dependent.

   . . .

   When $V(P_2r) = 1$, $V(M) = 1$, $\therefore$ M and P_2r are dependent.

3. M is dependent upon each of 2^r pure AND-FUNCTIONS of T.

Corollary 1 of Theorem IV. Two mixed AND-FUNCTIONS are dependent if their canonical forms each contain one or more pure AND-FUNCTIONS that are common to both. For example, if

$$M_1 = P_2 \oplus P_3 \oplus P_7 \oplus P_8 \oplus P_9 \oplus P_{12}, \text{ and}$$

$$M_3 = P_1 \oplus P_2 \oplus P_5 \oplus P_6 \oplus P_{11} \oplus P_{13},$$

then M_1 and M_3 are dependent since P_2 is common to both.

Corollary 2 of Theorem IV. A mixed AND-FUNCTION is dependent on each of the pure AND-FUNCTIONS contained in its canonical form.

Corollary 3 of Theorem IV. If a pure AND-FUNCTION and a mixed AND-FUNCTION are dependent, the canonical form of the mixed function contains the pure function.

Corollary 4 of Theorem IV. If two mixed AND-FUNCTIONS are dependent, there exists at least one pure AND-FUNCTION in their canonical forms that is common to both.

Theorem V. Table T, based on n conditions, contains one, and only one, set of 2^n independent AND-FUNCTIONS.

Proof of Theorem V.

1. Within a Table T, there exists exactly 2^n pure AND-FUNCTIONS (by corollary to Theorem II).

2. The 2^n pure AND-FUNCTIONS are independent of each other (by Theorem II). Thus, there exists one set of 2^n independent AND-FUNCTIONS.
3. Now, to show that this is the only set of 2^n independent AND-FUNCTIONS, we consider all the remaining sets of 2^n AND-FUNCTIONS that can be formed from the 3^n AND-FUNCTIONS in T. Denote their domain as R.
4. In each of these sets of R, let there be t mixed AND-FUNCTIONS and $(2^n - t)$ pure AND-FUNCTIONS; $1 \leq t \leq 2^n$.*
5. Those sets of R containing one or more pairs of mixed AND-FUNCTIONS that contain one or more pure AND-FUNCTIONS common to both, cannot contain 2^n independent AND-FUNCTIONS (by Corollary 1 to Theorem IV).
6. We therefore look at the Domain Q of R that comprises all those sets of R that contain no pairs of mixed AND-FUNCTIONS whose canonical forms contain pure AND-FUNCTIONS common to the pair.
7. In each of the sets of Q, the canonical form of the t mixed functions contains at least 2^t distinct pure functions (by Corollary 2 of Theorem III).
8. In each set of Q, there exists at least $2^n - t + 2t = 2^n - t$ pure AND-FUNCTIONS, where $t \geq 1$.
9. Since there are exactly 2^n pure AND-FUNCTIONS in T, at least one of them is repeated within each set of Q.

---

*1) For each t, we form all the possible sets of 2^n AND-FUNCTIONS.
2) The set of 2^n pure AND-FUNCTIONS is not in R, since $t \neq 0$.

10. If $t = 2^n$, there exists at least one pure AND-FUNCTION common to one pair of the 2^n mixed AND-FUNCTIONS in every set of Q. This violates 6 above.
11. Q then contains all sets of t mixed and $(2^n - t)$ pure AND-FUNCTIONS where $1 \leq t < 2^n$, i.e., there exists at least one pure AND-FUNCTION in every set of Q.
12. In every set of Q, the pair of identical pure AND-FUNCTIONS (see 9 above) exists as one of the functions and as part of a mixed function.
13. Hence, in every set of Q, there exists a pair of dependent AND-FUNCTIONS (by Corollary 2 of Theorem IV).
14. $\therefore$ There exists no set in Q that contains 2^n independent AND-FUNCTIONS.

DECISION TABLES

Notation and Definitions

Let us now turn our attention to decision tables. We assume that each table is based upon a number of independent conditions. The actual number and kinds of conditions vary from table to table. We now consider the general case of a decision table based on n independent conditions, C_1, C_2, C_3, ----, C_n, each of which can be true or false. All notation previously described applies here.

Let D_j denote Decision Rule j, a_j denote one action in D_j, and A_j denote the entire series of actions in D_j. Then

$$A_j = a_{1j} \odot a_{2j} \odot a_{3j} \odot \text{----} \odot a_{tj} ;$$

again $j = 3^n$. The symbol "$\odot$" signifies the Boolean "AND," that

specifies actions a_{1j}, a_{2j}, ---, a_{tj} must be executed serially. $D_j = B_j \rightarrow A_j$ says, "If $V(B_j) = 1$, execute actions A_j."

A decision table is a structure for describing the expression

$$DT = D_1 \oplus D_2 \oplus \text{---} \oplus D_{q-1} \oplus D_q \; ; \; q \leq 3^n.$$

Hereafter, the words "decision table" will refer to both the structure and the expression, unless otherwise stated.

Definitions

A decision rule is simple if it contains only one pure AND-FUNCTION. For example, if

$$P_1 = Y_1 + N_2 + N_3 + Y_4 + N_5,$$

then $D_1 = P_1 \rightarrow A_1$ is a simple decision rule. A decision rule is complex if it contains a mixed AND-FUNCTION in non-canonical form. For example, if

$$M_1 = I_1 + N_2 + Y_3 + N_4 + Y_5,$$

then $D_1 = M_1 \rightarrow A_1$ is a complex decision rule.

Implicit Decision Rules

Decision rules can be implied in a decision table by one of two types of decision rules, either complex decision rules or ELSE-decision-Rules.

Complex Decision Rules

Theorem VI. A complex decision rule that contains I in r positions of its AND-FUNCTION is equivalent to 2^r simple decision rules.

Proof of Theorem VI.

1. Let $D_k = M_k \rightarrow A_k$ where M_k contains I in r of its positions.

2. $M_k = P_1 \oplus P_2 \oplus \ldots \oplus P_2 r$ (by Theorem III)

3. $D_k = (P_1 \oplus P_2 \oplus \text{---} \oplus P_2 r) \rightarrow A_k$.

4. $D_k = (P_1 \rightarrow A_k) \oplus (P_2 \rightarrow A_k) \oplus \text{---} \oplus (P_2 r \rightarrow A_k)$.

5. Hence D_k is equivalent to 2^r simple decision rules.

A complex rule that contains I in r positions of its AND-FUNCTION implies 2^r simple decision rules.

The ELSE-Decision-Rule (D_L)

$$D_L = \text{ELSE} \rightarrow A_L$$

If a decision table contains s simple rules and c complex rules, actions A_L are executed when a particular set of values of the condition variables yields a truth value 0 for each of the s simple rules and the c complex rules.

One reason for using the ELSE-Decision-Rule (ELS) is to avoid having to write a set of rules that each contain the same series of actions. For example, suppose there are two conditions C_1 and C_2 such that if both are satisfied we want to execute actions A_1, otherwise we want to execute actions A_2. Without the ELSE-Decision-Rule, it would be written as follows:

	D_1	D_2	D_3	D_4
C_1	Y	Y	N	N
C_2	Y	N	Y	N
	A_1	A_2	A_2	A_2

With the ELSE-Decision-Rule, the above statements would be written:

	D_1	D_L
C_1	Y	-
C_2	Y	-
	A_1	A_2

Another possible use for the ELS is to enable the system designer to detect any omission of a condition or appropriate AND-FUNCTION. For example, suppose for 0-40 hours, an employee gets regular time; 40.1-50 hrs, $1\frac{1}{4}$ times the hrs. worked; 50.1-60 hrs., $1\frac{1}{2}$ times the hrs. worked Although it is inconceivable that any one in this plant will work more than 60 hours, the system designer wants to be notified if it happens. He therefore sets up the following:

Table 18

EMPLOYEE SALARY-HOURS

	D_1	D_2	D_3	D_L
Hrs-Worked $\geq$ 0	Y	Y	Y	-
Hrs-Worked $\leq$ 40	Y	N	N	-
Hrs-Worked $\leq$ 50	I	Y	N	-
Hrs-Worked $\leq$ 60	I	I	Y	-
Overtime-Hrs =	0	Hrs-Worked - 40	Hrs-Worked - 40	0
Salary-Hrs =	Hrs-Worked	40 + 1.25 * OT-Hrs	40 + 1.5 * OT-Hrs	0
Print Error	-	-	-	X

Thus, when a time card containing 65 hours-worked goes through this decision table, D_1, D_2, and D_3 will each have a truth value 0, and D_L will cause "ERROR" to be printed. The system designer will then learn that he has not taken care of the condition "HOURS-WORKED > 60" in this decision table.

DECISION TABLE REQUIREMENTS FOR DETAB-X

For DETAB-X, we use a special set of decision tables that meet the following requirements:

1. Every decision rule must have at least one action.
2. For any given set of values of the condition variables of a decision table, the AND-FUNCTION of one, and only one, decision rule in that table must assume the truth value 1.

The ELSE-Decision-Rule

A consequence of Requirement 2 is that the ELSE-Decision-Rule must appear in those decision tables that do not contain 2^n independent pure AND-FUNCTIONS. These particular AND-FUNCTIONS can appear as part of a simple decision rule or be implied by the AND-FUNCTION of a complex decision rule. For example,

	D_1	D_2	D_3	D_4
C_1	Y	N	Y	Y
C_2	N	I	I	Y
C_3	Y	I	N	Y
	A_1	A_2	A_3	A_4

Rules 1 and 4 each contain one pure AND-FUNCTION; Rule 2 implies 2^2 = 4 pure AND-FUNCTIONS; Rule 3 implies 2^1 = 2 pure AND-FUNCTIONS for a total of 8 pure AND-FUNCTIONS.

These 8 are also independent since each of the four rules are independent (each pair of rules has a Y in one rule and an N in the other in corresponding positions). This table is complete since $2^n = 2^3 = 8$. Hence, it does not require an ELSE-Decision-Rule. An example requiring an ELSE-Decision-Rule follows:

ELSE-Decision-Rule

	D_1	D_2	D_L
C_1	Y	N	D
C_2	N	I	-
C_3	Y	I	-
	A_1	A_2	A_3

AND-FUNCTION of Rule 1 = Y·N·Y

AND-FUNCTION of Rule 2 = N·Y·Y

$\oplus$ N·Y·N

$\oplus$ N·N·Y

$\oplus$ N·N·N

Then the AND-FUNCTION of D_L = Y·Y·Y

$\oplus$ Y·Y·N

$\oplus$ Y·N·N

A consequence of Requirement 2 is that the AND-FUNCTION of all decision rules other than the ELS can be tested in any order. For example, a set of values of the condition variables can be applied to the AND-FUNCTION of D_5, then D_3, and then D_6, etc. Or they can be applied to D_3, then D_5, then D_6, etc. In whatever order they are tested, the same series of actions will occur if Requirements 1 and 2 are not violated.

Since the ELSE decision-rule requires each of the other rules of the decision table to have a truth value 0, we execute it only after we have tested all the other decision rules and found their truth values equal 0.

Techniques for Discovering the Dependence of AND-FUNCTIONS

A contradiction between two decision rules exists when their AND-FUNCTIONS are dependent (a violation of Requirement 2) and their series of actions are not identical. If their series of actions are identical, redundancy exists. To find these contradictions or redundancies, it is necessary to examine the AND-FUNCTIONS in pairs. We suggest here two techniques for examining (in a computer) all pairs for dependence. The logical design of each computer will determine which of these two should be used or in fact whether some new technique ought to be devised for detecting the dependencies.

These two techniques are predicated on Theorem I which states that two AND-FUNCTIONS are dependent if there exists no position of each in which one contains Y, the other contains an N.

Technique Number 1. Pair one of these AND-FUNCTIONS with each of the other specified functions, scanning each pair for a Y in one function and an N in the other in a corresponding position. If no paired Y and N is found, the functions are dependent. In that case, check the pair's sequence of actions. If they differ, have the computer reject the pair as invalid. If one pair of Y and N is found, scan the next pair. Repeat this for each set of remaining AND-FUNCTIONS.

Consider the following decision table:

	D_1	D_2	D_3	D_4
C_1	Y	N	I	Y
C_2	N	N	N	N
C_3	I	Y	Y	Y
C_4	I	N	N	Y
Add a to	b	c	b	e
Go to	Table 11	Table 15	Table 11	Table 23

Consider the following pairs (B1, B2), (B1, B3), (B1, B4). B1 and B2 are valid. They are independent; there exists a Y,N pair. B1 and B3 are dependent since there is no Y,N pair; their series of actions are identical. One or more redundant decision rules exist in the table. B1 and B4 are dependent, but their actions are different. Hence, reject Rule 1 and Rule 4.

Next, consider the pairs (B2, B3), (B2, B4). B2 and B3 are dependent since they have no Y,N pair. Their actions are different. Reject Rule 2 and Rule 3. B2 and B4 are valid. They are independent; there is a paired Y,N. Finally, consider the pair (B3, B4). They are independent; there is a paired Y,N.

Technique Number 2. Some computers may do logical arithmetic more efficiently than scanning. Hence, we map the Y, N, and I's of the AND-FUNCTIONS into binary 01, 00, and 10 respectively. The above decision table becomes:

	D_1	D_2	D_3	D_4
C_1	01	00	10	01
C_2	00	00	00	00
C_3	10	01	01	01
C_4	10	00	00	01
Add a to	b	c	b	e
Go to	Table 11	Table 15	Table 11	Table 23

Form the logical sums of all pairs of Boolean function denoted by $L(B_i, B_j)$; $i \neq j$. If $L(B_r, B_t)$ contains one or more diads equal to 01, then B_r and B_t are independent. Otherwise, they are dependent.

$L(B_1, B_2) = 01 \quad 00 \quad 11 \quad 10$

$L(B_1, B_3) = 11 \quad 00 \quad 11 \quad 10$

$L(B_1, B_4) = 00 \quad 00 \quad 11 \quad 11$

$L(B_2, B_3) = 10 \quad 00 \quad 00 \quad 00$

$L(B_2, B_4) = 01 \quad 00 \quad 00 \quad 01$

$L(B_3, B_4) = 11 \quad 00 \quad 00 \quad 01$

Since $L(B_1, B_3)$, $L(B_1, B_4)$, and $L(B_2, B_3)$ have no 01 diad, B_1 and B_3, B_1 and B_4, and B_2 and B_3 are dependent. Again, check their series of actions and reject those pairs that do not have identical series of actions.

VI. OR-FUNCTION THEOREMS

EXTENSION OF DETAB-X DECISION TABLES

For tables based on n conditions, a need often exists to execute a series of action(s) if any one of p($p \leq n$) conditions is satisfied. While this can be handled in DETAB-X decision tables, it requires a great deal of writing that could be eliminated if it were possible to connect condition requirements with an inclusive "OR" operator.

For example, in input editing if a particular field is not numeric (C_1); and/or has more than 7 characters (C_2); and/or has a decimal point missing (C_3); then the field is invalid, and an error procedure must be executed (A_2). Currently in DETAB-X, we would require 3 decision rules:

$$D_2 = C_1 \rightarrow A_2$$
$$D_3 = (\overline{C}_1 \cdot C_2) \rightarrow A_2$$
$$D_4 = (\overline{C}_1 \cdot \overline{C}_2 \cdot C_3) \rightarrow A_2.$$

The table structure would be as follows:

	D_2	D_3	D_4
C_1	Y	N	N
C_2	I	Y	N
C_3	I	I	Y
	A_2	A_2	A_2

It should prove useful to express the above as a single decision rule:

$$D_2 = (C_1 + C_2 + C_3) \rightarrow A_2$$

where we denote the expression $C_1 + C_2 + C_3$ as an OR-FUNCTION. To

implement this in the decision table structure of DETAB-X, it becomes necessary to separate the decision rules that contain AND-FUNCTIONS (those whose condition requirements are separated by the "AND" operator) from the decision rules that contain OR-FUNCTIONS (those connected by the inclusive "OR" operator). A double line accomplishes this separation. For instance, the following decision table:

	D_1	D_2	D_3	D_4
C_1	N	Y	N	N
C_2	N	I	Y	N
C_3	N	I	I	Y
	A_1	A_2	A_2	A_2

can be expressed:

	D_1	D_2
C_1	N	Y
C_2	N	Y
C_3	N	Y
	A_1	A_2

In the latter, Decision Rule 1 says

if condition 1 is not satisfied

and condition 2 is not satisfied

and condition 3 is not satisfied,

then execute actions A_1.

Decision Rule 2 says

if condition 1 is satisfied

or* condition 2 is satisfied

or condition 3 is satisfied,

then execute actions A_2.

CONVERSION OF AN OR-FUNCTION TO AN AND-FUNCTION

Define $\phi_i = \overline{I_i} = (\overline{Y_i + N_i}) = (N_i \cdot Y_i)$. We denote ϕ as the null requirement. Denote the AND-FUNCTION as

$$B_j = W_{1j} \cdot W_{2j} \cdot \text{----} \cdot W_{n-1,j} \cdot W_{nj};$$

where W_i represents Y_i, N_i, or I_i; $j = 3^n$ and the OR-FUNCTION as

$$E_j = U_{1j} + \text{---} + U_{n-1,j} + U_{nj};$$

where U_i represents Y_i, N_i, or ϕ_i, $j = 3^n$. Since all our theorems have been postulated for AND-FUNCTIONS, we now explore the relations between AND-FUNCTIONS and OR-FUNCTIONS.

An OR-FUNCTION can be converted to an AND-FUNCTION with the following procedure, on the basis of the definition of the inclusive "OR" operator.

1. Suppose

   $E_j = U_{1j} + U_{2j} + \text{---} + U_{n-1,j} + U_{nj}$; U_i represents Y_i, N_i, or ϕ_i

2. If $U_{1j} \neq \phi$, $B_1 = U_{1j} \cdot I_2 \cdot I_3 \cdot \text{---} \cdot I_n$

   If $U_{1j} = \phi$, $B_1 = 0$

3. If $U_{2j} \neq \phi$, $B_2 = \overline{U_{1j}} \cdot U_{2j} \cdot I_3 \cdot I_4 \cdot \text{---} \cdot I_n$

   If $U_{2j} = \phi$, $B_2 = 0$.

---

*The "OR" is inclusive.

4. If $U_{3j} \neq \phi$, $B_3 = \overline{U_{1j}} \cdot \overline{U_{2j}} \cdot U_{3j} \cdot I_4 \cdot I_5 \cdot \text{---} \cdot I_n$

   If $U_{3j} = \phi$, $B_3 = 0$.

   . . . .
   . . . .
   . . . .
   . . . .

5. If $U_{nj} \neq \phi$, $B_n = \overline{U_{1j}} \cdot \overline{U_{2j}} \cdot \overline{U_{3j}} \cdot \text{---} \cdot \overline{U_{n-1,k}} \cdot U_{nj}$

   If $U_{nj} = \phi$, $B_n = 0$

   Let G denote E_j in converted form

6. Let $G = B_1 \oplus B_2 \oplus B_3 \oplus \text{------} \oplus B_{n-1} \oplus B_n$

   Where $B_i = 0$ if $U_{ij} = \phi$

   $B_i = \overline{U_{1j}} \cdot \overline{U_{2j}} \cdot \overline{U_{3j}} \cdot \text{---} \cdot \overline{U_{i-1,j}} \cdot U_{ij} \cdot I_{i+1,j} \cdot \text{---} \cdot I_n$

   if $U_{ij} \neq \phi$

Although we started with U_{1j} and continued with U_{2j} then U_{3j}, etc., to convert E to G, we could have started with any U, and taken away any of the remaining U's, etc., to convert E to G. This would have resulted in a G that looked different from the original G. They are, in fact, equivalent, i.e., they have the same truth table for all possible values of the condition variables. If each of the B's that contain I's are converted to their canonical forms, the two G's will be found to be identical. This is because both G's were derived on the basis of the definition of the inclusive "OR" operator.

For example: Let $E = Y_1 + N_2 + N_3$. If we use Y_1, then N_2, then N_3,

$$G_1 = (Y_1 \cdot I_2 \cdot I_3) \oplus (N_1 \cdot N_2 \cdot I_3) \oplus (N_1 \cdot Y_2 \cdot N_3)$$

$$
\begin{aligned}
G_1 &= Y_1 \cdot Y_2 \cdot N_3 && 1.\\
&\oplus Y_1 \cdot Y_2 \cdot Y_3 && 2.\\
&\oplus Y_1 \cdot N_2 \cdot N_3 && 3.\\
&\oplus Y_1 \cdot N_2 \cdot Y_3 && 4\\
&\oplus N_1 \cdot N_2 \cdot Y_3 && 5.\\
&\oplus N_1 \cdot N_2 \cdot N_3 && 6.\\
&\oplus N_1 \cdot Y_2 \cdot N_3 && 7.
\end{aligned}
$$

If we use N_2 then N_3 then Y_1,

$$
\begin{aligned}
G_2 &= Y_1 \cdot N_2 \cdot Y_3 && 4.\\
&\oplus N_1 \cdot N_2 \cdot Y_3 && 5.\\
&\oplus Y_1 \cdot N_2 \cdot N_3 && 3.\\
&\oplus N_1 \cdot N_2 \cdot N_3 && 6.\\
&\oplus Y_1 \cdot Y_2 \cdot N_3 && 1.\\
&\oplus N_1 \cdot Y_2 \cdot N_3 && 7.\\
&\oplus Y_1 \cdot Y_2 \cdot Y_3 && 2.
\end{aligned}
$$

Notice that G_1 and G_2 contain the same B's. Hence $G_1 = G_2$.

EXAMPLE OF CONVERSION

Let $E = Y_1 + N_2 + \phi_3 + \phi_4 + Y_5$,

then
$$G = (Y_1 \cdot I_2 \cdot I_3 \cdot I_4 \cdot I_5) \oplus (\overline{Y}_1 \cdot N_2 \cdot I_3 \cdot I_4 \cdot I_5) \oplus (\overline{Y}_1 \cdot \overline{N}_2 \cdot \overline{\phi}_3 \cdot \overline{\phi}_4 \cdot Y_5),$$

$$G = (Y_1 \cdot I_2 \cdot I_3 \cdot I_4 \cdot I_5) \oplus (N_1 \cdot N_2 \cdot I_3 \cdot I_4 \cdot I_5) \oplus (N_1 \cdot Y_2 \cdot I_3 \cdot I_4 \cdot Y_5).$$

Example 1 is equivalent to Example 2

	D_1	D_2	D_3	D_4	ELS
C_1	N	Y	N	N	-
C_2	Y	I	N	Y	-
C_3	Y	I	I	I	-
C_4	N	I	I	I	-
C_5	N	I	I	Y	-
	A_1	A_2	A_2	A_2	A_3

Example 1

	D_1	D_2	ELS
C_1	N	Y	-
C_2	Y	N	-
C_3	Y	ϕ	-
C_4	N	ϕ	-
C_5	N	Y	-
	A_1	A_2	A_3

Example 2

CONSEQUENCES OF CONVERSION

Every OR-FUNCTION can be converted (is equivalent) to the exclusive union of one or more independent AND-FUNCTIONS. The truth table of this converted form is the same as the original function for all possible values of the condition variables.

To illustrate, consider

$$E = U_1 + U_2 + U_3 + \text{---} + U_{n-1} + U_n$$

that converts to

$$G = B_1 \oplus B_2 \oplus B_3 \oplus \text{---} \oplus B_{n-1} \oplus B_n .$$

If one or more of the requirements, i.e., either Y or N is satisfied, V(E) = 1. For the same values of the condition variables, there exists one, and only one, B for which V(B) = 1. [This is true because each B that contains more than one U (that is either Y or N) contains one original U and the remaining U terms are negations of the original U's. The non-negated U is different for each B.] Since V(B) = 1, B(G) = 1.

V(E) = 0 if, and only if, the negation of every U occurs. If the negation of every U occurs V(G) = 0, since each B contains one non-negated U (that is either Y or N) and all U (that are either Y or N) are contained in G. Further, V(G) = 0 only if V(B) = 0 for every B. This can occur only if the negation of every U occurs.

E and G, therefore, have the same truth table for all possible values of the condition variables.

Theorem VI'.* An OR-FUNCTION that contains ϕ in r($o \leq r < n$) positions is equivalent to $(2^n - 2^r)$ distinct pure AND-FUNCTIONS.

Proof of Theorem VI'.

1. Consider an OR-FUNCTION that contains ϕ in the last r positions, and Z_i (which represents Y_i or N_i) in the first (n-r) positions:

   $$E = Z_1 + Z_2 + \text{---} + Z_{n-r} + \phi_{n-r+1} + \text{---} + \phi_n.$$

2. E can be expanded to:

   $$G = (Z_1 \cdot I_2 \cdot \text{---} \cdot I_n) \oplus (\overline{Z}_1 \cdot Z_2 \cdot I_3 \cdot \text{---} \cdot I_n) \oplus \ldots$$
   $$\oplus (\overline{Z}_1 \cdot \overline{Z}_2 \cdot \text{---} \cdot \overline{Z}_{n-r-1} \cdot Z_{n-r} \cdot I_{n-r+1} \cdot \text{---} \cdot I_n);$$

3. The first term contains (n-1) I's. It is equivalent to 2^{n-1} pure AND-FUNCTIONS. The second term contains (n-2) I's. It is equivalent to 2^{n-2} pure AND-FUNCTIONS. The (n-r)th term contains r I's. It is equivalent to 2^r pure AND-FUNCTIONS.

4. G is equivalent to $2^r + 2^{r-1} + \text{---} + 2^{n-2} + 2^{n-1}$ distinct pure AND-FUNCTIONS.

---

*The reader will find Theorems VI', IV' and I' presented out of sequence so that the OR-FUNCTION discussion can parallel the AND-FUNCTION argument.

5. G is equivalent to $(2^n - 2^r)$ distinct pure AND-FUNCTIONS.

6. This same reasoning could have been used if I appeared in any r positions. The theorem is therefore proved.

An AND-FUNCTION and an OR-FUNCTION are dependent if there exists at least one set of values of the condition variables such that both assume a truth value 1. Otherwise, they are independent. For example, suppose

$$B = Y_1 \cdot N_2 \cdot N_3, \text{ and } E = N_1 + Y_2 + Y_3.$$

Then there exists no S such that V(B) = 1 and V(E) = 1 since

$$E = N_1 + Y_2 + Y_3 = \overline{(Y_1 \cdot N_2 \cdot N_3)} = \overline{B}.$$

Therefore, B and E are independent.

Theorem IV'. Every OR-FUNCTION that contains ϕ in r positions $(0 \leq r < n)$ is dependent on $(2^n - 2^r)$ distinct pure AND-FUNCTIONS.

Proof of Theorem IV'.

1. Let $E = \phi_1 + \phi_2 + \text{---} + \phi_r + U_{r+1} + \text{---} + U_n$.

2. $E = P_1 \oplus P_2 \oplus \text{---} \oplus P_{(2^n - 2^r)}$

   where $P_1, P_2, \text{---}, P_{(2^n - 2^r)}$ are distinct pure AND-FUNCTIONS.

3. When $V(P_1) = 1$, $V(E) = 1$

   When $V(P_2) = 1$, $V(E) = 1$

   . .
   . .
   . .

   When $V(P_{2^n - 2^r}) = 1$, $V(E) = 1$.

4. Hence E is dependent upon $2^n - 2^r$ distinct pure AND-FUNCTIONS.

5. Although ϕ was in the first r positions, the logic of this proof applies no matter in what r positions they appear.

6. The theorem is therefore proved.

Theorem I'. Two OR-FUNCTIONS are dependent if, in at least one position, there exists a Y in both, or an N in both. Otherwise they are independent.

Proof of Theorem I'.

1. Assume there exists a Y in a particular position of two OR-FUNCTIONS, C_1 and C_2.
2. A transaction that satisfies the condition in that position of C_1 also satisfies the condition of C_2.
3. $V(C_1) = 1$, $V(C_2) = 1$ for a particular transaction.
4. Hence C_1 and C_2 are dependent.
5. The same proof holds for an N in a particular position of two OR-FUNCTIONS. Hence C_1 and C_2 are dependent.
6. Assume there exists no position of C_1 and C_2 such that both contain a Y or both contain an N, i.e., each position either contains a Y and N, a ϕ and Y, ϕ and N, or ϕ and ϕ.
7. Any transaction that has a condition value in the position where the ϕ appears contributes nothing to make the truth value 0 or 1. Hence those positions that contain ϕ can be ignored.
8. For those cases where one OR-FUNCTION contains a Y and the other contains an N, both functions cannot have a truth value 1 for any transaction. Hence C_1 and C_2 are independent and the theorem is proved.

Theorem I''. An AND-FUNCTION and an OR-FUNCTION are dependent if, in at least one position, there exists a Y in both, or an N in both, or there exists an I in the AND-FUNCTION and a Y or N in the OR-FUNCTION. Otherwise they are independent.

Examples:

1. $B = Y_1 \cdot N_2 \cdot Y_3 \cdot I_4$

   $E = N_1 + Y_2 + N_3 + \phi_4$

   B and E are independent.

2. $B = Y_1 \cdot N_2 \cdot Y_3 \cdot N_4$

   $E = N_1 + N_2 + N_3 + N_4$

   B and E are dependent.

3. $B = Y_1 \cdot N_2 \cdot I_3 \cdot N_4$

   $E = N_1 + Y_2 + Y_3 + \phi_4$

   B and E are dependent.

Proof of Theorem I''.

1. Let $B = Y_1 \cdot W_2 \cdot W_3 \cdot \text{---} \cdot W_n$

or $B = I_1 \cdot W_2 \cdot W_3 \cdot \text{---} \cdot W_n$

where $W_i = Y_i, N_i,$ or I_i

and $E = Y_1 + U_2 + U_3 + \text{---} + U_n$

where $U_i = Y_i, N_i,$ or ϕ_i

$V(B) = 1$ for $S = (1\ a_2\ a_3);\ a_i \rightarrow [V(Wi) = 1]$

$V(E) = 1$ for $S = (1\ a_2\ a_3\ \text{---})$.

Then B and E are dependent.

2. Similarly for $B = N_1 \cdot W_2 \cdot W_3 \cdot \text{---} \cdot W_n$

or $B = I_i \cdot W_2 \cdot W_3 \cdot \text{---} \cdot W_n$

and $E = N_1 + U_2 + U_3 + \text{---} + U_n$

$V(B) = 1$ for $S = (0\ a_2\ a_3);\ a_i \rightarrow [V(W_i) = 1]$

$V(E) = 1$ for $S = (0a_2\ a_3\ \text{---})$

and B and E are dependent.

3. Regardless where the pair of Y's or the pair of N's or the I in the AND-FUNCTION and the Y or N in the OR-FUNCTION appear, an S exists with the 1 or 0 in the corresponding position such that V(B) = 1. And for that same S, V(E) = 1. This proves the first part of the theorem.
4. Suppose there exists no pair of Y's or no pair of N's or no I in the AND-FUNCTION with a Y or N in the corresponding position of the OR-FUNCTION in any position of B and E.
5. In converting E to AND-FUNCTIONS, the ϕ contributes no AND-FUNCTION so that those positions containing ϕ's can be disregarded.
6. From those positions that contain Y or N, the conversion will yield AND-FUNCTIONS each of which either
   a. Contains at least one Y in the same position in which B contains an N, or
   b. Contains at least one N in the same position in which B contains a Y.
7. Hence if V(B) = 1, V(E) = 0 since each of the AND-FUNCTIONS of the converted form have a truth value 0.
8. Therefore B and E are independent Q.E.D.

APPENDIX

SUMMARY OF THEOREMS FOR AND-FUNCTIONS AND OR-FUNCTIONS

We now list and extend previous theorems to include OR-FUNCTIONS as well as the AND-FUNCTIONS. A Table T comprises all AND-FUNCTIONS and OR-FUNCTIONS.

Theorem I. Within Table T, two AND-FUNCTIONS are independent if, in at least one position, one function contains Y and the other function contains N. Otherwise, they are dependent.

Theorem I'. Within Table T, two OR-FUNCTIONS are dependent if, in at least one position, both functions contain a Y, or both functions contain an N. Otherwise, they are independent.

Theorem I''. Within Table T, an AND-FUNCTION and an OR-FUNCTION are dependent if, in at least one position, both functions contain a Y, or both functions contain an N, or the AND-FUNCTION contains an I, and the OR-FUNCTION contains a Y or N. Otherwise, they are independent.

Theorem II. Within Table T, each pure AND-FUNCTION is independent of every other pure AND-FUNCTION.

Corollary of Theorem II. Within Table T, there exists exactly 2^n pure AND-FUNCTIONS. Each of the remaining $[2(3^n)-2^n$ functions is either a mixed AND-FUNCTION or an OR-FUNCTION.

Theorem III. The form of a mixed AND-FUNCTION that contains I in r positions ($1 \leq r < n$) can be expanded into a canonical form that consists of 2^r pure AND-FUNCTIONS each connected by an exclusive "OR" operator.

Corollary 1 of Theorem III. The canonical form of a mixed AND-FUNCTION contains an even number of pure AND-FUNCTIONS.

Corollary 2 of Theorem III. The canonical form of a mixed AND-FUNCTION contains at least two pure AND-FUNCTIONS.

Theorem III'. An OR-FUNCTION that contains ϕ in r positions ($0 \leq r < n$) can be converted to (is equivalent) to ($2^n - 2^r$) distinct pure AND-FUNCTIONS.

Theorem IV. Within Table T, every mixed AND-FUNCTION that contains I in r positions ($1 \leq r < n$) is dependent on each of 2^r pure AND-FUNCTIONS of T.

Corollary 1 of Theorem IV. Two mixed AND-FUNCTIONS are dependent on each other if their canonical forms each contain one or more pure AND-FUNCTIONS that are common to both.

Corollary 2 of Theorem IV. A mixed AND-FUNCTION is dependent on each of the pure AND-FUNCTIONS contained in its canonical form.

Corollary 3 of Theorem IV. If a pure AND-FUNCTION and a mixed AND-FUNCTION are dependent, the pure AND-FUNCTION is contained in the canonical form of the mixed AND-FUNCTIONS.

Corollary 4 of Theorem IV. If two mixed AND-FUNCTIONS are dependent there exists at least one pure AND-FUNCTION in their canonical forms that is common to both.

Theorem IV'. Within Table T, every OR-FUNCTION that contains ϕ in r positions ($0 \leq r < n$) is dependent on ($2^n - 2^r$) distinct pure AND-FUNCTIONS.

Theorem V. Table T, based on n conditions, contains one, and only one, set of 2^n independent AND-FUNCTIONS; each of these is a pure AND-FUNCTION.

Theorem VI. A complex decision rule (a rule that contains mixed AND-FUNCTIONS) that contains I in r positions ($1 \leq r < n$) of its AND-FUNCTION is equivalent to 2^r simple decision rules (rules that contain pure AND-FUNCTIONS).

Theorem VI'. A decision rule that contains an OR-FUNCTION with ϕ in r positions ($0 \leq r < n$) is equivalent to $(2^n - 2^r)$ simple decision rules.

REFERENCES

1. Bromberg, Howard,"COBOL and Compatibility," Datamation, February, 1961, pp. 30-34.

2. Phillips, C. A., "Current Status of COBOL," Proceedings of the USAF World Data Systems and Statistics Conference, October 2-6, 1961.

3. Pollack, Soloman L., DETAB-X: An Improved Business-Oriented Computer Language, The RAND Corporation, RM-3273-PR, August, 1962.

4. Systems Group (CODASYL), Preliminary Specifications of DETAB-X, August, 1962.

5. Armerding, G. W., FORTAB: A Decision Table Language for Scientific Computing Applications, The RAND Corporation, RM-3306-PR, September, 1962.

6. Clarke, Dorothea S., "Structure Table Language," SHARE Secretary Distribution (SSD) 96, October 22, 1962; presented at the SHARE meeting in Toronto, Ontario, September 11-13, 1962.

7. Kavanagh, T. F., "TABSOL: A Fundamental Concept for Systems-Oriented Languages," Proceedings of the Eastern Joint Computer Conference, sponsored by IRE, AIEE, and ACM, December 14, 1960.

8. Evans, Orren Y., "Advanced Analysis Method for Integrated Electronic Data Processing," IBM General Information Manual, No. F20-8047, 1961.

9. Grad, Burton, "Tabular Form in Decision Logic," Datamation, July, 1961.

10. Montalbano, Michael, "Tables, Flow Charts and Program Logic," IBM Systems Journal, September, 1962.

11. Proceedings of the Decision Tables Symposium, sponsored by the CODASYL Systems Group and the Joint Users Group of ACM, September 19-21, 1962, New York.

12. Dolotta, T. A. and E. J. McCluskey, Jr., "Encoding of Incompletely Specified Boolean Matrices," Proceedings of the Western Joint Computer Conference, sponsored by IRE, AIEE, and ACM, May 3-5, 1960.

13. Hall, Frank B., "Boolean Prime Implicants by the Binary Sieve Method," Communications and Electronics, published by AIEE, January, 1962.

14. Hirschhorn, Edwin, "Simplification of a Class of Boolean Functions," Journal of the Association for Computing Machinery, Vol. 5, No. 1, January, 1958.

15. Suppes, Patrick, Axiomatic Set Theory, the D. Van Nostrand Company, Inc., New York, 1960.

16. Veitch, E. W., "A Chart Method for Simplifying Truth Functions," Proceedings of the Association for Computing Machinery, sponsored by ACM and the Mellon Institute, Pittsburgh, Pennsylvania, May 2-3, 1952.