THE LOGIC OF DECISION TABLES AND THEIR APPLICATIONS

BY

Albert Pinhas

Submitted to the Department of Electrical Engineering in partial fulfilment of the requirements for the degree

of

Master of Applied Science
Department of Electrical Engineering
Faculty of Science and Engineering
University of Ottawa

OTTAWA, ONTARIO.

JANUARY, 1971

ABSTRACT

The purpose of this thesis is to present an in depth study of Decision Tables. The study includes both, the investigation of the logical structure and the nature of decision tables, as well as the practical application of conversion procedures for electronic data processing (EDP).

In the first part of this thesis a detailed mathematical description in terms of matrices, and the characterization of decision tables are introduced. Based upon these basic rules operations are defined and used for the precise investigation of ambiguity and completeness of decision tables. The problems associated with splitting and merging the decision tables are discussed with the aid of different types of links. Finally, different relations between decision tables are defined, and based upon them algorithms for minimization of different types of decision talbes are included.

In the second part, a survey with detailed analysis of existing conversion procedures and algorithms is presented. This part also includes testing and diagnostic procedures with some powerful extensions.

The final part suggests an approach for implementing a decision table pre-compiler. The thesis is concluded by an attempt to design a comprehensive example that indicates the power and advantages of a good decision table pre-compiler.

ACKNOWLEDGEMENTS

I wish to thank Dr. C.D. Shepard for introducing me to the topic of decision tables and his encouragement to choose this topic for my Master's Thesis.

I am most deeply indebted to Prof. M. Kriger for guiding, encouraging and giving very generously of his time while serving as my advisor in the preparation of my Master's Thesis.

I am also very grateful to my wife for her encouragment understanding and her great deal of patience with which she typed this work.

TABLE OF CONTENTS

CHAPTER I: INTRODUCTION

The recent development of third generation hardware in the computers field and the increasing cost of the software necessary to support it has lead different scientists and mathematicians to design automated ways of software generation such as compiler generators. With the exception of report program generators (RPG'S) and sort program generators no real attempt was made to automate the generation of application programs, which could lead to the elimination (to some extent) of the costly job of the programmer. One possible approach for automation of user's program generation is decision tables.

The purpose of this thesis is to present an in-depth study of the logical structure of decision tables as well as the practical application of possible automated conversion procedures.

Decision tables, as such, represent an organized method for choosing a course of action upon the satisfaction of a set of specified conditions. The early use of decision tables was only as a systems analysis tool. With the advent of more advanced conversion techniques for converting decision tables into computer programs, today they are becoming an accepted tool for writing complex computer programs both in commercial and scientific applications. The main advantages of the use of decision tables are :

1. Direct communication language between the system analyst and the computer.
2. High level, computer independent, programming language.

3. An advanced systems analysis and design tool.
4. Enforces a clear problem statement and shows where information is ambiguous or missing.
5. Provides a precise means of communication among functional personnel.
6. Provides an effictive means of documentation and eliminates much unnecessary verbal description which can be easily misinterpreted.
7. Encourages modularity of programs.
8. Enables visualization of complex situations.
9. An easily learned technique.
10. Decision tables are problem-oriented design methods rather than machine-oriented.
11. Reduces significantly the amount of technical knowledge of the computer required by the systems designer.
12. Provides easier maintenance of a system.

Presently, there are available a large number of conversion processors for the automatic conversion of decision tables into computer programs. With the acceptance of these automated conversion techniques, the precise definition and characterization of decision tables is of vital importance. This is required to avoid ambiguity between the intended logic in the decision table and the corresponding computer program.

This thesis is divided into four parts:

1. Definition and characteristics of decision tables.

2. Manipulation of, and relations between decision tables.
3. A survey of existing conversion methods and algorithms.
4. A suggested decision table pre-compiler.

In the first part a generalized definition of decision tables, that provides consistent nomenclature and classification, is introduced. This is done in a matrix format which is adaptable to mathematical treatment. Exact definitions for decision tables with unrelated conditions as well as with related conditions, in terms of ambiguity and completeness, are presented. These definitions are in terms of the above defined matrix, and thus lend themselves to mathematical treatment. Furthermore, this part includes the fundamentals required for the mathematical manipulation of decision tables.

The second part is devoted to the problems involved in the splitting, merging, relations and minimization of decision tables. The problem of splitting decision tables is considered from the point of view of splitting by rules, by conditions and by branching. This is done by defining different types of linkages between decision tables. The resulting decision tables are either in the form of concatenated tables, or in the form of a tree of decision tables. It was found that merging, the inverse of splitting, could not be generalized.

The problem of relations between decision tables was considered from the point of view of isomorphism and equivalence. In line with this a few algorithms, with the associated problems, are

presented. Finally, three procedures for the minimization of decision tables were developed. One, corresponding to rule minimization, utilizes the concept of rule partitioning. The second corresponding to condition minimization, combines a set of logically related conditions into a single extended condition. The third, corresponding to action minimization, combines a set of actions into a single extended action.

The third part contains a critical survey of existing conversion procedures and algorithms. The procedures corresponding to three different conversion methods (the tree method, the mask method and the computed GO TO method) are convered separately. The advantages and the disadvantages of each conversion method are analysed together with the characteristics of each algorithm. This part also includes a set of theorems proving necessary and sufficient conditions for testing and diagnostic procedures for conversion time and a set of testing procedures for run time.

The last part suggests a flexible decision table pre-compiler based upon practical usage requirements. This scheme is presented in terms of syntactic and semantic description together with suggested conversion procedures. Finally an attempt was made to design a comprehensive example that indicates the usefulness of a decision table pre-compiler (or compiler). The actual conversion of this example into COBOL program code, using the steps of the suggested pre-compiler was done manually due to lack of funds necessary

for an implementation of a desired pre-compiler. The resulting program was run on a Burroughs B-5500 computer.

CHAPTER II. DEFINITION AND CHARACTERISTICS OF DECISION TABLES

The purpose of this chapter is to introduce the general background required in the study of decision tables and to provide a standard nomenclature. The latter is important as there are many uses of decision tables and thus, a number of different nomenclatures are used in existing literature.

In the first part of this chapter a general definition of decision tables is given and a classification based upon this is presented. The classification introduced here subdivides decision tables into 18 different classes, while the existing literature subdivides them into 2, 4, or 6 classes. In the same section a formal notation is introduced to allow detailed mathematical presentation.

In the second part of this chapter, a set of basic rule operations is defined to allow the detailed characterization of decision tables. This characterization, which is based on redundancy, contradiction and completeness, is done seperately for tables with unrelated conditions only as well as for tables with both related and unrelated conditions.

II-1 DEFINITION AND CLASSIFICATION OF DECISION TABLES.

In this section we shall introduce a general definition of decision tables, and based upon this we provide a classification of

decision tables. At the end of this section we also introduce a list of notations.

II-1-1 DEFINITION OF DECISION TABLES:

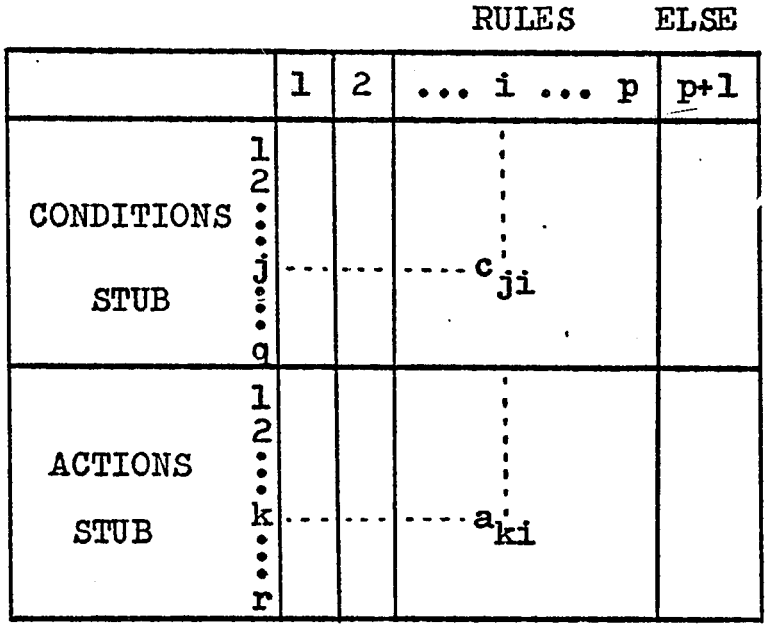


Fig. II-1-1

A decision table as indicated in Fig. II-1-1 has p (or p+1) columns and q+r rows. The resulting table is subdivided into two sub-tables: A conditions sub-table and an actions sub-table.

The conditions sub-table is divided into two parts: the CONDITION STUB containing the q CONDITIONS (condition statements) and the CONDITION ENTRY MATRIX $\{c_{ji}\}_{qxp}$.

A condition is LIMITED, c_j, whenever it is completely specified in the condition stub, or EXTENDED, C_j, whenever the specification of the condition is extended into the appropriate condition row in the condition entry matrix.

provides a means of specifying the actions to be taken if none of the other p rules is satisfied. Further discussion of the else rule will be presented later.

The action sub-table is also divided into two parts; the ACTION STUB containing the r ACTIONS(action statements) and the ACTION ENTRY MATRIX $\{a_{ki}\}_{rxp}$. The action statement can be limited, a_k, whenever the action is completely specified in the action stub, or extended, A_k, whenever the specification of the action is extended into the appropriate action row in the action entry matrix.

With a limited action, a_k, we associate the set $W_k = \{x\}$ and with an extended action we associate the set $W_k = \{W_{k1}, \ldots, W_{kn_k}\}$ with $n_k \geq 1$.

The i^{th} column δ^i of the action entry matrix represents the actions and the order in which they are to be taken whenever the i^{th} rule is satisfied. The k^{th} row of δ^i denoted by δ^i_k is "x"in the limited case, or an element of W_k in the extended case, wherever the k^{th} action is to be executed; otherwise, it will be blank.

Accordingly, the ki^{th} entry of the action entry matrix a_{ki} is given by: $a_{ki} = \delta^i_k \quad :: = \text{blank} | x | \in W_k$

Note: The order in which the conditions of the table are arranged is insignificant in the above definition. This excludes conditions such as testing a quantity for zero before performing other tests which use that quantity as a divisor. This freedom in

With the j^{th} condition we associate the set $V_j = \{V_{j1}, V_{j2}, \ldots, V_{jn_j}\}$ with $n_j \geq 2$. The j^{th} row of the condition entry matrix, P^j, associated with j^{th} condition contains elements from the set V_j or a don't care (sometimes denoted by "d", or left blank, or by "-"). It should be noted that only for extended conditions can $n \geq 2$. However in the case of limited condition $V_j = \{Y,N\}$, $n_j = 2$.

The i^{th} rule is represented by the i^{th} column of the condition entry matrix, γ^i, whose j^{th} entry γ^i_j is either a don't care entry or an element from the set V_j. Such a column γ^i is called an INCOMPLETELY SPECIFIED COLUMN or INCOMPLETE RULE (providing at least one of the entries γ^i_j is a don't care entry and not more than q-1 entries are don't care entries).

Thus the ji^{th} entry of the condition entry matrix is given by;

$$c_{ji} = \gamma^i_j = P^j_i \quad :: = \text{don't care} \mid \in V_j$$

Equivalently γ^i can be represented by a set of columns $\omega^i = \{\omega^{i1}, \omega^{i2}, \ldots, \omega^{i\nu}, \ldots, \omega^{il_i}\}$. Such a column ω^i is called a COMPLETELY SPECIFIED COLUMN (completely specified rule) and its j^{th} entry $\omega^{i\nu}_j$ is an element of V_j.

Note that:

$$1 \leq l_i \leq \max\{n_2 x n_3 x \ldots n_p, n_1 x n_3 x \ldots x n_p, \ldots, n_1 x n_2 \ldots n_{p-1}\}$$

A decision table may include the $p+1^{st}$ column which is associated with the so called ELSE RULE. This is a special rule which

the order of conditions will allow us later to design efficient algorithms for treating decision tables.

To clarify the above definition of decision tables we introduce now an example of a decision table which includes the different types of conditions, actions and rules.

		RULE AND ACTION NUMBERS						
CONDITION STUB		1	2	...	i	...	p	p+1
1	A = B	Y	N	...	N	...	Y	E
2	P =	3	-		15		35	
3	SIG Q	"+"	"0"	(b)	"-"	(c)	-	
⋮	⋮	⋮	⋮	(a)	⋮		⋮	L
j	C > D	Y	N	...	Y	...	-	
j+1	R =	S or P	-		S and P		S	
⋮	⋮	⋮	⋮		⋮		⋮	S
q-2	W =	3 or 4	5.5		-			
q-1	REL U,V	U > V	U<V		-			
q	T	> A	<A	...	< A	...		E
ACTION STUB								
1	A ← B		x	...	x	...		
2	Q ← 10		x	...		...	x	
⋮	⋮	⋮	⋮		⋮		⋮	⋮
K	PERFORM	x1	x2	...	x1	...	x5	
K+1	Q ← E/Q	x	x	...	x	...	-	
⋮	⋮	⋮	⋮		⋮		⋮	⋮
r	GO TO	P1	P2	...	P2	...	P1	ERR

FIG. II-1-2

Example of IS-MCE-MAE decison table.

a. Completely specified rule (providing the entries not shown exclude don't care).
b. Incompletely specified rules.
c. The else rule.
d. Limited condition entries.
e. Extended condition entries.
f. Limited action entries.
g. Extended action entries.

Note: The extended condition entries and action entries do not necessarily represent all different possible conditions and actions. In existing precompilers, interpreters and compilers there are some limitations imposed on the user.

Many other condition can be used, for example, testing a field for numeric or alphabetic, etc. Some of the most commonly used conditions and actions are summarized in later sections.

II-1-2 CLASSIFICATION OF DECISION TABLES.

Decision tables are classified with respect to three major characteristics: The don't care condition, the number of elements in the sets V_j and the number of elements in the sets W_k.

i) A decision table is called COMPLETELY SPECIFIED if it excludes don't care conditions in its condition entries c_{ji}; otherwise it is called INCOMPLETELY SPECIFIED.

ii) A decision table is called LIMITED CONDITION ENTRY if all the

sets V_j, $j = 1,2,\ldots ,q$ are of the type $V_j = \{Y,N\}$ and it is called EXTENDED CONDITION ENTRY if all sets V_j are of the type $V_j = \{V_{j1},\ldots ,V_{jn_j}\}$ $n \geq 2$ and $V_j \neq \{Y,N\}$ $j = 1,2,\ldots ,q$. Otherwise it is a MIXED CONDITION ENTRY decision table.

iii) A decision table is called LIMITED ACTION ENTRY if the sets W_k, $k = 1,2,\ldots r$ are of the type $W_k = \{x\}$ and it is called EXTENDED ACTION ENTRY if all the sets W_k, $k = 1,2,\ldots ,r$ are of the type $W_k = \{W_{k1},W_{k2},\ldots , W_{kn_k}\}$, with $n_k \geq 1$. Otherwise it is called a MIXED ACTION ENTRY decision table.

Accordingly we can group the decision tables into the following classes:

1. CS-ICE-LAE :: = COMPLETELY SPECIFIED - LIMITED CONDITION ENTRY - LIMITED ACTION ENTRY.
2. CS-LCE-EAE :: = COMPLETELY SPECIFIED - LIMITED CONDITION ENTRY - EXTENDED ACTION ENTRY.
3. CS-LCE-MAE :: = COMPLETELY SPECIFIED - LIMITED CONDITION ENTRY - MIXED ACTION ENTRY.
4. IS-LCE-LAE :: = INCOMPLETELY SPECIFIED - LIMITED CONDITION ENTRY - LIMITED ACTION ENTRY.
5. IS-LCE-EAE :: = INCOMPLETELY SPECIFIED - LIMITED CONDITION ENTRY - EXTENDED ACTION ENTRY.
6. IS-LCE-MAE :: = INCOMPLETELY SPECIFIED - LIMITED CONDITION ENTRY - MIXED ACTION ENTRY.
7. CS-ECE-LAE :: = COMPLETELY SPECIFIED - EXTENDED CONDITION ENTRY - LIMITED ACTION ENTRY.
8. CS-ECE-EAE :: = COMPLETELY SPECIFIED - EXTENDED CONDITION ENTRY - EXTENDED ACTION ENTRY.
9. CS-ECE-MAE :: = COMPLETELY SPECIFIED - EXTENDED CONDITION ENTRY - MIXED ACTION ENTRY.
10. IS-ECE-LAE :: = INCOMPLETELY SPECIFIED - EXTENDED CONDITION ENTRY - LIMITED ACTION ENTRY.

matic translation from a formatted decision table into high level (or machine code) computer language.

II-1-3 NOTATION.

To simplify the future discussion on decision tables we will adopt the following notations:

c_i :: = LIMITED CONDITION

C_i :: = EXTENDED CONDITION

d = " " = "-" :: = don't care condition/don't do action

V_j :: = The set of values associated with the j^{th} condition

$V_j = \{V_{ji}, \cdots, V_{jn_j}\}, \quad n_j \geq 2$

Y,N :: = The two different values a limited entry can have

$V_j = \{Y,N\}$ (with exception of the don't care)

p :: = The number of columns in a decision table (except the ELSE rule) equal to the number of rules.

q :: = The number of rows in the condition sub-table equal to the number of conditions.

r :: = The number of rows in the action sub-table equal to the number of actions.

γ^i :: = The i^{th} column in the condition entry matrix representing the i^{th} rule.

γ^i_j :: = The j^{th} entry in the γ^i column (can be a don't care or a value from the set V_j).

ρ^j :: = The j^{th} row in the condition entry matrix.

ρ^j_i :: = The i^{th} entry in the j^{th} row of the condition entry matrix.

11. IS-ECE-EAE :: = INCOMPLETELY SPECIFIED - EXTENDED CONDITION ENTRY - EXTENDED ACTION ENTRY.
12. IS-ECE-MAE :: = INCOMPLETELY SPECIFIED - EXTENDED CONDITION ENTRY - MIXED ACTION ENTRY.
13. CS-MCE-LAE :: = COMPLETELY SPECIFIED - MIXED CONDITION ENTRY - LIMITED ACTION ENTRY.
14. CS-MCE-EAE :: = COMPLETELY SPECIFIED - MIXED CONDITION ENTRY - EXTENDED ACTION ENTRY.
15. CS-MCE-MAE :: = COMPLETELY SPECIFIED - MIXED CONDITION ENTRY - MIXED ACTION ENTRY.
16. IS-MCE-LAE :: = INCOMPLETELY SPECIFIED - MIXED CONDITION ENTRY - LIMITED ACTION ENTRY.
17. IS-MCE-EAE :: = INCOMPLETELY SPECIFIED - MIXED CONDITION ENTRY - EXTENDED ACTION ENTRY.
18. IS-MCE-MAE :: = INCOMPLETELY SPECIFIED - MIXED CONDITION ENTRY - MIXED ACTION ENTRY.

The hierarchy, then, in terms of complexity, can be represented by the following diagram:

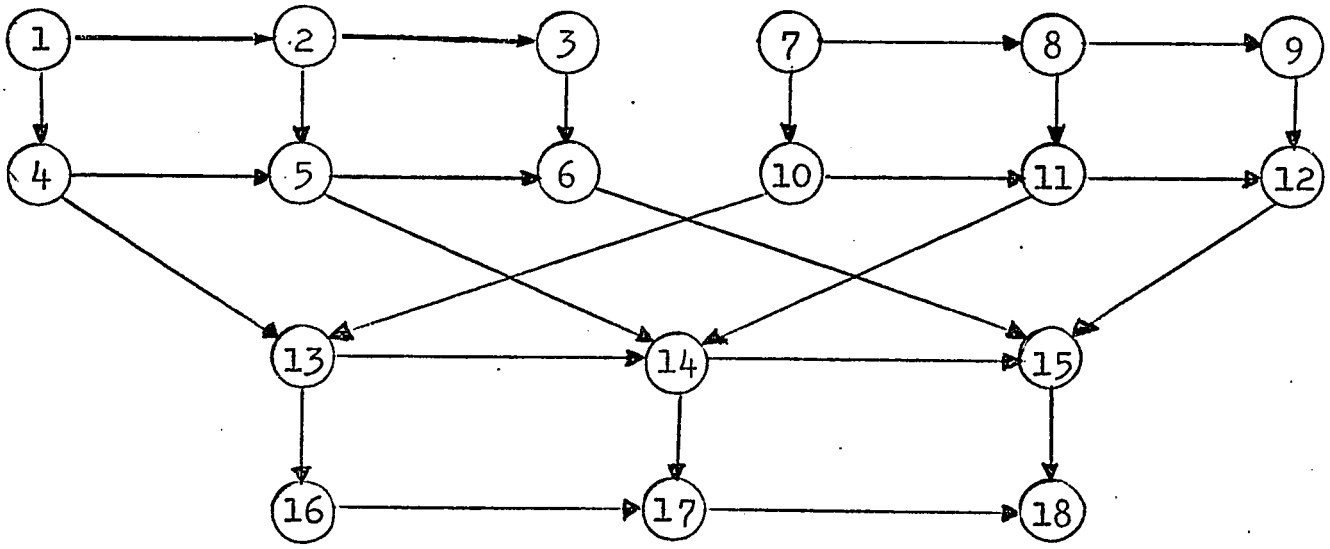


Fig. II-1-3

Note: This hierarchical diagram, as we shall see later, is quite important in developing or adapting algorithms for auto-

$c_{ji} = \gamma^i_j = \rho^j_i$ :: = The ji^{th} entry of the condition entry matrix.

ω^i :: = The set of completely specified columns (or rules) which represent the i^{th} column (or rule) γ^i.

$\omega^{i\nu}$ :: = A completely specified rule from the set ω^i.

$\{c_{ji}\}qxp$ :: = The condition entry matrix.

$\{a_{ki}\}rxp$ :: = the action entry matrix.

a_k :: = A limited action (action statement)

A_k :: = An extended action (action statement)

W_k :: = The set associated with the k^{th} action.

$W_k = \{x\}$ for limited actions.

$W_k = \{W_{k1}, \ldots, W_{kn_k}\}$ for extended actions $n_k \geq 2$.

δ^i :: = The i^{th} column of the action entry matrix.

$\delta^i_k = a_{ki}$ :: = The k^{th} entry in the δ^i column.

Finally, a general decision table can be expressed in terms of the matrix definition given above, as follows:

$$\left[\begin{array}{c|c} \begin{matrix}\sigma_1\\ \vdots\\ \sigma_j\\ \vdots\\ \sigma_q\end{matrix} & \begin{matrix} c_{11} & \cdots & c_{1i} & \cdots & c_{1p}\\ \vdots & & \vdots & & \vdots\\ c_{j1} & & c_{ji} & & c_{jp}\\ \vdots & & \vdots & & \vdots\\ c_{q1} & & c_{qi} & & c_{qp}\end{matrix} \\ \hline \begin{matrix}\alpha_1\\ \vdots\\ \alpha_k\\ \vdots\\ \alpha_r\end{matrix} & \begin{matrix} a_{11} & \cdots & a_{1i} & \cdots & a_{1p}\\ \vdots & & \vdots & & \vdots\\ a_{k1} & & a_{ki} & & a_{kp}\\ \vdots & & \vdots & & \vdots\\ a_{r1} & & a_{ri} & & a_{rp}\end{matrix} \end{array}\right] = \left\{\begin{array}{c|c|c|c} \sum & \gamma^1 & \gamma^2 & \gamma^p \\ \hline \alpha & \delta^1 & \delta^2 & \delta^p \end{array}\right\}.$$

$$= \left[\begin{array}{c|c} \Sigma & \begin{matrix}\rho^1 \\ \vdots \\ \rho^q\end{matrix} \\ \hline \alpha & A \end{array}\right] = \left[\begin{array}{c|c} \Sigma & C \\ \hline \alpha & A \end{array}\right]$$

Where: $\sigma_j :: = c_j/C_j$

$\alpha_k :: = a_k/A_k$

$\Sigma^T = \{\sigma_1, \ldots, \sigma_q\}$

$\alpha^T = \{\alpha_1, \ldots, \alpha_r\}$

II-2 CHARACTERISTICS OF DECISION TABLES.

The major characteristics of decision tables are ambiguity and completeness. Ambiguity in decision tables deals with the problems of Redundancy, Contradiction and Inconsistency. Completeness, on the other hand, deals with the number of rules included in a decision table. In order to define the characteristics we introduce some basic "rule-operations" in the first part of this section. The actual characteristics are first considered for decision tables with unrelated conditions and then extended to include decision tables with related and unrelated conditions.

II-2-1 BASIC RULE OPERATIONS.

Here we shall only introduce those "rule-operations" which are required in the definition of ambiguity and completeness. Other rule-

operations are defined wherever neccessary.

Definition II-2-1; - "Simple Rule Splitting" - Let γ^i be an incompletely specified rule with c_{ji} as a don't care entry and the associated set $V_j = \{V_{j1}, V_{j2}, \dots, V_{jn_j}\}$. Then a simple rule splitting of c_{ji} (with respect to V_j), is given by:

$$\gamma^i = {}_{ij}\begin{bmatrix} c_{1i} \\ c_{2i} \\ \vdots \\ - \\ \vdots \\ c_{qi} \end{bmatrix} \xrightarrow[V_j]{} \left\{ \begin{bmatrix} c_{1i} \\ c_{2i} \\ \vdots \\ V_{ji} \\ \vdots \\ c_{qi} \end{bmatrix} ; \begin{bmatrix} c_{1i} \\ c_{2i} \\ \vdots \\ V_{j2} \\ \vdots \\ c_{qi} \end{bmatrix} ; \dots , \begin{bmatrix} c_{1i} \\ c_{2i} \\ \vdots \\ V_{jn_j} \\ \vdots \\ c_{qi} \end{bmatrix} \right\} = \bigcup_{p=1}^{n_j} \begin{bmatrix} c_{1i} \\ c_{2i} \\ \vdots \\ V_{jp} \\ \vdots \\ c_{qi} \end{bmatrix}$$

Note: The remaining entries c_{ji} in γ^i may or may not be don't care entries.

Definition II-2-2; - "Complete Rule Splitting" - A complete rule splitting is defined as a multiple simple rule splitting of all the don't care entries of an incompletly specified rule γ^i. That is, from the definition of the decision table in II-1-1, γ^i can be expressed by the set of completely specified rules $\{\omega^i\}$. If, for example, c_{ji} and $c_{j+k,i}$ are the only don't care entries in some rule γ^i with the associated sets V_j and V_{j+k} $(j+k < q)$ respectively, we denote this transformation as follows:

$$\gamma^i \xrightarrow[V_j, V_{j+k}]{*} \{\omega^i\} = \{\omega^{i1}, \omega^{i2}, \dots, \omega^{i\nu}, \dots, \omega^{il_i}\}$$

Where $\omega^{i\nu}$ is a completely specified rule and $\omega^{i\nu} \neq \omega^{i\mu}$ if $\nu \neq \mu$. Furthermore, note that li is the number of completely specified rules corresponding to γ^i.

Example:

Let $\gamma^i = \begin{matrix} 1 \\ 2 \\ 3 \\ 4 \end{matrix}\begin{bmatrix} Y \\ - \\ - \\ RED \end{bmatrix}$ where $\{V_2\} = \{Y,N\}$, $\{V_3\} = \{100,110,120\}$

then simple rule splitting with respect of c_{2i} is given by:

$$\gamma^i \xrightarrow[V_2]{} \left\{ \begin{bmatrix} Y \\ Y \\ - \\ RED \end{bmatrix} ; \begin{bmatrix} Y \\ N \\ - \\ RED \end{bmatrix} \right\}$$

and a complete rule splitting is given by:

$$\gamma^i \xrightarrow[V_2,V_3]{*} \left\{ \begin{bmatrix} Y \\ Y \\ 100 \\ RED \end{bmatrix} ; \begin{bmatrix} Y \\ Y \\ 110 \\ RED \end{bmatrix} ; \begin{bmatrix} Y \\ Y \\ 120 \\ RED \end{bmatrix} ; \begin{bmatrix} Y \\ N \\ 100 \\ RED \end{bmatrix} ; \begin{bmatrix} Y \\ N \\ 110 \\ RED \end{bmatrix} ; \begin{bmatrix} Y \\ N \\ 120 \\ RED \end{bmatrix} \right\}$$

Note: In splitting a rule in an actual decision table, we have to assign the original action column to every one of the completely specified rules associated with the split rule. Thus:

$$\begin{bmatrix} \gamma^i \\ \hline \delta^i \end{bmatrix} \xrightarrow{*} \left\{ \begin{bmatrix} \omega^{i1}_{\cdot} \\ \hline \delta^i \end{bmatrix} ; \begin{bmatrix} \omega^{i2} \\ \hline \delta^i \end{bmatrix} ; \ldots ; \begin{bmatrix} \omega^{il} \\ \hline \delta^i \end{bmatrix} \right\}$$

We shall consider now the decision table characteristics separately for independent and dependent conditions.

II-2-2 CHARACTERISTICS OF DECISION TABLES WITH UNRELATED CONDITIONS

Definition II-2-4; - Two conditions(or condition statements) are called independent if and only if the variables in the respective condition stubs are logically independent, otherwise they are called dependent,or related conditions.

Using the definition II-2-4 we can define redundancy and contradiction in the case of limited condition entry decision tables (LCE) with independent conditions as follows:

Definition II-2-5; - If two (or more) rules γ^i and γ^k have identical actions $\delta^i = \delta^k$ respectively, and do not include at least one pair of Y,N in any of the ρ^p rows (p = 1,2,... ,q), then redundancy exists in the LCE decision table.

Definition II-2-6; - If two (or more) rules γ^i and γ^k have different actions $\delta^i \neq \delta^k$ respectively, and do not include at least one pair of Y,N in any of the ρ^p rows (p = 1,2,... ,q) then there is contradiction in the LCE decision table.

To include decision tables with mixed condition entries, these last two definitions can be generalized as follows:

Definition II-2-3; - "Rule Intersection" - Let γ^i and γ^k be represented by the following sets of completely specified rules:

$$\gamma^i = \omega^i = \{\omega^{i1}, \omega^{i2}, \ldots, \omega^{1i}\}$$

$$\gamma^k = \omega^k = \{\omega^{k1}, \omega^{k2}, \ldots, \omega^{1k}\}$$

then the intersection of γ^i and γ^k is given by:

$$\gamma^i \cap \gamma^k = \omega^i \cap \omega^k = \{\Omega^1_{ik}, \Omega^2_{ik}, \ldots, \Omega^{l_{ik}}_{ik}\} = \Omega_{ik}$$

where $\Omega_{ik} \subseteq \omega^i$ and $\Omega_{ik} \subseteq \omega^k$ (in the usual meaning of sub-sets).

Also note that the intersection is defined only between two rules of the same order (namely, with the same number of entries) and usually used for rules taken from the same table.

Furthermore, $\gamma^i \cap \gamma^k = \gamma^k \cap \gamma^i = \Omega_{ik} = \Omega_{ki}$ and $l_{ik} = l_{ki}$ represents the number of completely specified rules in the set $\{\Omega_{ik}\}$.

Example:

Let $\gamma^i = \begin{bmatrix} Y \\ - \\ - \end{bmatrix}$ and $\gamma^k = \begin{bmatrix} - \\ Y \\ - \end{bmatrix}$ where $\{V_3\} = \{\text{RED}, \text{BLUE}, \text{PINK}\}$

then $\gamma^i \cap \gamma^k = \left\{ \begin{bmatrix} Y \\ Y \\ \text{RED} \end{bmatrix} ; \begin{bmatrix} Y \\ Y \\ \text{BLUE} \end{bmatrix} ; \begin{bmatrix} Y \\ Y \\ \text{PINK} \end{bmatrix} \right\} = \{\Omega_{ik}\}$

Definition II-2-7; - If two (or more) rules γ^i and γ^k have identical actions $\delta^i = \delta^k$ respectively, and do not include at least one pair of values $\gamma_p^i \neq \gamma_p^k$ in any of the rows ρ^p ($p = 1,2,\ldots ,q$) there is redundancy in the MCE decision table.

Definition II-2-8; - If two (or more) rules γ^i and γ^k have different actions $\delta^i \neq \delta^k$ respectively, and do not include at least one pair of values $\gamma_p^i \neq \gamma_p^k$ in any of the rows ρ^p ($p = 1,2,\ldots ,q$) then a contradiction exists in the MCE decision table.

Lemma II-2-1: Given any two rules γ^i and γ^k of a decision table, their intersection is empty ($\{\Omega_{ik}\} = \emptyset$) if and only if they contain at least one pair of different values $\gamma_p^i, \gamma_p^k \neq$ don't care in any of the rows ρ^p, ($p = 1,2,\ldots ,q$)

Proof: Let $\gamma^i \xrightarrow{*} \{\omega^i\} = \{\omega^{i1}, \omega^{i2}, \ldots , \omega^{il_i}\}$

and $\gamma^k \xrightarrow{*} \{\omega^k\} = \{\omega^{k1}, \omega^{k2}, \ldots , \omega^{kl_i}\}$

a. First, let's prove that: If there is at least one pair $\gamma_p^i \neq \gamma_p^k$, ($p = 1,2,\ldots ,q$) then $\gamma^i \cap \gamma^k = \{\Omega_{ik}\} = \emptyset$.
Since, by assumption $\gamma_p^i \neq \gamma_p^k$ (and they are not don't care) then, $\omega^{i\nu} \neq \omega^{k\mu}$ ($\nu = 1,2,\ldots ,l_i$, $\mu = 1,2,\ldots ,l_k$) and therefore, $\{\Omega_{ik}\} = \emptyset$.

b. If $\{\Omega_{ik}\} = \emptyset$ then there is at least one $\gamma_p^i \neq \gamma_p^k$ ($p = 1,2,\ldots q$). Here, for simplicity, we shall start from completely specified rules:

i. For completely specified rules $\omega^j = \omega^{jl} = \gamma^j$, $j = i,k$ thus if $\omega^i \cap \omega^k = \Omega_{ik} = \emptyset$ then $\omega^{il} \neq \omega^{kl}$ and therefore, $\gamma^i \neq \gamma^k$ which means there is at least one pair of values $\gamma_p^i \neq \gamma_p^k$ $p = 1,2,\ldots q$.

ii. For incompletely specified rules, assuming $\{\Omega_{ik}\} = \emptyset$ we have this for every $p = 1,2,\ldots ,l_i$.
$\omega^i \cap \{\omega^{k1},\omega^{k2},\ldots ,\omega^{kl_k}\} = \emptyset$ thus, $\omega^i \cap \gamma^k = \emptyset$ which means there is at least one pair of values $\omega_p^{i\nu} \neq \omega_p^{k\mu}$ $p = 1,2,\ldots ,q$ such that $\gamma_p^k \neq$ don't care. This implies the existence of at least one pair of values $\gamma_p^i \neq \gamma_p^k$ (γ_p^i, $\gamma_p^k \neq$ don't care).

Theorem II-2-1; Given a decision table with unrelated conditions, the table is ambiguous if and only if there is at least one set $\{\Omega_{ik}\} \neq \emptyset$ for $i,k = 1,2,\ldots ,p$. Furthermore, if ambiguity exists then:

i. If $\delta^i = \delta^k$ the table includes redundancy.

ii. If $\delta^i \neq \delta^k$ the table includes contradiction.

Proof:

a. If there is at least one set $\{\Omega_{ik}\} \neq \emptyset$, $i,k = 1,2,\ldots ,p$, then by Lemma II-2-1, there can be no pair of values $\gamma_p^i \neq \gamma_p^k$ in any of the rows ρ^p, $p = 1,2,\ldots ,q$. Therefore:

   i. By definition II-2-7 if $\delta^i = \delta^k$ the table includes redundancy.

   ii. By definition II-2-7 if $\delta^i \neq \delta^k$ the table includes contradiction.

This concludes the first part of the proof.

b. If the table includes ambiguity, then, by definition II-2-7 and II-2-8, the table contains at least one pair of rules γ^i and γ^k $i,k = 1,2,\ldots,p$ such that they do not include any pairs of different values $\gamma_p^i \neq \gamma_p^k$ $p = 1,2,\ldots,q$. Thus by Lemma II-2-1 there is at least one set $\{\Omega_{ik}\} \neq \emptyset$ for $i,k = 1,2,\ldots,p$.

This concludes the proof.

In a decision table where all the conditions are independent only data dependent inconsistency can exist.

Definition II-2-9; - A rule γ^i is said to be data dependent inconsistenty if it includes $c_{ij} \notin V_j$ or if the combination of its entries cannot be satisfied by real data. This is a semantics problem and is beyond the scope of this document.

Example:

	RULE NUMBER				
CONDITIONS	1	2	3	4	5
SIZE-OF-JACKET	42	42	42	82	ELSE
SIZE-OF-PANTS	40	42	48	46	
ACTIONS					
ORDER	x	x	x	x	
DO NOT ORDER					x

$V_1 = \{40,42,44\}$

Fig. II-2-1

In Fig. II-2-1 the third rule is data inconsistent of the second type because the combination of its entries cannot be satisfied by real data. Rule number 4 is also data inconsistent because $82 \notin V_1$.

Note that this type of inconsistency cannot be identified by any mechanical procedure.

In decision tables with related conditions, logical inconsistency may also exist. This type of inconsistency is presented in section II-2-3.

Definition II-2-10: - The Else Rule - Wherever included, represents all the possible rules not explicitely stated in the condition entry matrix.

Definition II-2-11: - A decision table with independent conditions is said to be complete whenever:

1. It includes the else rule (in this case, the table may include ambiguities)

or

2. It does not include any ambiguity and,

$$\sum_{i=1}^{p} l_i = \prod_{j=1}^{q} n_j$$

where n_j is the number of elementsin the set V_j .

Note that in the case of LCE decision tables, $n_j = 2$ and

$$\sum_{i=1}^{p} l_i = 2^q$$

II-2-3 CHARACTERISTICS OF DECISION TABLES WITH RELATED AND UNRELATED CONDITIONS.

The above definitions were developed for decision tables with unrelated conditions. Actually, they can also be applied to decision tables with related conditions, but, in this case, they become too restrictive for practical use. At this point, we shall introduce an example to indicate the restrictive nature of these definitions and then give a modified definition of ambiguity.

Example II-2-2;

In Electrical Engineering, it could happen that the following decision table with related conditions has to be used:

	RULE NUMBER		
CONDITIONS	1	2	3
VOLTAGE < 110	Y	-	-
VOLTAGE = 110	-	Y	-
VOLTAGE > 110	-	-	Y
ACTIONS			
GO TO	LOW	LOW	HIGH

Fig. II-2-2:

Decision table with related conditions.

Obviously, the conditions in the example are related and the rules are well designed and do not leave a place for ambiguity of

any kind. On the other hand, applying the rule for ambiguity and completeness just defined, we can easily determine that:

1. Rule number 1 and rule number 2 are redundant.
2. Rule number 2 and rule number 3 are contradictory.
3. Splitting the rules completely we get $l_i = 12 \neq 2^3 = 8$
4. Every rule includes inconsistency. For example:

$$\begin{bmatrix} Y \\ - \\ - \end{bmatrix} \xrightarrow{*} \left\{ \begin{bmatrix} Y \\ Y \\ Y \end{bmatrix} ; \begin{bmatrix} Y \\ Y \\ N \end{bmatrix} ; \begin{bmatrix} Y \\ N \\ Y \end{bmatrix} ; \begin{bmatrix} Y \\ N \\ N \end{bmatrix} \right\}$$

The first three competely specified rules are inconsistent.

In this case the don't care entries have a different meaning: they specify that the condition entry with a don't care is irrelevant to the decision if the particular rule holds, and therefore, in making the decision, that condition should not be tested.

Actually, we could have designed a decision table that satisfies all the above definitions performing the same logical function as the one in the given example, Fig. II-2-3:

	RULE NUMBER			
CONDITIONS	1	2	3	4
VOLTAGE < 110	Y	N	N	E
VOLTAGE = 110	N	Y	N	L
VOLTAGE > 110	N	N	Y	S
				E
ACTIONS				
GO TO	LOW	LOW	HIGH	ERROR

Fig. II-2-3
Decision table with related conditions

Note that, in this case, we must include the else rule for completeness (which will never take place in the example because it contains all the inconsistent rules which are logically impossible). Furthermore, in converting that table to a program, we have to make three tests in order to find the rule, which is an unnecessary increase in program running time. Actually to save program running time the designer may introduce in the decision tables with related conditions, irrelevant don't care entries, as in the above Example.

Definition II-2-12; - Given a mixed condition entry decision table with two (or more) rules γ^i and γ^k that have identical actions $\delta^i = \delta^k$, then:

i) Possible redundancy exists if γ^i and γ^k do not contain at least one pair γ_p^i, γ_p^k of different values in any row ρ^p $p = 1,2,\ldots,q$ but have at least one pair γ_p^i,- or -, γ_p^k.

ii) If they contain at least one pair γ_p^i, γ_p^k of different values the rules are different and not redundant.

iii) If neither i) nor ii) holds, the rules are identical and therefore trivially redundant.

Definition II-2-13; - Given a mixed condition entry decision table with two or more rules γ^i and γ^k that have different actions $\delta^i \neq \delta^k$ then:

i. Possible contradiction exists if γ^i and γ^k do not contain at least one pair γ_p^i, γ_p^k of different values in any row ρ^p,

2. If the number of elements in the set $\{\omega\} = \bigcup_{i=1}^{p} \{\omega^i\}$ is equal to $\prod_{j=1}^{q} n_j$.

Note: If the table includes extended condition entries, inconsistent data may give unpredicted errors when running a program embodying such a table (depending on the way the table was translated into a computer program) therefore, the inclusion of the else rule is recommended whenever possible.

$p = 1,2,\ldots,q$ but have at least one pair γ_p^i - or - γ_p^k.

ii) If the table contains at least one pair γ_p^i, γ_p^k of different values the rules are different and do not contain contradictions.

iii) If neither i) nor ii) holds, the rules are identical and therefore trivially contradictory.

As stated before, in decision tables with related conditions outside data dependent inconsistency (see definition II-2-9), logical inconsistency may also exist

Definition II-2-14; - In decision tables with related conditions, a rule γ^i is said to include logical contradiction, if it can technically exist but implies logical contradiction.

Example II-2-3:

In the table shown in Fig. II-2-4 the first and last rules are technically possible, but obviously, logically impossible. In this case, the conditions are clearly logically related or dependent.

	RULE NUMBER			
CONDITIONS	1	2	3	4
A = B	Y	Y	N	N
A ≠ B	Y	N	Y	N

Fig. II-2-4

Note that even this trivial type of logical inconsistency cannot be, in general, identified by a mechanical procedure.

Definition II-2-15; - A decision table with related and unrelated conditions is said to be complete whenever:

1. It includes the Else Rule (even if it includes real ambiguities).

CHAPTER III. MANIPULATION OF AND RELATIONS BETWEEN DECISION TABLES

The purpose of this chapter is to consider some basic procedures for obtaining simpler decision tables from a given table.

In the first part of this chapter we consider the basic procedures for splitting decision tables. After defining the different linkages that can exist between decision tables, the decision table is split with respect to its rules, or conditions, or by branching. Furthermore, it is indicated that the merging of decision tables can be done only as each specific case dictates.

In the second part of this chapter, isomorphism and equivalence of decision tables is defined and then simplification of decision tables is done by minimization. The minimization techniques developed in this chapter are accomplished along three lines: rule minimization; condition stub minimization and action stub minimization.

III-1 SPLITTING AND MERGING OF DECISION TABLES.

In the previous section we presented the definition and the charactristics of a single decision table. In the present section we consider sets of a single decision table that are functionally interconnected. First we shall define a few procedures for splitting

a (large) decision table into a set of smaller decision tables such that the resulting set of interconnected decision tables will have the same function as the original one. Finally, we shall deal with the problem involved in merging sets of interconnected decision tables.

III-1-1 SPLITTING OF DECISION TABLES.

Splitting of decision tables can be obtained along two lines: Splitting by rules and/or splitting by conditions. When splitting by rules the result is two or more decision tables with the same number of conditions, each containing else rules. On the other hand, splitting by conditions results in two or more decision tables with a reduced number of rules and conditions. It should be noted that in both cases, an extra action must be used to allow linkage between the resulting tables.

The linkage between decision tables can have one of the following three forms:

1. Simple Linkage: - A decision table T_1 is said to be linked to a decision table T_2 by a "Simple Linkage" if it contains in its action stub a statement which transfers the control to T_2 upon satisfaction of one or more of its rules. A simple linkage may exist between a single decision table and a set of decision tables.

2. Concatenated linkage: - A decision table T_1 is said to be concatenated to decision table T_2 if it contains in its action stub a statement which transfer control to T_2 only if none of its explicit rules is satisfied. This implies that T_1 includes the Else rule.

3. Reference linkage: - A decision table T_1 is said to be linked by a "Reference Linkage" or simply, to refer to T_2 if it contains in its action stub a statement which transfers temporary control to T_2, such that upon completion of T_2 the control is transferred back to T_1 to the next action in sequence (if any). This implies that T_2 may not be linked to any other table by type 1 or type 2 linkes. That type of link allows for a single decision table to refer to a set of decision tables.

The third type of linkage is presented here only for the sake of completeness. It will be utilized only in later chapters. The usage of this type of linkage implies certain restriction which will be discussed later.

Example III-1-1:

Simple Linkage - In this example we shall use the statement "GO TO" to transfer control from one decision table to another. As shown below, a decision table can be linked by a "Simple Linkage" to one or more decision tables.

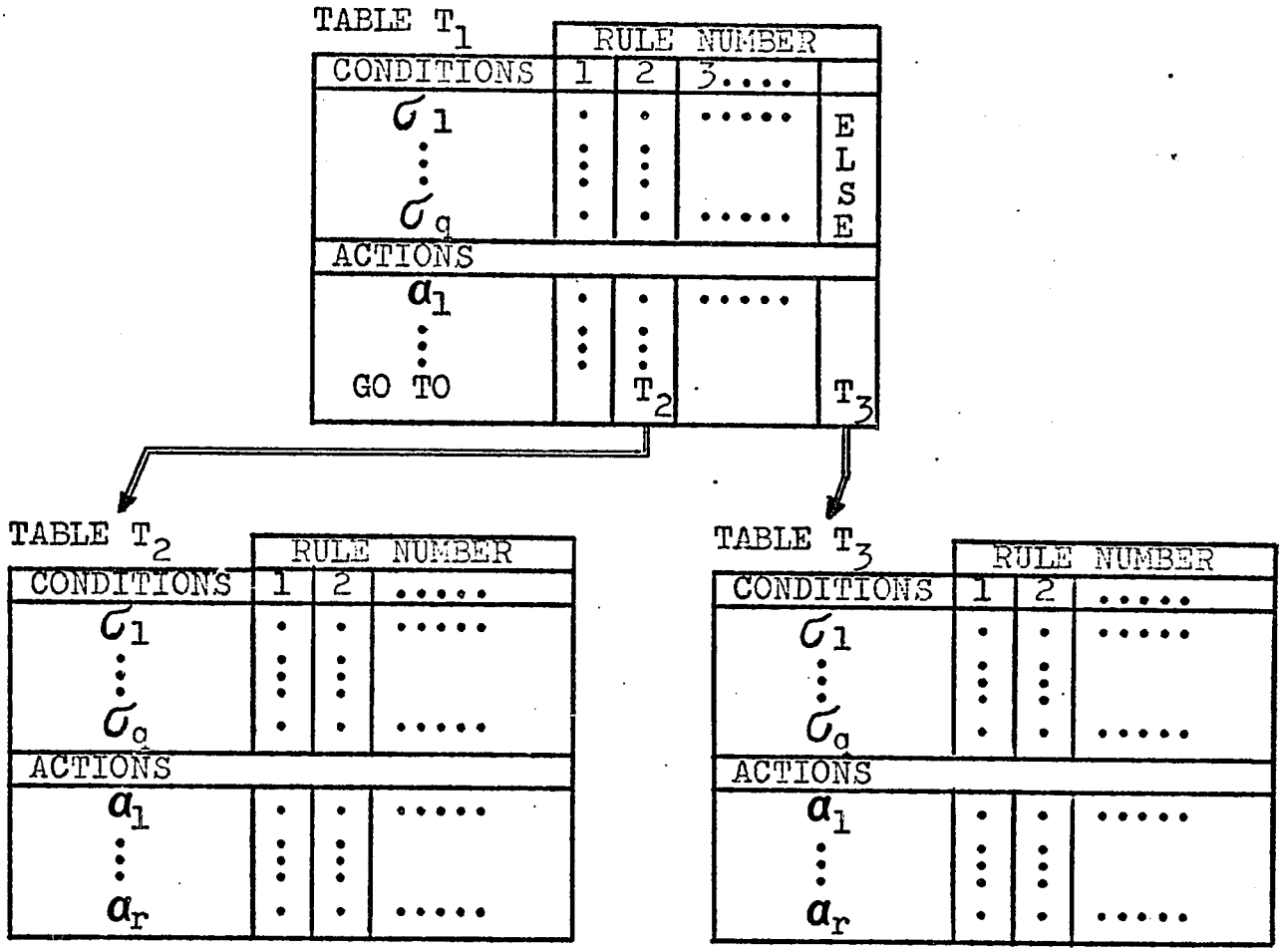


Fig. III-1-1: Simple Linkage

Example III-1-3: Concatenation Linkage

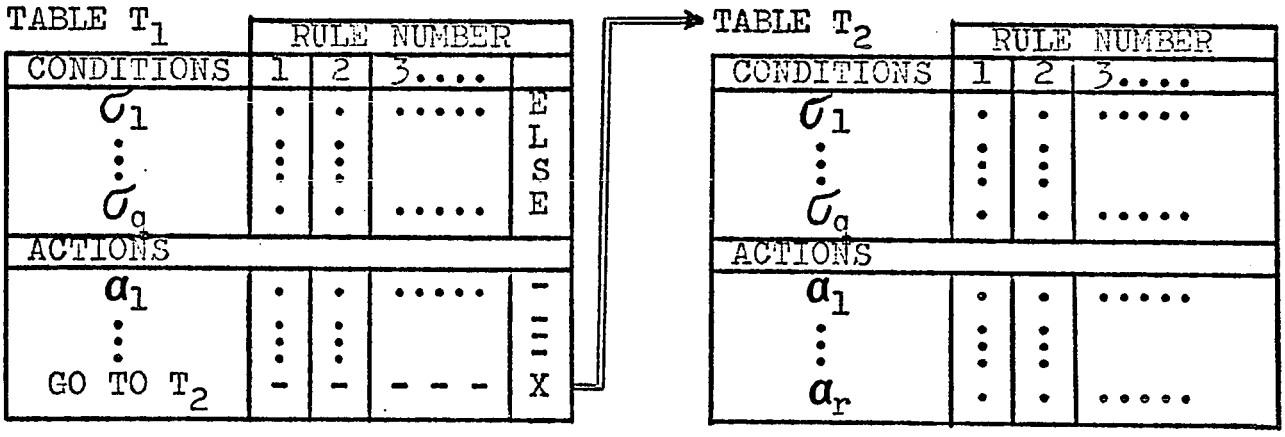


Fig III-1-2: Concatenation Linkage

Example III-1-3:

Reference Linkage - In this example, we shall use the statment "PERFORM" for "REFERENCE LINKAGE".

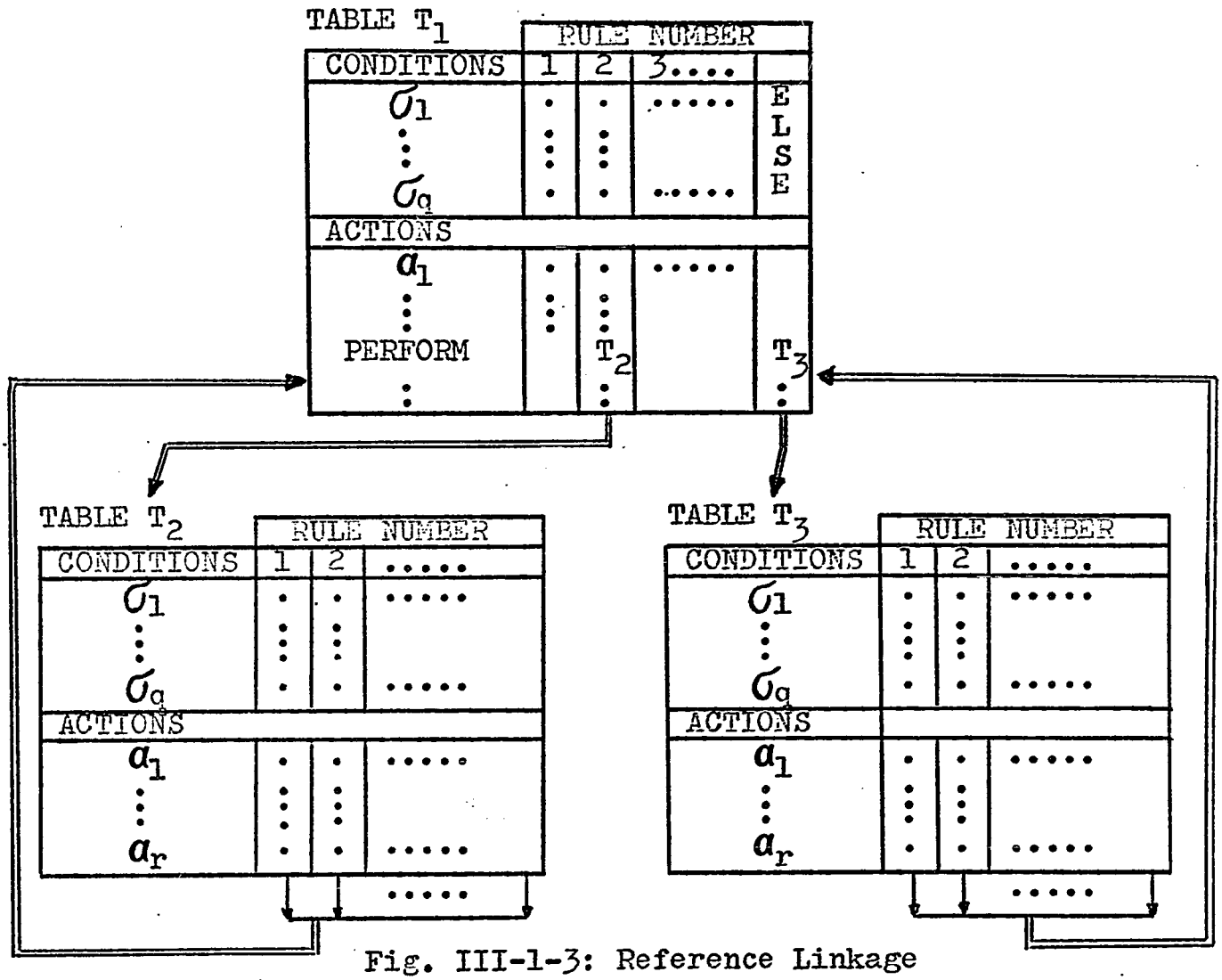


Fig. III-1-3: Reference Linkage

The splitting of a decision table by its rules is equivalent to breaking a given decision table into a set of concatenated decision tables. The actual procedure, described below, breaks the set of rules $\{\gamma^1, \gamma^2, \ldots, \gamma^{p+1}\}$ into two or more concatenated sets

$\{\gamma^1, \gamma^2, \ldots, \gamma^i, \gamma^{E_1}\} \ldots \{\gamma^{i+1}, \ldots, \gamma^{p+1}\}$ of rules, where the additional rules denoted by $\gamma^{E_1}, \gamma^{E_2}, \ldots$ serve as concatenation links. A decision table is usually split by rules whenever the number of rules included in a single table are too large for practical usage.

SPLITTING BY RULES:

Given a decision table T_1 with the set $\{\gamma^1, \gamma^2, \ldots, \gamma^p\}$ of rules. To split this table into two decision tables T_{11} and T_{12} after the i^{th} rule and without altering the logic, the following rules should be applied:

TABLE T_1

	RULE NUMBER						
CONDITIONS	1	2	3	...	i	...	p
σ_1 ⋮ σ_q	⋮	⋮	⋮	...	⋮	...	⋮
ACTIONS							
a_1 ⋮ a_r	⋮	⋮	⋮	...	⋮	...	⋮

Fig. III-1-4

1. Reordering of rules and/or conditions, before and/or after the splitting is allowed, as long as the logic remains unchanged.
2. Conditions and/or actions which become inapplicable after the splitting, may be excluded.
3. Every decision table (except the last one) contains an additional rule which is an else rule, and an additional action for concatenation linkage purpose (in the case of splitting into two, only the first decision table contains the linkage).

The resulting decision tables after the splitting are shown below.

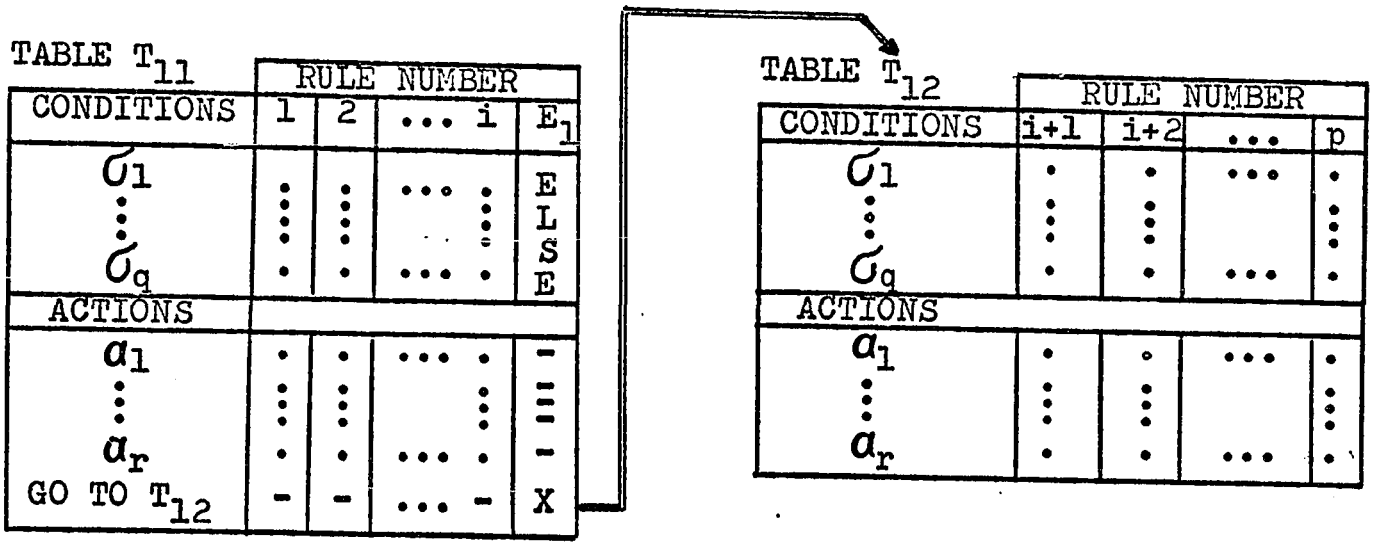


Fig. III-1-5: Splitting by Rules

Splitting into more than two decision tables is done in the same way, keeping the same rules.

Note that any decision table can be split by rules up to p concatenated decision tables, each containing a single rule and an else rule for linkage (except the last one, which contains a single rule plus the original else rule if it existed).

Example III-1-4: - "Splitting by Rule"

In this example we shall split a given decision table into two along the 16^{th} rule.

It should be noted that in the process of splitting, we excluded the second condition σ_2 and the second action a_2 from T_{11}, since they originally had only don't care entries in their condition entry and action entry matrices up to number 16. This, obviously, did not change the logic.

TABLE T_1

CONDITIONS	RULE NUMBER 1	2	...	16	17	...	p
σ_1	c_{11}	c_{12}	...	-	c_{117}	...	c_{1p}
σ_2	-	-	...	-	c_{217}	...	c_{2p}
σ_3	c_{31}	c_{32}	...	c_{316}	.	...	.
⋮	⋮	⋮		⋮	⋮		⋮
σ_q	c_{q1}	c_{q2}	...	c_{q16}	c_{q17}	...	c_{qp}
ACTIONS							
a_1	a_{11}	a_{12}	...	a_{116}	a_{117}	...	a_{1p}
a_2	-	-	...	-	a_{217}	...	a_{2p}
a_3	a_{31}	.		.	.		.
⋮	⋮	⋮		⋮	⋮		⋮
a_r	a_{r1}	.	...	a_{r16}	.	...	a_{rp}

Splitting along the 16^{th} rule we get:

TABLE T_{11}

CONDITIONS	RULE NUMBER 1	2	...	16	17
σ_1	c_{11}	c_{12}	...	c_{116}	E
σ_3	c_{31}	c_{32}	...	c_{316}	L
⋮	⋮	⋮		⋮	S
σ_q	.	.	...	.	E
ACTIONS					
a_1	a_{11}	a_{12}	...	a_{116}	-
a_3	a_{31}	a_{32}	...	.	-
⋮	⋮	⋮		⋮	⋮
a_r	.	.	...	.	.
GO TO T_{12}	-	-	-	-	X

TABLE T_{12}

CONDITIONS	RULE NUMBER 17	...	p
σ_1	c_{117}	...	c_{1p}
σ_2	c_{217}	...	c_{2p}
⋮	⋮		⋮
σ_q	.	...	.
ACTIONS			
a_1	a_{117}	...	a_{1p}
a_2	a_{217}	...	a_{2p}
⋮	⋮		⋮
a_r	a_{r17}		a_{rp}

Fig. III-1-6: Splitting by Rules.

Splitting of a decision table by rules is definitely not recommended, unless it is absolutely necessary to do so. Splitting a

decision table by rules, into a set of K concatenated decision tables, multiplies the number of conditions that have to be tested by a factor of Q where $1 < Q < K$.

The splitting of a decision table by its conditions is also equivalent to breaking a given decision table into a set of concatenated decision tables. The actual splitting can be done only if the condition entry matrix C can be partitioned into blocks, such that all nondiagonal blocks (cells) contain only don't care entries as in Fig. III-1-7.

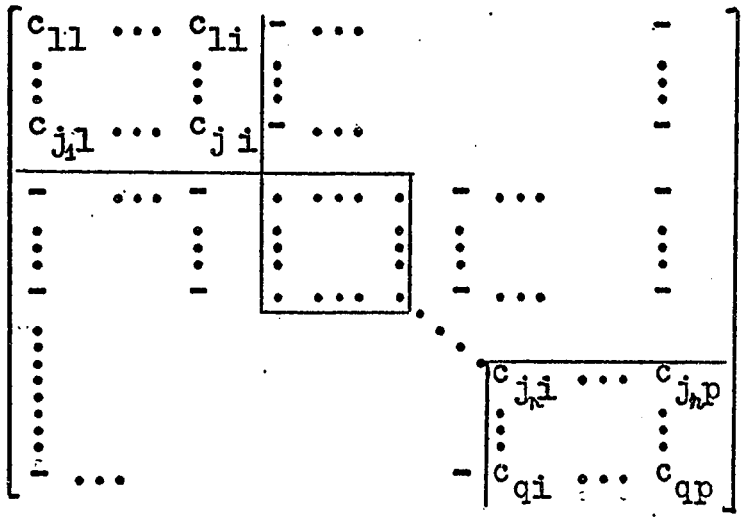


Fig. III-1-7

A partitioned condition entry natrix with don't care entries in nondiagonal blocks.

SPLITTING BY CONDITIONS

Given a decision table with a partitioned condition entry matrix, in which all the non-diagonal blocks contain only don't care entries, and with K diagonal blocks:

TABLE T_1

CONDITIONS	RULE NUMBER: 1	2	...	i_1	....	i_k ...	p
σ_1	c_{11}	c_{12}	...	c_{1i}	- ...		-
⋮	⋮	⋮		⋮			⋮
σ_{j_1}	c_{j1}	c_{j2}	...	c_{ji}	- ...		-
⋮	- ...			-	⋮ ... ⋮	- ...	⋮
	⋮			⋮	⋮ ⋮		
	-			-	⋮ ... ⋮	- ...	
	⋮					⋱	
σ_{j_k}						$c_{j_k i}$ ...	$c_{j p}$
⋮						⋮	⋮
σ_q	- ...					- $c_{q i}$ ...	$c_{q p}$
ACTIONS							
a_1	a_{11}	a_{12}	...				a_{1p}
a_2	a_{21}	a_{22}	...				a_{2p}
⋮	⋮	⋮					⋮
a_r	a_{r1}	a_{r2}	...				a_{rp}

Fig. III-1-8.

The result of splitting the above given table by conditions will be K concatenated decision tables. The set of conditions σ_1, σ_2,... , σ_q is broken into K sets of σ_1,... , σ_{j_1}, σ_{j_1+1},... , σ_{j_2},... , σ_{j_k},... , σ_q. The decision table will be represented by a set of concatenated decision tables; every one containing one set of actions respectively, and all the actions. The linkage is

done by concatenation.

TABLE T_{11}

CONDITIONS	RULE NUMBER 1	2	...	i_1	
σ_1	c_{11}	c_{12}	...	c_{1i_1}	E
⋮	⋮	⋮		⋮	L
$\sigma_{j_1 1}$	$c_{j_1 1}$	$c_{j_1 2}$	...	$c_{j_1 i}$	S
ACTIONS					E
a_1	a_{11}	a_{12}	...	a_{1i}	-
⋮	⋮	⋮		⋮	=
a_r	a_r	a_{r2}		$a_{r i}$	-
GO TO T_{12}	-	-	...	-	X

TABLE T_{12}

CONDITIONS	RULE NUMBER	
⋮	⋮ ... ⋮	E
		L
		S
		E
ACTIONS		
⋮	⋮ ... ⋮	=
		=
		=
GO TO T_{13}	- ... -	X

TABLE T_{13}

CONDITIONS	RULE NUMBER i	...	p
σ_{j_n}	$c_{j_n i}$	...	$c_{j p}$
⋮	⋮		⋮
σ_a	$c_{a i}$	...	$c_{a p}$
ACTIONS			
a_1	.	...	a_{1p}
a_2	.	...	a_{2p}
⋮	⋮		⋮
a_r	.	...	a_{rp}

Fig. III-1-9: Splitting by Conditions

Note that the splitting procedure can be executed only in special cases, e.g., whenever the problem could have been originally expressed by a set of concatenated decision tables. Whenever rearranging rules does not alter the logic, it is possible to start from a decision table in which the condition entry matrix does not allow splitting by conditions. Then, after reordering the rules and the conditions, to generate a decision table in which splitting by conditions is possible. (As we shall see in section II-5 it is possible to do the same by changing the set of rules by merging rules, or as we have seen before, by splitting rules).

Another case in which splitting by conditions is possible, is of an extended condition entry decision table which is translated into a limited entry decision table, e.g.,

Example III-1-5:

Given the extended condition entry decision table:

	RULE NUMBER			
CONDITIONS	1	2	3	4
IF A =	B	C	D	ELSE
IF P =	Q	R	T	
ACTIONS				
a_1	•	•	•	•
a_2	•	•	•	•

Fig. III-1-10

The equivalent limited entry decision table is:

	RULE NUMBER			
CONDITIONS	1	2	3	
A = B	Y	-	-	E
P = Q	Y	-	-	
A = C	-	Y	-	L
P = R	-	Y	-	
A = D	-	-	Y	S
P = T	-	-	Y	E
ACTIONS				
a_1	•	•	•	•
a_2	•	•	•	•

Fig. III-1-11

Note that splitting by rules and excluding inapplicable conditions would have resulted in the same set of concatenated decision tables.

Another type of decision table splitting by its conditions is also possible. To differentiate it from the previous one, we shall refer to it as "Splitting by Branching". In this case, any decision table may be split with respect to any one of its conditions, providing the order of the rules is immaterial.

SPLITTING BY BRANCHING

Splitting by branching can be done if and only if, the order of the rules in a decision table that has to be split is immaterial and reordering does not change the logic. Otherwise, splitting by branching can be done only with respect to the conditions c_j, in which the condition row entries ρ^j are sorted by values from V_j and do not contain don't care entries.

Assuming that reordering of rules is allowed, the splitting is done as follows:

If the splitting is done with respect to a limited condition entry the result will be three different tables. The first table contains only the condition which controls the splitting, transferring control to the second or the third decision table, depending on the outcome of this condition testing.. In the second and the third tables, however, that condition is absent and also there are less

TABLE T_{11}

CONDITIONS	RULE NO. 1	2
c_1	Y	N
ACTIONS		
GO TO	T_{12}	T_{13}

TABLE T_{12}

CONDITIONS	RULE NUMBER 1	2	4	5	7	ELSE
σ_2	c_{21}	c_{22}	c_{24}	c_{25}	c_{27}	
σ_3	c_{31}	-	c_{34}	-	-	
ACTIONS						
a_1	·	·	·	·	·	·
a_3	·	·	·	·	·	·
⋮						
a_p	·	·	·	·	·	·

TABLE T_{13}

CONDITIONS	RULE NUMBER 3	5	6	7	ELSE
σ_2	c_{23}	c_{25}	c_{26}	c_{27}	
ACTIONS					
a_1	·	·	·	·	·
a_2	·	·	·	·	·
⋮					
a_p	·	·	·	·	·

Fig. III-1-13

Note that the action a_2 was excluded in T_{12} and the conditions σ_3 was excluded in T_{13}. Every rule which had a don't care entry in its first condition entry row ρ^1, appears in T_{12} and T_{13}:

If the splitting is done with respect to an extended condition entry the result will be $n_j + 1$ decision tables, otherwise, everything else is done by complete analogy.

Note that this process of splitting by branching can be continued with each subtable until the resulting set of subtables contains subtables with only one condition or one rule (and one else rule if the else rule was included in the original table). Tests for ambiguity and completess in the final stage of splitting become trivial.

Splitting by branching can also be done with respect to more

rules in each. If the else rule was included in the original table, the second and the third decision tables will contain it. Conditions and actions which become inapplicable in the process of splitting may be excluded.

Assuming that the splitting is done with respect to the first condition then the actual splitting is done as illustrated in Fig. III-1-13 below.

TABLE T_1	RULE NUMBER							
CONDITIONS	1	2	3	4	5	6	7	
σ_1	Y	Y	N	Y	-	N	-	E
σ_2	c_{21}	c_{22}	c_{23}	c_{24}	c_{25}	c_{26}	c_{27}	L
σ_3	c_{31}	-	-	c_{34}	-	-	-	S
								E
ACTIONS								
a_1	.	.	.	.	.	.	.	.
a_2	-	-	a_{23}	-	-	a_{26}	-	-
a_3	.	.	.	.	.	.	.	.
⋮								
a_p	.	.	.	.	.	.	.	.

Fig. III-1-12

than one condition as illustrated in the following example.

Example III-1-6: Splitting by branching with respect to two conditions

TABLE T_1

CONDITIONS	RULE NUMBER 1	2	3	4	5	6	7	8
c_1	Y	Y	Y	Y	N	N	N	N
c_2	Y	Y	N	N	Y	Y	N	N
σ_3	c_{31}	c_{32}	-	c_{34}	c_{35}	-	-	-
σ_4	-	c_{42}	c_{43}	-	c_{45}	c_{46}	c_{47}	c_{48}
ACTIONS								
a_1	·	·	·	·	·	·	·	·
⋮								
a_r	·	·	·	·	·	·	·	·

Fig. III-1-14

After splitting with respect to c_1 and c_2 we get the following set of tables.

TABLE T_{11}

CONDITIONS	RULE NUMBER 2	3	4	5
c_1	Y	Y	Y	N
c_2	Y	N	N	Y
GO TO	T_{12}	T_{13}	T_{14}	T_{15}

T_{12}

CON.	RULE NO. 1	2
σ_3	c_{31}	c_{32}
σ_4	-	c_{42}
ACT.		
a_1	·	·
⋮		
a_r	·	·

T_{13}

CON.	RULE NO. 3	4
σ_3	-	c_{34}
σ_4	c_{43}	-
ACT.		
a_1	·	·
⋮		
a_r	·	·

T_{14}

CON.	RULE NO. 5	6
σ_3	c_{35}	-
σ_4	c_{45}	c_{46}
ACT.		
a_1	·	·
⋮		
a_r	·	·

T_{15}

CON.	RULE NO. 7	8
σ_3	-	-
σ_4	c_{47}	c_{48}
ACT.		
a_1	·	·
⋮		
a_r	·	·

Fig. III-1-15

Even though the splitting procedures introduced in this section were presented separately, in practice they can be utilized in some combined way.

III-1-2 MERGING OF DECISION TABLES.

Merging of decision tables refers to the combination of two or more decision tables into one decision table. The question that must be ascertained first, is whether or not two decision tables (or more) can be merged without producing ambiguity and without changing the logical function of the original decision tables.

From this point of view, one can state that, given a set of decision tables, which was derived by using one or more of the splitting procedures, introduced in section III-1-1, they can be merged. The problem is that there is no mechanized procedure for testing whether the decision tables were derived by splitting a simple decision table or not. Therefore, even in this case, no generalized merging procedure can be given.

Now, with considering any two decision tables, it was found that even under severe restrictions there is no generalized procedure which will not produce, in general, ambiguities or other logical errors. The only statement that can be made is that merging of two decision tables (or more) can be considered only on an individual basis.

III-2 RELATIONS BETWEEN DECISION TABLES.

In this section we shall define the different relations possible between any two decision tables namely; isomorphsm, equivalence, and coverage. As we shall see in the next section the equivalence relation is of special importance mainly in practical usage.

III-2-1 ISOMORPHISM AND EQUIVALENCE OF DECISION TABLES.

Definition III-2-1; Two decision tables are called isomorphic if:

1. They have the same set of conditions in the conditions stub, (not necessarily in the same order).
2. They have identical action stubs.
3. They have the same set of rules with the same actions associated respectively.
4. And by reorganizing the rules and the conditions in one decision table, providing the logic remains unchanged, it can be transformed into the second.

The above definition of isomorphism defines an equivalence relation. The only problem is how to check whether reorganizing of the rules changes the logic of a table or not. If the conditions are unrelated, the procedure for checking isomorphism can be mechanized.

Checking two decision tables (with dependent conditions) for isomorphism can be accomplished manually. This is done by checking after every rule reorganization to see that the logic was not changed. Another way is simply to check if all possible combinations of data will result in the same actions in both tables.

Definition III-2-2: Two decision tables T_1 and T_2 are said to be simply equivalent if:

1. They have the same set of conditions.
2. They have identical action stubs.

And as well, we can replace one decision table by the other without changing the logical function of the total system.

We shall introduce at this point another rule operation which, together with the rule splitting operations that were introduced in II-2-1, will simply allow us to treat equivalent tables more systematically.

Definition III-2-3: - "Rule Combination" - Let $\{\gamma^{n_1}, \gamma^{n_2}, \ldots, \gamma^{n_\ell}\}$ be a subset of rules in a decision table with identical actions $\delta^{n_u} \equiv \delta^{n_v}$ $u,v = 1,2,\ldots,j$. Let $V_j = \{V_{j1}, V_{j2}, \ldots, V_{jn}\}$ be the set of values associated with the j^{th} condition. Then if:

1. The subset contains rules which are identical except for the j^{th} entry; and
2. Every element $V_{ji} \in V_j$ appears at least in one of the rules $(n_\ell > n_j)$.

This subset of rules can be combined into one rule in which the j^{th} entry is a don't care entry.

Example: "Rule Combination"

$V_j = \{1,2,3,4\}$

$$j \left\{ \begin{bmatrix} c_{1n} \\ c_{2n} \\ \vdots \\ 1 \\ \vdots \\ c_{qn} \end{bmatrix} ; \begin{bmatrix} c_{1n} \\ c_{2n} \\ \vdots \\ 2 \\ \vdots \\ c_{qn} \end{bmatrix} ; \begin{bmatrix} c_{1n} \\ c_{2n} \\ \vdots \\ 3 \\ \vdots \\ c_{qn} \end{bmatrix} ; \begin{bmatrix} c_{1n} \\ c_{2n} \\ \vdots \\ 4 \\ \vdots \\ c_{qn} \end{bmatrix} \right\} V_j^* \rightarrow \begin{bmatrix} c_{1n} \\ c_{2n} \\ \vdots \\ - \\ \vdots \\ c_{qn} \end{bmatrix}$$

Now it becomes quite obvious how two decision tables can be simply equivalent. To illustrate this, we shall introduce the following example.

Example III-2-1: - "Simple equivalent tables"

Suppose the following table is given

	RULE NO.			
CONDITIONS	1	2	3	4
c_1	Y	Y	Y	E
c_2	Y	N	N	L
c_3	Y	Y	N	S E
ACTIONS				
a_1	X	X	X	
a_2				X

Fig. III-2-1

It is easy to see that we can combine rules 1 and 2 or rules 2 and 3 giving two different decision tables which are simply equivalent by definition II-4-2.

	RULE NO.		
CONDITIONS	1	2	3
c_1	Y	Y	E
c_2	-	N	L
c_3	Y	N	S E
ACTIONS			
a_1	X	X	
a_2			X

	RULE NO.		
CONDITIONS	1	2	3
c_1	Y	Y	E
c_2	Y	N	L
c_3	Y	-	S E
ACTIONS			
a_1	X	X	
a_2			X

Fig. III-2-2

For checking if two decision tables with unrelated conditions are simply equivalent the following procedure can be used:

Procedure III-2-1: Given two decision tables T_1 and T_2.

STEP 1: Check if the conditions are the same

1. If they are different the tables are not simply equivalent, END.
2. GO to STEP 2.

STEP 2: Check if the actions stubs are identical

1. If not, the tables are not simply equivalent, END.
2. If yes GO to STEP 3

STEP 3: Transfer each table T_i $(i = 1,2)$ to a completely specified condition entry decision table T_i^c, GO to STEP 4.

STEP 4: Check if T_1^c and T_2^c are isomorphic.

1. If Yes, the tables are simply equivalent, END.
2. If Not, the tables are not simply equivalent, END.

Checking for simple equivalence in the general case, when the decision tables may include related rules, may be considered only on an individual basis. The only mechanized procedure that can be applied in every case is simply generating all possible combinations of data and checking on both tables to see if the resulting actions are the same for every combination. But even in this case, since many combinations might be logically, or data inconsistent, the results of the test may not prove a thing.

Definition III-2-4: Two decision tables are called equivalent if, and only if, we can replace one table by the other without changing the logical function of the total system.

In order to introduce a few comprehensive examples of equivalent decision tables, we shall define a transformation of an extended condition entry into a set of limited condition entries.

Definition III-2-5: - Let C_j be an extended condition associated with the set of values $V_j = \{V_{j1}, V_{j2}, \ldots, V_{jn_j}\}$ then the transformation of C_j and the associated row ρ^j of condition entries into a set of limited conditions, and limited condition entries is done as follows:

1. The extended condition C_j is transformed into n_j limited conditions (obviously related).
2. Every don't care entry in c_j will result in a complete column of don't care entries after the transformation.
3. If as a result of the transformation, we get a condition c_p

associated with a row ρ^p full of don't care entries, it can be suppressed.

Example:

Given the extended condition A

A = 1 2 3 4 - 2

associated with the set $V_j = \{1,2,3,4,5\}$

then after the transformation we get

	1	2	3	4	5	6
A = 1	Y	-	-	-	-	-
A = 2	-	Y	-	-	-	Y
A = 3	-	-	Y	-	-	-
A = 4	-	-	-	Y	-	-
A = 5	-	-	-	-	-	-

Obviously, the 5^{th} condition may be suppressed, column number 5 is the result of having a don't care entry in the original row. Columns 2 and 6 must both be included.

Attention should be paid to the fact that the number of columns after the transformation remains the same and the number of limited entry conditions is n_j or less. Therefore, this transformation does not affect the action entry matrix.

Definition III-2-6: Let A_k be an extended action associated with the set $W_k = \{W_{k1}, W_{k2}, \ldots, W_{kn_k}\}$ then the transformation of A_k and the associated action entries row into a set of limited entry actions and limited action entries, is done as follows:

1. The extended action A_k is transformed into n_k limited actions.
2. Every blank entry in the action entries row results in a complete columns of blanks.

Example:

Given the extended action:

GO TO L1 L2 L3 - L2 - -

then after the transformation we get

GO TO L1 X

GO TO L2 X X

GO TO L3 X

Here again the number of columns remains the same and the number of limited entry actions is exactly n_k. Therefore this transformation does not effect in any way the condition entry matrix.

In the following example we shall see two equivalent decision tables and using the transformations we just defined, we shall prove that they are equivalent. Unfortunately, a general procedure for testing for equivalence cannot be given. In general case, the equivalence or the difference between two decision tables may even depend on information not explicitely stated in the decision tables under consideration, and again, every case must be treated on an individual basis.

Example:

The following two decision tables are equivalent

TABLE T_1

ACTIONS	RULE NUMBER 1	2	3	4	5	6	7
A =	1	2	3	4	-	2	ELSE
c_2	Y	Y	-	Y	N	N	
c_3	-	Y	N	Y	-	Y	
ACTIONS							
GO TO L_1	X						
GO TO L_2		X			X		
GO TO L_3			X				
B ← A				X			
T ← A					X		

TABLE T_2

ACTIONS	RULE NUMBER 1	2	3	4	5	6	7
A = 1	Y	-	-	-	-	-	E
A = 2	-	Y	-	-	-	Y	
A = 3	-	-	Y	-	-	-	L
A = 4	-	-	-	Y	-	-	S
c_2	Y	Y	-	Y	N	N	
c_3	-	Y	N	Y	-	Y	E
ACTIONS							
GO TO	L_1	L_2	L_3		L_2		
B ← A				X		X	
T ← A							X

Fig. III-2-3

In the above given example (which is relatively simple) we can deduce the equivalence of the tables simply by showing that by using the transformations just defined, we can transform every one of them to the table given below which is equivalent to T_1 and to T_2. (The equivalence of decision tables is an equivalence relation i.e. Reflexive symmetric and transitive).

TABLE T

ACTIONS	RULE NUMBER 1	2	3	4	5	6	7
A = 1	Y	-	-	-	-	-	E
A = 2	-	Y	-	-	-	Y	
A = 3	-	-	Y	-	-	-	L
A = 4	-	-	-	Y	-	-	S
c_2	Y	Y	-	Y	N	N	
c_3	-	Y	N	Y	-	Y	E
ACTIONS							
GO TO L_1	X						
GO TO L_2		X			X		
GO TO L_3			X				
B ← A				X		X	
T ← A							X

Fig. III-2-4

In general, two equivalent decision tables may contain completely different conditions or actions which are equivalent in some particular type of

application. If, for example, the variable ALPHA is redefined somewere as Beta (in COBOL) then a condition such as "IF ALPHA > 0" is logically equivalent to the statement "IF BETA > 0". But it is very difficult to know just from the information given in the decision tables. Moreover, in ALGOL, for example, if a statement like "A ← B ← C" is executed previous to executing the same decision table, then a condition like "IF A > 0" is equivalent to the condition "IF B > 0". But this equivalence is not apparent and requires additional knowlege not included in the decision tables.

III-2-2 MINIMIZATIN OF DECISION TABLES.

Minimization of decision tables can be done along three lines: Rule minimization, done by rule combination; conditions stub minimization, done by transforming sets of limited entry conditions into extended entry conditions and action stub minimization, done by transformation of sets of limited entry actions into extended entry actions.

RULE MINIMIZATION: - Here we try to minimize a given decision table by minimizing the number of rules included in the condition entry matrix by combining or merging sets of rules whenever possible.

A necessary condition for rule combination is identical actions associated with the rules we are trying to combine. Therefore the first step in minimizing the table by rules is partitioning the set of rules into sub-tables with identical actions. If the

partitioning requires reorganizing rules then afterwards additional tests should be conducted to verify that the logic was not changed. If, by reorganizing the rules, the logic was changed and the original order of the rules is significant, then only partial partitioning is possible. In some cases some reorganization of rules is possible without altering the logic of the decision tables and should be done, if neccessary, before partioning. The goal, in every case, is to reach a partion with the smallest number of sub-sets possible. Suppose the following decision table is given:

$$\left[\frac{\Sigma}{a}\middle\| \begin{matrix}\gamma^1 & \gamma^2 & \gamma^3 & \gamma^4 & \gamma^5 & \gamma^6 & \gamma^7 & \gamma^8 & \gamma^9 & \gamma^{10}\\ \delta^1 & \delta^2 & \delta^3 & \delta^4 & \delta^5 & \delta^6 & \delta^7 & \delta^8 & \delta^9 & \delta^{10}\end{matrix}\right]$$

where $\delta^1 = \delta^2 = \delta^6$

$\delta^3 = \delta^4 = \delta^5 = \delta^7 = \delta^8$

$\delta^9 = \delta^{10}$

then, if reorganization of rules is possible, the partioning will give:

$$\{\gamma^1,\gamma^2,\gamma^6\} \; ; \{\gamma^3,\gamma^4,\gamma^5,\gamma^7,\gamma^8\} \; ; \{\gamma^9,\gamma^{10}\}$$

if the original order of the rules is significant, the partitioning will give:

$$\{\gamma^1,\gamma^2\} \; ; \{\gamma^3,\gamma^4,\gamma^5\} \; ; \{\gamma^6\} \; ; \{\gamma^7,\gamma^8\} \; ; \{\gamma^9,\gamma^{10}\}$$

if, for example, some reorganizing should take place without chan-

ging the logic, for example if shifting γ^6 to the 8^{th} position was possible, then the partioning will give:

$$\{\gamma^1,\gamma^2\} \; ; \{\gamma^3,\gamma^4,\gamma^5,\gamma^7,\gamma^8\} \; ; \{\gamma^6\} \; ; \{\gamma^9,\gamma^{10}\}$$

The second step in rule minimization is to combine within every subset, as many rules as possible. To do that we shall partition again every sub-set by condition entry rows. As a first step in partioning again, we have to find which are the possible condition entry rows that can be used. If the i^{th} sub-set includes N_i rules, then only the condition entry row associated with condition σ_j which, in turn, is associated with the set $V_j = \{V_{j1}, V_{j2}, \ldots, V_{jn_j}\}$ where $n_j < N_i$, should be considered. For simplicity, we shall call the number n_j associated with the set V_j the index of the condition σ_j.

We start by trying to partition every sub-set with respect to the largest index $n_j < N_i$. Every n_j rules or more, included in the same sub-set, will form a new sub-set in the secondary partition if the j^{th} condition entry row contains the complete set $V_j = \{V_{j1}, V_{j2}, \ldots, V_{jn_j}\}$ and all the other condition row entries contain identical values, respectively. Such a sub-set is called COMPLETE. If the original order of the rules is significant, then the rules of that sub-set must appear in a continuous sequence.

Now every complete sub-set of rules may be combined into a single incompletely specified rule with respect to the j^{th} entry*.

* See also the definition of rule combination; Definition III-2-3

Here again we must differ between two cases. If a sub-set can be partioned into a set of complete sub-sets, then the secondary partition is called COMPLETE; otherwise, PARTIAL SECONDARY PARTITON.

A complete secondary partition with respect to any index $n_j < N_i$ will decrease the number of rules in the sub-set by a factor of n_j.

That procedure of secondary partitioning and rule combination, can be continued with respect to the other conditions until no more secondary partioning is possible. The following example will clarify the procedure of rule minimization.

Example:

Let the $\gamma^1, \gamma^2, \ldots, \gamma^{16}$ be a sub-set of rules with identical actions as given below

$$\{\gamma^1, \gamma^2, \ldots, \gamma^{16}\} = \begin{bmatrix} a\ a\ a\ a & b\ b\ b\ b & c\ c\ c\ c & d\ d\ d\ d \\ Y\ Y\ N\ N & Y\ Y\ N\ N & Y\ Y\ N\ N & Y\ Y\ N\ N \\ Y\ N\ Y\ N & Y\ N\ Y\ N & Y\ N\ Y\ N & Y\ N\ Y\ N \\ e\ e\ e\ e & e\ e\ e\ e & e\ e\ e\ e & e\ e\ e\ e \end{bmatrix}$$

Where the first condition entry row is associated with the index $n_1 = 4$ (and $V_1 = \{V_{11}, V_{12}, V_{13}, V_{14}\} = \{a,b,c,d\}$ and the next two are limited condition entry rows associated with the index $n_2 = n_3 = 2$ (and $V_2 = V_3 = \{Y,N\}$).

If the original order of the rules is not significant, starting from the largest index $n_1 = 4$, we can derive the following complete secondary partition:

$\{\gamma^1, \gamma^5, \gamma^9, \gamma^{13}\}$; $\{\gamma^2, \gamma^6, \gamma^{10}, \gamma^{14}\}$; $\{\gamma^3, \gamma^7, \gamma^{11}, \gamma^{15}\}$; $\{\gamma^4, \gamma^8, \gamma^{12}, \gamma^{16}\}$.

and after rule combination with respect to the above secondary partition, the following set of rules will result.

$$\{\gamma_1^1, \gamma_1^2, \gamma_1^3, \gamma_1^4\}^* = \begin{bmatrix} - & - & - & - \\ Y & Y & N & N \\ Y & N & Y & N \\ e & e & e & e \end{bmatrix}$$

As can be easily seen, the number of rules is decreased by a factor $n_1 = 4$ (from 16 to 4), because the secondary partition is complete.

Now we can go one step farther and try to find another partition with the largest index n_j possible, which is, in this case, $n_3 = 2$, and the partition is:

$$\{\gamma_1^1, \gamma_1^2\} \; ; \; \{\gamma_1^3, \gamma_1^4\}$$

after rule combimation we get:

$$\{\gamma_2^1, \gamma_2^2\} = \begin{bmatrix} - & - \\ Y & N \\ - & - \\ e & e \end{bmatrix}$$

In the final step we arrive at a single rule which is:

---

* The notation γ_n^i is used to denote the i^{th} rule after n rule combination took place

$$\{\gamma_3^1\} = \begin{bmatrix} - \\ - \\ - \\ e \end{bmatrix}$$

Note: The above example was so designed as to give, in every step, a complete secondary partition. In practical tables, this is not always the case. Sometimes it will be even difficult to decide which index to use for partitioning (if they are equal) as for example the following case:

$$\{\gamma^1\ \gamma^2\ \gamma^3\} = \begin{bmatrix} Y & Y & N \\ N & Y & Y \\ Y & Y & Y \end{bmatrix}$$

By using $n_1 = 2$ we get $\{\gamma^1\}$; $\{\gamma^2, \gamma^3\}$, and by using $n_2 = 2$ we get $\{\gamma^1, \gamma^2\}$; $\{\gamma^3\}$. As a result, whenever the secondary partitions are not complete, the above described procedure may not neccessarily lead to the minimal number of rules, but will always decrease the number of rules. In such a case, the derivation of the minimal number of rules can be done by exhausting all possibilities, trying in every step, to find a complete secondary partition with respect to the maximal index.

CONDITION STUB and ACTION STUB MINIMIZATION.

The condition stub and action stub minimization are the opposite transformations of the transformations defined in III-2-5/6. By their very nature they are possible only in special cases. The usefulness of that kind of minimization is mainly for documentation

purposes and whenever a precompiler , interpreter or compiler, which accept ECE-EAE decision tables, is available. The procedure of minimizing the condition stub or the action stub (wherever possible) is simple and done step by step in the opposite order as defined in definitions III-2-5 and III-2-6 and for that reason will be omitted.

Note: that the nature of rule minimization is different from the condition stub and action stub minimization. In rule minimization we actually decrease the size of the decision table (not only physically) by reducing the number of rules and in terms of electronic data processing we decrease, in general, the compile time and run time. The condition stub and action stub minimization on the other hand decreases the size of the decision table only physically but usually increases compile or procompiler time due to the increase of complexity and does not save, in general, run time.

CHAPTER IV. CONVERSION METHODS

The methods used in converting decision tables to computer programs (either manual or automatic) fall into three main categories, depending upon the translation method. In this chapter we shall introduce the TREE METHOD, the MASK METHOD and the COMPUTED GO TO METHOD. The purpose of this chapter is to present existing procedures corresponding to the above methods and to print out their advantages, disadvantages and the limitations.

IV-1 THE TREE METHOD:

In this section, the tree method for conversion of decision tables to computer programs is described. Many variations of this method have been published, and based upon this method a large number of precompilers, interpreters and compilers have been implemented. All the procedures and algorithms based upon the tree technique were designed mainly for IS-LCE decision tables. At the end of the section we shall analyse the advantages and disadvantages of the tree method.

IV-1-1 THE TECHNIQUE:

The simplest technique for conversion of decision tables to computer programs is to translate the decision table into an equivalent flow chart, rule by rule, based upon "if... then..." relation-

ships. To show this we shall use the decision table given in Fig. IV-1-1:

	RULE NUMBER			
CONDITIONS	1	2	3	4
c_1	Y	Y	N	E
c_2	Y	Y	-	L
c_3	Y	N	N	E
ACTIONS				
a_1	x	x		x
a_2	x		x	x
a_3				x

Fig. IV-1-1

The corresponding flow chart is shown in Fig. IV-1-2. Note that this technique produces as many branch points as the number of condition entries which are not "don't care" entries.

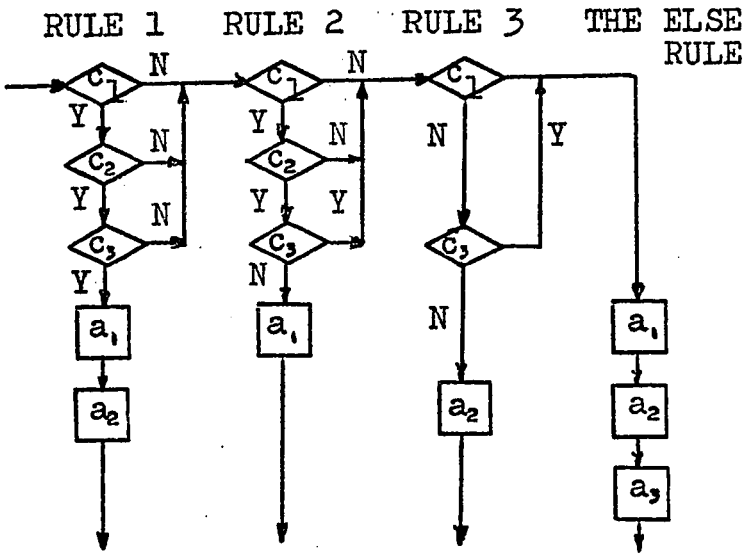


Fig. IV-1-2

In this section the conversion is presented using a flow chart to avoid the explicit use of a given programming language. Once a flow chart is obtained it can be written in assembly or any other high level language.

As seen from Fig. IV-1-2, in this translation a given condition is tested for every rule that includes it. For example condition c_1 is tested in every rule except the else rule. The "tree method" alleviates this by eliminating unnecessary redundancy. This is shown in the "decision tree" in Fig. IV-1-3.

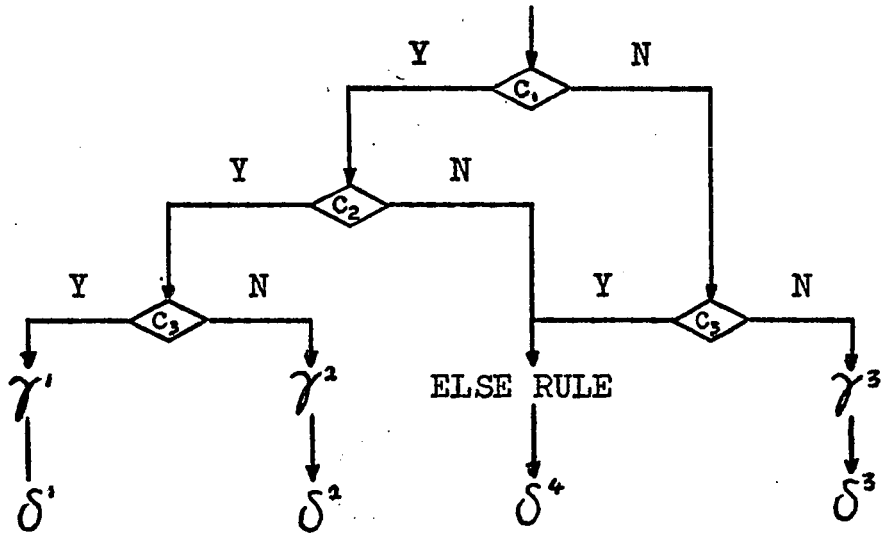


Fig. IV-1-3: Decision Tree

Each condition is tested never more than once and, therefore, the number of tests in a decision tree necessary to satisfy a particular condition, never can exceed the total number of conditions in the condition stub of a decision table.

In comparing the flow chart in Fig. IV-1-2 with the Decision Tree in Fig. IV-1-3, we can see that, in the worst cases, the former

must test eight times, while the latter requires three. Furthermore, a savins of over 50% is realized in storage required utilizing the Decision Tree Technique. Thus the superiority of the Tree echnique over the usual flow chart a certainty.

Note that only if the order in which the conditions are tested must be kept the same as in the condition stub, then the Decision Tree is unique. In general the form of the Decision Tree for any decision table is not unique and depends on the order in which the conditions are tested:

1. If, in every level of the decision tree, we test the same condition (if required), the number of different decision trees for a given decision table is equal to the number of permutations of the q conditions and therefore equal to q!.
2. If for every branch in the decision tree we are free to choose the next condition to be tested (in every level), then the number of decision trees for a given decision table is much larger. In this case, the upper limit of the number of different decision trees for a given decision table is:

$$q!\ [(q-1)!\ ]^2 \cdot [(q-2)!\ ]^3 \cdots [2!]^{q-1} = \prod_{u=1}^{q} (u!)^{q-u}$$

This upper limit is obtained whenever the decision table does not include don't care conditions.

Even though many of the outcoming decision trees may look similar to each other, but taking into account that different tests may have different program code storage requirements and different

run times, the different trees may differ in total storage or run time requirments.

IV-1-2 EFFICIENCY CONSIDERATIONS.

In converting decision tables to computer programs the main goal is to generate the most efficient program code in the least amount of conversion time (by conversion time we mean the precompile, the interpret or the compile time). The efficiency of the program code is measured in terms of:

1. Computer run time.
2. Computer storage required.

As usual, we expect to use any program for some length of time before any changes or updates are necessary. For this reason the minimization of the conversion time is of less or importance and is not considered in this discussion.

The tree method can be used in many variations, and generate program code which takes minimum run time or minimum storage or a desired combination of both. This is done by choosing one of the many decision trees corresponding to a given decision table.

In the rest of this section we shall introduce briefly some of the most advanced techniques. The different techniques using the tree method fall in two categories:

1. Techniques for optimization of standard decision tables.

2. Techniques that require additional information not usually included in standard decision tables, such as apriori knowledge on data distribution, or the amount of code generated for a given condition etc.

PRESS'S [A5] PROCEDURE.

Press's procedure falls into the first category and can be used for hand-coding or as an algorithm for a decision tables processor. The leading idea is to choose, in every level and for every branch in the decision tree, the next condition to be tested, called by Press the "key row" or "key condition", based upon certain criteria. The table is then split by branching as defined in III-1-1.

The actual procedure, as stated by Press, is given below. After every step a brief explanation is also included.

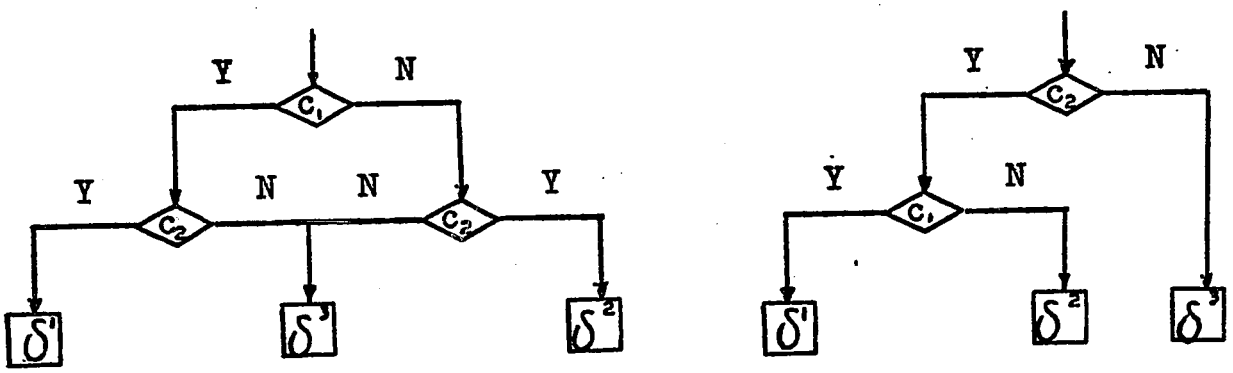
STEP 1: CHOOSE THE KEY CONDITION FROM THE SET OF ROWS (condition entry rows ρ^i) WITH THE MINIMUM NUMBER OF DON'T CARE ENTRIES.

As defined in III-1-1 splitting a table by branching on a given condition c_k that has y "Y" entries, x "N" entries and i don't care entries, results in two sub-tables with q-1 conditions and with x+i and y+i rules respectively. Choosing the key condition as the one with the minimum number of don't care entries obviously leads to smallest sub-table.

STEP 2: IF ONE OF THE ROWS CONTAINS ALL Y'S IN ALL ENTRIES DISCRIMINATE ON THAT ROW.

This can be best seen by the foolowing example.

	1	2	3
c_1	Y	N	ELSE
c_2	Y	Y	



In this example, both rules c_1 and c_2 are qualified to be the key condition by step 1 Choosing c_2 as a key condition, results in a decision tree which requires less storage.

STEP 3: DISCRIMINATE ON THE CONDITION ROW WHICH MAXIMIZES CS_i WHERE

$$CS_i = \sum_{j=1}^{q} \rho_{ij} - \rho_{ii}$$

AND WHERE q EQUALS THE NUMBER OF CONDITIONS IN THE TABLE.

The first two steps of the procedure can be applied by looking at the condition entry rows individually. In the third step of the procedure a look ahead strategy is applied by taking under consideration the relation among the key condition in such a way that that future sub-tables will contain condition entry rows with all Y's or all N's. To understand this look ahead strategy, con-

sider the following:

Divide the entries of each condition entry row ρ^i into two non disjoint sub-set:

1. A positive sub-set including all the Y and don't care entries.
2. A negative sub-set including all the N and don't care entries.

A condition entry row ρ^j is said to be the complement of a condition entry row ρ^i if corresponding to the negative elements and/or the positive element in ρ^i the elements of ρ^j are only Y's or only N's. The number of complementary elements, which is called the count of ρ^i with respect to ρ^j, is denoted by ρ_{ij}.

$$CS_i = \sum_{j=1}^{q} \rho_{ij} - \rho_{ii} = \sum_{\substack{j=1 \\ j\neq i}}^{q} \rho_{ij}$$

$$\rho^i = \{\underbrace{Y \quad Y}_{+\text{'ve elements}} \quad \overbrace{- \quad N}^{-\text{'ve elements}}\}$$

$$\rho^i = [Y \quad Y \quad - \quad N]$$

$$\rho^j = [N \quad N \quad N \quad -]$$

$$\rho_{ij} = 3$$

Thus choosing the row with the largest CS_i may lead at a future splitting to a subtable with a condition entry row with all Y's or all N's. This procedure does not produce necessarily the optimum code possible but gives a way of converting decision tables into efficient program in terms of run time and storage required.

It should be noted that the above procedure is good only for LCE non ambiguous decision tables. In this procedure no means were provided for testing for ambiguity or for completeness.

POLLACK'S [A1] PROCEDURES.

Two procedures were developed by S.L. POLLACK. One which minimizes the required computer storage space and the other which minimizes computer run time. The former falls into first category, while the second falls into the second category which requires information not usually included in a decision table.

POLLACK'S FIRST PROCEDURE: MINIMUM COMPUTER STORAGE REQUIRED

This procedure is based (as in Press) upon a selection criteria that chooses, in every level and for every branch in the decision tree, the next condition or condition entry row upon which the table or the sub-table is split by branching (see definition III-1-1). The selected condition is called by Pollack the "K condition" or the "K row". Pollack does not prove that the procedure actually leads to an optimum decision tree in terms of minimum computer storage required.

The procedure as stated by Pollack is given below. After every step whenever necessary a brief explanation is given.

STEP 1: CHECK FOR REDUNDANCY OR CONTRADICTION. IF, AT ANY STAGE, A PAIR OF RULES DOES NOT CONTAIN AT LEAST ONE Y,N PAIR IN ANY OF ITS ROWS, REDUNDANCY OR CONTRADICTION EXISTS.

Note: Pollack suggests performing the test for redundancy or contradiction when the table is reduced to sub-tables with one

condition.

STEP 2: DETERMINE THOSE ROWS THAT HAVE A MINIMUM DASH-COUNT.

A don't care entry that appears in a rule (i.e. column) containing r don't care entries is counted as 2^r dashes. For each rule, we denoted the 2^r as the column-count. In each row, the sum of the column-counts corresponding to the don't care entries in that row is denoted as dash-count. A row's dash-count is the sum of the column-count of those rules that have don't care entries in the row.

STEP 3: IF TWO OR MORE ROWS HAVE A MINIMUM DASH-COUNT, SELECT THE ROW THAT HAS THE MAXIMUM DELTA.

Delta is the absolute value of the difference of the Y-count and the N-count. The Y-count is the sum of the column-counts corresponding to the Y's in the row. The N-count is similar.

STEP 4: DISCRIMINATE ON THE CONDITION IN ROW K; CALL IT c_k. THIS DISCRIMINATION HAS BRANCHES, EACH OF WHICH LEADS TO A SUB-TABLE CONTAINING ONE MORE RULES, WITH ONE ROW LESS THAN THE ORIGINAL TABLE (ROW K IS DELETED).

The K row is the selected row. The selection is based on criteria described in the first three steps. Step 4 simply splits the decision table on the selected condition K by branching.

STEP 5: IF THE BRANCH LEADS TO A SUB-TABLE CONTAINING MORE THAN ONE RULE GO BACK TO STEP 1.

Step 5 simply states that the splitting by branching is repeated until the sub-tables contain only one rule.

STEP 6a: IF A BRANCH LEADS TO A SUB-TABLE CONTAINING EXACTLY ONE RULE, AND IF THAT RULE CONTAINS ALL DON'T CARE ENTRIES REPLACE THE SUB-TABLE WITH THE RULE ITSELF.

STEP 6b: IF A BRANCH LEADS TO A SUB-TABLE WITH ONE RULE, AND THAT RULE CONTAINS ONE OR MORE DON'T CARE ENTRIES BUT DOES NOT CONTAIN ALL DON'T CARE ENTRIES, CHOOSE AS ROW K ANY ROW THAT HAS NO DON'T CARE ENTRIES.

The selected branch will indicate a sub-table with one less row in it. The opposing branch of the selected row will be indicated as an ELSE-RULE.

STEP 6c: IF A BRANCH DOES NOT LEAD TO A SUB-TABLE, IT LEADS TO AN ELSE-RULE.

STEP 6d: IF THE BRANCH LEADS TO A SUBTABLE CONTAINING ONLY ONE RULE AND THAT RULE CONTAINS ONLY ONE CONDITION WHOSE VALUE IS Y (OR N), THEN ONE BRANCH OF THE DISCRIMINATION ON THAT CONDITION LEADS TO THE RULE, THE OTHER BRANCH LEADS TO THE ELSE-RULE.

POLLACK'S SECOND PROCEDURE: MINIMUM COMPUTER RUN TIME.

In his second procedure, Pollack uses the same technique of splitting by branching but the criteria for choosing the K condition upon which the splitting is done by introducing a weighted

dash count. This procedure can be used if and only if the following restrictions and conditions are satisfied.

1. The table is complete, whether by an implicit ELSE-RULE, or an explicit ELSE-RULE.
2. The average probability to satisfy a rule can be provided for every rule.
3. Only a small percentage of the transactions will satisfy the else rule.

This procedure falls into the second category, for which additional information is needed. The actual procedure, as stated by Pollack, is given below and explanations are added whenever needed.

STEP 1: SAME AS FOR THE FIRST PROCEDURE.

STEP 2: DETERMINE THOSE ROWS THAT HAVE MINIMUM WEIGHTED DASH COUNT.

If we denote by (p^i) the frequency of satisfying the i^{th} rule the weighted dash-count (WDC) for any row is the sum of the products $[p^i x$ (column count)] for those column that contain a dash in that row.

STEP 3: IF TWO OR MORE ROWS HAVE A MINIMUM WDC, SELECT, FROM AMONG THEM, THE ROW THAT HAS THE MINIMUM DELTA. IF, AMONG THESE, THERE STILL EXISTS TWO OR MORE ROWS, SELECT THE ROW WITH THE MINIMUM DASH-COUNT. IF THERE ARE MORE THAN TWO SUCH ROWS, SELECT ANY ONE OF THEM.

STEP 4,5, and 6: Same as the first procedure.

The procedure imposes the following three restrictions :

1. The decision table must be unambiguous.
2. The decision table must be complete without including an explicit else rule.
3. There is complete freedom in order in which the conditions in the decision table are being tested.

Furthermore it requires the folloing additional information:

$P_i = P(\gamma^i)$ the probability distribution of the rules.

$C_i = C(c_i)$ the cost function for testing the i^{th} condition expressed in terms of run time.

The rules are partitioned by actions.

At the the begining of this section we found that the number of decision trees corresponding to a given decision table is $\prod_{K=1}^{q} (K!)^{q-K}$. Every decision tree is refered to as a Sequential Testing procedure (STP). The criterion of efficiency employed is "expected processing time". The expected processing cost for an STP is the weighted sum of the path costs where the weights are the respective path probabilities. The procedure described [A6] will find the minimum expected cost STP for a given LCE decision table. This procedure uses a similar algorithm like the Branch and Bound algorithm given by Little et al. [D4] for solving the traveling Salesman Problem. The authors have developed an extension which can be used in conversion of decision tables with related conditions.

REMARKS ON THE PROCEDURES DEVELOPED BY L.T. REINWALD AND R.M. SOLAND [A6], [A7]

A detailed description of these two procedures are excluded because of their length, but for the sake of completeness some remarks are presented. L.T. REINWALD and R.M. SOLAND developed two procedures for converting decision tables to optimal computer programs. The first converts decision tables to computer programs with minimum average run time and the second converts to programs with minimum storage required. Both procedures were developed for conversion of LCE-decision tables, and fall into the second category in which additional information is required during conversion time.

FIRST PROCEDURE: MINIMUM RUN TIME [A6]

This procedure is designed to convert decision tables of the form given in Fig. IV-1-4 to program code which on the average will require minimum run time.

	RULE NUMBER						
CONDITIONS	1	2	...	i	...	p	COST
c_1	•	•	...	•	...	•	$C(c_1)$
c_2	•	•		•		•	$C(c_2)$
⋮	⋮	⋮		⋮		⋮	⋮
c_q	•	•		•		•	$C(c_q)$
PROBABILITIES	P_1	P_2	...	P_i	...	P_p	
ACTIONS	δ^1	...		δ^u	...	δ^v	

Fig. IV-1-4.

SECOND PROCEDURE: MINIMUM STORAGE REQUIRED [A7]

This procedure converts decision tables of the form given in Fig. IV-1-5 to program code with minimum storage requirements. The storage required for every individual test must be given.

	RULE NUMBER						
CONDITIONS	1	2	...	i	...	p	S
c_1	•	•	...	•	...	•	$S(c_1)$
c_2	•	•		•		•	$S(c_2)$
⋮	⋮	⋮		⋮		⋮	⋮
c_q	•	•		•		•	$S(c_q)$
ACTIONS	δ^1		...	δ^u	...	δ^v	

Fig. IV-1-5

$S_i = S(c_i)$ is the storage required for i^{th} test.

The rules are partitioned according to the actions.

The algorithm is similar to the Branch and Bound algorithm given by Little et al [D4] and used in the first procedure too.

The author developed a combination of both procedures to give a total minimum cost in terms of minimum average time and storage.

IV-1-3 ADVANTAGES, DISADVANTAGES AND COMPARISON OF TREE METHODS

1. THE ADVANTAGES OF THE TREE METHODS:
   a. Easy to use for CS-LCE and IS-LCE decision tables with and without related conditions.
   b. If used in a pre-compiler the generated code is clear and

easy to follow, (for debugging purposes).

c. Using the last two procedures, whenever the probability function and the storage required per test is known, an optimum program code is generated.

2. THE DISADVANTAGES OF THE TREE METHODS:

a. Not easily adaptable for conversion of ECE or MCE decision tables. Any extention of existing procedures to cover MCE becomes extremely complicated.

b. The method does not include effective means for testing ambiguity and completeness, in conversion and/or run time.

c. If order of the conditions is important no optimization is possible.

d. If the order in which the rules have to be tested is significant the tree method cannot be used efficiently.

e. The last two procedures require the probability function and/or the storage and cost per test as additional information, which in most cases are unknown to the user.

3. COMPARISON.

The comparison study of the different procedures presented in this section will be done by converting two illustrative examples to decision trees using the appropriate procedures as applicable.

Example IV-1-1; Minimum Average Processing Time.

Given the decision table:

	RULE NUMBER								
CONDITIONS	1	2	3	4	5	6	7	8	COST
c_1	Y	Y	N	Y	Y	N	N	N	$C(c_1)=50$
c_2	Y	Y	Y	N	N	Y	N	N	$C(c_2)=68$
c_3	Y	N	N	Y	N	Y	Y	N	$C(c_3)=25$
PROBABILITIES	.00	.15	.20	.05	.25	.20	.00	.15	
ACTIONS	δ^1			δ^2		δ^3			

Fig. IV-1-6

Using the minimization techniques defined in III-2-2 we can obtain the following minimized condition entry matrices (with the appropriate probability and actions):

$$\begin{bmatrix} Y & N & Y & N & N \\ Y & Y & N & - & N \\ - & N & - & Y & N \\ .15 & .20 & .30 & .20 & .15 \\ \delta^1 & \delta^1 & \delta^2 & \delta^3 & \delta^3 \end{bmatrix}$$

(a)

$$\begin{bmatrix} Y & - & Y & N & N \\ Y & Y & N & Y & N \\ Y & N & - & Y & - \\ .00 & .35 & .30 & .20 & .15 \\ \delta^1 & \delta^1 & \delta^2 & \delta^3 & \delta^3 \end{bmatrix}$$

(b)

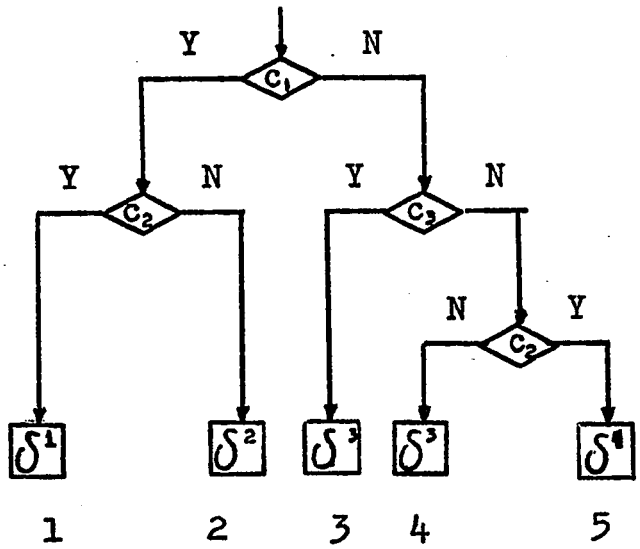
$$\begin{bmatrix} Y & N & Y & N & N \\ Y & Y & N & Y & N \\ - & N & - & Y & - \\ .15 & .20 & .30 & .20 & .15 \\ \delta^1 & \delta^1 & \delta^2 & \delta^3 & \delta^3 \end{bmatrix}$$

(c)

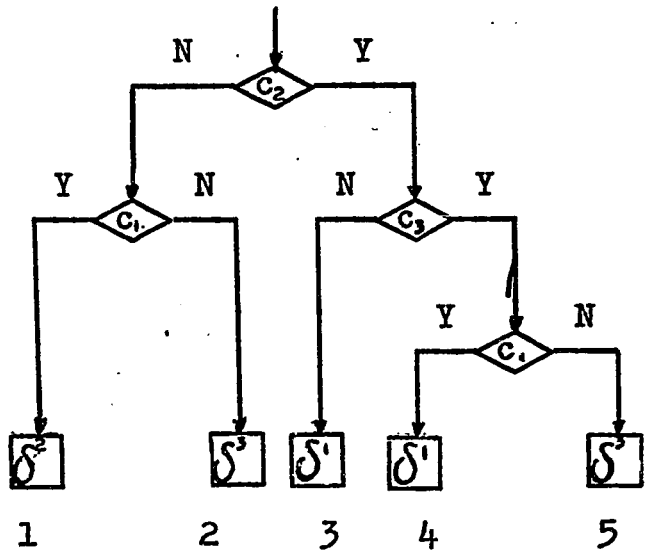
$$\begin{bmatrix} Y & - & Y & N & N \\ Y & Y & N & - & N \\ Y & N & - & Y & N \\ .00 & .35 & .30 & .20 & .15 \\ \delta^1 & \delta^1 & \delta^2 & \delta^3 & \delta^3 \end{bmatrix}$$

(d)

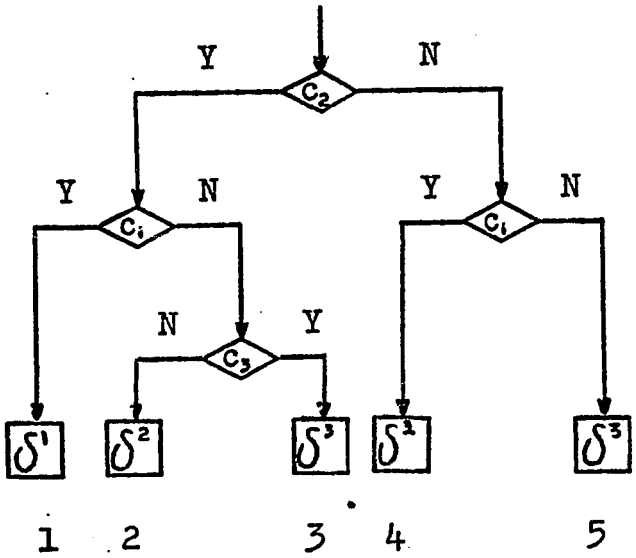
A. To take advantage of Press's procedure, we have to apply it to each of the minimized version of the decision table. The resulting decision trees are given below.



(a)



(b)



(a)

Press's criteria for choosing the condition for splitting do not give a unique result in this case. There are three different possibilities.

(d)

Fig. IV-1-7

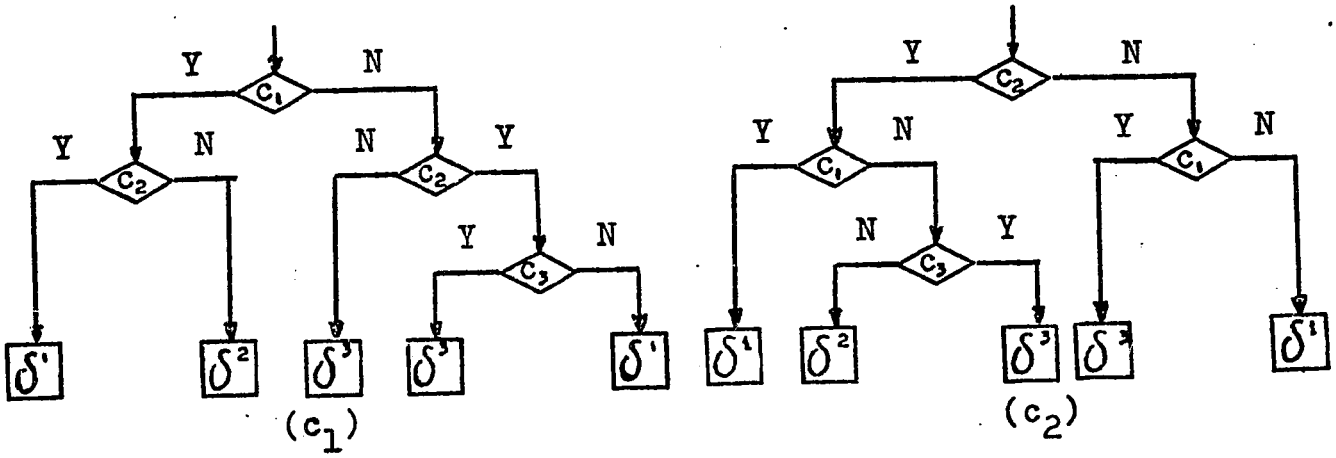
In the following table the expected costs of (a) (b) and (c) are given;

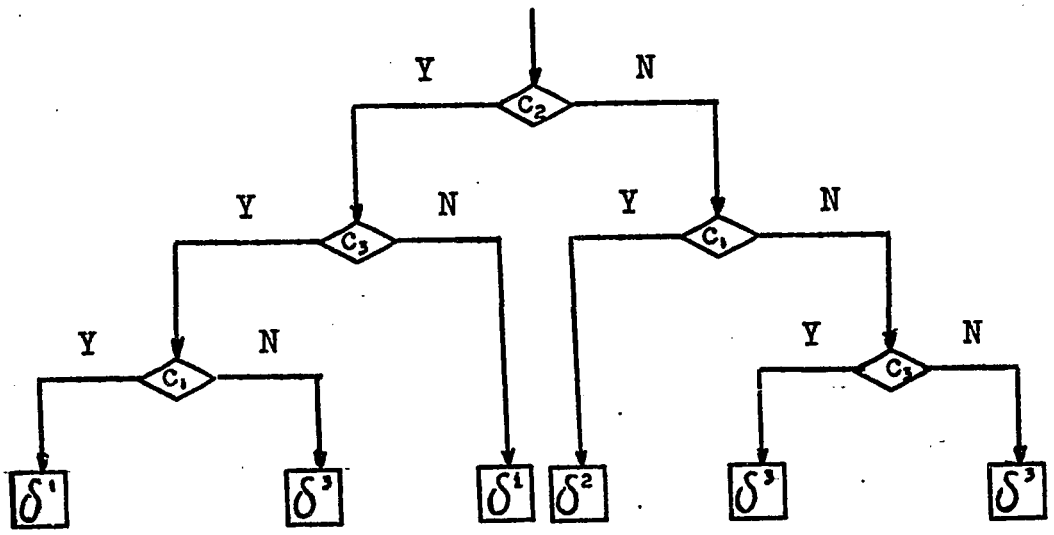
PATH	PROBABILITY			COST			EXPECTED COST		
	(a)	(b)	(c)	(a)	(b)	(c)	(a)	(b)	(c)
1	.15	.30	.15	118	118	118	17.70	35.40	17.70
2	.30	.15	.20	118	118	143	35.40	17.70	23.60
3	.20	.35	.20	75	93	143	15.00	32.55	28.60
4	.15	.00	.30	143	143	118	21.45	00.00	35.40
5	.20	.20	.15	143	143	118	28.60	28.60	17.70
TOTAL EXPECTED COST							118.15	104.25	123.00

TABLE IV-1-1

Note: It should be noted that using Press's procedure, any one of the above decision trees could have been the result, depending on the choice of the minimized form of the decision table.

B. Applying Pollack's second procedure on the minimized versions of the decision table given in Fig. IV-1-6, the resulting decision trees for (a) and (b) are exactly the same as in the case where Press's procedure is used. For (c) we arrive at two possible decision trees as shown below in (c_1) and (c_2). Pollack's procedure gives a unique decision tree shown in (d) below.





(d)

Fig. IV-1-8

In the following table the expected costs of (a),(b),(c_1), (c_2), and (d) are given:

PATH	PROBABILITY					COST					EXPECTED COST				
	a	b	c_1	c_2	d	a	b	c_1	c_2	d	a	b	c_1	c_2	d
1	.15	.30	.15	.15	.00	118	118	118	118	143	17.70	35.40	17.70	17.70	00.00
2	.30	.15	.30	.20	.20	118	118	118	143	143	35.40	17.70	35.40	23.60	28.60
3	.20	.35	.15	.20	.35	75	93	118	143	93	15.00	32.55	17.70	28.60	32.55
4	.15	.00	.20	.30	.30	143	143	143	118	118	21.45	00.00	28.60	35.40	35.40
5	.20	.20	.20	.15	.00	143	143	143	118	143	28.60	28.60	28.60	17.70	00.00
6					.15					143					
TOTAL EXPECTED COST											118.15	104.25	128.00	123.00	118.00

TABLE IV-1-2

Note: It should be noted that in this case too, the resulting decision tree could have been any one of the five different decision trees, depending on the minimized version of the decision table used.

Using the first procedure developed by L.T. Reinwald and R.M.

tree is not unique when using Press's or Pollack's procedure exhausting all possible minimal decision tables and trees, the optimum decision tree can be found. (see tables IV-1-1 and IV-1-2. Both include the minimal expected cost 104.25)

3. Press's procedure does not require any additional information such as probabilities or costs, which is an advantage over the other procedures. The procedure is simple and automatic conversion is relatively simple and efficient. To evaluate the maximum amount of expected overhead cost we shall use:

$$\frac{\text{MAXEC-MINEC}}{\text{MINEC}} \times 100 = \frac{123.00-104.25}{104.25} \times 100 \approx 19\%$$

Where: MAXEC = MAXimum Expected Cost and MINEC = MINImum Expected Cost.

The cost for a straight forward conversion is obviously 143 and the absolutemaximum overhead is $\frac{143.00-104.25}{104.25} \approx 37.2\%$

Taking into account cases in which the costs for the individual tests cannot be supplied, Press's procedure still gives an improvement of about 37.2-19 = 18.2% in the worst case.

If the costs are given, Press's procedure can be used in an exhausting mode in which the expected cost is calculated for all the minimized decision tables. The decision tree with the minimal expected cost is the optimal sequential path resulting from Reinwald and Soland's first procedure.

4. Pollack's second procedure used in this example, makes use of the rule probabilities but does not require or uses the cost for the individual test. In this case also, the optimim deci-

Soland [A6], we obtain directly from the original decision table given in Fig. IV-1-6, the decision tree shown in Fig. IV-1-9(a) and using the modified form of that procedure we obtain decision tree shown in Fig. IV-1-9(b)

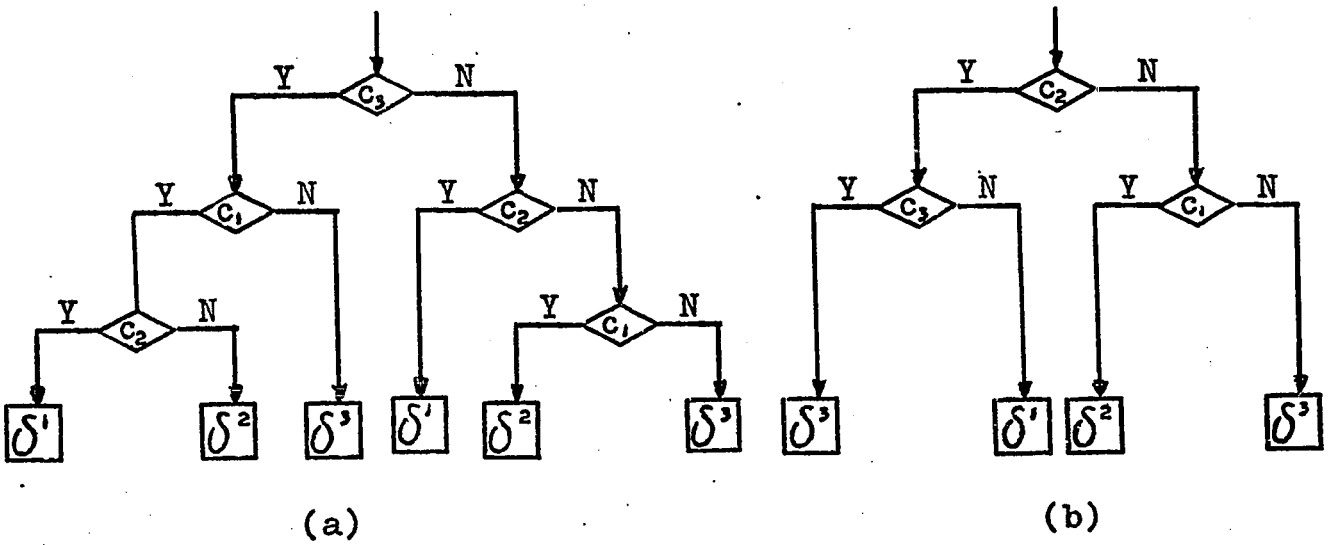


Fig. IV-1-9

Optimal decision tree for the decision table given in Fig. IV-1-6. The expected cost is 111.9.

Optimal decision tree resulting from the modified procedure. The expected cost is 104.25.

At this point we are ready to analyse the results from this example;

1. Neither Press's procedure nor Pollack's could produce a unique, efficient decision tree from the original decision table given in Fig. IV-1-6. Both procedures require some additional steps, namely to minimize the original decision table.

2. Minimization techniques do not lead necessarily to the appropriate minimal decision table and therefore the resulting decision

sion tree can be found (if the costs are supplied in an exhaustive mode. The maximum overhead in this case is

$$\frac{\text{MAXEC-MINEC}}{\text{MINEC}} = \frac{128.00-104.25}{104.25} = 23.8\%$$

Taking into account that the maximum overhead is 37.2% we still save about 37.8-23.8 = 14% in the worst case.

5. The greatest limitation of Reinwald and Soland's first procedure is that the rule probabilities and the costs for individual test must be supplied. On the other hand, this procedure leads to an optimal decision tree without going into a phase of minimization. Note that for using this procedure it is required to complete the table if it is given in an incompletely specified form.

$$\frac{\text{MAXEC-MINEC}}{\text{MINEC}} = \frac{111.9-104.25}{104.25} \approx 7.3\%$$

The modified procedure leads to a decision tree with an absolute minimal expected cost. The only weak point is that branches (or rules) with probability 0 do not appear in the decision tree. If these rules are logically inconsistant, then by omitting them no harm is done. However they are data inconsistent or simply with probability 0, and if data satisfying these rules is submitted by mistake, the result may be unpredictable.

Example IV-1-2: Minimum Storage Requirement.

Given the decision table:

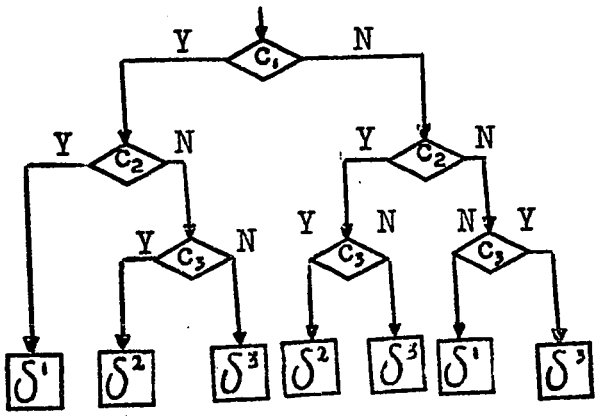
	RULE NUMBERS								
CONDITIONS	1	2	3	4	5	6	7	8	S
c_1	Y	Y	N	Y	N	Y	N	N	$S(c_1)=3$
c_2	Y	Y	N	N	Y	N	Y	N	$S(c_2)=7$
c_3	Y	N	N	Y	Y	N	N	Y	$S(c_3)=8$
ACTIONS	δ^1			δ^2		δ^3			

Fig. IV-1-10

Using again the minimization techniques defined in III-2-2, we arrive at the following simple minimized condition entry matrix (with the appropriate actions)

$$\begin{bmatrix} Y & N & Y & N & Y & N & N \\ Y & N & N & Y & N & Y & N \\ - & N & Y & Y & N & N & Y \end{bmatrix}$$
$$\begin{bmatrix} \delta^1 & \delta^1 & \delta^2 & \delta^2 & \delta^3 & \delta^3 & \delta^3 \end{bmatrix}$$

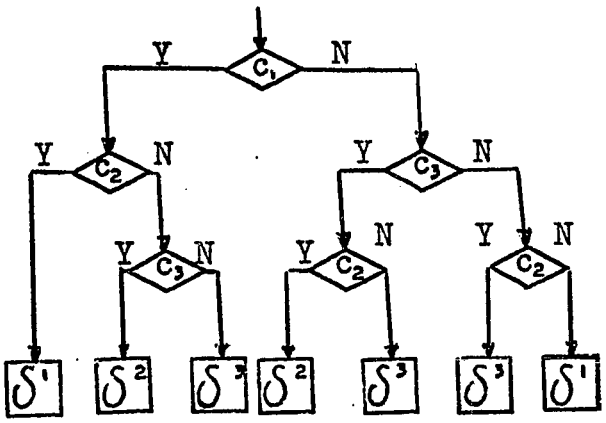
A. Using Press's procedure we obtain four decision trees, as shown in Fig. IV-1-11.



The storage required is

$3 + 2x7 + 3x8 = 41$

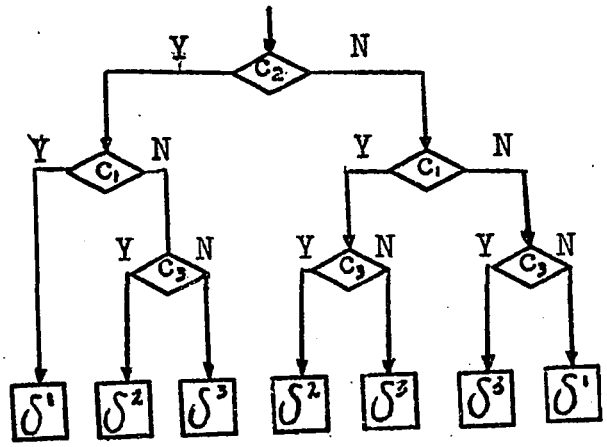
(a)



The storage required is

$3 + 2x8 + 3x7 = 40$

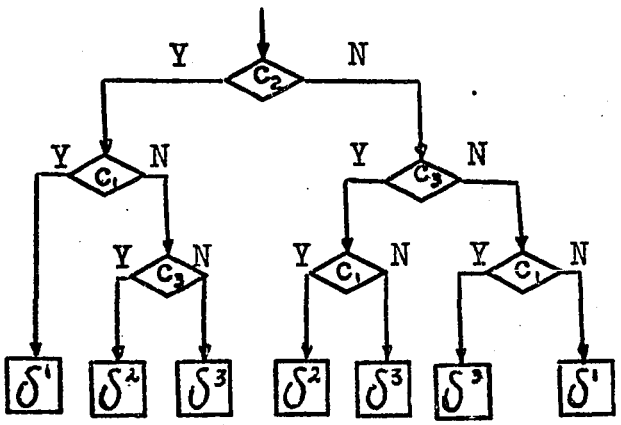
(b)



The storage required is

$7 + 2x3 + 3x8 = 37$

(c)



The storage required is

$7 + 3x3 + 2x8 = 32$

(d)

Fig. IV-1-11

B. using Pollack's first procedure on the minimized decision table, the resulting decision trees are exactly the same as those obtained by Press's procedure.

C. Using Reinwald and Soland's procedure, the resulting tree is unique and is given in Fig. IV-1-11(d).

The results of this example lead to the following conclusion:

1. The resulting decision trees after applying Press's or Pollack's procedures are not unique and depend on the minimized versions of the decision tables. Even if there is only one minimized decision table, the resulting decision tree is not unique. If the storage required for every individual test is known, then by exhausting search the decision tree with minimum storage required can be found. But, in any case, both procedures minimize to some extent the required storage.

2. Reinwald and Soland's second procedure leads to a unique decision tree with minimum storage required. The only weak point is that the individual storage required has to be known for every test, which becomes extremly hard in high language as it is compiler dependent.

3. Reinwald and Soland's procedure is applicable for tree methods only. It is possible to obtain further reduction in the storage requirement by using the mask method.

IV-2 THE MASK METHOD

In this section, a fundamentally different method for conversion of decision tables to computer programs is presented. The mask method can be regarded as a semi-interpretive method. The basic technique as first described by H.W. Kirk [B1] together with some variations, modifications and extensions will be discussed. At the end, a general comparison between the tree method and the mask method is presented.

IV-2-1 THE TECHNIQUE.

The technique, as Kirk [B1] describes it, is based upon the creation of a binary image of a limited entry decision table (we shall see later that this can be extended and used for LCE and MCE decision tables, too), and a binary image vector of the results after testing a given set of conditions for given data. This binary image vector is used then to scan the binary image of the decision

table in order to find which rule is being satisfied and which action has to be taken.

In introducting the basic Mask technique we shall use the following IS-LCE decision table (given in Fig. IV-2-1).

	RULE NUMBER						
CONDITIONS	1	2	3	4	5	6	7
1	Y	Y	Y	N	N	N	E
2	Y	-	-	Y	N	N	L
3	Y	N	Y	-	Y	N	S
4	N	Y	Y	-	-	N	E
ACTIONS	δ^1	δ^2	δ^3	δ^4	δ^5	δ^6	δ^7

Fig. IV-2-1: Illustrative example for presenting the Mask technique

To create a binary image of a given decision table (like the one in Fig.IV-2-1) we have to build two binary matrices: a "Decision Matrix" and a "Mask Matrix."

The Decision Matrix: - Given a LCE decision table, the respective decision matrix, denoted by D, is constructed by the following rule:

$$d_{ij} = \begin{cases} 1 \text{ if } c_{ij} = \text{Y or Yes} \\ 0 \text{ otherwise} \end{cases}$$

The decision matrix which corresponds to the decision table of Fig. IV-2-1 is:

$$D = [d_{ij}] = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 \end{bmatrix}$$

The Mask Matrix: - Given a LCE decision table, the respective mask matrix, denoted by M, is contructed by the following rule:

$$m_{ij} = \begin{cases} 0 \text{ if } c_{ij} = \text{don't care} \\ 1 \text{ otherwise} \end{cases}$$

The mask matrix which corresponds to the decision table of Fig. IV-2-1 is:

$$M = [m_{ij}] = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 1 \end{bmatrix}$$

These two binary matrices include all the information given in a limited condition entry matrix. In order to decide which action is to be taken when certain data is presented, upon testing the different conditions, we build concurrently a result binary vector denoted by R, which is constructed by the following rule:

$$r_i = \begin{cases} 1 \text{ If the result of testing } c_i \text{ is YES} \\ 0 \text{ otherwise} \end{cases}$$

This next operation is a scanning operation in which, using the result vector and the mask matrix , we scan the decision matrix to find which rule is satisfied.

The Scanning operation: - Starting from the first column vector in the mask matrix, we multiply it logically by the result binary vector and test for equality with the corresponding column vector in the decision matrix.

i. If $r_i x m_{ij} = d_{ij}$ for all $i = 1,2,\ldots,q$ then the j^{th} rule γ^j is satisfied and the action to be taken is δ^j.

ii. If $r_i x m_{ij} \neq d_{ij}$ for a least one i, $i = 1,2,\ldots,q$. then the next column vector has to be tested.

iii. If upon completion of the scanning operation, no rule was satisfied, the action corresponding to the "else rule" has to be taken. If no "else rule" is included (and the decision table is not complete) a standard error procedure has to be performed.

In the basic mask technique, as originally presented by Kirk, in order to find which rule is satisfied, we first have to test all the conditions. In addition, p/2 scanning operations, on the average, are also required. Therefore, a straight application of this method as a conversion procedure will give very inefficient program with respect to run time.

On the other hand, testing every condition in a continuous sequence, and recording the result in a binary form, gives a very compact program procedure requiring the least possible amount of storage. Also, no code for branching is required and therefore some run time together with many paragraph names (or labels) are saved (very important especially for computers in which the number of paragraph names or labels per program is limited as in the Burroughs B-5500 computer).

Note that since both matrices and the result vector are binary, very little additional storage is required.

KIRK'S MODIFICATION.

For computers which are character oriented and are lacking bit manipulation operations, Kirk [B1] has suggested the following

modification of the mask technique. In the modified technique, only a modified decision matrix and a modified result vector are used. The decision matrix is created by the following rule:

$$d_{ij} = \begin{cases} Y & \text{if } c_{ij} = Y \text{ or YES} \\ N & \text{if } c_{ij} = N \text{ or NO} \\ D \text{ or "-"} & \text{if } c_{ij} = \text{"-" or don't care.} \end{cases}$$

In other words $d_{ij} = c_{ij}$ or $D = C$ the condition entry matrix as defined. The result vector is created by the following rule:

$$r_i = \begin{cases} Y & \text{if testing the condition } c_i \text{ results in Yes} \\ N & \text{otherwise} \end{cases}$$

The function originally performed by the mask matrix in the basic mask technique, is replaced by moving the result vector to a work area and then only inserting a D or "-" into the corresponding position whenever a D exists in the corresponding column vector in the decision matrix. The resulting modified result vector is denated by $D_{ij}(r_i)$. Only then the comparison is done.

Let us use now the example given in Fig. IV-2-1. The modified decision matrix would be simply the original condition entry matrix C = D.

$$D = C = [r_{ij}] = \begin{bmatrix} Y & Y & Y & N & N & N \\ Y & - & - & Y & N & N \\ Y & N & Y & - & Y & N \\ N & Y & Y & - & - & N \end{bmatrix}$$

And if the result vector is:

$$R = \begin{bmatrix} Y \\ N \\ N \\ Y \end{bmatrix}$$

Then:

$$D_{i1}(r_i) = \begin{bmatrix} Y \\ N \\ N \\ Y \end{bmatrix} \neq \begin{bmatrix} Y \\ Y \\ Y \\ N \end{bmatrix} = [d_{i1}] = [c_{i1}]$$

therefore the data does not satisfy the first rule

$$D_{i2}(r_i) = \begin{bmatrix} Y \\ - \\ N \\ Y \end{bmatrix} = [d_{i2}] = [c_{i2}]$$

the second rule γ^2 is satisfied and therefore the action δ^2 has to be taken.

The mask method, unlike the tree method, can be easily extended and used for ECE and MCE decision tables as well. This extension was done in 1969 by R.Peel [B6] and is given next.

PEEL'S EXTENDED MASK TECHNIQUE

In the extended mask technique introduced by Peel [B6] the decision matrix is modified and some additional vectors containing information pertinent to extended condition are utilized. In order to illustrate extended mask method, the following IS-MCE decision table will be used.

	RULE NUMBER						
CONDITIONS	1	2	3	4	5	6	7
1	A	B	A	C	C	B	ELSE
2	Y	-	N	-	Y	N	
3	20	10	-	-	20	30	
ACTIONS	δ^1	δ^2	δ^3	δ^4	δ^5	δ^6	δ^7

Fig. IV-2-2: Illustrative example for presenting the extended mask technique

In the extended mask technique we first create a row vector, denote by RV, for every condition by the following rule:

$$RV(c_j) = [V_{j1}, V_{j2}, \ldots, V_{jn_j}]$$

For the illustrative example given in Fig. IV-2-2 we have to create two such row vectors for the first and the third conditions which are extended.

$$RV(c_1) = [A,B,C]$$
$$RV(c_3) = [20,10,30]$$

Then the decision matrix is built, by using two different rules, depending on the type of condition. For a limited condition c_i the corresponding row in the decision matrix is built using the rule:

$$d_{ij}(c_i) = \begin{cases} 0 & \text{if } c_{ij} = \text{don't care/"-"} \\ 1 & \text{if } c_{ij} = \text{YES/Y} \\ 2 & \text{if } c_{ij} = \text{NO/N} \end{cases}$$

For an extended condition C_i the corresponding row in the decision matrix is built according to the following rule:

$$d_{ij}(C_i) = \begin{cases} 0 & \text{if } C_{ij} = \text{don't care/"-"} \\ 1 & \text{if } C_{ij} = 1^{st} \text{ element in the appropriate row vector} \\ 2 & \text{if } C_{ij} = 2^{nd} \text{ element in the appropriate row vector} \\ \vdots & \\ \text{etc} \ldots & \end{cases}$$

Using these rules the decision matrix for the decision table given in Fig. IV-2-2 is:

$$D = \begin{bmatrix} 1 & 2 & 1 & 3 & 3 & 2 \\ 1 & 0 & 2 & 0 & 1 & 2 \\ 1 & 2 & 0 & 0 & 1 & 3 \end{bmatrix}$$

The mask matrix is created by the same rule as in the basic mask technique and therefore the corresponding mask matrix is :

$$M = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 & 1 & 1 \end{bmatrix}$$

The result vector contains, as before, a single element for each condition. For limited conditions, the result value after the test is:

$$r_i(c_i) = \begin{cases} 1 & \text{if the result is YES} \\ 2 & \text{if the result is NO} \end{cases}$$

For extended conditions the data is first compared against each element in the appropreate row vector, and the value is assigned according to the following rule.

$$r_i(C_i) = \begin{cases} 0 & \text{if no successful compare results.} \\ 1 & \text{if the first comparison is successful} \\ 2 & \text{if the second comparison is successful} \\ \text{etc} \ldots & \end{cases}$$

The scanning procedure is done in the same way as in the basic mask technique. As shown by Peel, the mask method can be easily adapted and used for all types of decision tables defined in chapter II. This is its big advantage. In the next section we shall analyse the efficiency of the mask technique and introduce an additional variation which will increase slightly the storage required, but increase the efficiency of the technique in run time.

EFFICIENCY CONSIDERATIONS

Here, as in the tree technique, the efficiency of the method

or the technique would be measured in terms of: Conversion time, Run time and Storage requirement.

In addition, the flexibility of the method to include processing of ECE and MCE decision tables, without overcomplicating the techniques and increasing the conversion time significantly, should be also considered as a strong point in measuring the efficiency of the method.

Conversion time can be kept to a minimum due to the simplicity of the method and the different techniques. The modified Kirk technique for example can make use of the original decision table as part of the converted tables. Therefore the conversion time using the mask method, as presented so far, can be classified as low (the exact efficiency must be measured on an individual basis and depends on the skill and ability of the system programmer writing the processor).

Storagewise, the method allows implementation of techniques which will give program code with minimum storage requirements relative to any tree technique which attempls to do the same. As an example if we calculate again the storage requirement for the decision table given in Fig. IV-1-10 we get: $S(c_i) = 3+7+8 = 15$ plus some small additional storage for the different tables and vectors.. Using the best tree technique the minimum storage required is 32.

The only disadvantage of the mask method lies in the fact that the efficiency measured in terms of run time is less than in

any tree technique.

Kirk, himself, noticed this drawback of his method and suggested splitting large decision tables and taking into account, in every partial decision table, the rule probabilities. The run time saved in this way is relatively small and the main advantage of decision tables as a tool for describing complex logical problems in a single tabular form is lost.

P.J.H. King [B2] is the only one who has given serious considerations to the mask technique and suggested a modification which is called by King "The interrupted rule mask procedure".

THE INTERRUPTED RULE MASK PROCEDURE

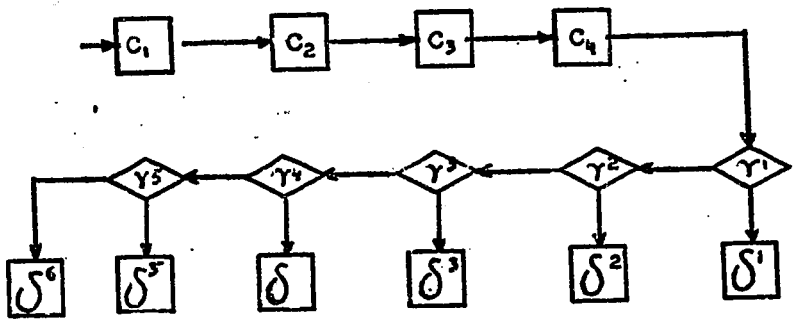
As stated before, the only drawback of the different mask techniques and procedures is the inefficiency in run time. King, in his techniques attempts to improve the run time by taking into account the rule probabilities and the relative cost for evaluating the different conditions.

We shall introduce the interrupted rule mask procedure through the illustrative example given in Fig. IV-2-3. King developed the procedure only for LCE decision tables. For the sake of simplicity we shall also keep this restriction although that procedure can be extended and combined with Peel's procedure for ECE and MCE decision tables.

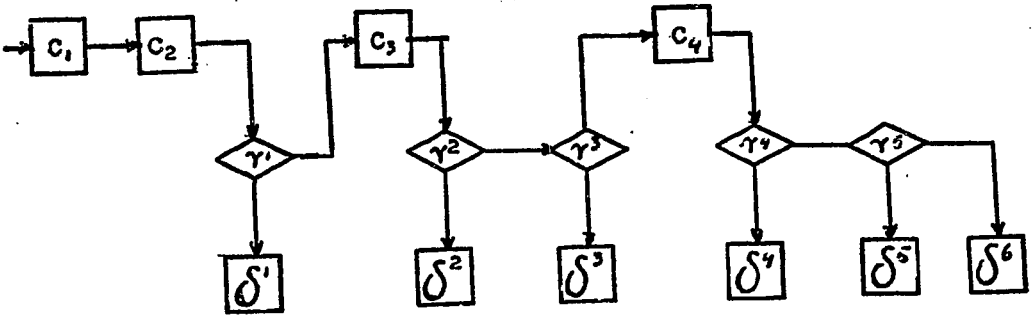
	RULE NUMBER						
CONDITIONS	1	2	3	4	5	6	COST
c_1	Y	Y	-	N	N	E	$C(c_1)=.6$
c_2	N	Y	Y	Y	Y	L	$C(c_2)=.1$
c_3	-	Y	N	Y	Y	S	$C(c_3)=.3$
c_4	-	-	-	N	Y	E	$C(c_4)=.3$
PROBABILITIES	.1	.1	.4	.2	.1	.1	
ACTIONS	δ^1	δ^2	δ^3	δ^4	δ^5	δ^6	

Fig. IV-2-3

If we express by a flow chart the basic mask technique, we get the flow chart presented in Fig. IV-2-4(a). In this flow chart we first test the conditions and then scan the rules.



(a)



(b)

Fig. IV-2-4.

However, it can be easily seen that in order to decide if γ^1 is satisfied, it is enough if we test only first two conditions, and in order to decide whether γ^2 is satisfied, only the first three conditions should be tested. This is expressed in Fig. IV-2-4(b).

The procedure described in IV 2-4(b) is refered to as the interrupted rule mask technique.

As suggessted by King [B2], the order of the tests of the different conditions and scanning of the different rules can be specified in a 2x(q+p) array where q is the number of conditions and p the number of rules (excluding the else rule). For example, the order of operations for the flow chart in Fig. IV-2-4(a) and Fig. IV-2-4(b) is given in Fig. IV-2-5 (a) and (b) respectively.

$$\underset{(a)}{\begin{bmatrix} c_1 & c_2 & c_3 & c_4 & - & - & - & - & - \\ - & - & - & - & \gamma^1 & \gamma^2 & \gamma^3 & \gamma^4 & \gamma^5 \end{bmatrix}} \qquad \underset{(b)}{\begin{bmatrix} c_1 & c_2 & - & c_3 & - & - & c_4 & - & - \\ - & - & \gamma^1 & - & \gamma^2 & \gamma^3 & - & \gamma^4 & \gamma^5 \end{bmatrix}}$$

Fig. IV-2-5

Using the additional information supplied in the decision table given in Fig. II-2-3 we are in a position to calculate the average expected cost in terms of run time. Let denote by Sc the time (or cost) required for scanning a rule. Then the average cost for (a) and (b) are:

$Ta = (.6+.1+.3+.3) + Sc \times .1 + 2Sc \times .1 + 3Sc \times .4 + 4Sc \times .2 +$
$5Sc \times .1 = 1.30 + 2.8Sc$

$Tb = (.6+.1+Sc) \times .1 + (.6+.1+.3+2Sc) \times .1 + (.6+.1+.3+3Sc) \times .4 +$

$$+ (.6+.1+.3+.3+4Sc) \times .2 + (.6+.1+.3\ .3+5Sc) \times .1$$

$$= .96 + 28 \cdot Sc$$

Both expressions contain the constant 2.8Sc which depends on the way the scanning procedure was implemented, and remains a constant, independent of the order in which the different operations are performed. On the other hand, the first portion of the cost is different and is reduced in the interruped rule mask technique.

Let us introduce the following vector notation.

$$P = \begin{bmatrix} P_1 \\ P_2 \\ \vdots \\ P_p \end{bmatrix} \qquad \overline{C} = \begin{bmatrix} C(c_1) \\ C(c_2) \\ \vdots \\ C(c_q) \end{bmatrix} \qquad A = \begin{bmatrix} a_{11} & \cdots & a_{p1} \\ \cdot & & \cdot \\ \vdots & & \vdots \\ a_{1q} & \cdots & a_{pq} \end{bmatrix} \qquad \overline{V} = \begin{bmatrix} 1 \\ 2 \\ \vdots \\ P \end{bmatrix}$$

Where $\overline{P}$ is the vector of probabilities, $\overline{C}$ the cost vector and A is a matrix in which every column vector conditions that have to be tested for the corresponding rule are represented by 1's. Finally lets denote by $\overline{G}$, a vector of the probabilities rearranged in the order in which the rules are tested.

Then:

$$T = \overline{C}^T A\overline{P} + Pc\left\{\sum_{i=1}^{q} C(c_i) + PSc\right\} + Sc\ \overline{G}^T\ \overline{V}$$

The middle term in the above expression is constant, and therefore the procedure with the minimum average cost is the one for which $\overline{C}^T A\overline{P} + Sc\overline{G}^T\ \overline{V}$ is minimum.

King does not suggest any numerical or analytical procedure

to solve this problem (except an exhausting search which is of limited usage in the case of large tables). Instead, he offers four different strategies, in which the additional information of cost and probability is used in a limited mode and do not lead to optimum procedures. More detailed discussion on these techniques will be given in chapter VI.

IV-2-3 ADVANTAGES, DISADVANTAGES AND COMPARISON OF THE MASK METHODS

1. THE ADVANTAGES OF THE MASK METHODS:

a. Easy to use for any kind of decision tables with and without related conditions.

b. If used in a pre-complier, a compiler or an interpreter, the conversion time is very low.

c. The generated program code requires minimum storage space.

d. If the order in which the rules that have to be tested is significant, the mask method is the one to be used.

e. Does not require manual translation from ECE and MCE decision tables to LCE decision tables.

f. The mask method lends itself to efficient testing for ambiguity and completeness in conversion time and/or run time. (See chapter V).

2. THE DISADVANTAGES OF THE MASK METHODS

a. If used in a pre-compiler or a compiler, the generated code can be some times confusing and hard to follow (when debu-

gging is neccessary).

b. Less efficient in terms of run time compared to the tree method.

c. The scanning operation adds a significant factor to the inefficiency in run time.

3. COMPARISON.

At this point, general discussion rather than analytical or numerical comparison is done. This because the complete presentation of the mask method together with different techniques for optimizing will be given in chapter VI.

In general, the mask method can be considered superior to the tree method. The mask method is easier to implement in an automatic translator or compiler, even when used with most complicated decision tables. Using the mask method in an advanced precompiler or interpreter can give us the possibility of skipping the function of the programmer; going from the system analyst straight to the computer, sacrificing, in some cases, more machine time which is becoming cheaper and cheaper. As a semi-interpertive method, it can be applied in time sharing and real time systems.

IV-3 THE COMPUTED GO TO METHOD:

This last method is also a semi-interpretive method and with some modifications it could be used in interpreters for time sharing systems. The method was first introduced by C.G. Veinott [C1] and will be referred to as "Veinott's Method", or "Veinott's Technique".

Veinott's method is based upon the well known computed GO TO in COBOL, FORTRAN and ALGOL.

IV-3-1 THE VEINOTT'S TECHNIQUE

The Veinott's technique differ from the mask technique, in the recording method of the results after testing, and in the scanning function required to find the right action. We shall introduce the technique through an example of Fig IV-2-1. The basic idea in Veinott's technique is to calculate a unique number for every possible combination of conditions, with a single restriction that the resulting number are in an unbroken sequence; such that they can be used as the value of a branching variable.

As a first step we shall assign a weighted value 2^{j-1} to the j^{th} condition and extend the decision table to its completely specified form. The resulting decision table (which is equivalent to the one given in Fig. IV-2-1) is:

	W	RULE NUMBER															
CONDITIONS	VAL.	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
c_1	1	Y	Y	Y	Y	Y	N	N	N	N	N	N	N	Y	Y	Y	N
c_2	2	Y	Y	N	Y	N	Y	Y	Y	Y	N	N	N	N	Y	N	N
c_3	4	Y	N	N	Y	Y	Y	Y	N	N	Y	Y	N	N	N	Y	N
c_4	8	N	Y	Y	Y	Y	Y	N	Y	N	Y	N	N	N	N	N	Y
ACTIONS		δ^1	δ^2	δ^2	δ^3	δ^3	δ^4	δ^4	δ^4	δ^4	δ^5	δ^5	δ^6	δ^7	δ^7	δ^7	δ^7

The Else Rule

Fig. IV-3-1

It should be noted, that in this case, it was relatively easy to derive the completely specified decision table with the correct

number of rules because the original decision table was unambiguous and complete. For decision tables in which apparent ambiguity (in particular, when some or all of the conditions are related) the process of going from a decision table given in IS form to a decision table which is in CS form can be more complicated, ambiguous and cannot be done using an automated procedure. In order to simplify the procedure, avoid ambiguity and keep completeness, it is preferable to arrange the rules in an increasing sequence denoting by "X", wherever the result of a test should be "Y" and leaving the entry blank when "N" should appear. The reformatted decision table is in Fig. IV-3-2

	W	RULE NUMBER															
CONDITIONS	VAL.	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
c_1	1		X		X		X		X		X		X		X		X
c_2	2			X	X			X	X			X	X			X	X
c_3	4					X	X	X	X					X	X	X	X
c_4	8									X	X	X	X	X	X	X	X
ACTIONS		δ^6	δ^7	δ^4	δ^7	δ^5	δ^7	δ^4	δ^1	δ^7	δ^2	δ^4	δ^2	δ^5	δ^3	δ^4	δ^3

Fig. IV-3-2

The technique now uses a result vector R with q entries in which the outcoming results for the respective conditions tests are recorded, "1" if the outcome is yes, and "0" otherwise. The next operation is to compute a single number for the satisfied rule, and the last operation is to branch to the appropriate action to be taken. If we assign labels L1, L2, L3 etc to the different actions the last two operations can be done in COBOL, FORTRAN IV, ALGOL and PL/1 as follows:

In COBOL:

```
COMPUTE JUMP = 1 + R(1) + 2*R(2) + 4*R(3)
GO TO  L6  L7  L4  L7  L5  L7  L4  L1  L7  L2  L4  L2  L5  L3  L4
       L3  DEPENDING ON JUMP.
```

In FORTRAN:

```
JUMP = 1 + R(1) + 2*R(2) + 4*R(3)
GO TO  (L6,L7,L4,L7,L5,L7,L4,L1,L7,L2,L4,L2,L5,L3,L4,L3),JUMP
```

In FORTRAN IV:

```
GO TO  (L6,L7,L4,L7,L5,L7,L4,L1,L7,L2,L4,L2,L5,L3,L4,L3)
       1 + R(1) + 2*R(2) + 4*R(3)
```

In ALGOL or PL/1:

```
SWITCH SW = L6,L7,...
:
JUMP = 1 + R(1) + 2*R(2) + 4*(3);
GO TO SW (JUMP)
or
SWITCH SW = L6,L7,...
:
GO TO SW (1 + R(1) + 2*R(2) + 4*R(3));
```

Using this technique, the problem of ambiguous and incomplete decision tables does not arise in the case of limited condition entry decision tables.

The same technique can be applied for ECE and MCE decision

tables. The technique generates again for every rule, a unique number which is used as a branching variable with the same restriction as for the case of LCE decision tables. Let's assume now that the table contains q conditions and every condition σ_j is associated with the set $V_{ji},\ldots,V_{jn_j}$ of n_j valus. Then the j^{th} position in the result vector R(j) can take the values 0,1,2,... ... ,n_j-1 depending on the outcome of the rest of the j^{th} condition.

Now, the branching variable is calculated according to the following formula:

$$JUMP = 1 + R(1) + n_1*R(2) + n_1*n_2*R(3) + \ldots + R(q)* \prod_{j=1}^{q-1} n_j$$

Everything else remains the same as in the case of LCE decision tables. This technique implies again that the ECE or MCE decision table has to be given in an CS-LCE decision table, excluding the impossible combinations, due to the resulting related limited conditions, after the conversion from MCE to LCE.

The following example given in Fig. IV-3-3 will clarify the utilization of Veinott's technique used for extended condition entry decision tables.

	RULE NUMBER											
CONDITIONS	0	1	2	3	4	5	6	7	8	9	10	11
σ_1	a	a	a	a	b	b	b	b	c	c	c	c
σ_2	1	2	3	4	1	2	3	4	1	2	3	4
ACTIONS	δ^0	δ^1	δ^2	δ^3	δ^4	δ^5	δ^6	δ^7	δ^8	δ^9	δ^{10}	δ^{11}

Fig. IV-3-3

The condition c_1 is associated with the set a,b,c with $n_1 = 3$
The condition c_2 is associated with the set 1,2,3,4 with $n_2 = 4$

The decision table in Fig. IV-3-3 is already given in a completely specified form. In order to make use of the Veinott's procedure for extended condition entry decision tables, the table should be converted into the following form:

	RULE NUMVER												LOCAL	MUTI-	NET
CONDITIONS	0	1	2	3	4	5	6	7	8	9	10	11	VALUE	PLIER	VALUE
$c_1(a)$	X			X			X			X			0		0
$c_1(b)$		X			X			X			X		1	1	1
$c_1(c)$			X			X			X			X	2		2
$c_2(1)$	X	X	X										0		0
$c_2(2)$				X	X	X							1	3	3
$c_2(3)$							X	X	X				2		6
$c_2(4)$										X	X	X	3		9
ACTIONS	δ^0	δ^1	δ^2	δ^3	δ^4	δ^5	δ^6	δ^7	δ^8	δ^9	δ^{10}	δ^{11}			

Fig. IV-3-4

It can be easily seen that the table does not contain all possibilities; only those which are included in the original CS-ECE decision table.

IV-3-2 EFFICIENCY CONSIDERATIONS.

If we limit the technique to CS-LCE-LAE nonambigous and complete decision tables, rather than MCE-MAE decision tables then the conversion time required (manually or automatically) is low. In other words if the decision tables are such that most of the work is done ahead of conversion time then obviously the conversion time is

low. On the other hand if we allow any kind of decision tables up to IS-MCE-MAE then the conversion time will be increased significantly. In some cases in particular when IS-MCE decision tables which include apparent ambiguty bu are incomplete, automated conversion becomes impossible.

The run time of the generated code using this technique obviously is maximum for testing all conditions. In addition some computation time is required for computing the proper rule number and executing a computed GO TO, which is, in general, relatively less than the time required for the scanning function in the mask techniques. In the case of ECE decision table, first they have to be converted to LCE. In run time for every condition which was originally extended c_j, n_j tests are made disregarding the fact that they are dependant.

Obviously, the storage required for the generated code is minimal as in the case of the mask method and may be even less.

Finally, it should be noted that, in terms of efficiency, this method is not recommended.

IV-3-3 ADVANTAGES AND DISADVANTAGES

1. THE ADVANTAGES OF THE VEINOTT'S METHOD
   a. Simple to use as a manual conversion technique.
   b. If used in pre-compiler or interpreter, within the limitations mentioned before the conversion time is relatively small.

c. The generated code requires minimum storage.

2. THE DISADVANTAGES OF THE VEINOTT'S METHOD

a. The method generates inefficient code with respect to run time.

b. Does not allow natural extensions to ECE decision tables in the real sense.

c. Cannot deal with apparent ambiguities.

d. If the order in which the rules have to be tested is significant there is no way to do so using Veinott's method.

e. Uses in every case 2^q branch tables, even in cases where this is unneccessary.

CHAPTER V. TESTING AND DIAGNOSTIC METHODS

This chapter is devoted to one of the most important aspects in the usage of decision tables, namely, testing and diagnostic methods. The efficiency and usefulness of decision tables in EDP (Electronic Data Processing) becomes doubtfull without incorporating advanced and efficient methods and/or procedures for testing decision tables and generating, if necessary, comprehensive diagnostic messages. By testing, we mean testing for ambiguity of any kind and testing for completeness.

The testing of tables for ambiguity and completeness can be done at two different times; prior to conversion to computer programs and/or at run time. If a decision tables processor is used in time sharing or real time in an interpretive mode, the possibility of checking at run time is very desirable feature. Different approaches have been suggested and taken. The most common approach, defined by Pollack [A9], which assumes unrelated conditions in the condition stub, can be implemented in any LCE decision table processor, with relative simplicity, using the definitions for ambiguity given in II-2-2. If testing for completeness is required, one may use, also with relative simplicity, the definition given in II-2-11. Many of the existing decision table processors ignore completely the test for completeness, and that automatically adds the requirement of incorporating the ELSE RULE in every decision table. It should be noted that if definitions II-2-5 and II-2-6 are used as guide

rules for writing testing and diagnostic procedures (even with related conditions) the resulting tables and programs are correct (see the decision tables in Fig. II-2-2 and Fig. II-2-3). These definitions for ambiguity were used in many LCE decision table processors.

In general the testing and diagnostic methods fall into two main categories:

1. Testing methods in conversion time.
2. Testing methods in run time.

V-1 TESTING AND DIAGNOSTIC METHODS IN CONVERSION TIME.

In this section we shall introduce necessary and sufficient conditions for ambiguity in LCE and MCE decision tables which can serve as guide rules in writing and diagnostic procedures in conversion time. The conditions are based on the definitions given in sections II-1 and II-2. We shall start with LCE decision tables with unrelated conditions, extend it to LCE decision tables with related and unrelated conditions. Finally we shall consider MCE decision tables. It should be noted that P.H.J. King was the first to give a serious consideration to this subject and he introduced necessary and sufficient condition for testing LCE decision tables.

Before going any farther let us introduce the following definitions and notations:

1. We denote by d_i the i^{th} column of the decision matrix defined in IV-2-1.
2. We denote by m_j the j^{th} column vector of the mask matrix defined

in IV-2-1.

3. We denote by I a unity vector, a q dimensional column vector in which every component equals 1.

Definition V-1-1: - Given two q dimensional vectors m_i and d_j, the intersection $m_i \cap d_j$ is the row by row logical intersection and the union $m_i \cup d_j$ is the row by row logical union.

Bessed upon definitions II-2-5 and II-2-6, necessary and sufficient conditions for ambiguity are derived in the following theorems.

Theorem V-1-1: - Given a LCE decision table with unrelated conditions, the necessary and sufficient conditions for redundancy between two rules γ^i and γ^j are:

1. $m_i \cap d_j = m_j \cap d_i$

2. $\delta^i = \delta^j$

Where m_i, m_j are the i^{th} and the j^{th} vectors from the mask matrix, and d_i, d_j are the i^{th} and the j^{th} vectors from the decision matrix associated with the given LCE decision table.

Proof: First let us prove that the conditions are necessary. The necessity of the second condition is obvious from the definition of redundancy given in II-2-5.

From the definitions of the LCE decision matrix and the LCE mask matrix, the set of conditions that has to be tested in order to satisfy γ^i, γ^j or both, is given by the $1^{'s}$ in the vector $m_i \cup m_j$.

$$= m_j \cap (m_i \cap d_j) =$$
$$= m_j \cap d_i$$

$$(m_i \cap m_j) \cap d_j = m_i \cap (m_j \cap d_j)$$
$$= m_i \cap d_j$$

if ambiguity exists $m_i \cap d_j = m_j \cap d_i$
and if the ambiguity is redundancy $\delta^i = \delta^j$
This concludes the proof that the conditions are necessary.

To prove that the conditions are sufficient we reverse the steps used in proving the necessity of the conditions. Therefore if the two conditions hold, so does:

$$(m_i \cap m_j) \cap d_i = (m_i \cap m_j) \cap d_j$$

and therefore redundancy exists.

Corollary V-1-1: - If, for a given LCE decision table, two rules γ^i and γ^j are found to satisfy the two conditions stated in theorm V-1-1, then redundancy will occur whenever the data satisfies the conditions represented by the 1's in the column vector, $(m_i \cup m_j) \cap (d_i \cup d_j)$ and every 0 in $(m_i \cup m_j)$ represents an irrelevant or a don't care condition.

This corollary simply states that if two rules γ^i and γ^j both have to be satisfied, the conditions relevant to γ^i or γ^j or both are given by the 1's in the column vector $(m_i \cup m_j)$. This set can be divided into three disjoint sets (as shown in theorem VI-1-1) which, when united, give :

This set of conditions can be divided into three disjoint subsets:

1. Conditions relevant to γ^i but not to γ^j, represented by the 1's in the column vectors $m_i \cap (I-m_j)$, respectively.
2. Conditions relevant to γ^j but not to γ^i, represented by the 1's in the column vector $m_j \cap (I-m_i)$, respectively.
3. Conditions relevant to γ^i and to γ^j, represented by the 1's in the coulmn vector $m_i \cap m_j$, respectively.

To satisfy γ^i, the conditions designated by the 1's in the column vector $m_i \cap (I-m_j) \cap d_i$ have to be satisfied (This is a subset of the set of conditions that has to be satisfied in order to satisfy γ^i. The complete set correspondence to the 1's in the column vector $m_i \cap d_i$). Note that this does not prevent the satisfaction of rule γ^j also. Similarly to satisfy the rule γ^j, the conditions designated by the 1's in the column vector $m_j \cap (I-m_i) \cap d_j$ have to be satisfied. That, again, does not prevent the satisfaction of the rule γ^i also.

Using the third subset $m_i \cap m_j$, in order to satisfy the rule γ^i, the conditions designated by the 1's in the column vector $(m_i \cap m_j) \cap d_i$ have to be satisfied, and to satisfy the rule γ^j, the conditions designated by the 1's in the column vector $(m_i \cap m_j) \cap d_j$ have to be satisfied.

If ambiguity exists:

$$(m_i \cap m_j) \cap d_i = (m_i \cap m_j) \cap d_j$$

but by definition $d_i \subset m_i$ and $d_j \subset m_j$; therefore:

$$(m_i \cap m_j) \cap d_i = (m_j \cap m_i) \cap d_i =$$

$$\{[m_j \cap (I-m_i)] \cap d_j\} \cup \{[m_i \cap (I-m_j)] \cap d_i\} \cup \{[m_j \cap m_i] \cap [d_i \cup d_j]\}$$

$$= (m_i \cup m_j) \cap (d_i \cup d_j)$$

Theorem V-1-2: - Given a LCE decision table with unrelated conditions, the necessary and sufficient conditions for contradiction between two rules γ^i and γ^j are:

1. $m_i \cap d_j = m_j \cap d_i$
2. $\delta^i \neq \delta^j$

Where m_i, m_j, d_i and d_j have the same meaning as in theorem V-1-1. The proof of this theorem is completely analogous to the proof in theorem V-1-1 and thus not included.

Corollary V-1-2: - If, for a given LCE decision table, two rules γ^i and γ^j are found to satisfy the two conditions stated in theorem V-1-2, then contradiction will occur whenever the data satisfies the conditions represented by the 1's in the column vector:

$$(m_i \cup m_j) \cap (d_i \cup d_j).$$

These two theorems can actually serve as guide rules for implementation of an efficient and comprehensive procedures for testing LCE decision tables (with unrelated conditions) for ambiguity.

Actually, the same criteria for testing ambiguity of LCE decision tables with unrelated conditions can be used for testing ambiguty for LCE decision tables with related and unrelated conditions. This is stated in the following theorems.

Theorem V-1-3: - Given an LCE decision table with related and unrelated conditions, the necessary and sufficient conditions for possible redundancy between two rules γ^i and γ^j are:

1. $m_i \cap d_j = m_j \cap d_i$
2. $\delta^i = \delta^j$

Actual redundancy exists if, and only if, there exists consistant data which satisfies the conditions represented by the 1's in the column vector $(m_i \cup m_j) \cap (d_i \cup d_j)$.

The proof of this theorem is completely analogous to the proof of theorem V-1-1 and corollary V-1-1 and therefore not included.

Theorem V-1-4: - Given a LCE decision table with related and unrelated conditions, the necessary and sufficient conditions for possible contradiction between two rules γ^i and γ^j are:

1. $m_i \cap d_j = m_j \cap d_i$
2. $\delta^i \neq \delta^j$

Real contradiction exists if, and only if, there exists consistent data which satisfies the conditions represented by the 1's in the column vector $(m_i \cup m_j) \cap (d_i \cup d_j)$.

The proof of this theorem is again completely analogous to the proof of theorem V-1-1 and corollary V-1-1 and thus not included.

Example V-1-1: - Given the rules $\gamma^i = \begin{bmatrix} Y \\ Y \\ - \\ N \\ - \end{bmatrix}$ and $\gamma^j = \begin{bmatrix} - \\ Y \\ N \\ N \\ - \end{bmatrix}$

the respective decision and mask vectors are:

$$m_i = \begin{bmatrix}1\\1\\0\\1\\0\end{bmatrix} \qquad d_i = \begin{bmatrix}1\\1\\0\\0\\0\end{bmatrix} \qquad m_j = \begin{bmatrix}0\\1\\1\\1\\0\end{bmatrix} \qquad d_j = \begin{bmatrix}0\\1\\0\\0\\0\end{bmatrix}$$

$$m_i \cap d_j = \begin{bmatrix}1\\1\\0\\1\\0\end{bmatrix} \cap \begin{bmatrix}0\\1\\0\\0\\0\end{bmatrix} = \begin{bmatrix}0\\1\\0\\0\\0\end{bmatrix} = \begin{bmatrix}0\\1\\1\\1\\0\end{bmatrix} \cap \begin{bmatrix}1\\1\\0\\0\\0\end{bmatrix} = m_j \cap d_i$$

$\delta^i = \delta^j$; then:

If the decision table contains unrelated conditions, a definite redundancy exists. (Theorem V-1-1)

If the decision table contains related (and unrelated) conditions (as well), a possible redundancy exists. (Theorem V-1-3)

$\delta^i \neq \delta^j$; then:

If the decision table contains unrelated conditions, a definite contradiction exists. (Theorem V-1-2)

If the decision table contains related (and unrelated) conditions (as well), a possible redundancy exists. (theorem V-1-4)

Based upon corollaries V-1-1, V-1-2, and the last condition theorems V-1-3 and V-1-4, the ambiguity will occur with the following result:

$$(m_i \cup m_j) \cap (d_i \cup d_j) = \left\{\begin{bmatrix}1\\1\\0\\1\\0\end{bmatrix} \cup \begin{bmatrix}0\\1\\1\\1\\0\end{bmatrix}\right\} \cap \left\{\begin{bmatrix}0\\1\\0\\0\\0\end{bmatrix} \cup \begin{bmatrix}1\\1\\0\\0\\0\end{bmatrix}\right\} = \begin{bmatrix}1\\1\\1\\1\\0\end{bmatrix} \cap \begin{bmatrix}1\\1\\0\\0\\0\end{bmatrix} = \begin{bmatrix}1\\1\\0\\0\\0\end{bmatrix} \Rightarrow \begin{bmatrix}Y\\Y\\N\\N\\N\end{bmatrix} = R$$

; can be easily seen that $R \epsilon \{W^i\}$ as well as $R \epsilon \{W^j\}$.

The last four theorems can be extended, with some modification

and definition, to cover the MCE decision table. In order to introduce these extensions we shall define the following two operations.

Definition V-1-2: - Given a mask vector m_i and a decision vector d_j of the same order, the multiplication $m_i x d_j$ is a vector whose components are the respective row by row multiplication

$$R_{ik} = m_{ik} \times d_{jk} \; ; \; k = 1,2,\ldots ,q$$

This operation has the following property $(m_i \cap m_j) \times d_j = m_i \times (m_j \times d_j)$

Definition V-1-3: - The operator $\overset{*}{U}$ has the following meanings:

$$d_{ik} \overset{*}{U} d_{jk} = \begin{cases} d_{ik} & \text{if } d_{ik} = d_{jk} \\ 0 & \text{if } d_{ik} = 0 \text{ and/or } d_{jk} = 0 \\ \text{not defined otherwise.} & \end{cases}$$

the operator $\overset{*}{U}$ between two vectors d_i and d_j is defined as if it is between every one of the corresponding components.

Theorem V-1-5: - Given an MCE decision table with unrelated conditions, these necessary and sufficient conditions for redundancy between two rules γ^i and γ^j are:

1. $m_i \times d_j = m_j \times d_i$
2. $\delta^i = \delta^j$

where m_i, m_j are the i^{th} and the j^{th} vectors from the MCE mask matrix and d_i, d_j are the i^{th} and the j^{th} vectors from the MCE decision matrix associated with a given MCE decision table.

Proof: First let us prove that the conditions are necessary. The necessity of the second condition is obvious from the definition of redundancy for MCE decision tables.

From the definitions of the MCE decision matrix and the MCE mask matrix the set of conditions that has to be tested in order to satisfy γ^i, γ^j or both is given by the 1's in the vector $m_i \cup m_j$. This set of conditions can be divided into three disjoint subsets:

1. Conditions relevant to γ^i but not to γ^j, represented by the 1's in the column vector $m_i \cap (I-m_j)$ respectively.
2. Conditions relevant to γ^j but not to γ^i, represented by the 1's in the column vector $m_j \cap (I-m_i)$ respectively.
3. Conditions relevant to γ^i and to γ^j represented by the column vector $m_i \cap m_j$ respectively.

To satisfy γ^i, the conditions designated by the non-zero entries in the column vector $[m_i \cap (I-m_j)] \times d_i$ have to be satisfied (This is a subset of the set of conditions that has to be satisfied in order to satisfy γ^i. The complete set corresponds to the non-zero entries in the column vector $m_i \times d_i$). Note that this does not prevent the satisfaction of the rule γ^j also. Similarly to satisfy the rule γ^j the conditions designated by the non-zero entries in the column vector $[m_j \cap (I-m_i)] \times d_j$ have to be satisfied. That, again, does not prevent the satisfaction of the rule γ^i also.

From the third subset $m_i \cap m_j$, in order to satisfy the rule γ^i, the conditions designated by the mon-zero entries in the column vector $(m_i \cap m_j) \times d_i$ have to be satisfied, and to satisfy the rule γ^j the conditions designated by the non-zero entries in the column vector $(m_i \cap m_j) \times d_j$ have to be satisfied.

If ambiguity exists

$$(m_i \cap m_j) \times d_i = (m_i \cap m_j) \times d_j$$

But the set of the non-zero entries in d_j is a proper subset of the non-zero entries in m_j, and the set of the non-zero entries in d_i is a proper subset of the non-zero entries in m_i therefore

$$\begin{aligned}(m_i \cap m_j) \times d_i &= (m_j \cap m_i) \times d_i \\ &= m_j \times (m_i \times d_i) \\ &= m_j \times d_i\end{aligned}$$

$$\begin{aligned}(m_i \cap m_j) \times d_j &= m_i \times (m_j \times d_j) \\ &= m_i \times d_j\end{aligned}$$

If ambiguity exists $m_i \times d_j = m_j \times d_i$

and if the ambiguity is redundancy $\delta^i = \delta^j$

This concludes the proof that the conditions are necessary.

To prove that the conditions are sufficient we reverse the steps used in proving the necessity of the conditions. Therefore if the two conditions hold so does:

$$(m_i \cap m_j) \times d_i = (m_i \cap m_j) \times d_j$$

and therefore redundancy exists.

Corollary V-1-3: - If, for a given MCE decision table, two rules γ^i and γ^j satisfy the conditions stated in theorem V-1-5; then redundancy will occur whenever the conditions represented by the vector:

$(m_i U m_j) \times (d_i \overset{*}{U} d_j)$ are satisfied.

The meaning of $d_i \overset{*}{U} d_j$ is as previously defined in definition V-1-3.

Proof: This corollary states that if two rules γ^i and γ^j both have to be satisfied, the conditions relevant only to γ^i have to be satisfied, the conditions relevant only to γ^j have to be satisfied and the conditions relevant to both γ^i and γ^j have to be satisfied by the same values respectively. The set of all the conditions that have to be satisfied can be divided into three disjoint sets (as shown in theorem VI-1-5) which when united give:

$$\{[m_j \cap (I-m_i)] \times d_j\} \overset{*}{U} \{[m_i \cap (I-m_j)] \times d_i\} \overset{*}{U} \{[m_j \cap m_i] \times [d_i \overset{*}{U} d_j]\}$$
$$= (m_i U m_j) \times (d_i \overset{*}{U} d_j)$$

Note that if the conditions of theorem V-1-5 hold $d_i \overset{*}{U} d_j$ is always defined and the operation $\overset{*}{U}$ is valid and can be applied.

Theorem V-1-6: - Given an MCE decision table with unrelated conditions, the necessary and sufficient conditions for contradiction between two rules γ^i and γ^j are:

1. $m_i \times d_j = m_j \times d_i$
2. $\delta^i \neq \delta^j$

Proof: The proof of this theorem is completely analogous to the proof of theorem V-1-5 and thus omitted.

Corollary V-1-4: - If, for a given MCE decision table two rules γ^i and γ^j satisfy the conditions stated in theorem V-1-6; then contradiction will occur whenever the data satisfies the conditions represented by the vector:

$$(m_i U m_j) \times (d_i \overset{*}{U} d_j).$$

Proof: The proof of this corollary is completely analogous to the proof of corollary V-1-3 and thus not included.

Theorem V-1-7: - Given a MCE decision table with related and unrelated conditions, the necessary and sufficient conditions for possible redundancy between two rules γ^i and γ^j are:

1. $m_i \times d_j = m_j \times d_i$
2. $\delta^i = \delta^j$

Real redundancy exists if and only if there exists consistent data which satisfy the conditions represented by the vector:

$$(m_i \overset{*}{U} m_j) \times (d_i U d_j).$$

Proof: The proof of this theorem is completely analogous to the proof of theorem V-1-5 and corollary V-1-3 and therefore not included.

Theorem V-1-8: - Given a MCE decision table with related and unrelated conditions, the necessary and sufficient conditions for

possible contradiction between two rules γ^i and γ^j are:

1. $m_i \times d_j = m_j \times d_i$
2. $\delta^i \neq \delta^j$.

Real contradiction exists if and only if, there exists consistent data which satisfy the conditions represented by the vector:

$$(m_i U m_j) \times (d_i \overset{*}{U} d_j).$$

Proof: The proof of this theorem is completely analogous to the proof of theorem V-1-5 and corollary V-1-3 and therefore not included.

Example V-1-2: - Given the foolowing MCE decision table:

	RULE NUMBER							
CONDITIONS	1	2	3	4	5	6	7	8
c_1	27	27	30	35	35	35	40	ELSE
c_2	Y	-	-	Y	-	Y	-	
c_3	-	-	-	=	=	<	>	
ACTIONS	δ^1	δ^2	δ^3	δ^4	δ^4	δ^5	δ^6	δ^7

Fig. V-1-1

$V_1 = [27, 30, 35, 40]$

$V_2 = [=, <, >]$

$$D = \begin{bmatrix} 1 & 1 & 2 & 3 & 3 & 3 & 4 \\ 1 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 2 & 3 \end{bmatrix} \qquad M = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{bmatrix}$$

$$m_1 \times d_2 = \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix} \times \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \times \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix} = m_2 \times d_1$$

Since γ^1 and γ^2 stand for two different actions, and the actual conditions are not explicitely stated, a possible contradiction exists. If there exists consistent data which satisfy the conditions represented by the following vector, real contradiction exists

$$(m_1 U m_2) x (d_1 \overset{*}{U} d_2) = \left\{\begin{bmatrix}1\\1\\0\end{bmatrix} U \begin{bmatrix}1\\0\\0\end{bmatrix}\right\} x \left\{\begin{bmatrix}1\\1\\0\end{bmatrix} \overset{*}{U} \begin{bmatrix}1\\0\\0\end{bmatrix}\right\} = \begin{bmatrix}1\\1\\0\end{bmatrix} x \begin{bmatrix}1\\1\\0\end{bmatrix} = \begin{bmatrix}1\\1\\0\end{bmatrix} \Rightarrow \begin{bmatrix}27\\Y\\-\end{bmatrix}$$

It can easily be verified that data which satisfies [27,Y,-] satisfies both γ^1 and γ^2.

Testing for the 4^{th} and the 5^{th} rules we get:

$$m_4 \; x \; d_5 = \begin{bmatrix}1\\1\\1\end{bmatrix} x \begin{bmatrix}3\\0\\1\end{bmatrix} = \begin{bmatrix}3\\0\\1\end{bmatrix} = \begin{bmatrix}1\\0\\1\end{bmatrix} x \begin{bmatrix}3\\1\\1\end{bmatrix} = m_5 \; x \; d_4$$

And since γ^4 and γ^5 have identical actions, a possible redundancy for data which satisfies the conditions represented by the following vector, exists:

$$(m_4 U m_5) x (d_4 \overset{*}{U} d_5) = \left\{\begin{bmatrix}1\\1\\1\end{bmatrix} U \begin{bmatrix}1\\0\\1\end{bmatrix}\right\} x \left\{\begin{bmatrix}3\\1\\1\end{bmatrix} \overset{*}{U} \begin{bmatrix}3\\0\\1\end{bmatrix}\right\} = \begin{bmatrix}1\\1\\1\end{bmatrix} x \begin{bmatrix}3\\1\\1\end{bmatrix} = \begin{bmatrix}3\\1\\1\end{bmatrix} \Rightarrow \begin{bmatrix}35\\Y\\=\end{bmatrix}$$

Data which satisfies [35,Y,=] satisfies both γ^4 and γ^5.

Conclusions: Any one of the previous theorems can serve as a guide rule in implementing testing procedures for any kind of decision tables, starting with LCE decision tables with unrelated conditions and going up to MCE decision tables with related and unrelated conditions.

Since in every case, real or apparent ambiguity should result in the generation and printing of the appropriate messages, and since the most flexible decision tables for actual use are the MCE decision tables with related and unrelated conditions, the use of theorem V-1-8 is recommended. A good decision table processor should not stop the conversion if possible ambiguity is found. It should continue the conversion generating the appropriate messages and diagnostics, leaving the final decision to the systems analyst. If, after reviewing the generated messages, actual ambiguity is found then an appropriate correction and an additional conversion pass should be made.

The testing procedures which can be designed, using the former theorems, are most adaptable to the mask method, in which the same mask and decision matrices can be used, The same procedures can be adapted for testing, even if the tree methods are used; this at the expense of some additional storage and conversion time.

V-2 TESTING AND DIAGNOSTIC METHODS IN RUN TIME

In the last section, we delt with possible ambiguity, because in actual use of decision tables, we use MCE decision tables with related and unrelated conditions. We were only able to comment on the data (if any) for which ambiguity could occur. This, because at conversion time, we are not able to judge whether or not this type of data is consistant and/or possible. The big advantage of the previously discussed testing and diagnostic methods, is the fact

that at conversion time, they can be used with the mask method as well as with the tree method.

The main problem is that a possible ambiguity cannot always be identified, by the user, as a real ambiguity or not. A simple example, given in Fig. V-2-1 shows that redundancy exists between the rules γ^1 and γ^2, and possible contradiction exists between the rules γ^3 and γ^4. But whether the ambiguity contained in this table is real or not it is sometimes quite difficult to decide.

	RULE NUMBER				
CON.	1	2	3	4	5
A<B	Y	-	Y	-	ELSE
A>C	-	Y	-	Y	
A=D	N	N	Y	Y	
ACT.	δ'	δ'	δ^2	δ^2	δ^3

Fig. V-2-1

The present section will deal with testing and diagnostic methods at run time. These procedures were first published in an algorithmic form by C.R. Muthukrishnan and V. Rajaraman [B7] and can serve as a complementary way to the previously suggested methods to solve completely the problem of ambiguity in decision table logic. Note that these algorithms can be used with masked method only.

The following two algorithms can serve as conversion techniques as well as testing and diagnostic techniques.

ALGORITHM V-2-1: - For converting and testing LCE decision table.

STEP 1: GENERATE A MASK MATRIX M BY SUBSTITUTING THE FOLLOWING CODES FOR THE CONDITION ENTRIES IN THE CONDITION ENTRY MATRIX.

$$M_{ij} = \begin{cases} 1 \\ 0 & \text{if } c_{ij} = Y \\ 0 \\ 1 & \text{if } c_{ij} = N \\ 1 \\ 1 & \text{if } c_{ij} = \text{"-" or don't care.} \end{cases}$$

For example, the corresponding mask matrix for the table given in Fig V-2-1 is:

$$M = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 \end{bmatrix}$$

STEP 2: FOR GIVEN DATA, THE CONDITIONS ARE TESTED AND THE RESULTS ARE RECORDED IN A RESULT VECTOR D USING THE SAME CODES AS IN STEP 1.

STEP 3: SELECT THE ROWS OF M WHICH CORRESPOND TO THE 1 ENTRIES IN D, AND LOGICALLY "AND" THE RESULTING ELEMENTS IN A GIVEN POSITION TO A RESULT ROW VECTOR R OF ORDER p.

STEP 4: IF $R_j = 0 \quad j = 1,2,\ldots,p.$ THE "ELSE RULE" APPLIES. IF $R_k = 1$ for a unique k, THEN γ^k RULE APPLIES. IF MORE THAN ONE ELEMENT is equal to 1, A REAL AMBIGUITY BETWEEN THE CORRESPONDING RULES EXISTS; REDUNDANCY IF THE CORRESPONDING ACTIONS ARE THE SAME AND CONTRADICTION IF THE CORRESPONDING ACTIONS ARE DIFFERENT.

Example V-2-1: - Using the decision table given in Fig. V-2-1;

1. Suppose c_1 is false c_2 true and c_3 is false; then:

$$D = \begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

The corresponding rows in M are

$$\begin{array}{cccc} 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 \\ \underline{1} & \underline{1} & \underline{0} & \underline{0} \\ 0 & 1 & 0 & 0 \end{array}$$

$$R = [0\ 1\ 0\ 0]$$

which points to the fact that γ^2 is satisfied and there is no ambiguity.

2. If c_1 is true, c_2 is true and c_3 is false; then:

$$D = \begin{bmatrix} 1 \\ 0 \\ 1 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

the corresponding rows in M are

$$\begin{array}{cccc} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ \underline{1} & \underline{1} & \underline{0} & \underline{0} \\ 1 & 1 & 0 & 0 \end{array}$$

$$R = [1\ 1\ 0\ 0]$$

which points out that ambiguity between γ^1 and γ^2 exists and since they have the same action, redundancy exists.

3. If c_1 is true, c_2 is true and c_3 is true; then:

$$D = \begin{bmatrix} 1 \\ 0 \\ 1 \\ 0 \\ 1 \\ 0 \end{bmatrix}$$

the corresponding rows in M are

$$\begin{array}{cccc} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ \underline{0} & \underline{0} & \underline{1} & \underline{1} \\ 0 & 0 & 1 & 1 \end{array}$$

$$R = [0\ 0\ 1\ 1]$$

$$D_{ij} = \begin{cases} 1 & \text{if the } i^{th} \text{ condition is satisfied in } \gamma^j \\ 0 & \text{otherwise (including the don't care case)} \end{cases}$$

STEP 3: LET T = D OR M

That is $t_{ij} = d_{ij}$ OR m_{ij}

STEP 4: LOGICALLY "AND" THE RESULTING ELEMENTS OF THE MATRIX T COLUMNWISE.

$$\bigcap_{i=1}^{q} t_{ij} = R_j$$

THE RESULTING ROW VECTOR R HAS 1's IN EVERY POSITION FOR WHICH THE DATA SATISFIES THE RESPECTIVE RULE.

Example V-2-3: - Refering to the MCE decision table given in Fig. V-1-1;

1. Suppose c_1 is 35, c_2 is true and c_3 is "-" then

$$D = \begin{bmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 \end{bmatrix}$$

$$D \text{ or } M = \begin{bmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 \end{bmatrix} \text{ or } \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 \end{bmatrix}$$

$R = [0\ 0\ 0\ 1\ 1\ 0\ 0]$

which points out that ambiguity between γ^4 and γ^5 exists. Since the actions of these two rules are different, contradiction exists. Note that this was discovered in Example V-1-2 as a possible ambiguity.

which points out that ambiguity between γ^3 and γ^4 exists and since they have different actions, the rules contain contradiction.

The proof of the algorithm is based on the fact that in order to satisfy a certain rule the corresponding outcome after the AND operation must be "1". Since the code for a don't care entry in the condition entry matrix is $\binom{1}{1}$ this will not affet result of the AND operation.

ALGORITHM V-2-2: - For Conversion and Testing MCE Decision Tables. The steps of this algorithm will be illustrated, whenever it is felt necessary, by applying them to the MCE decision table given in Fig. V-1-1.

STEP 1: GENERATE A qxp MASK MATRIX M BY SUBSTITUTING THE FOLLOWING CODES FOR THE CONDITION ENTRIES c_{ij} IN THE CONDITION ENTRY MATRIX.

$$m_{ij} = \begin{cases} 0 & \text{if } c_{ij} = Y \text{ or } c_{ij} = N \text{ or } V_{ij} \\ 1 & \text{otherwise} \end{cases}$$

The corresponding M matrix to the decision table given in Fig. V-1-1 is:

$$M = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 \end{bmatrix}$$

STEP 2: FOR GIVEN DATA PERFORM THE TESTS AND GENERATE A qxp MATRIX USING THE FOLLOWING CODE.

2. Suppose $c_1 = 30$, c_2 is false and c_3 is ">"; then:

$$D = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

$$D \text{ or } M = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \text{ or } \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 \end{bmatrix}$$

$$R = [0\ 0\ 1\ 0\ 0\ 0\ 0]$$

This points out that the γ^3 is satisfied.

Note: Conditions that have the same value in the condition entry matrix do not have to be tested more than once. For example, as soon as we found that c_1 is satisfied by 35 we can insert 1 in c_{14}, c_{15} and c_{16}.

Now we shall show that with some small modification the same algorithm can be used for testing at conversion time and at run time.

MODIFICATION FOR ALGORITHM V-2-1:

The only modification that has to be made in order to use this algorithm at conversion time for checking possible ambiguities, is in the 2nd step. Insted of generating a result vector after testing real data, we can generate in a loop all the possible outcomes (2^q) of result vectors and apply them to the mask matrix in the same manner as described in algorithm V-2-1. This will give us another

way to indicate possible ambiguities at conversion time.

MODIFICATION FOR ALGORITHM V-2-2:

Here the modification can be made in the same way as proposed previously for algorithm V-2-1. But here, instesd of generating a possible result vector, we have to generate in a loop, a possible result matrix. In this case the number of possible matrices becomes very large ($\prod_{j=1}^{q} n_j$ - see definition II-2-11) and time consuming. Therefore, for MCE decision tables the use of theorems V-1-7 and V-1-8 is highly recommended.

REMARLS:

1. The two algorithms presented above are again interpretive in nature, and therefore more suited for interpreters to be used in time sharing or real time systems.
2. Both alorithms require testing all the conditions before deciding which rule is being satiafied and whether or not ambiguity of any kind exists. Therefore, a price of longer run time is paid for the advantage of being able to test for ambiguities at run time.
3. Both algorithms can be modified as suggested by King [B2] in his interrupted rule mask procedure, for the Kirk's procedure. But by doing so, we lose the ability of detecting ambiguities in run time since, in this procedure, the first rule satisfied will be taken. On the other hand this provides a way

to save run time.

4. In algorithm V-1-2 we could eliminate the T = (D or M) operation by initiating D to the values of M and save some time by eliminating further testing of conditions for a rule γ^i as the condition entry c_{ij} is found to be unsatisfied.
5. These two algorithms, as the other, discussed in the mask method, in chapter IV, are very efficient as far as the required storage is concerned.

V-3 TESTING AND DIAGNOSTIC METHODS IN CONVERSION TIME AND IN RUN TIME

In this section we shall introduce another two algorithms capable to perform testing in conversion time and run time as well as to serve as conversion algorithms. The use of conversion algorithms such as described in this section will allow conversion together with debugging runs to be run in a continuous sequence giving all the necessary diagnostics for fast and efficient debugging.

ALGORITHM V-2-3: - For conversion of LCE decision tables (with and without related conditions) and testing in conversion time and run time whenever the table contains possible ambiguities. This algorithm is based upon the mask technique given in IV-2-1 and theorems V-1-3, V-1-4.

The steps of the algorithm are divided into two groups, the first froup is executed in conversion time while the second, based upon the first, is executed in run time.

GROUP 1: IN CONVERSION TIME

STEP 1: GENERATE A DECISION MATRIX D BY SUBSTITUTING THE FOLLOWING CODES FOR THE CONDITION ENTRIES IN THE CONDITION ENTRY MATRIX.

$$d_{ij} = \begin{cases} 1 & \text{if } c_{ij} = \text{Y or YES} \\ 0 & \text{otherwise} \end{cases}$$

STEP 2: GENERATE A MASK NATRIX M BY SUBSTITUTING THE FOLLOWING CODES FOR THE CONDITION ENTRIES IN THE CONDITION ENTRY MATRIX.

$$m_{ij} = \begin{cases} 0 & \text{if } c_{ij} = \text{"-" or don't care} \\ 1 & \text{otherwise} \end{cases}$$

STEP 3: CHECK FOR EVERY TWO RULES γ^i, γ^j IF $m_i \cap d_j = m_j \cap d_i$

1. IF $m_i \cap d_j = m_j \cap d_i$ AND IF $\delta^i = \delta^j$ GENERATE A DIAGNOSTIC MESSAGE FOR POSSIBLE REDUNDANCY BETWEEN RULES γ^i AND γ^j FOR DATA WHICH SATISFYS THE CONDITIONS REPRESENTED BY THE 1'S IN THE COLUMN VECTOR $(m_i \cup m_j) \cap (d_i \cup d_j)$. GENERATE A TEST RESULT VECTOR TR IDENTICAL TO $(m_i \cup m_j) \cap (d_i \cup d_j)$ WITH AN ADDITIONAL TAG (i,j,k) WHERE i,j ARE THE RULE NUMBERS AND

$$k = \begin{cases} 0 & \text{FOR REDUNDANCY} \\ 1 & \text{FOR CONTRADICTION} \end{cases}$$

TO BE USED IN RUN TIME. NOTE THAT HERE k = 0. STORE THIS RECORD IN A TABLE.

2. IF $m_i \cap d_j = m_j \cap d_i$ AND $\delta^i \neq \delta^j$ GENERATE A DIAGNOSTIC MESSAGE FOR POSSIBLE CONTRADICTION BETWEEN RULES γ^i

AND γ^j FOR DATA WHICH SATISFYS THE CONDITIONS REPRESENTED BY THE 1'S IN THE COLUMN VECTOR $(m_i \cup m_j) \cap (d_i \cup d_j)$. GENERATE A TEST RESULT VECTOR TR IDENTICAL TO $(m_i \cup m_j) \cap (d_i \cup d_j)$ WITH AN ADDITIONAL TAG (i,j,k) AS FOR 1, HERE k = 1. STORE THAT VECTOR IN A TABLE.

GROUP 2: IN RUN TIME

STEP 4: FOR GIVEN DATA, THE CONDITIONS ARE TESTED AND THE RESULTS ARE RECORDED IN A RESULT VECTOR RV USING THE SAME CODES AS FOR THE DECISION MATRIX.

STEP 5: IF THE TABLE IN WHICH THE TR VECTORS ARE STORED IN CONVERSION TIME IS NOT EMPTY COMPARE THE RESULT VECTOR RV WITH EVERY VECTOR IN THE TABLE. FOR EVERY MATCH GENERATE A RUN TIME DIAGNOSTIC MESSAGE LISTING THE AMBIGUOUS DATA, REDUNDANCY IF k = 0, CONTRADICTION IF k = 1. STATE ALSO THE RULE NUMBER i AND j.

STEP 6: IF THE TEST RESULT VECTOR TABLE IS EMPTY OR NO MATCH OCCUR IN STEP 5 PERFORM THE SCANNING OPERATION (DESCRIBED IN IV-2-1) IN ORDER TO FIND WHICH RULE IS SATISFIED AND TAKE THE APPROPRIATE ACTION.

ALGORITHM V-2-4: - For conversion of MCE decision tables (with and without related conditions) and testing in conversion time and run time whenever the table contains possible ambiguities. This algorithm is based upon Peel's Extended Mask Technique given in page 92 and theorems V-1-7, V-1-8.

The steps of this algorithm are also divided into two groups, the first group is executed in conversion time and the second in run time.

GROUP 1: IN CONVERSION TIME.

STEP 1: GENERATE A DECISION MATRIX D AS IN PEEL'S EXTENDED MASK TECHNIQUE.

STEP 2: GENERATE A MASK MATRIX M AS IN PEEL'S EXTENDED MASK TECHNIQUE.

STEP 3: CHECK FOR EVERY TWO RULES γ^i, γ^j IF $m_i \times d_j = m_j \times d_i$

1. IF $m_i \times d_j = m_j \times d_i$ AND IF $\delta^i = \delta^j$ GENERATE A DIAGNOSTIC MESSAGE FOR POSSIBLE REDUNDANCY BETWEEN RULES γ^i AND γ^j FOR DATA WHICH SATISFIES THE CONDITIONS REPRESENTED BY THE VECTOR $(m_i U\ m_j) \times (d_i \overset{*}{U}\ d_j)$. GENERATE A TEST RESULT VECTOR TR IDENTICAL TO $(m_i\ U\ m_j) \times (d_i \overset{*}{U} d_j)$ WITH AN ADDITIONAL TAG (i,j,k) SAME AS IN STEP 3(1) IN ALGORITHM V-2-3, AND STORE IT IN A TABLE.
2. IF $m_i \times d_j = m_j \times d_i$ AND $\delta^i \neq \delta^i$ GENERATE A MESSAGE FOR POSSIBLE CONTRADICTION BETWEEN RULES γ^i AND γ^j FOR DATA WHICH SATISFIES THE CONDITIONS REPRESENTED BY THE VECTOR $(m_i\ U\ m_j) \times (d_i \overset{*}{U}\ d_j)$. GENERATE A TEST RESULT VECTOR TR SAME AS IN STEP 3(2) IN ALGORITHM V-2-3, AND STORE IT IN A TABLE.

GROUP 2: IN RUN TIME.

STEP 4: FOR GIVEN DATA, THE CONDITIONS ARE TESTED AND THE RESULTS

ARE RECORDED IN A RESULT VECTOR RV USING THE SAME CODES AS FOR THE RESULT VECTOR IN PEEL'S EXTENDED MASK TECHNIQUE.

STEP 5: SAME AS STEP 5 IN ALGORITHM V-1-3.

STEP 6: SAME AS STEP 6 IN ALGORITHM V-1-3.

REMARKS:

1. The main advantage in the last two algorithm is the fact that they incorporate in one; conversion, conversion time testing and run time testing almost without increasing the conversion time or the program run time and storage required compare to Kirk's mask technique or Peel's extended mask technique.
2. The only disadvantage is that Kirk's mask technique as well as Peel's extended mask technique require the testing of every condition, which is inefficient in run time.
3. One way to make use of these last algorithms is by inclusion two additional rules denoted by A (for ambiguity) and R (for redundancy) which when satisfied specify the actions to be taken in either case.

CHAPTER VI. PROPOSED CONVERSION METHOD

In this chapter, an attempt is made to propose an efficient and flexible conversion method based upon the experience accumulated in the previous chapters. We can clearly state that none of the previously suggested methods, procedures and algorithms, for conversion of decision tables to program code and for checking ambiguity and completeness, can be used separately in a real attempt to write a good decision table processor. We shall first state the main requirements of a good processor in order of priority. This list of requirements will be the guiding list and the reason standing behind the suggested appraoch. Finally, we shall give an example in which the conversion will be made manually as if done by a preprocessor converting the tables into COBOL program code.

VI-1 REQUIREMENTS:

1. Minimum requirement limitations and restrictions imposed on the user.
2. Efficient conversion time checking procedures and a comprehensive system of diagnostic messages.
3. Efficient program code with optimal combinations of minimal run time and storage requirements.
4. Possible run time checking procedures for debugging mode runs.
5. Possible inclusion of additional information (such as probability distribution of the rules and storage required) for optimization

of the program code generated.

6. Possible elimination of the programming job.
7. Efficient decision table processor.

The most essential requirement, as stated above, is maximum flexibility. Fulfilling this requirment can eventually lead to possible elimination of the programming job. Using third generation computers, where machine time and memory are becoming cheaper and cheaper, and the price paid for programming becomes more and more expensive, these requirements simple state that the reduction and/or the elimination of the programming job is most desirable even if this means relatively longer conversion time and execution time. We would like to design and implement a decision table translator which can cope with the following functions:

1. Accept MCE decision tables, with and without auxiliary information. Easy to learn and easy to use.
2. Be able to copy standard portions of program code from standard library.
3. Have the necessary interface to data base generator.
4. Generate as efficient program code as possible, without imposing heavy restrictions or requirements upon the user.
5. Have efficient procedure for testing and generating diagnostic messages in conversion time, and upon requirement in run time as well, in a special debugging mode run.
6. Serve as an additional means of documentation.
7. The decision table processor should be written in a modular form

(as far as possible) to allow easy modification by the potential user.

VI-2 SUGGESTED DECISION TABLE COBOL PRE-COMPILER SYNTACTIC STRUCTURE

In this section we shall deal with the formation of rules for writing decision tables utilizing the COBOL language. The COBOL language was chosen in this particular example because this is the most useful language in today's computer applications. Besides a decision table processor written in a high level language is relatively machine independent. As stated before, an actual pre-compiler was not written because of lack of resources, but the presented suggestions can serve as nuclei in implementing one, and were used to generate the example of this chapter.

A COBOL pre-compiler must be able to accept any COBOL program intermixed with decision tables as input, from which an efficient COBOL program will be generated.

The syntax of the precompiler may contain few reserved words and symbols which must not be reserved nor have any specific meaning in any standard COBOL language specifications (COBOL 60, COBOL 61, COBOL 65, USASI and CIDASYL COBOL).

The pre-compiler must be able to accept three types of input data (or statements).

1. CONTROL STATEMENTS.
2. DECISION TABLES.
3. STANDARD COBOL STATEMENTS.

In the rest of the section we will make use of the Bachus normal form (with some modifications) to describe the syntax required by the pre-compiler. The conversions used are:

a. < > left and right broken brackets are used to contain a meta-variable.
b. ::= Means "IS DEFINED AS".
c. | Means "OR"
d. [ ] Left and right brackets are used to indicate "AND"

Note: Some of the following is Burroughs oriented since I am most familiar with the Burroughs computers.

CONTROL STATEMENTS

Control statements can be supplied to the precompiler by means of punch cards and/or card images on magnetic tape or disk. The control statements supply important information to the pre-compiler as well as set and reset different options. Most of the control options are optional and must have a default value. The general format of a control card is as follows:

COLUMN:

1-6 STANDARD COBOL CARD SEQUENCE
7 BLANK
8 # (CONTROL CARD ID)

9-72 CONTROL OPTIONS (in free format)

73-80 STANDARD COBOL IDENTIFICATION.

SYNTAX:

```
<CONTROL OPTION>        :: = <INPUT OPTIONS><OUTPUT OPTIONS>
                             <TESTING OPTIONS><EFFICIENCY OPTIONS>
                             <DEBUGGING OPTIONS><WORK-FILES OPTIONS>
                             <COMPILE OPTIONS>
<INPUT OPTIONS>         :: = <MAIN INPUT OPTIONS><LIBRARY OPTIONS>
<MAIN INPUT OPTIONS>    :: =  CARD | TAPE | DISK | EMPTY
<LIBRARY OPTIONS>       :: =  EMPTY | TAPELIB | DISKLIB
<OUTPUT OPTIONS>        :: = <LISTING OPTIONS><CARD-OUTPUT OPTIONS>
                             <TAPE-OUTPUT OPTIONS><DISK-OUTPUT OPTIONS>
                             <XREF OPTIONS>
<LISTING OPTIONS>       :: = <REPORT OPTIONS><EDITING OPTIONS>
<REPORT OPTIONS>        :: =  EMPTY|LIST|LIST-1
<EDITING OPTIONS>       :: =  EMPTY|<LINE OPTIONS><PAGE OPTIONS>
<LINE OPTIONS>          :: =  EMPTY| SINGLE|DOUBLE
<PAGE OPTION>           :: =  EMPTY| NEWPAGE
<CARD-OUTPUT OPTIONS>   :: =  EMPTY|  PUNCH | PUNCH-1
<TAPE-OUTPUT OPTIONS>   :: =  EMPTY|NEWTAPE|  NEWTAPE-1
<DISK-OUTPUT OPTIONS>   :: =  EMPTY | NEWDISK | NEWDISK-1
<XREF OPTIONS>          :: =  EMPTY| XREF
<TESTING OPTIONS>       :: =  EMPTY|CT-TEST|RT-TEST

<EFFICIENCY OPTIONS>    :: =  EMPTY <FREQUENCY OPTIONS><MEMORY OPTIONS>
                              <MINIMIZATION OPTIONS><COST OPTIONS>
```

<FREQUENCY OPTIONS> :: = EMPTY|FREQ

<MEMORY OPTION :: = EMPTY|MEM.

<MINIMIZATION OPTIONS> :: = EMPTY|MINIMIZE

<COST OPTIONS> :: = EMPTY|COST

<DEBUGING OPTIONS> :: = EMPTY|DEBUG

<WORK-FILES OPTIONS> :: = EMPTY|WD3|WD2TP1 | WD1TP2 | TP3

<COMPILE OPTIONS> :: = EMPTY| SYNCOM|RUNCOM|LIBCOM

REMARKS:

1. Control cards can be inserted anywhere in the input string to reset different options with one exception. The work-files option may appear only once in the first set of control cards in front of the pre-compiler input.
2. All the options can be pre-set at the time of the pre-compiler compilation. A suggested list of pre-set values is: CARD, LIST, SINGLE, CT-TEST, TP3, SYNCON.
3. If necessary more than one control card may be used.

SEMANTICS:

The functions of the different control options available are as follows:

INPUT OPTIONS: The precompiler must be able to accept its primary input form cards, tape or decision tables and may be copied from a standard library.

OUTPUT OPTIONS: Up to four different output files may be produced

simultaniously. LIST stands for a listing of the generated program excluding the original decision tables. LIST-1 stands for a complete listing including the decision tables listed as notes for documentation purposes. SINGLE and DOUBLE stand for single and double line skipping, NEWPAGE stands for a newpage for every new decision tables. PUNCH and PUNCH-1, NEWTAPE and NEWTAPE-1, DISK and NEWDISK-1 are the same as LIST and LIST-1. The XREF option allows the user to receive crossreference of the generated program.

TESTING OPTIONS: On request test for ambiguity can be done in conversion time or different procedures will be incorporated to do so in run time as well. This is a debugging mode of operation and as soon as the user is sure that the program is free of ambiguities, it should perform another conversion and compile pass to return back to normal mode of operation. Using the RT-TEST option additional and inefficient code will be incorporated to perform the required test.

EFFICIENCY OPTIONS: If the probability distribution of the rules and the storage required for testing the different conditions is known, they may be supplied to the pre-compiler and taken into account in the conversion pass, on request, to generate a more effi-

cient code whenever possible. Possible minimization of tables should be also available.

DEBUGGING OPTIONS: If the DEBUG option is set together with LIST-1 some additional notes (including the mask and the decision matrices) will be listed for every decision table.

WORKING-FILES OPTIONS: THREE WORKING FILES are used by the pre-compiler and they can be any combination of disk and tapes.

COMPILE OPTIONS: Three different options are available. The pre-compiler may start the COBOL compiler, after conversion, for syntax check, for compile and run or for compile for library.

DECISION TABLES; PROGRAM FORMAT:

The format of the program must be according to the COBOL requirement except from the control cards which can appear in one place and the decision tables which are inserted in the procedure division as sections and referred to by an appropriate COBOL statement (GO TO, PERFORM or simply in line code).

The IDENTIFICATION DIVISION and the ENVIROMENT DIVISION are treated as in any standard COBOL program. The DATA DIVISION in a program containing decision tables must contain every data name, constant or conditional variable referenced in the procedure division code or decision tables. Inconsistencies will be discovered only in the compilation pass.

CODE AND TABLE LINKAGE

In the procedure division COBOL statement can be intermixed with decision tables. Every decision table is translated into a COBOL section code and decision tables can be linked by any of the linkages described in chapter III.

FORMAT OF THE TABLES

The suggested format for a decision table is demonstrated in the table given at the end of this chapter. Due to the limitations, in size, of punched cards (80 columns) a special way was designed for extended condition entries and extended action entries.

The program generated from a decision table is illustrated in the example given at the end of this chapter.

ALGORITHMS WHICH SHOULD BE USED

An efficient decision table precompiler must be able to use different procedures with different decision tables and according to the user request. Therefore the following is suggested:

1. For standard LCE-MAE-decision tables: Press's or Pollack's procedures should be used if run-time sfficiency is desired, and Kirk's mask procedure if minimum storage is desired.
2. For LCE-MAE decision tables where the cost for testing different conditions together with the rule probability distribution is supplied: Reinwald and Soland's procedure is to be used if run

time efficiency is desired; otherwise, the interrupted rule mask procedure (by King) should be used.

Note: In every one of the previous cases the procedures given by section V-2 should be used if run time diagnostics are required.

3. For MCE-MAE decision tables with and without cost and probability information, a special procedure is suggested here (and demonstrated at the end of this chapter.

STEP 1: Sort the conditions into two groups, LCE and ECE.

STEP 2: Use one of the tree procedures suggested in 1 or 2 for the LCE (depending on the type of the table) to partition the ECE into groups. The table is split by branching.

STEP 3: If cost and probability is given, use interrupted rule mask procedure accordingly; otherwise, use interrupted rule mask procedure dictated by the don't care entries.

STEP 4: Unsuccessful tests in any of the ECE groups leads to the ELSE RULE (if any); otherwise, to a standard error exit.

Note: If probabilities are given, the probability of every branch in the LCE is the sum of the probabilities of the rules in the appropriate group.

This procedure cannot produce run time diagnostics but can give the possible ambiguities included in the table.

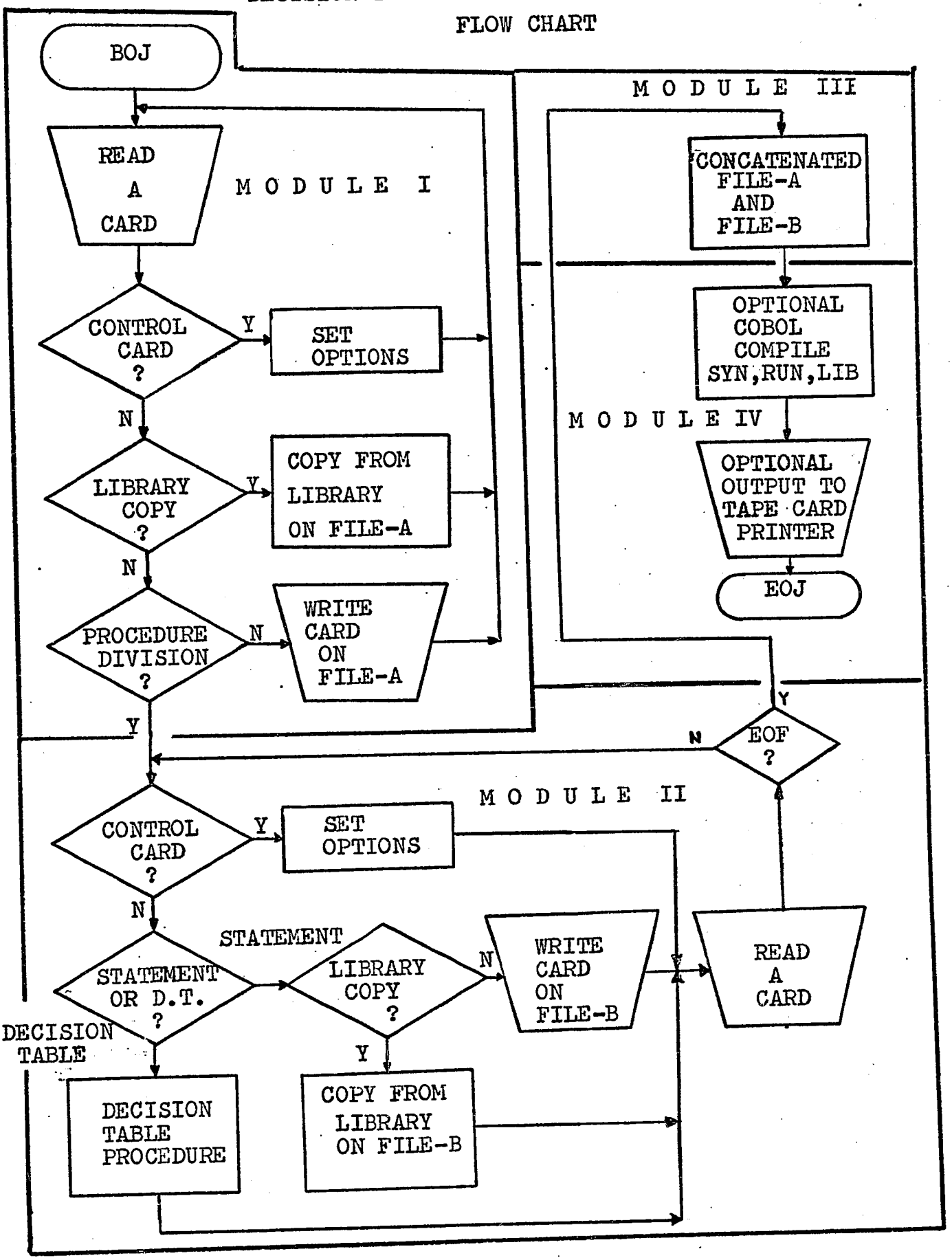
If run time diagnostics are required, the procedures described in section V-2 must be used.

VI-3 DECISION TABLE PRE-COMPILER GENERAL BLOCK DIAGRAM AND FILE CHART.

The first of the following two diagrams is a macro flow chart describing the different modules of the suggested COBOL pre-compiler. No doubt that the real implementation will probably require significant modifications even to this general block diagram.

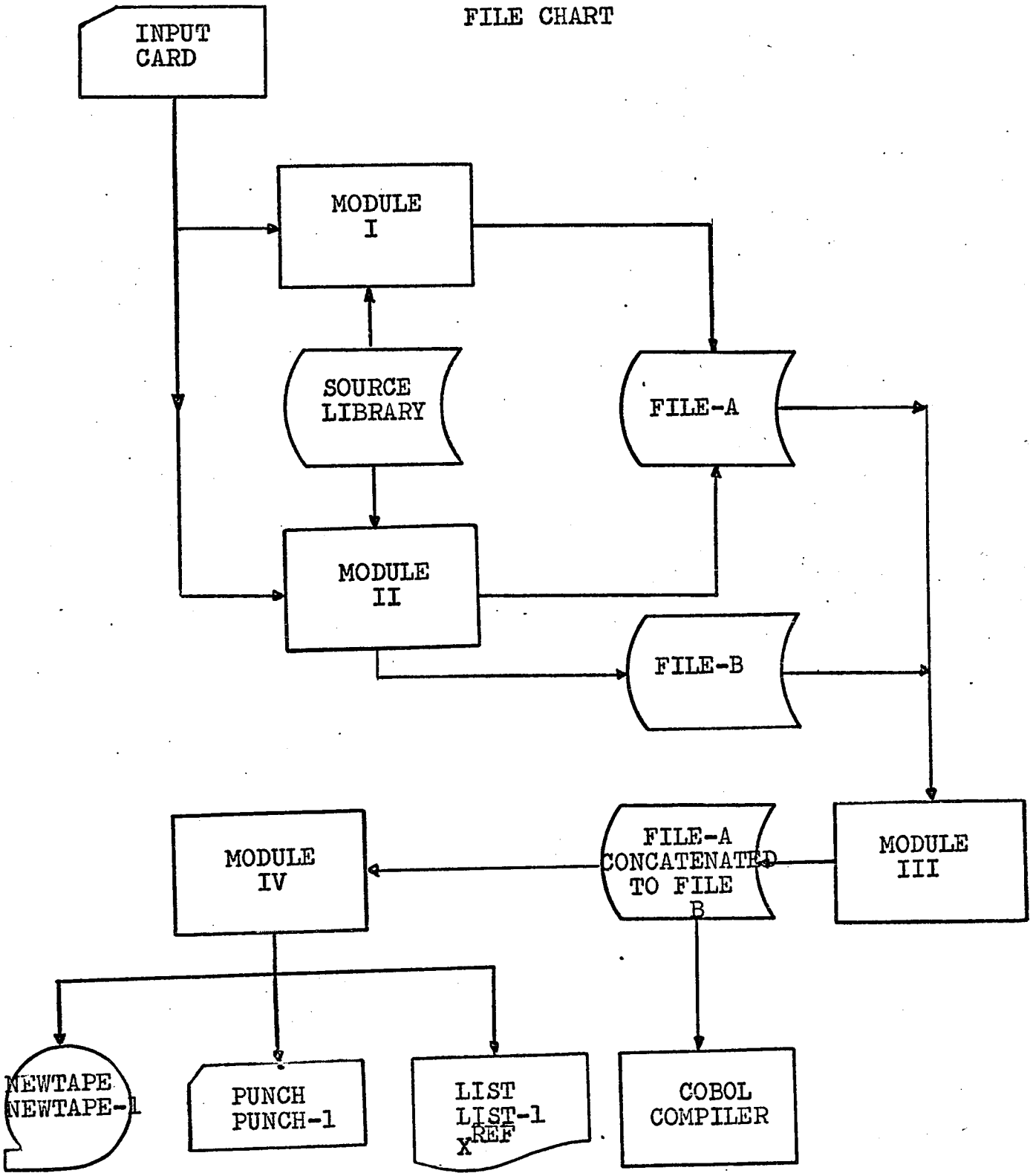
The file chart suggests a possible way of file utilization. All the files are described as disk files, actually any kind of combination of disk files and magnetic tape files can be used.

DECISION TABLE COBOL PRE-COMPILER MACRO
FLOW CHART



DECISION TABLE COBOL PRE-COMPILER

FILE CHART



VI-4 EXAMPLE:

The following example was designed to illustrate a possible use of the suggested COBOL pre-compiler. The example does not contain all the suggested options but can serve as sample. The example contains the following:

1. Input to the (suggested) precompiler.
2. LIST-1 output from the pre-compiler which was prepared by simulating the pre-compiler operations manually.
3. An actual compilation of the generated program (done on the Burroughs B-5500 computer).
4. Test data.
5. Run.

Note that run time diagnostics where not demonstrated.

VI-4-1 PRE-COMPILER INPUT.

The pre-compiler input was desiged to illustrate the simplicity of the input, and the suggested format for writing decision tables with extended condition entry and extended action entry.

```
00000  # CARD TAPELIB LIST-1 SINGLE XREF CT-TEST FREQ COST RUNCOM                    DT-EXAMP
00101  IDENTIFICATION DIVISION.                                                    DT-EXAMP
00102  PROGRAM-ID. "TEST MCE-MAE".                                                 DT-EXAMP
00103  AUTHOR. A PINHAS.                                                           DT-EXAMP
00104  DATE-WRITTEN. 28 DEC 1970.                                                  DT-EXAMP
00105  DATE-COMPILED.                                                              DT-EXAMP
00106  REMARKS. THIS PROGRAM WAS WRITTEN MANUALLY SIMULATING A                     DT-EXAMP
00107           SUGGESTED DECISION TABLE PRE-COMPILER. THE PRE-                     DT-EXAMP
00108           COMPILATION PASS WAS PRESUMABLY DONE WITH THE                       DT-EXAMP
00109           FOLLOWING OPTIONS SET:                                              DT-EXAMP
00110             1. INPUT FROM CARDS                                               DT-EXAMP
00111             2. OUTPUT LIST-1 PUNCH-L                                          DT-EXAMP
00112             3. TEST    CT-TEST                                                DT-EXAMP
00113             4.         FREQ                                                   DT-EXAMP
00114             5.         COST                                                   DT-EXAMP
00115  ENVIRONMENT DIVISION  COPY ENV-DIV-1 FROM TAPELIB.
00116  DATA DIVISION         COPY DAT-DIV-1 FROM TAPELIB.
00117  PROCEDURE DIVISION.
00118  INIT.
00119      OPEN INPUT CARDIN.
00120      OPEN OUTPUT LISTING. PERFORM HEADINGS.
00121  READ-LOOP.
00122      READ  CARDIN AT END GO TO LAST.
00123      PERFORM IS-MCE-MAE-DT.
00124      PERFORM WRITE-LISTING.
00125      GO TO READ-LOOP.

01018      TABLE-ID. "IS-MCE-MAE-DT" MCE-MAE.
01019      RL1                       000000000111111111122222222223E    COST
01020      RL2                       123456789012345678901234567890L
01021      CONDITIONS
01022      L IF G="PINK"             Y-YNYY-NYNNN-YNYN-YYNNY--YN-Y-        50
01023      E IF A***B                                                      20
01024           <                    Y-YY---Y---------Y--Y-YY----Y-
01025           =                    -Y--Y-Y--Y------Y-Y------Y---Y
01101           >                    -----Y-----Y-Y-Y---Y----Y-----
01102      E IF E =                                                        30
01103                20              Y-------Y--Y------Y-----YY-Y--
01104                15              ----Y----Y----------Y-Y----YY
01105                30              ------YY----YY--Y--Y----------
01106                35              -Y-----------------Y-Y-------
01107      L IF (X**2-3)=97          YYNNYNYNYNYNYNYYNYNYNNNYYNYNNY        68
01108      E IF C IS                                                       65
01109                NUMERIC         Y----Y--Y----Y-----------Y-Y-
01110                ALPHABETIC      ----Y-----Y-Y-YY--------Y-Y----
01111                ALPHANUMERIC    ----------------Y---Y------
01112      E IF COULOR =                                                   50
01113                     "BLUE "    ----Y--Y---------------------
01114                     "RED  "    -Y-Y-Y--Y-YY-------Y----------
01115                     "GREEN"    ----------Y--YYYY-YY-----Y--
01116                     "WHITE"    ------Y--Y--------------------
01117      L IF VALUE-1              YN--Y-NNY-Y-N-YY-N-Y---NN-Y--N        25
01118      E IF NUMBER IS                                                  45
01119                POSITIVE        Y-------YY--Y-Y--YY----Y-Y-Y--
01120                NEGATIVE        ---Y---Y--YY---Y-----YY---Y---
01121                ZERO            ----Y-Y---------Y---Y---------
01122      ACTIONS
01123      L MOVE "0" TO RL1         XXXXXXXXX
01124      L MOVE "1" TO RL1                  XXXXXXXXXX
01125      L MOVE "2" TO RL1                           XXXXXXXXXX
01201      L MOVE "3" TO RL1                                    X
```

```
01202      L MOVE "1"TO RL2
01203      L MOVE "2" TO RL2
01204      L MOVE "3" TO RL2
01205      L MOVE "4" TO RL2
01206      L MOVE "5" TO RL2
01207      L MOVE "6" TO RL2
01208      L MOVE "7" TO RL2
01209      L MOVE "8" TO RL2
01210      L MOVE "9" TO RL2
01212      L MOVE "ELSE" TO RULE-NO
01213      E ADD 1 TO
01214          CNT1
01215              CNT2
01216              CNT3
01217              CNT4
01218              CNT5
01219              CNT6
01220      E ADD 1 TO
01221              COUNTER-1
01222              COUNTER-2
01223              COUNTER-3
01224              COUNTER-4
01225              COUNTER-5
01301              COUNTER-6
01302              COUNTER-7
01303              COUNTER-8
01304              COUNTER-9
01305              COUNTER-10
01306              COUNTER-11
01307              COUNTER-12
01308              COUNTER-13
01309              COUNTER-14
01310              COUNTER-15
01311              COUNTER-16
01312              COUNTER-17
01313              COUNTER-18
01314              COUNTER-19
01315              COUNTER-20
01316              COUNTER-21
01317              COUNTER-22
01318              COUNTER-23
01319              COUNTER-24
01320              COUNTER-25
01321              COUNTER-26
01322              COUNTER-27
01323              COUNTER-28
01324              COUNTER-29
01325              COUNTER-30
01401      PROBABILITY          00000100000000000000000000000
01402      RULE-DISTRIBUTION    06110542027323803840125373511 5
01403      FINI.

02808  REPORTEX SECTION.
02809  HEADINGS.
02810      WRITE PRINTER-LINE FROM HDR1 AFTER ADVANCING TO CHANNEL 1.
02811      WRITE PRINTER-LINE FROM HDR2 AFTER ADVANCING 2 LINES.
02812  WRITE-LISTING.
02813      MOVE INPUT-CARD TO WS-CARD-IMAGE.
02814      WRITE PRINTER-LINE FROM PRINT-LINE AFTER ADVANCING 1 LINES.
028141     MOVE SPACES TO RULE-NO.
02815  LAST.
```

```
02816      WRITE PRINTER-LINE FROM GROUP-CNT-LINE    AFTER ADVANCING 2.
02817      WRITE PRINTER-LINE FROM RULE-COUNT-LINE-1 AFTER ADVANCING 2.
02818      WRITE PRINTER-LINE FROM RULE-COUNT-LINE-2 AFTER ADVANCING 2.
02819      WRITE PRINTER-LINE FROM RULE-COUNT-LINE-3 AFTER ADVANCING 2.
02820      WRITE PRINTER-LINE FROM RULE-COUNT-LINE-4 AFTER ADVANCING 2.
02821      WRITE PRINTER-LINE FROM RULE-COUNT-LINE-5 AFTER ADVANCING 2.
02822      CLOSE CARDIN LISTING WITH RELEASE.
02823      STOP RUN.
999999 END-OF-JOB.
```

VI-4-2 COBOL PRE-COMPILER OUTPUT.

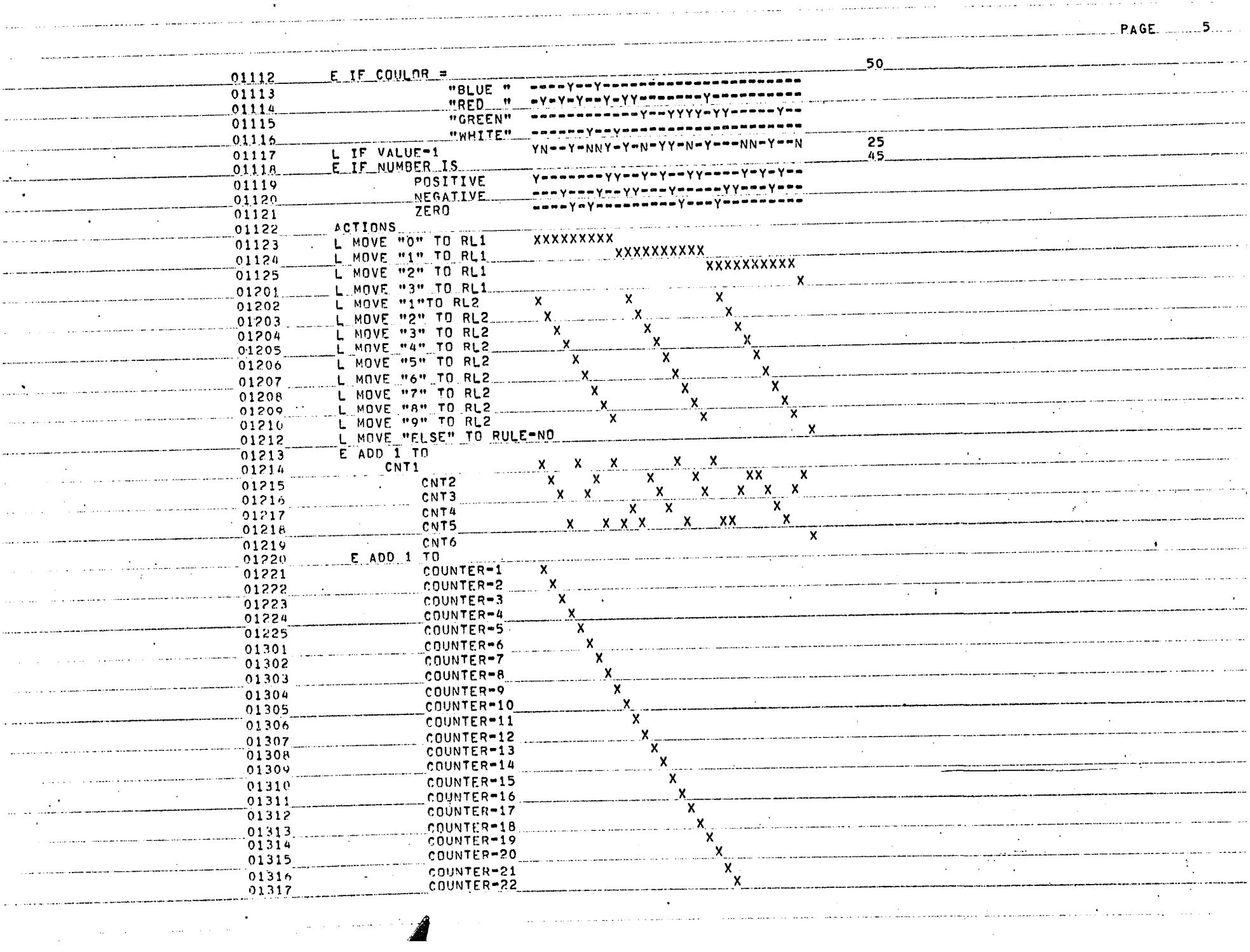
The pre-compiler output includes LIST-1, in which the original decision table together with conversion time diagnostics are listed under COBOL NOTES, as well as a cross reference of the program.

```
00101  IDENTIFICATION DIVISION.                                          DT=EXAMP
00102  PROGRAM-ID. "TEST MCE-MAE".                                       DT=EXAMP
00103  AUTHOR. A PINHAS.                                                 DT=EXAMP
00104  DATE-WRITTEN. 28 DEC 1970.                                        DT=EXAMP
00105  DATE-COMPILED.                                                    DT=EXAMP
00106  REMARKS. THIS PROGRAM WAS WRITTEN MANUALLY SIMULATING A           DT=EXAMP
00107           SUGGESTED DECISION TABLE PRE-COMPILER. THE PRE-           DT=EXAMP
00108           COMPILATION PASS WAS PRESUMABLY DONE WITH THE             DT=EXAMP
00109           FOLLOWING OPTIONS SET:                                    DT=EXAMP
00110              1. INPUT FROM CARDS                                    DT=EXAMP
00111              2. OUTPUT LIST=1 PUNCH=L                               DT=EXAMP
00112              3. TEST   CT=TEST                                      DT=EXAMP
00113              4.        FREQ                                         DT=EXAMP
00114              5.        COST                                         DT=EXAMP
00115  ENVIRONMENT DIVISION.                                             DT=EXAMP
00116  CONFIGURATION SECTION.                                            DT=EXAMP
00117  SOURCE-COMPUTER. B-5500.                                          DT=EXAMP
00118  OBJECT-COMPUTER. B-5500.                                          DT=EXAMP
00119  INPUT-OUTPUT SECTION.                                             DT=EXAMP
00120  FILE-CONTROL.                                                     DT=EXAMP
00121      SELECT CARDIN ASSIGN TO CARD-READER.                          DT=EXAMP
00122      SELECT LISTING ASSIGN TO PRINTER.                             DT=EXAMP
00123  DATA DIVISION.                                                    DT=EXAMP
00124  FILE SECTION.                                                     DT=EXAMP
00125  FD  CARDIN                                                        DT=EXAMP
00201      VALUE OF ID IS "CARDS"                                        DT=EXAMP
00202      DATA RECORD IS INPUT-CARD.                                    DT=EXAMP
00203  01  INPUT-CARD      SIZE 80.                                      DT=EXAMP
00204      02  A           PICTURE 9.                                    DT=EXAMP
00205      02  FILLER      PICTURE X(5).                                 DT=EXAMP
00206      02  B           PICTURE 9.                                    DT=EXAMP
00207      02  FILLER      PICTURE X(5).                                 DT=EXAMP
00208      02  E           PICTURE 99.   ← VJ=[20,15,30,35]      ECVDT=EXAMP
00209      02  FILLER      PICTURE X(5).                                 DT=EXAMP
00210      02  G           PICTURE X(4). ← PINK VJ=[Y,N]         LCVDT=EXAMP
00211      02  FILLER      PICTURE X(5).                                 DT=EXAMP
00212      02  X           PICTURE 99.   ← VALUES FROM 00 TO 99          DT=EXAMP
00213      02  FILLER      PICTURE X(5).                                 DT=EXAMP
00214      02  COULOR      PICTURE X(5). ← VJ=[BLUE ,RED  ,GREEN,WHITE]  DT=EXAMP
00215      02  FILLER      PICTURE X(5).                                 DT=EXAMP
00216      02  NUMBER      PICTURE J99.  ← VJ=[POSITIVE,NEGATIVE,ZERO]   DT=EXAMP
00217      02  FILLER      PICTURE X(5).                                 DT=EXAMP
00218      02  VAL         PICTURE XX.                                   DT=EXAMP
00219          88 VALUE=1  VALUE "AA".   ← AA   VJ=[Y,N]                 DT=EXAMP
00220      02  FILLER      PICTURE X(5).                                 DT=EXAMP
00221      02  C           PICTURE XXX.  ← VJ=[NUMERIC,ALPHABETI,        DT=EXAMP
00222      02  FILLER      PICTURE X(9). ←                ALPHANUMERIC]  DT=EXAMP
00223      02  RULE-ID     PICTURE X(9).                                 DT=EXAMP
00224                                                                    DT=EXAMP
00225                                                                    DT=EXAMP
00301  FD  LISTING                                                       DT=EXAMP
00302      VALUE OF ID IS "REPORT"                                       DT=EXAMP
00303      DATA RECORD IS PRINTER-LINE.                                  DT=EXAMP
00304  01  PRINTER-LINE    SIZE 132.                                     DT=EXAMP
```

```
00305
00400  WORKING-STORAGE SECTION.
00401  01  HDR1.
00402      02  FILLER                  PICTURE X(9)  VALUE SPACES.
00403      02  FILLER                  PICTURE X(14) VALUE
                                                    "RULE SATISFIED".
00404
00405      02  FILLER                  PICTURE X(50) VALUE SPACES.
00406      02  FILLER                  PICTURE X(10) VALUE "INPUT-CARD".
00407      02  FILLER                  PICTURE X(49) VALUE SPACES.
00408  01  HDR2.
00409      02  FILLER                  PICTURE X(40) VALUE SPACES.
00410      02  FILLER                  PICTURE X(6)  VALUE "A     ".
00411      02  FILLER                  PICTURE X(6)  VALUE "B     ".
00412      02  FILLER                  PICTURE X(7)  VALUE "E     ".
00501      02  FILLER                  PICTURE X(9) VALUE "G       ".
00502      02  FILLER                  PICTURE X(6) VALUE "X     ".
00503      02  FILLER                  PICTURE X(8) VALUE "COULOR  ".
00504      02  FILLER                  PICTURE X(8) VALUE "NUMBER  ".
00505      02  FILLER                  PICTURE X(10) VALUE "VALUE-1   ".
00506      02  FILLER                  PICTURE X(6)  VALUE "C     ".
00507      02  FILLER                  PICTURE X(13) VALUE "INTENDED RULE".
00508      02  FILLER                  PICTURE X(13) VALUE SPACES.
00509  01  PRINT-LINE.
00510      02  FILLER                  PICTURE X(9)  VALUE SPACES.
00511      02  FILLER                  PICTURE X(15) VALUE
                                                    "SATISFIES RULE=".
005111
00512      02  RULE-NO.
00513          03  RL1                 PICTURE X.
00514          03  RL2                 PICTURE X.
00515      02  FILLER                  PICTURE X(11) VALUE SPACES.
00516      02  WS-CARD-IMAGE           PICTURE X(80).
00517      02  FILLER                  PICTURE X(13) VALUE SPACES.
00518  01  GROUP-CNT-LINE.
00519      02  FILLER                  PICTURE X(16) VALUE
                                                    "  CNT1 = ".
00520
00521      02  CNT1                    PICTURE 999 VALUE 0.
00522      02  FILLER                  PICTURE X(17) VALUE
                                                    "  CNT2 = ".
00523
00524      02  CNT2                    PICTURE 999 VALUE 0.
00525      02  FILLER                  PICTURE X(17) VALUE
                                                    "  CNT3 = ".
00601
00602      02  CNT3                    PICTURE 999 VALUE 0.
00603      02  FILLER                  PICTURE X(17) VALUE
                                                    "  CNT4 = ".
00604
00605      02  CNT4                    PICTURE 999 VALUE 0.
00606      02  FILLER                  PICTURE X(17) VALUE
                                                    "  CNT5 = ".
00607
00608      02  CNT5                    PICTURE 999 VALUE 0.
00609      02  FILLER                  PICTURE X(17) VALUE
                                                    "  CNT6 = ".
00610
00611      02  CNT6                    PICTURE 999 VALUE 0.
00612      02  FILLER                  PICTURE X(13) VALUE SPACES.
00613
00614  01  RULE-COUNT-LINE-1.
00615      02  FILLER  PICTURE X(22) VALUE "  COUNTER-1  = ".
```

```
00616       02  COUNTER-1   PICTURE   9(3) VALUE 0.
00617       02  FILLER  PICTURE X(17) VALUE "    COUNTER-2  = ".
00618       02  COUNTER-2   PICTURE   9(3) VALUE 0.
00619       02  FILLER  PICTURE X(17) VALUE "    COUNTER-3  = ".
00620       02  COUNTER-3   PICTURE   9(3) VALUE 0.
00621       02  FILLER  PICTURE X(17) VALUE "    COUNTER-4  = ".
00622       02  COUNTER-4   PICTURE   9(3) VALUE 0.
00623       02  FILLER  PICTURE X(17) VALUE "    COUNTER-5  = ".
00624       02  COUNTER-5   PICTURE   9(3) VALUE 0.
00625       02  FILLER  PICTURE X(17) VALUE "    COUNTER-6  = ".
00701       02  COUNTER-6   PICTURE 9(3) VALUE 0.
00702       02  FILLER  PICTURE X(3) VALUE SPACES.
00703   01  RULE-COUNT-LINE-2.
00704       02  FILLER  PICTURE X(22) VALUE "          COUNTER-7  = ".
00705       02  COUNTER-7   PICTURE 9(3) VALUE 0.
00706       02  FILLER  PICTURE X(17) VALUE "    COUNTER-8  = ".
00707       02  COUNTER-8   PICTURE 9(3) VALUE 0.
00708       02  FILLER  PICTURE X(17) VALUE "    COUNTER-9  = ".
00709       02  COUNTER-9   PICTURE 9(3) VALUE 0.
00710       02  FILLER  PICTURE X(17) VALUE "    COUNTER-10 = ".
00711       02  COUNTER-10  PICTURE 9(3) VALUE 0.
00712       02  FILLER  PICTURE X(17) VALUE "    COUNTER-11 = ".
00713       02  COUNTER-11  PICTURE 9(3) VALUE 0.
00714       02  FILLER  PICTURE X(17) VALUE "    COUNTER-12 = ".
00715       02  COUNTER-12  PICTURE 9(3) VALUE 0.
00716       02  FILLER  PICTURE X(3)  VALUE SPACES.
00717   01  RULE-COUNT-LINE-3.
00718       02  FILLER  PICTURE X(22) VALUE "          COUNTER-13 = ".
00719       02  COUNTER-13  PICTURE 9(3) VALUE 0.
00720       02  FILLER  PICTURE X(17) VALUE "    COUNTER-14 = ".
00721       02  COUNTER-14  PICTURE 9(3) VALUE 0.
00722       02  FILLER  PICTURE X(17) VALUE "    COUNTER-15 = ".
00723       02  COUNTER-15  PICTURE 9(3) VALUE 0.
00724       02  FILLER  PICTURE X(17) VALUE "    COUNTER-16 = ".
00725       02  COUNTER-16  PICTURE 9(3) VALUE 0.
00801       02  FILLER  PICTURE X(17) VALUE "    COUNTER-17 = ".
00802       02  COUNTER-17  PICTURE 9(3) VALUE 0.
00803       02  FILLER  PICTURE X(17) VALUE "    COUNTER-18 = ".
00804       02  COUNTER-18  PICTURE 9(3) VALUE 0.
00805       02  FILLER  PICTURE X(3) VALUE SPACES.
00806   01  RULE-COUNT-LINE-4.
00807       02  FILLER  PICTURE X(22) VALUE "          COUNTER-19 = ".
00808       02  COUNTER-19  PICTURE 9(3) VALUE 0.
00809       02  FILLER  PICTURE X(17) VALUE "    COUNTER-20 = ".
00810       02  COUNTER-20  PICTURE 9(3) VALUE 0.
00811       02  FILLER  PICTURE X(17) VALUE "    COUNTER-21 = ".
00812       02  COUNTER-21  PICTURE 9(3) VALUE 0.
00813       02  FILLER  PICTURE X(17) VALUE "    COUNTER-22 = ".
00814       02  COUNTER-22  PICTURE 9(3) VALUE 0.
00815       02  FILLER  PICTURE X(17) VALUE "    COUNTER-23 = ".
00816       02  COUNTER-23  PICTURE 9(3) VALUE 0.
00817       02  FILLER  PICTURE X(17) VALUE "    COUNTER-24 = ".
00818       02  COUNTER-24  PICTURE 9(3) VALUE 0.
00819       02  FILLER  PICTURE X(3) VALUE SPACES.
00820   01  RULE-COUNT-LINE-5.
```

```
00821       02  FILLER  PICTURE X(22) VALUE "          COUNTER-25 = ".
00822       02  COUNTER-25  PICTURE 9(3) VALUE 0.
00823       02  FILLER  PICTURE X(17) VALUE "    COUNTER-25 = ".
00824       02  COUNTER-26  PICTURE 9(3) VALUE 0.
00825       02  FILLER  PICTURE X(17) VALUE "    COUNTER-27 = ".
00901       02  COUNTER-27  PICTURE 9(3) VALUE 0.
00902       02  FILLER  PICTURE X(17) VALUE "    COUNTER-28 = ".
00903       02  COUNTER-28  PICTURE 9(3) VALUE 0.
00904       02  FILLER  PICTURE X(17) VALUE "    COUNTER-29 = ".
00905       02  COUNTER-29  PICTURE 9(3) VALUE 0.
00906       02  FILLER  PICTURE X(17) VALUE "    COUNTER-30 = ".
00907       02  COUNTER-30  PICTURE 9(3) VALUE 0.
00908       02  FILLER  PICTURE X(3) VALUE SPACES.
009061 01   TABLE-1.
00909       02  RESULT-VECTOR-1.
00910           03  RV-1    PICTURE  9  OCCURS 5 TIMES.
00911
00912       02  WORK-VECTOR-1.
00913           03  WV-1    PICTURE  9  OCCURS 5 TIMES.
01001  PROCEDURE DIVISION.
01002  INIT.
01003       OPEN INPUT CARDIN.
01004       OPEN OUTPUT LISTING. PERFORM HEADINGS.
01005  READ-LOOP.
01006       READ CARDIN AT END GO TO LAST.
01007       PERFORM IS-MCE-MAE-DT.
01008       PERFORM WRITE-LISTING.
01009       GO TO READ-LOOP.
01010
01011  IS-MCE-MAE-DT SECTION.
01012  IS-MCE-MAE-DT-NOTES.
01013       NOTE THE ORIGINAL DECISIO TABLE TOGETHER WITH THE REQUIRED
01014            DIAGNOSTICS (ACCORDING TO THE SPECIFIED OPTIONS IN THE
01015            # CARD) ARE LISTED HERE FOR YOUR INSPECTION. IF CORRECT-
01016            IONS ARE NECESSARY EXECUTE ANOTHER PRE-COMPLE RUN.
01017
01018       TABLE-ID. "IS-MCE-MAE-DT" MCE-MAE.
01019       RL1                       00000000011111111112222222223E    COST
01020       RL2                       12345678901234567890123456789OL
01021       CONDITIONS
01022       L IF G="PINK"             Y-YNYY-NYNNN-YNYN-YYNNY--YN-Y-      50
01023       E IF A***B                                                    20
01024                  <              Y-YY---Y---------Y--Y-YY----Y-
01025                  =              -Y--Y-Y--Y------Y-Y-----Y---Y
01101                  >              -----Y-----Y-Y-Y---Y----Y-----
01102       E IF E =                                                      30
01103                    20           Y-------Y--Y------Y-----YY-Y--
01104                    15           ----Y----Y----------Y-Y----YY
01105                    30           -----YY----YY--Y--Y----------
01106                    35           -Y-----------------Y-Y-------
01107       L IF (X**2-3)=97          YYNNYNYNYNYNYNYYNYNYNNNYYNYNNY      68
01108       E IF C IS                                                     65
01109                  NUMERIC        Y----Y--Y----Y-----------Y-Y-
01110                  ALPHABETIC     ----Y-----Y-Y-YY-------Y-Y----
01111                  ALPHANUMERIC   -----------------Y---Y-------
```


50
01112 E IF COULOR =
"BLUE "
"RED "
"GREEN"
"WHITE"
01113
01114
01115
01116
01117 L IF VALUE=1
01118 E IF NUMBER IS
POSITIVE
NEGATIVE
ZERO
25
45
01119
01120
01121
01122 ACTIONS
01123 L MOVE "0" TO RL1
01124 L MOVE "1" TO RL1
01125 L MOVE "2" TO RL1
01201 L MOVE "3" TO RL1
01202 L MOVE "1"TO RL2
01203 L MOVE "2" TO RL2
01204 L MOVE "3" TO RL2
01205 L MOVE "4" TO RL2
01206 L MOVE "5" TO RL2
01207 L MOVE "6" TO RL2
01208 L MOVE "7" TO RL2
01209 L MOVE "8" TO RL2
01210 L MOVE "9" TO RL2
01212 L MOVE "ELSE" TO RULE-NO
01213 E ADD 1 TO
01214 CNT1
01215 CNT2
01216 CNT3
01217 CNT4
01218 CNT5
01219 CNT6
01220 E ADD 1 TO
01221 COUNTER-1
01222 COUNTER-2
01223 COUNTER-3
01224 COUNTER-4
01225 COUNTER-5
01301 COUNTER-6
01302 COUNTER-7
01303 COUNTER-8
01304 COUNTER-9
01305 COUNTER-10
01306 COUNTER-11
01307 COUNTER-12
01308 COUNTER-13
01309 COUNTER-14
01310 COUNTER-15
01311 COUNTER-16
01312 COUNTER-17
01313 COUNTER-18
01314 COUNTER-19
01315 COUNTER-20
01316 COUNTER-21
01317 COUNTER-22

```
01318              COUNTER-23                                                X
01319              COUNTER-24                                                 X
01320              COUNTER-25                                                  X
01321              COUNTER-26                                                   X
01322              COUNTER-27                                                    X
01323              COUNTER-28                                                     X
01324              COUNTER-29                                                      X
01325              COUNTER-30
01401     PROBABILITY          00000100000000000000000000000000
01402     RULE-DISTRIBUTION    06110542027323803840125373511 5
01403     FINI.
01404  IS-MCE-MAE-DT-OPTIONS.
01405     NOTE FOR THIS DECISION TABLE THE FOLLOWING OPTIONS WERE SET:
01406          1 INPUT OPTIONS: CARD, TAPELIB
01407          2 OUTPUT OPTIONS: LIST-1, SINGLE
01408          3 TESTING OPTIONS: CT-TEST
01409          4 EFFICENCY OPTIONS: FREQ, COST
01410          5 COMPILE OPTIONS: RUNCOM
01411     FOR OTHER OPTIONS THE DEFAULT VALUES WERE USED.
01412
01413  IS-MCE-MAE-DT-DIAGNOSTICS.
01414     NOTE
01415     1 POSSIBLE REDUNDANCIES:
01416          NONE
01417
01418     2 POSSIBLE CONTRADICTIONS:

01419          FOR RULE  NO-1      NO-13     NO-18     NO-3      NO-3
01420          AND RULE  NO-9      NO-18     NO-24     NO-29     NO-26
01421                    Y         -         -         Y         Y
01422                    <         <         <         <         <
01423                    20        30        15        15        35
01424                    Y         Y         Y         N         N
01425                    NUMERIC   ALPHABETIC ALPHABETIC NUMERIC  ALPHANUERC
01501                    RED       GREEN     GREEN     -         -
01502                    Y         N         N         -         -
01503                    POSITIVE  POSITIVE  POSITIVE  -         NEGATIVE
01504
01505          FOR RULE  NO-6      NO-19
01506          AND RULE  NO-14     NO-26
01507                    Y         Y
01508                    >         =
01509                    30        20
01510                    N         N
01511                    NUMERIC   ALPHABETIC
01512                    RED       GREEN
01513                    -         -
01514                    -         POSITIVE
01515
01516     3 COMPLETNESS: ACHIEVED BY THE ELSE RULE
01517
01518     4 GENERAL:
01519          THE SUM OF THE PROBABILITY IS NOT  1
01520
01521  *** NO FATAL ERRORS WERE FOUND
```

```
01522      *** IN OPTIMIZING THE FOLLOWING PROCEDURES WERE USED:
01523           1  CONDITION PARTIONING BY LCE  FOLLOWD BY ECE
01524           2  TABLE SPLITING BY BRANCHING DONE ON THE LCE
01525           3  OPTIMIZING THE LCE BY A TREE METHOD
01601           4 OPTIMIZIN THE ECE FIRST BY PROBABILITIES FOLOWED BY
01602             COST.
01603
01604  IS-MCE-MAE-DT-SPLIT.
01605       IF (X**2 - 3) = 97
01606          NEXT SENTENCE
01607       ELSE
01608       IF G="PINK"
01609          GO TO IS-MCE-MAE-DT-GRP-3
01610       ELSE
01611          GO TO IS-MCE-MAE-DT-GRP-5.
01612       IF VALUE-1
01613          NEXT SENTENCE
01614       ELSE
01615          GO TO IS-MCE-MAE-DT-GRP-2.
01616       IF G="PINK"
01617          GO TO IS-MCE-MAE-DT-GRP-1
01618       ELSE
01619          GO TO IS-MCE-MAE-DT-GRP-4.
01620
01701  IS-MCE-MAE-DT-CON-2.
01702       IF A < B MOVE 1 TO RV-1(1)  ELSE
01703       IF A = B MOVE 2 TO RV-1(1)  ELSE
01704                MOVE 3 TO RV-1(1).
01705  IS-MCE-MAE-DT-CON-3.
01706       IF E = 20 MOVE 1 TO RV-1(2) ELSE
01707       IF E = 15 MOVE 2 TO RV-1(2) ELSE
01708       IF E = 30 MOVE 3 TO RV-1(2) ELSE
01709       IF E = 35 MOVE 4 TO RV-1(2).
01711  IS-MCE-MAE-DT-CON-5.
01712       IF C IS NUMERIC      MOVE 1 TO RV-1(3) ELSE
01713       IF C IS ALPHABETIC   MOVE 2 TO RV-1(3) ELSE
01714                            MOVE 3 TO RV-1(3).
01715  IS-MCE-MAE-DT-CON-6.
01716       IF COULOR = "BLUE "  MOVE 1 TO RV-1(4) ELSE
01717       IF COULOR = "RED  "  MOVE 2 TO RV-1(4) ELSE
01718       IF COULOR = "GREEN"  MOVE 3 TO RV-1(4) ELSE
01719       IF COULOR = "WHITE"  MOVE 4 TO RV-1(4).
01721  IS-MCE-MAE-DT-CON-8.
01722       IF NUMBER IS POSITIVE MOVE 1 TO RV-1(5) ELSE
01723       IF NUMBER IS NEGATIVE MOVE 2 TO RV-1(5) ELSE
01724                             MOVE 3 TO RV-1(5).
01725
01801  IS-MCE-MAE-DT-GRP-1.
01802       MOVE ALL ZEROS TO RESULT-VECTOR-1.
01803            PERFORM IS-MCE-MAE-DT-CON-2.
01804            PERFORM IS-MCE-MAE-DT-CON-3.
01805            PERFORM IS-MCE-MAE-DT-CON-8.
01806            PERFORM IS-MCE-MAE-DT-CON-6.
01807                MOVE RESULT-VECTOR-1 TO WORK-VECTOR-1
01808                MOVE ZERO TO WV-1(3) WV-1(5)
```

```
01809                IF WORK-VECTOR-1 = "33020"                  ← RULE-20
01810                    MOVE "2" TO RL1
01811                    MOVE "0" TO RL2
01812                    ADD   1 TO CNT1
01813                    ADD   1  TO COUNTER-1
01814                        GO TO IS-MCE-MAE-DT-END.
01815           PERFORM IS-MCE-MAE-DT-CON-5.
01816               MOVE RESULT-VECTOR-1 TO WORK-VECTOR-1
01817               MOVE ZERO TO WV-1(4)
01818                   IF WORK-VECTOR-1 = "11101"                ← RULE-01
01819                       MOVE "0" TO RL1
01820                       MOVE "1" TO RL2
01821                       ADD   1  TO CNT1
01822                       ADD   1  TO COUNTER-1
01823                           GO TO IS-MCE-MAE-DT-END.
01824               MOVE RESULT-VECTOR-1 TO  WORK-VECTOR-1
01825                   IF WORK-VECTOR-1 ="22321"                 ← RUEL-03
01901                       MOVE "0" TO RL1
01902                       MOVE "3" TO RL2
01903                       ADD   1  TO CNT1
01904                       ADD   1  TO COUNTER-3
01905                           GO TO IS-MCE-MAE-DT-END.
01906               MOVE RESULT-VECTOR-1 TO WORK-VECTOR-1
01907               MOVE ZERO TO WV-1(1)
01908                   IF WORK-VECTOR-1 = "01121"                ← RULE-09
01909                       MOVE "0" TO RL1
01910                       MOVE "9" TO RL2
01911                       ADD   1  TO CNT1
01912                       ADD   1  TO COUNTER-9
01913                           GO TO IS-MCE-MAE-DT-END.
01914               MOVE RESULT-VECTOR-1 TO WORK-VECTOR-1
01915               MOVE ZERO TO WV-1(2)
01916                   IF WORK-VECTOR-1 = "30232"                ← RULE-16
01917                       MOVE "1" TO RL1
01918                       MOVE "6" TO RL2
01919                       ADD   1  TO CNT1
01920                       ADD   1  TO COUNTER-16
01921                           GO TO IS-MCE-MAE-DT-END.
01922                   GO TO IS-MCE-MAE-DT-ELSE-RULE.            ← ELSE-RULE
01923  IS-MCE-MAE-DT-GRP-2.
01924      MOVE ALL ZEROS TO RESULT-VECTOR-1.
01925           PERFORM IS-MCE-MAE-DT-CON-2.
02001           PERFORM IS-MCE-MAE-DT-CON-8.
02002           PERFORM IS-MCE-MAE-DT-CON-6.
02003                   IF RESULT-VECTOR-1 = "10031"              ← RULE-18
02004                       MOVE "1" TO RL1
02005                       MOVE "8" TO RL2
02006                       ADD   1  TO CNT2
02007                       ADD   1  TO COUNTER-18
02008                           GO TO IS-MCE-MAE-DT-END.
02009           PERFORM IS-MCE-MAE-DT-CON-3.
02010               MOVE RESULT-VECTOR-1 TO WORK-VECTOR-1
02011               MOVE ZERO TO WV-1(3) WV-1(4) WV-1(5)
02012                   IF WORK-VECTOR-1 = "31000"                ← RULE-25
02013                       MOVE "2" TO RL1
```

```
02014                    MOVE "5" TO RL2
02015                    ADD   1  TO CNT2
02016                    ADD   1  TO COUNTER-25
02017                       GO TO IS-MCE-MAE-DT-END.
02018           MOVE RESULT-VECTOR-1 TO WORK-VECTOR-1
02019           MOVE ZERO TO WV-1(3) WV-1(5)
02020              IF WORK-VECTOR-1 = "24020"            ← RULE-02
02021                    MOVE "0" TO RL1
02022                    MOVE "2" TO RL2
02023                    ADD   1  TO CNT2
02024                    ADD   1  TO COUNTER-2
02025                       GO TO IS-MCE-MAE-DT-END.
02101           MOVE RESULT-VECTOR-1 TO WORK-VECTOR-1
02102           MOVE ZERO TO WV-1(3) WV-1(4) WV-1(5)
02103              IF WORK-VECTOR-1 = "22000"            ← RULE-30
02104                    MOVE "3" TO RL1
02105                    MOVE "0" TO RL2
02106                    ADD   1  TO CNT2
02107                    ADD   1  TO COUNTER-30
02108                       GO TO IS-MCE-MAE-DT-END.
02109           IF RESULT-VECTOR-1 = "23043"             ← RULE-07
02110                    MOVE "0" TO RL1
02111                    MOVE "7" TO RL2
02112                    ADD   1  TO CNT2
02113                    ADD   1  TO COUNTER-7
02114                       GO TO IS-MCE-MAE-DT-END.
02115     PERFORM IS-MCE-MAE-DT-CON-5.
02116        MOVE RESULT-VECTOR-1 TO WORK-VECTOR-1
02117        MOVE ZERO TO WV-1(4)
02118           IF WORK-VECTOR-1 = "12201"             ← RULE-24
02119                    MOVE "2" TO RL1
02120                    MOVE "4" TO RL2
02121                    ADD   1  TO CNT1
02122                    ADD   1  TO COUNTER-24
02123                       GO TO IS-MCE-MAE-DT-END.
02124           MOVE RESULT-VECTOR-1 TO WORK-VECTOR-1
02125           MOVE ZERO TO WV-1(1)
02201              IF WORK-VECTOR-1 = "03231"            ← RULE-13
02202                    MOVE "1" TO RL1
02203                    MOVE "3" TO RL2
02204                    ADD   1  TO CNT2
02205                    ADD   1  TO COUNTER-13
02206                       GO TO IS-MCE-MAE-DT-END.
02207                 GO TO IS-MCE-MAE-DT-ELSE-RULE.
02208  IS-MCE-MAE-DT-GRP-3.
02209      MOVE ALL ZEROS TO RESULT-VECTOR-1.
02210           PERFORM IS-MCE-MAE-DT-CON-2.
02211           PERFORM IS-MCE-MAE-DT-CON-5.
02212           PERFORM IS-MCE-MAE-DT-CON-6.
02213              IF RESULT-VECTOR-1 = "30120"          ← RULE-06
02214                    MOVE "0" TO RL1
02215                    MOVE "6" TO RL2
02216                    ADD   1  TO CNT3
02217                    ADD   1  TO COUNTER-6
02218                       GO TO IS-MCE-MAE-DT-END.
```

```
02219          PERFORM IS-MCE-MAE-DT-CON-3.
02220          PERFORM IS-MCE-MAE-DT-CON-8.
02221              MOVE RESULT-VECTOR-1 TO WORK-VECTOR-1
02222              MOVE ZERO TO WV-1(4) WV-1(5)
02223              IF WORK-VECTOR-1 = "14302"                    <- RULE-23
02224                  MOVE "2" TO RL1-1(3)
02225                  MOVE "3" TO RL2
02301                  ADD "1" TO CNT3
02302                  ADD "1" TO COUNTER-23
02303                  GO TO IS-MCE-MAE-DT-END.
02304              IF RESULT-VECTOR-1 = "21231"                  <- RULE-19
02305                  MOVE "1" TO RL1
02306                  MOVE "9" TO RL2
02307                  ADD "1" TO CNT3
02308                  ADD "1" TO COUNTER-19                     <- RULE-30
02309                  GO TO IS-MCE-MAE-DT-END.
02310          MOVE RESULT-VECTOR-1 TO WORK-VECTOR-1
02311          MOVE ZERO TO WV-1(4) WV-1(5)
02312              IF WORK-VECTOR-1 = "38100"                    <- RULE-14
02313                  MOVE "1" TO RL1
02314                  MOVE "4" TO RL2
02315                  ADD "1" TO CNT3
02316                  ADD "1" TO COUNTER-14
02317                  GO TO IS-MCE-MAE-DT-END.
02318          MOVE RESULT-VECTOR-1 TO WORK-VECTOR-1
02319          MOVE ZERO TO WV-1(4)
02320              IF WORK-VECTOR-1 = "21201"                    <- RULE-26
02321                  MOVE "2" TO RL1
02322                  MOVE "6" TO RL2
02323                  ADD "1" TO CNT3
02324                  ADD "1" TO COUNTER-26
02325                  GO TO IS-MCE-MAE-DT-END.
02401          MOVE RESULT-VECTOR-1 TO WORK-VECTOR-1
02402          MOVE ZERO TO WV-1(4) WV-1(5)
02403              IF WORK-VECTOR-1 = "12100"                    <- RULE-29
02404                  MOVE "2" TO RL1
02405                  MOVE "9" TO RL2
02406                  ADD "1" TO CNT3
02407                  ADD "1" TO COUNTER-29
02408                  GO TO IS-MCE-MAE-DT-END.
02409          MOVE RESULT-VECTOR-1 TO WORK-VECTOR-1
02410          MOVE ZERO TO WV-1(2) WV-1(3) WV-1(4) WV-1(5)
02411              IF WORK-VECTOR-1 = "10000"                    <- RULE-03
02412                  GO TO MOVE-"0" TO RL1
02413                  MOVE "3" TO RL2
02414                  ADD "1" TO CNT3.
02415                  ADD "1" TO COUNTER-3
02416                  GO TO IS-MCE-MAE-DT-END.
02417                  GO TO IS-MCE-MAE-DT-ELSE-RULE.            <- ELSE-RULE
02418  IS-MCE-MAE-DT-GRP-4.
02419      MOVE ALL ZEROS TO RESULT-VECTOR-1.
02420          PERFORM IS-MCE-MAE-DT-CON-5.
02421          PERFORM IS-MCE-MAE-DT-CON-8.
02422              IF RESULT-VECTOR-1 = "00201"                  <- RULE-15
02423                  MOVE "1" TO RL1
```

```
02424                    MOVE "5" TO RL2.
02425                    ADD   1  TO CNT4
02501                    ADD   1  TO COUNTER-15
02502                       GO TO IS-MCE-MAE-DT-END.
02503           PERFORM IS-MCE-MAE-DT-CON-6.
02504                 IF RESULT-VECTOR-1 = "00222"          ← RULE-11
02505                    MOVE "1" TO RL1
02506                    MOVE "1" TO RL2
02507                    ADD   1  TO CNT4
02508                    ADD   1  TO COUNTER-11
02509                       GO TO IS-MCE-MAE-DT-END.
02510              MOVE ZERO TO RV-1(4)
02511                 IF RESULT-VECTOR-1 = "00103"          ← RULE-27
02512                    MOVE "2" TO RL1
02513                    MOVE "7" TO RL2
02514                    ADD   1  TO CNT4
02515                    ADD   1  TO COUNTER-27
02516                       GO TO IS-MCE-MAE-DT-END.
02517                 GO TO IS-MCE-MAE-DT-ELSE-RULE.        ← ELSE-RULE
02518  IS-MCE-MAE-DT-GRP-5.
02519      MOVE ALL ZEROS TO RESULT-VECTOR-1.
02520           PERFORM IS-MCE-MAE-DT-CON-2.
02521           PERFORM IS-MCE-MAE-DT-CON-3.
02522           PERFORM IS-MCE-MAE-DT-CON-6.
02523           PERFORM IS-MCE-MAE-DT-CON-8.
02524                 IF RESULT-VECTOR-1 = "31022"          ← RULE-12
02525                    MOVE "1" TO RL1
02601                    MOVE "2" TO RL2
02602                    ADD   1  TO CNT5
02603                    ADD   1  TO COUNTER-12
02604                       GO TO IS-MCE-MAE-DT-END.
02605              IF RESULT-VECTOR-1 = "23033"          ← RULE-16
02606                 MOVE "1" TO RL1
02607                 MOVE ZERO TO RL2-1(5)
02608                 IF WORK-VECTOR-1 = "12100"          ← RULE-29
02609                    ADD   1  TO CNT5
02610                    ADD   1  TO COUNTER-16
02611                       GO TO IS-MCE-MAE-DT-END.
02612              IF  RESULT-VECTOR-1 = "13012"          ← RULE-08
02613                    MOVE "6" TO RL2
02614                    MOVE ZERO TO WORK-VECTOR-1
02615                    ADD   1  TO CNT5
02616                    ADD   1  TO COUNTER-8
02617                 IF WORK-VECTOR-1 = "02032"          ← RULE-03
02618                    MOVE "2" TO RL1
02619                    MOVE "2" TO RL2
02620                    ADD   1  TO CNT5
02621                    ADD   1  TO COUNTER-10
02622                       GO TO IS-MCE-MAE-DT-END.
02623              IF RESULT-VECTOR-1 = "22041"          ← RULE-10
02624                    MOVE "1" TO RL1
02625                    MOVE "0" TO RL2
02701                    ADD   1  TO CNT5
02702                    ADD   1  TO COUNTER-3
02703                 GO TO IS-MCE-MAE-DT-END.          ← ELSE-RULE
```

```
02704                       ADD   1  TO COUNTER-22
02705                         GO TO IS-MCE-MAE-DT-END.
02706                 MOVE RESULT-VECTOR-1 TO WORK-VECTOR-1.
02707                 MOVE ZERO TO WV-1(2)
02708                     IF WORK-VECTOR-1 = "10022"            ← RULE-04
02709                       MOVE "0" TO RL1
02710                       MOVE "4" TO RL2
02711                       ADD   1  TO CNT5
02712                       ADD   1  TO COUNTER-4
02713                         GO TO IS-MCE-MAE-DT-END.
02714                   IF RESULT-VECTOR-1 = "14033"            ← RULE-21
02715                       MOVE "2" TO RL1
02716                       MOVE "1" TO RL2
02717                       ADD   1  TO CNT5
02718                       ADD   1  TO COUNTER-21
02719                         GO TO IS-MCE-MAE-DT-END.
02720                 MOVE ZERO TO RV-1(1)
02721                     IF RESULT-VECTOR-1 = "01031"            ← RULE-28
02722                       MOVE "2" TO RL1
02723                       MOVE "8" TO RL2
02724                       ADD   1  TO CNT5
02725                       ADD   1  COUNTER-28
02801                         GO TO IS-MCE-MAE-DT-END.
02802                   GO TO IS-MCE-MAE-DT-ELSE-RULE.          ← ELSE-RULE
02803  IS-MCE-MAE-DT-ELSE-RULE.
02804       MOVE "ELSE" TO RULE-NO.
02805       ADD 1 TO CNT6.
028051      GO TO IS-MCE-MAE-DT-END.
02806  IS-MCE-MAE-DT-END.
02807       EXIT.
02808  REPORTEX SECTION.
02809  HEADINGS.
02810       WRITE PRINTER-LINE FROM HDR1 AFTER ADVANCING TO CHANNEL 1.
02811       WRITE PRINTER-LINE FROM HDR2 AFTER ADVANCING 2 LINES.
02812  WRITE-LISTING.
02813       MOVE INPUT-CARD TO WS-CARD-IMAGE.
02814       WRITE PRINTER-LINE FROM PRINT-LINE AFTER ADVANCING 1 LINES.
028141      MOVE SPACES TO RULE-NO.
02815  LAST.
02816       WRITE PRINTER-LINE FROM GROUP-CNT-LINE    AFTER ADVANCING 2.
02817       WRITE PRINTER-LINE FROM RULE-COUNT-LINE-1 AFTER ADVANCING 2.
02818       WRITE PRINTER-LINE FROM RULE-COUNT-LINE-2 AFTER ADVANCING 2.
02819       WRITE PRINTER-LINE FROM RULE-COUNT-LINE-3 AFTER ADVANCING 2.
02820       WRITE PRINTER-LINE FROM RULE-COUNT-LINE-4 AFTER ADVANCING 2.
02821       WRITE PRINTER-LINE FROM RULE-COUNT-LINE-5 AFTER ADVANCING 2.
02822       CLOSE CARDIN LISTING WITH RELEASE.
02823       STOP RUN.
999999 END-OF-JOB.
```

DEFINED ON:	NAME	REFERENCED ON:	001 NO RECDS WRITTEN	001 NO BLKS WRITTEN	OF
					3
00204	A	01023 01702 01703			3
00206	B	01023 01702 01703			
FILE 00125	CARDIN	DUPLICATE NAME			3
		01003 01006 02822			7
00521	CNT1	01214 01812 01821 01903 01911 01919 02121			7
00524	CNT2	01215 02006 02015 02023 02106 02112 02204			8
00602	CNT3	01216 02216 02301 02307 02315 02323 02406 02414			4
00605	CNT4	01217 02425 02507 02514			9
00608	CNT5	01218 02602 02608 02614 02620 02703 02711 02717 02724			2
00611	CNT6	01219 02805			5
00214	COULOR	01112 01716 01717 01718 01719			2
00711	COUNTER-10	01305 02621			2
00713	COUNTER-11	01306 02508			2
00715	COUNTER-12	01307 02603			2
00719	COUNTER-13	01308 02205			2
00721	COUNTER-14	01309 02316			2
00723	COUNTER-15	01310 02501			3
00725	COUNTER-16	01311 01920 02609			1
00802	COUNTER-17	01312			2
00804	COUNTER-18	01313 02007			2
00808	COUNTER-19	01314 02308			3
00616	COUNTER-1	01221 01813 01822			1
00810	COUNTER-20	01315			2
00812	COUNTER-21	01316 02718			2
00814	COUNTER-22	01317 02704			2
00816	COUNTER-23	01318 02302			

	00818	COUNTER-24	01319 02122	2
	00822	COUNTER-25	01320 02016	2
	00824	COUNTER-26	01321 02324	2
	00901	COUNTER-27	01322 02515	2
	00903	COUNTER-28	01323 02725	2
	00905	COUNTER-29	01324 02407	2
	00618	COUNTER-2	01222 02024	2
	00907	COUNTER-30	01325 02107	3
	00620	COUNTER-3	01223 01904 02415	2
	00622	COUNTER-4	01224 02712	1
	00624	COUNTER-5	01225	2
	00701	COUNTER-6	01301 02217	2
	00705	COUNTER-7	01302 02113	2
	00707	COUNTER-8	01303 02615	2
	00709	COUNTER-9	01304 01912	3
	00221	C	01108 01712 01713	
PARA	999999	END-OF-JOB	NO EXPLICIT REFERENCE	12
	00208	E	01023 01102 01102 01108 01112 01118 01213 01220 01706 01707 01708 01709	1
	00518	GROUP-CNT-LINE	02816	3
	00210	G	01022 01608 01616	1
	00401	HDR1	02810	1
	00408	HDR2	02811	1
PARA	02809	HEADINGS	"PERFORM"-ED AT: 01004	1
PARA	01002	INIT	NO EXPLICIT REFERENCE	1
	00203	INPUT-CARD	02813	4
PARA	01701	IS-MCE-MAE-DT-CON-2	"PERFORM"-ED AT: 01803 01925 02210 02520	

PARA	01705	IS-MCE-MAE-DT-CON-3	"PERFORM"-ED AT: 01804 02009 02219 02521	4
PARA	01711	IS-MCE-MAE-DT-CON-5	"PERFORM"-ED AT: 01815 02115 02211 02420	4
PARA	01715	IS-MCE-MAE-DT-CON-6	"PERFORM"-ED AT: 01806 02002 02212 02503 02522	5
PARA	01721	IS-MCE-MAE-DT-CON-8	"PERFORM"-ED AT: 01805 02001 02220 02421 02523	5
PARA	01413	IS-MCE-MAE-DT-DIAGNOSTICS	NO EXPLICIT REFERENCE	
PARA	02803	IS-MCE-MAE-DT-ELSE-RULE	"GO TO"-ED AT: 01922 02207 02417 02517 02802	5
PARA	02806	IS-MCE-MAE-DT-END	"GO TO"-ED AT: 01814 01823 01905 01913 01921 02008 02017 02025 02108 02114 02123 02206 02218 02303 02309 02317 02325 02408 02416 02502 02509 02516 02604 02610 02616 02622 02705 02713 02719 02801 028051	31
PARA	01801	IS-MCE-MAE-DT-GRP-1	"GO TO"-ED AT: 01617	1
PARA	01923	IS-MCE-MAE-DT-GRP-2	"GO TO"-ED AT: 01615	1
PARA	02208	IS-MCE-MAE-DT-GRP-3	"GO TO"-ED AT: 01609	1
PARA	02418	IS-MCE-MAE-DT-GRP-4	"GO TO"-ED AT: 01619	1
PARA	02518	IS-MCE-MAE-DT-GRP-5	"GO TO"-ED AT: 01611	1
PARA	01012	IS-MCE-MAE-DT-NOTES	NO EXPLICIT REFERENCE	
PARA	01404	IS-MCE-MAE-DT-OPTIONS	NO EXPLICIT REFERENCE	
PARA	01604	IS-MCE-MAE-DT-SPLIT	NO EXPLICIT REFERENCE	
SECT	01011	IS-MCE-MAE-DT	01011 "PERFORM"-ED AT: 01007	2
PARA	02815	LAST	"GO TO"-ED AT: 01006	1
FILE	00301	LISTING	DUPLICATE NAME 01004 02822	2

	00216	NUMBER	01118 01722 01723	3
	00304	PRINTER-LINE	02810 02811 02814 02816 02817 02818 02819 02820 02821	9
	00509	PRINT-LINE	02814	1
PARA	01005	READ-LOOP	"GO TO"-ED AT: 01009	1
SECT	02808	REPORTEX	02808	1
	00909	RESULT-VECTOR-1	01802 01807 01816 01824 01906 01914 01924 02003 02010 02018 02101 02109 02116 02124 02209 02213 02221 02304 02310 02318 02401 02409 02419 02422 02504 02511 02519 02524 02605 02611 02617 02623 02706 02714 02721	35
	00513	RL1	01019 01123 01124 01125 01201 01810 01819 01901 01909 01917 02004 02013 02021 02104 02110 02119 02202 02214 02224 02305 02313 02321 02404 02412 02423 02505 02512 02525 02606 02612 02618 02701 02709 02715 02722	35
	00514	RL2	01020 01202 01203 01204 01205 01206 01207 01208 01209 01210 01811 01820 01902 01910 01918 02005 02014 02022 02105 02111 02120 02203 02215 02225 02306 02314 02322 02405 02413 02424 02506 02513 02601 02607 02613 02619 02702 02710 02716 02723	40
	00614	RULE-COUNT-LINE-1	02817	1
	00703	RULE-COUNT-LINE-2	02818	1
	00717	RULE-COUNT-LINE-3	02819	1
	00806	RULE-COUNT-LINE-4	02820	1
	00820	RULE-COUNT-LINE-5	02821	1
	00223	RULE-ID	NO EXPLICIT REFERENCE	
	00512	RULE-NO	01212 02804 028141	3
	00910	RV-1	01702 01703 01704 01706 01707 01708 01709 01712 01713 01714 01716 01717 01718 01719 01722 01723 01724 02510 02720	19
	009081	TABLE-1	NO EXPLICIT REFERENCE	
	00218	VAL	NO EXPLICIT REFERENCE	
	00219	VALUE-1	01117 01612	2
	00912	WORK-VECTOR-1	01807 01809 01816 01818 01824 01825 01906 01908 01914 01916 02010 02012 02018 02020 02101 02103 02116 02118 02124 02201 02221 02223 02310 02312 02318 02320 02401 02403 02409 02411 02623 02625 02706 02708	34
PARA	02812	WRITE-LISTING	"PERFORM"-ED AT:	

01008 1

00516 WS-CARD-IMAGE 02813 1

00913 WV-1

01808 01808 01817 01907 01915 02011 02011 02011 02019 02019
02102 02102 02102 02117 02125 02222 02311 02311 02319 02402
02402 02410 02410 02410 02410 02624 02707 27

00212 X

01107 01201 01202 01202 01202 01203 01203 01203 01204 01204
01204 01205 01205 01205 01206 01206 01206 01207 01207 01207
01208 01208 01208 01209 01209 01209 01210 01210 01210 01212
01214 01214 01214 01214 01214 01215 01215 01215 01215 01215
01216 01216 01216 01216 01216 01216 01216 01217 01217 01217
01218 01218 01218 01218 01218 01218 01219 01221 01222 01223
01224 01225 01301 01302 01303 01304 01305 01306 01307 01308
01309 01310 01311 01312 01313 01314 01315 01316 01317 01318
01319 01320 01321 01322 01323 01324 01325 01605 88

NO. OF DATA-NAMES 000069
NO. OF PARAGRAPHS 000024
NO. OF FILES DECL 000002

RESERVED WORD LISTING

NAME	REFERENCED ON:	
		88
00000000011111111112222222223E	01019	1
00000100000000000000000000000000	01401	1
06110542027323803840125373511 5	01402	1
12345678901234567890123456789 0L	01020	1
15	01104 01707	2
1	01213 01220 01702 01702 01703 01704 01706 01712 01716 01722 01812 01813 01821 01822 01903 01904 01907 01911 01912 01919 01920 02006 02007 02015 02016 02023 02024 02106 02107 02112 02113 02121 02122 02125 02204 02205 02216 02217 02301 02302 02307 02308 02315 02316 02323 02324 02406 02407 02414 02415 02425 02501 02507 02508 02514 02515 02602 02603 02608 02609 02614 02615 02620 02621 02624 02703 02704 02711 02712 02717 02718 02720 02724 02725 02805 02810 02814	77
20	01023 01103 01706	3
25	01117	1
2=3	01107	1
2	01605 01703 01706 01707 01707 01708 01709 01713 01717 01723 01915 02410 02707 02811 02816 02817 02818 02819 02820 02821	20
30	01102 01105 01708	3
35	01106 01709	2
3	01605 01704 01708 01712 01713 01714 01714 01718 01724 01808 02011 02019 02102 02410	14
45	01118	1
4	01709 01716 01717 01718 01719 01719 01817 02011 02102 02117 02222 02311 02319 02402 02410 02510	16
50	01022 01112	2
5	01722 01723 01724 01808 02011 02019 02102 02311 02402 02410	10
65	01108	1
68	01107	1
97	01107 01605	2

POSITIVE	01119 01722 02005	2
PRE-COMPLE	01015 01603 01102 01107 01108 01112 01117 01119 01605 01608 01611 01616 01702 01703 01706 01707 01708 01709 01712	1
PROBABILITY	01401 01717 01719 01719 01723 01723 01728 01818 01822 01908 01912 02020 02101 02109 02111 02201 02213	1
READ	01006 02114 02120 02203 02411 02505 02508 02511 02520 02521	1
RELEASE	02822	1
RUEL-03	01825	1
RULE-01	01818 01119 01712 01717 01722 01724	1
RULE-02	02020 02811 02819	1
RULE-03	02411 01102 01117 01123 01124 01125 01201 01202 01203 01204	1
RULE-04	02708 01204 01207 01208 01209 01210 01212	1
RULE-06	02213	1
RULE-07	02109 01119 01125 01201 01202 01203 01204 01205 01206 01207 01208 01209 01210 01212 01702 01703 01706 01707 01708	1
RULE-08	02611 01713 01718 01716 01717 01718 01719 01722 01723 01724 01807 01808 01810 01811 01814 01817 01819 01820	1
RULE-09	01908 01912 01914 01916 01917 01918 01919 01920 01921 01922 02010 02011 02012 02013 02015 02017	1
RULE-10	02617 02101 02102 02108 02109 02110 02111 02112 02113 02114 02201 02202 02203 02204 02205 02206	1
RULE-11	02504 02502 02504 02506 02507 02508 02509 02510 02511 02512 02513 02514 02515	1
RULE-12	02524 02501 02502 02503 02504 02505 02506 02507 02508 02509 02510 02511 02512 02513	1
RULE-13	02201 02203 02204 02205 02206 02207 02208 02209 02210 02211 02212 02213 02214 02215 02216 02217 02218	1
RULE-14	02312 01016	1
RULE-15	02422 01119 01722	1
RULE-16	01916 02605 01611	2
RULE-18	02003 01813 01819 01818	1
RULE-19	02304 01109 01719	1
RULE-20	01809 01803 01804	1
RULE-21	02714 01708	1
RULE-22	02625 01802 01803 01803 01804 01805 01806 01815 01825 02001	1
RULE-23	02223 02115 02210 02211 02213 02219 02220 02226 02231 02232	1
RULE-24	02118	1

XX!	01213 01218	2
XXXXXXXXX	01128	1
XXXXXXXXXX	01124 01125	2
YN--Y-NNY-Y-N-YY-N-Y---NN-Y--N	01117	1
Y-------Y--Y------Y-----YY-Y--	01103	1
Y-------YY--Y-Y--YY----Y-Y-Y--	01119	1
Y----Y--Y----Y----------Y-Y-	01109	1
Y-YNYY-NYNNN-YNYN-YYNNY--YN-Y-	01022	1
Y-YY---Y--------Y--Y-YY----Y-	01024	1
YYNNYNYNYNYNYNYYNYNYNNNYYNYNNY	01107	1
ZERO	01121 01808 01817 01907 01915 02011 02019 02102 02117 02125 02222 02311 02319 02402 02410 02510 02624 02707 02720	19
ZEROS	01802 01924 02209 02419 02519	1
	[illegible]	1
	[illegible]	1
	[illegible]	1
	[illegible]	1
	[illegible]	1
	[illegible]	1
	[illegible]	1
	[illegible]	1
	[illegible]	1
	[illegible]	2
	[illegible]	1
	[illegible]	1
	[illegible]	1
	[illegible]	1
	[illegible]	0

READ PARA PROCESSOR TIME = 0000690 SECS
SCANNER PROCESSOR TIME = 0004180 SECS
OVERALL ELAPSED TIME = 0009010 SECS
OVERALL PROCESSOR TIME = 0006081 SECS
OVERALL I - O TIME = 0005466 SECS
NO. RECDS RELEASED TO SORT 1297

VI-4-3 PROGRAM COMPILATION.

This part contains the actual COBOL compilation which can be initiated by the precompiler on an optional basis. Three different options can be used:

1. Compile for syntax (COMSYN)
2. Compile and Run (COMRUN)
3. Compile for Library(COMLIB)

Here the second option was illustrated.

```
00101    IDENTIFICATION DIVISION.                                          DT-EXAMP  0002-0001
00102    PROGRAM-ID. "TEST MCE-MAE".                                       DT-EXAMP  0002-0001
00103    AUTHOR. A PINHAS.                                                 DT-EXAMP  0002-0001
00104    DATE-WRITTEN. 28 DEC 1970.                                        DT-EXAMP  0002-0001
00105    DATE-COMPILED.  11- 1-71, 11:30 AM.                               DT-EXAMP  0002-0001
00106    REMARKS. THIS PROGRAM WAS WRITTEN MANUALLY SIMULATING A           DT-EXAMP  0002-0001
00107             SUGGESTED DECISION TABLE PRE-COMPILER. THE PRE-          DT-EXAMP  0002-0001
00108             COMPILATION PASS WAS PRESUMABLY DONE WITH THE            DT-EXAMP  0002-0001
00109             FOLLOWING OPTIONS SET:                                   DT-EXAMP  0002-0001
00110                1. INPUT FROM CARDS                                   DT-EXAMP  0002-0001
00111                2. OUTPUT LIST-1 PUNCH-L                              DT-EXAMP  0002-0001
00112                3. TEST    CT-TEST                                    DT-EXAMP  0002-0001
00113                4.         FREQ                                       DT-EXAMP  0002-0001
00114                5.         COST                                       DT-EXAMP  0002-0001
                                                                          PROCEDURE 0002 SIZE 0089
                                                                          PROCEDURE 0004 SIZE 0012
                                                                          PROCEDURE 0005 SIZE 0043
                                                                          PROCEDURE 0006 SIZE 0046

00115    ENVIRONMENT DIVISION.                                             DT-EXAMP  0007-0001
00116    CONFIGURATION SECTION.                                            DT-EXAMP  0007-0001
00117    SOURCE-COMPUTER. B-5500.                                          DT-EXAMP  0007-0001
00118    OBJECT-COMPUTER. B-5500.                                          DT-EXAMP  0007-0001
00119    INPUT-OUTPUT SECTION.                                             DT-EXAMP  0007-0001
00120    FILE-CONTROL.                                                     DT-EXAMP  0007-0001
00121        SELECT CARDIN ASSIGN TO CARD-READER.                          DT-EXAMP  0007-0001
   PRT 0072
   PRT 0073
   PRT 0074
   PRT 0075
   PRT 0076
00122        SELECT LISTING ASSIGN TO PRINTER.                             DT-EXAMP  0007-0001
   PRT 0077
   PRT 0100
   PRT 0101
   PRT 0102
   PRT 0103
00123    DATA DIVISION.                                                    DT-EXAMP  0007-0001
00124    FILE SECTION.                                                     DT-EXAMP  0007-0001
00125    FD  CARDIN                                                        DT-EXAMP  0007-0001
00201        VALUE OF ID IS "CARDS"                                        DT-EXAMP  0007-0001
00202        DATA RECORD IS INPUT-CARD.                                    DT-EXAMP  0007-0001
00203    01  INPUT-CARD      SIZE 80.                                      DT-EXAMP  0007-0001
00204        02  A           PICTURE 9.                                    DT-EXAMP  0007-0001
00205        02  FILLER      PICTURE X(5).                                 DT-EXAMP  0007-0001
00206        02  B           PICTURE 9.                                    DT-EXAMP  0007-0001
00207        02  FILLER      PICTURE X(5).                                 DT-EXAMP  0007-0001
00208        02  E           PICTURE 99.   ← VJ=[20,15,30,35]        ECV   DT-EXAMP  0007-0001
00209        02  FILLER      PICTURE X(5).                                 DT-EXAMP  0007-0001
00210        02  G           PICTURE X(4). ← PINK VJ=[Y,N]           LCV   DT-EXAMP  0007-0001
00211        02  FILLER      PICTURE X(5).                                 DT-EXAMP  0007-0001
00212        02  X           PICTURE 99.   ← VALUES FROM 00 TO 99          DT-EXAMP  0007-0001
00213        02  FILLER      PICTURE X(5).                                 DT-EXAMP  0007-0001
00214        02  COULOR      PICTURE X(5). ← VJ=[BLUE ,RED  ,GREEN,WHITE]  DT-EXAMP  0007-0001
00215        02  FILLER      PICTURE X(5).                                 DT-EXAMP  0007-0001
00216        02  NUMBER      PICTURE 999.  ← VJ=[POSITIVE,NEGATIVE,ZERO]   DT-EXAMP  0007-0001
00217        02  FILLER      PICTURE X(5).                                 DT-EXAMP  0007-0001
00218        02  VAL         PICTURE XX.                                   DT-EXAMP  0007-0001
00219            88 VALUE-1  VALUE "AA".   ← AA    VJ=[Y,N]               DT-EXAMP  0007-0001
00220        02  FILLER      PICTURE X(5).                                 DT-EXAMP  0007-0001
00221        02  C           PICTURE XXX.  ← VJ=[NUMERIC,ALPHABETI,        DT-EXAMP  0007-0001
```

```
00222          02  FILLER      PICTURE X(9). *                    ALPHANUMERIC]  DT-EXAMP    0007-0001
00223          02  RULE-ID     PICTURE X(9).                                     DT-EXAMP    0007-0001
00224                                                                             DT-EXAMP    0007-0001
00225                                                                             DT-EXAMP    0007-0001
00301      FD  LISTING                                                            DT-EXAMP    0007-0001
00302          VALUE OF ID IS "REPORT"                                            DT-EXAMP    0007-0001
00303          DATA RECORD IS PRINTER-LINE.                                       DT-EXAMP    0007-0001
00304      01  PRINTER-LINE    SIZE 132.                                         DT-EXAMP    0007-0001
00305                                                                             DT-EXAMP    0007-0001
00400      WORKING-STORAGE SECTION .                                              DT-EXAMP    0007-0001
    PRT 0104
00401      01  HDR1.                                                                         0007-0001
    PRT 0105
00402          02  FILLER              PICTURE X(9)  VALUE SPACES.,                      0007-0001
00403          02  FILLER              PICTURE X(14) VALUE                               0007-0004
00404                                                "RULE SATISFIED".                   0007-0004
00405          02  FILLER              PICTURE X(50) VALUE SPACES.                       0007-0006
00406          02  FILLER              PICTURE X(10) VALUE "INPUT-CARD".                 0007-0009
00407          02  FILLER              PICTURE X(49) VALUE SPACES.                       0007-0012
00408      01  HDR2.                                                                     0007-0015
    PRT 0106
00409          02  FILLER              PICTURE X(40) VALUE SPACES.                       0007-0015
00410          02  FILLER              PICTURE X(6)  VALUE "A     ".                     0007-0018
00411          02  FILLER              PICTURE X(6)  VALUE "B     ".                     0007-0020
00412          02  FILLER              PICTURE X(7)  VALUE "E     ".                     0007-0023
00501          02  FILLER              PICTURE X(9) VALUE "G       ".                    0007-0026
00502          02  FILLER              PICTURE X(6) VALUE "X     ".                      0007-0029
00503          02  FILLER              PICTURE X(8) VALUE "COULOR  ".                    0007-0032
00504          02  FILLER              PICTURE X(8) VALUE "NUMBER  ".                    0007-0034
00505          02  FILLER              PICTURE X(10) VALUE "VALUE-1    ".                0007-0037
00506          02  FILLER              PICTURE X(6)  VALUE "C     ".                     0007-0040
00507          02  FILLER              PICTURE X(13) VALUE "INTENDED RULE".              0007-0043
00508          02  FILLER              PICTURE X(13) VALUE SPACES.                       0007-0045
00509      01  PRINT-LINE.                                                               0007-0049
    PRT 0107
00510          02  FILLER              PICTURE X(9)  VALUE SPACES.                       0007-0049
00511          02  FILLER              PICTURE X(15) VALUE                               0007-0052
005111                                                "SATISFIES RULE=".                0007-0052
00512          02  RULE-NO.                                                             0007-0054
00513              03  RL1             PICTURE X.                                       0007-0054
00514              03  RL2 PICTURE X. 03 FILLER PC XX VA SPACES.                        0007-0054
00515          02  FILLER              PICTURE X(11) VALUE SPACES.                      0007-0057
00516          02  WS-CARD-IMAGE       PICTURE X(80).                                   0007-0060
00517          02  FILLER              PICTURE X(13) VALUE SPACES.                      0007-0060
00518      01  GROUP-CNT-LINE.                                                           0007-0063
    PRT 0110
00519          02  FILLER              PICTURE X(16) VALUE                              0007-0063
00520                                                "      CNT1 = ".                  0007-0063
00521          02  CNT1                PICTURE 999 VALUE 0.                             0007-0066
00522          02  FILLER              PICTURE X(17) VALUE                              0007-0069
00523                                                "      CNT2 = ".                  0007-0069
00524          02  CNT2                PICTURE 999 VALUE 0.                             0007-0072
00525          02  FILLER              PICTURE X(17) VALUE                              0007-0075
00601                                                "      CNT3 = ".                  0007-0075
00602          02  CNT3                PICTURE 999 VALUE 0.                             0007-0078
00603          02  FILLER              PICTURE X(17) VALUE                              0007-0081
00604                                                "      CNT4 = ".                  0007-0081
00605          02  CNT4                PICTURE 999 VALUE 0.                             0007-0085
00606          02  FILLER              PICTURE X(17) VALUE                              0007-0088
00607                                                "      CNT5 = ".                  0007-0088
00608          02  CNT5                PICTURE 999 VALUE 0.                             0007-0091
```

```
00609          02  FILLER                  PICTURE X(17) VALUE                    0007-0094
00610                                                  "          CNT6 = ".        0007-0094
00611          02  CNT6                    PICTURE 999 VALUE 0.                   0007-0097
00612          02  FILLER                  PICTURE X(13) VALUE SPACES.            0007-0101
00613                                                                              0007-0101
00614      01  RULE-COUNT-LINE-1.                                                  0007-0104
    PRT 0111
00615          02  FILLER  PICTURE X(22) VALUE "          COUNTER-1  = ".         0007-0104
00616          02  COUNTER-1   PICTURE   9(3) VALUE 0.                            0007-0107
00617          02  FILLER  PICTURE X(17) VALUE "    COUNTER-2  = ".               0007-0110
00618          02  COUNTER-2   PICTURE   9(3) VALUE 0.                            0007-0113
00619          02  FILLER  PICTURE X(17) VALUE "    COUNTER-3  = ".               0007-0116
00620          02  COUNTER-3   PICTURE   9(3) VALUE 0.                            0007-0119
00621          02  FILLER  PICTURE X(17) VALUE "    COUNTER-4  = ".               0007-0122
00622          02  COUNTER-4   PICTURE   9(3) VALUE 0.                            0007-0125
00623          02  FILLER  PICTURE X(17) VALUE "    COUNTER-5  = ".               0007-0128
00624          02  COUNTER-5   PICTURE   9(3) VALUE 0.                            0007-0131
00625          02  FILLER  PICTURE X(17) VALUE "    COUNTER-6  = ".               0007-0134
00701          02  COUNTER-6   PICTURE 9(3) VALUE 0.                              0007-0137
00702          02  FILLER  PICTURE X(3) VALUE SPACES.                             0007-0140
00703      01  RULE-COUNT-LINE-2.                                                  0007-0142
    PRT 0112
00704          02  FILLER  PICTURE X(22) VALUE "          COUNTER-7  = ".         0007-0142
00705          02  COUNTER-7   PICTURE 9(3) VALUE 0.                              0007-0145
00706          02  FILLER  PICTURE X(17) VALUE "    COUNTER-8  = ".               0007-0148
00707          02  COUNTER-8   PICTURE 9(3) VALUE 0.                              0007-0151
00708          02  FILLER  PICTURE X(17) VALUE "    COUNTER-9  = ".               0007-0154
00709          02  COUNTER-9   PICTURE 9(3) VALUE 0.                              0007-0157
00710          02  FILLER  PICTURE X(17) VALUE "    COUNTER-10 = ".               0007-0160
00711          02  COUNTER-10  PICTURE 9(3) VALUE 0.                              0007-0163
00712          02  FILLER  PICTURE X(17) VALUE "    COUNTER-11 = ".               0007-0166
00713          02  COUNTER-11  PICTURE 9(3) VALUE 0.                              0007-0169
00714          02  FILLER  PICTURE X(17) VALUE "    COUNTER-12 = ".               0007-0172
00715          02  COUNTER-12  PICTURE 9(3) VALUE 0.                              0007-0174
00716          02  FILLER  PICTURE X(3)  VALUE SPACES.                            0007-0178
00717      01  RULE-COUNT-LINE-3.                                                  0007-0180
    PRT 0113
00718          02  FILLER  PICTURE X(22) VALUE "          COUNTER-13 = ".         0007-0180
00719          02  COUNTER-13  PICTURE 9(3) VALUE 0.                              0007-0183
00720          02  FILLER  PICTURE X(17) VALUE "    COUNTER-14 = ".               0007-0186
00721          02  COUNTER-14  PICTURE 9(3) VALUE 0.                              0007-0189
00722          02  FILLER  PICTURE X(17) VALUE "    COUNTER-15 = ".               0007-0192
00723          02  COUNTER-15  PICTURE 9(3) VALUE 0.                              0007-0195
00724          02  FILLER  PICTURE X(17) VALUE "    COUNTER-16 = ".               0007-0198
00725          02  COUNTER-16  PICTURE 9(3) VALUE 0.                              0007-0201
00801          02  FILLER  PICTURE X(17) VALUE "    COUNTER-17 = ".               0007-0204
00802          02  COUNTER-17  PICTURE 9(3) VALUE 0.                              0007-0207
00803          02  FILLER  PICTURE X(17) VALUE "    COUNTER-18 = ".               0007-0210
00804          02  COUNTER-18  PICTURE 9(3) VALUE 0.                              0007-0213
00805          02  FILLER  PICTURE X(3) VALUE SPACES.                             0007-0216
00806      01  RULE-COUNT-LINE-4.                                                  0007-0219
    PRT 0114
00807          02  FILLER  PICTURE X(22) VALUE "          COUNTER-19 = ".         0007-0219
00808          02  COUNTER-19  PICTURE 9(3) VALUE 0.                              0007-0221
00809          02  FILLER  PICTURE X(17) VALUE "    COUNTER-20 = ".               0007-0225
00810          02  COUNTER-20  PICTURE 9(3) VALUE 0.                              0007-0227
00811          02  FILLER  PICTURE X(17) VALUE "    COUNTER-21 = ".               0007-0231
00812          02  COUNTER-21  PICTURE 9(3) VALUE 0.                              0007-0233
00813          02  FILLER  PICTURE X(17) VALUE "    COUNTER-22 = ".               0007-0237
00814          02  COUNTER-22  PICTURE 9(3) VALUE 0.                              0007-0239
00815          02  FILLER  PICTURE X(17) VALUE "    COUNTER-23 = ".               0007-0243
```

```
00816          02  COUNTER-23  PICTURE 9(3) VALUE 0.                                0007-0245
00817          02  FILLER  PICTURE X(17) VALUE "    COUNTER-24 = ".                0007-0249
00818          02  COUNTER-24  PICTURE 9(3) VALUE 0.                                0007-0251
00819          02  FILLER  PICTURE X(3) VALUE SPACES.                               0007-0255
00820      01  RULE-COUNT-LINE-5.                                                   0007-0257
    PRT 0115
00821          02  FILLER  PICTURE X(22) VALUE "        COUNTER-25 = ".             0007-0257
00822          02  COUNTER-25  PICTURE 9(3) VALUE 0.                                0007-0260
00823          02  FILLER  PICTURE X(17) VALUE "    COUNTER-25 = ".                 0007-0263
00824          02  COUNTER-26  PICTURE 9(3) VALUE 0.                                0007-0266
00825          02  FILLER  PICTURE X(17) VALUE "    COUNTER-27 = ".                 0007-0269
00901          02  COUNTER-27  PICTURE 9(3) VALUE 0.                                0007-0272
00902          02  FILLER  PICTURE X(17) VALUE "    COUNTER-28 = ".                 0007-0275
00903          02  COUNTER-28  PICTURE 9(3) VALUE 0.                                0007-0278
00904          02  FILLER  PICTURE X(17) VALUE "    COUNTER-29 = ".                 0007-0281
00905          02  COUNTER-29  PICTURE 9(3) VALUE 0.                                0007-0284
00906          02  FILLER  PICTURE X(17) VALUE "    COUNTER-30 = ".                 0007-0287
00907          02  COUNTER-30  PICTURE 9(3) VALUE 0.                                0007-0290
00908          02  FILLER  PICTURE X(3) VALUE SPACES.                               0007-0293
000001     01  TABLE-1.                                                             0007-0296
    PRT 0116
00909          02  RESULT-VECTOR-1.                                                0007-0296
00910              03  RV-1    PICTURE  9  OCCURS 5 TIMES.                         0007-0296
00911                                                                             0007-0296
00912          02  WORK-VECTOR-1.                                                  0007-0296
00913              03  WV-1    PICTURE  9  OCCURS 5 TIMES.                         0007-0296
                                                                                  PROCEDURE 0003 SIZE 0112
    PRT 0117
                                                                                  PROCEDURE 0007 SIZE 0296
    PRT 0120
    PRT 0121
    PRT 0122
    PRT 0123
    PRT 0124
01001      PROCEDURE DIVISION.                                                    0009-0001
01002      INIT.                                                                  0009-0001
    PRT 0125
    PRT 0126
                                                                                  PROCEDURE 0009 SIZE 0001
01003          OPEN INPUT CARDIN.                                                 0010-0001
01004          OPEN OUTPUT LISTING. PERFORM HEADINGS.                             0010-0003
    PRT 0127
01005      READ-LOOP.                                                             0010-0005
    PRT 0130
                                                                                  PROCEDURE 0010 SIZE 0007
01006          READ  CARDIN AT END GO TO LAST.                                    0011-0003
    PRT 0131
01007          PERFORM IS-MCE-MAE-DT.                                             0011-0004
    PRT 0132
01008          PERFORM WRITE-LISTING.                                             0011-0006
    PRT 0133
01009          GO TO READ-LOOP.                                                   0011-0007
01010                                                                             0011-0007
01011      IS-MCE-MAE-DT SECTION.                                                 0011-0008
01012      IS-MCE-MAE-DT-NOTES.                                                   0011-0008
                                                                                  PROCEDURE 0011 SIZE 0008
01013          NOTE THE ORIGINAL DECISIO TABLE TOGETHER WITH THE REQUIRED         0012-0001
01014               DIAGNOSTICS (ACCORDING TO THE SPECIFIED OPTIONS IN THE        0012-0001
01015               # CARD) ARE LISTED HERE FOR YOUR INSPECTION. IF CORRECT-       0012-0001
01016               IONS ARE NECESSARY EXECUTE ANOTHER PRE-COMPILE RUN.           0012-0001
01017                                                                             0012-0001
```

```
TABLE-ID. "IS-MCE-MAE-DT" MCE-MAE.                                                     0012-0001
RL1                                 000000000111111111122222222223E   COST             0012-0001
RL2                                 1234567890123456789012345678901L                   0012-0001
CONDITIONS                                                                             0012-0001
L IF G="PINK"                       Y-YNYY-NYNNN-YNYN-YYNNY--YN-Y-    50               0012-0001
E IF A***B                                                            20               0012-0001
         <                          Y-YY---Y---------Y--Y-YY----Y-                     0012-0001
         =                          -Y--Y-Y--Y------Y-Y------Y---Y                     0012-0001
         >                          -----Y-----Y-Y-Y---Y----Y-----                     0012-0001
E IF E =                                                              30               0012-0001
             20                     Y-------Y--Y------Y-----YY-Y--                     0012-0001
             15                     ----Y----Y-----------Y-Y----YY                     0012-0001
             30                     ------YY----YY--Y--Y----------                     0012-0001
             35                     -Y------------------Y-Y-------                     0012-0001
L IF (X**2-3)=97                    YYNNYNYNYNYNYNYYNYNYNNNYYNYNNY    68               0012-0001
E IF C IS                                                             65               0012-0001
             NUMERIC                Y--Y----Y----Y------------Y-Y-                     0012-0001
             ALPHABETIC             ----Y-----Y-Y-YY-------Y-Y----                     0012-0001
             ALPHANUMERIC           ------------------Y---Y-------                     0012-0001
E IF COULOR =                                                         50               0012-0001
                "BLUE "             ----Y--Y----------------------                     0012-0001
                "RED  "             -Y-Y-Y--Y-YY-------Y----------                     0012-0001
                "GREEN"             ------------Y--YYYY-YY-----Y--                     0012-0001
                "WHITE"             ------Y--Y--------------------                     0012-0001
L IF VALUE=1                        YN--Y-NNY-Y-N-YY-N-Y---NN-Y--N    25               0012-0001
E IF NUMBER IS                                                        45               0012-0001
             POSITIVE               Y-------YY--Y-Y--YY----Y-Y-Y--                     0012-0001
             NEGATIVE               ---Y---Y--YY---Y-----YY---Y---                     0012-0001
             ZERO                   ----Y-Y---------Y---Y---------                     0012-0001
ACTIONS                                                                                0012-0001
L MOVE "0" TO RL1                   XXXXXXXXX                                          0012-0001
L MOVE "1" TO RL1                            XXXXXXXXXX                                0012-0001
L MOVE "2" TO RL1                                      XXXXXXXXXX                      0012-0001
L MOVE "3" TO RL1                                                X                     0012-0001
L MOVE "1"TO RL2                    X         X         X                              0012-0001
L MOVE "2" TO RL2                    X         X         X                             0012-0001
L MOVE "3" TO RL2                     X         X         X                            0012-0001
L MOVE "4" TO RL2                      X         X         X                           0012-0001
L MOVE "5" TO RL2                       X         X         X                          0012-0001
L MOVE "6" TO RL2                        X         X         X                         0012-0001
L MOVE "7" TO RL2                         X         X         X                        0012-0001
L MOVE "8" TO RL2                          X         X         X                       0012-0001
L MOVE "9" TO RL2                           X         X         X                      0012-0001
L MOVE "ELSE" TO RULE-NO                                          X                    0012-0001
E ADD 1 TO                                                                             0012-0001
     CNT1                           X   X   X      X   X                               0012-0001
          CNT2                       X    X     X    X     XX    X                     0012-0001
          CNT3                        X  X       X    X   X  X  X                      0012-0001
          CNT4                                X   X           X                        0012-0001
          CNT5                         X   X X X    X   XX     X                       0012-0001
          CNT6                                                    X                    0012-0001
 E ADD 1 TO                                                                            0012-0001
          COUNTER-1                 X                                                  0012-0001
          COUNTER-2                  X                                                 0012-0001
          COUNTER-3                   X                                                0012-0001
          COUNTER-4                    X                                               0012-0001
          COUNTER-5                     X                                              0012-0001
          COUNTER-6                      X                                             0012-0001
          COUNTER-7                       X                                            0012-0001
          COUNTER-8                        X                                           0012-0001
          COUNTER-9                         X
```

```
01305               COUNTER-10       X                                        0012-0001
01306               COUNTER-11        X                                       0012-0001
01307               COUNTER-12         X                                      0012-0001
01308               COUNTER-13          X                                     0012-0001
01309               COUNTER-14           X                                    0012-0001
01310               COUNTER-15            X                                   0012-0001
01311               COUNTER-16             X                                  0012-0001
01312               COUNTER-17              X                                 0012-0001
01313               COUNTER-18               X                                0012-0001
01314               COUNTER-19                X                               0012-0001
01315               COUNTER-20                 X                              0012-0001
01316               COUNTER-21                  X                             0012-0001
01317               COUNTER-22                   X                            0012-0001
01318               COUNTER-23                    X                           0012-0001
01319               COUNTER-24                     X                          0012-0001
01320               COUNTER-25                      X                         0012-0001
01321               COUNTER-26                       X                        0012-0001
01322               COUNTER-27                        X                       0012-0001
01323               COUNTER-28                         X                      0012-0001
01324               COUNTER-29                          X                     0012-0001
01325               COUNTER-30                           X                    0012-0001
01401          PROBABILITY          000001000000000000000000000000            0012-0001
01402          RULE-DISTRIBUTION    061105420273238038401253735115            0012-0001
01403          FINI.                                                          0012-0001
01404      IS-MCE-MAE-DT-OPTIONS.                                             0012-0001
   PRT 0134                                                                   PROCEDURE 0012 SIZE 0003
01405          NOTE FOR THIS DECISION TABLE THE FOLLOWING OPTIONS WERE SET:   0013-0001
01406               1 INPUT OPTIONS:  CARD, TAPELIB                           0013-0001
01407               2 OUTPUT OPTIONS: LIST-1, SINGLE                          0013-0001
01408               3 TESTING OPTIONS: CT-TEST                                0013-0001
01409               4 EFFICENCY OPTIONS: FREQ, COST                           0013-0001
01410               5 COMPILE OPTIONS: RUNCOM                                 0013-0001
01411          FOR OTHER OPTIONS THE DEFAULT VALUES WERE USED.                0013-0001
01412                                                                         0013-0001
01413      IS-MCE-MAE-DT-DIAGNOSTICS.                                         0013-0001
   PRT 0135                                                                   0013-0001
01414          NOTE                                                           PROCEDURE 0013 SIZE 0003
                                                                              0014-0001
01415          1 POSSIBLE REDUNDANCIES:                                       0014-0001
01416                 NONE                                                    0014-0001
01417                                                                         0014-0001
01418          2 POSSIBLE CONTRADICTIONS:                                     0014-0001
01419            FOR RULE  NO-1      NO-13      NO-18      NO-3     NO-3      0014-0001
01420            AND RULE  NO-9      NO-18      NO-24      NO-29    NO-26     0014-0001
01421                      Y         -          -          Y        Y         0014-0001
01422                      <         <          <          <        <         0014-0001
01423                      20        30         15         15       35        0014-0001
01424                      Y         Y          Y          N        N         0014-0001
01425                      NUMERIC   ALPHABETIC ALPHABETIC NUMERIC  ALPHANUERC 0014-0001
01501                      RED       GREEN      GREEN      -        -         0014-0001
01502                      Y         N          N          -        -         0014-0001
01503                      POSITIVE  POSITIVE   POSITIVE   -        NEGATIVE  0014-0001
01504                                                                         0014-0001
01505            FOR RULE  NO-6      NO-19                                    0014-0001
01506            AND RULE  NO-14     NO-26                                    0014-0001
01507                      Y         Y                                        0014-0001
01508                      >         =                                        0014-0001
01509                      30        20                                       0014-0001
01510                      N         N
```

```
01511                    NUMERIC  ALPHABETIC                                0014-0001
01512                    RED      GREEN                                    0014-0001
01513                    -        -                                        0014-0001
01514                    -        POSITIVE                                 0014-0001
01515                                                                      0014-0001
01516          3 COMPLETNESS: ACHIEVED BY THE ELSE RULE                    0014-0001
01517                                                                      0014-0001
01518          4 GENERAL:                                                  0014-0001
01519              THE SUM OF THE PROBABILITY IS NOT 1                     0014-0001
01520                                                                      0014-0001
01521          *** NO FATAL ERRORS WERE FOUND                              0014-0001
01522          *** IN OPTIMIZING THE FOLLOWING PROCEDURES WERE USED:       0014-0001
01523              1  CONDITION PARTIONING BY LCE  FOLLOWD BY ECE         0014-0001
01524              2  TABLE SPLITING BY BRANCHING DONE ON THE LCE          0014-0001
01525              3  OPTIMIZING THE LCE BY A TREE METHOD                  0014-0001
01601              4 OPTIMIZIN THE ECE FIRST BY PROBABILITIES FOLOWED BY   0014-0001
01602                COST.                                                 0014-0001
01603                                                                      0014-0001
01604      IS-MCE-MAE-DT-SPLIT.                                            0014-0001
     PRT 0136                                                             PROCEDURE 0014 SIZE 0003
01605          IF (X**2 - 3) = 97                                          0015-0006
01606             NEXT SENTENCE                                            0015-0006
01607          ELSE                                                        0015-0007
01608          IF G="PINK"                                                 0015-0007
01609             GO TO IS-MCE-MAE-DT-GRP-3                                0015-0011
01610          ELSE                                                        0015-0011
     PRT 0142                                                             0015-0013
01611             GO TO IS-MCE-MAE-DT-GRP-5.
     PRT 0143                                                             0015-0014
01612          IF VALUE=1                                                  0015-0015
01613             NEXT SENTENCE                                            0015-0019
01614          ELSE                                                        0015-0020
01615             GO TO IS-MCE-MAE-DT-GRP-2.
     PRT 0144                                                             0015-0021
01616          IF G="PINK"                                                 0015-0025
01617             GO TO IS-MCE-MAE-DT-GRP-1                                0015-0025
01618          ELSE
     PRT 0145                                                             0015-0027
01619             GO TO IS-MCE-MAE-DT-GRP-4.                               0015-0027
01620
     PRT 0146                                                             0015-0028
01701      IS-MCE-MAE-DT-CON-2.
     PRT 0147                                                             PROCEDURE 0015 SIZE 0031
01702          IF A < B MOVE 1 TO RV-1(1)  ELSE                            0016-0008
01703          IF A = B MOVE 2 TO RV-1(1)  ELSE                            0016-0030
01704                   MOVE 3 TO RV-1(1).                                 0016-0046
01705      IS-MCE-MAE-DT-CON-3.                                            0016-0062
     PRT 0150                                                             PROCEDURE 0016 SIZE 0064
01706          IF E = 20 MOVE 1 TO RV-1(2) ELSE                            0017-0008
01707          IF E = 15 MOVE 2 TO RV-1(2) ELSE                            0017-0031
01708          IF E = 30 MOVE 3 TO RV-1(2) ELSE                            0017-0054
01709          IF E = 35 MOVE 4 TO RV-1(2).                                0017-0077
01711      IS-MCE-MAE-DT-CON-5.                                            0017-0092
     PRT 0151                                                             PROCEDURE 0017 SIZE 0094
01712          IF C IS NUMERIC      MOVE 1 TO RV-1(3) ELSE                 0018-0003
01713          IF C IS ALPHABETIC   MOVE 2 TO RV-1(3) ELSE                 0018-0022
01714                               MOVE 3 TO RV-1(3).                     0018-0038
```

```
                                                                        0018-0053
01715 .    IS-MCE-MAE-DT-CON-6.
    PRT 0152                                                            PROCEDURE 0018 SIZE 0056
                                                                        0019-0005
01716          IF COULOR = "BLUE "  MOVE 1 TO RV-1(4) ELSE               0019-0025
01717          IF COULOR = "RED  "  MOVE 2 TO RV-1(4) ELSE               0019-0046
01718          IF COULOR = "GREEN"  MOVE 3 TO RV-1(4) ELSE               0019-0066
01719          IF COULOR = "WHITE"  MOVE 4 TO RV-1(4).                   0019-0081
01721      IS-MCE-MAE-DT-CON-8.
    PRT 0153                                                            PROCEDURE 0019 SIZE 0084
                                                                        0020-0007
01722          IF NUMBER IS POSITIVE MOVE 1 TO RV-1(5) ELSE              0020-0028
01723          IF NUMBER IS NEGATIVE MOVE 2 TO RV-1(5) ELSE              0020-0044
01724                                MOVE 3 TO RV-1(5).                  0020-0044
01725                                                                   0020-0060
01801    IS-MCE-MAE-DT-GRP-1.                                           PROCEDURE 0020 SIZE 0062
                                                                        0021-0001
01802        MOVE ALL ZEROS TO RESULT-VECTOR-1.                         0021-0003
01803            PERFORM IS-MCE-MAE-DT-CON-2.                           0021-0004
01804            PERFORM IS-MCE-MAE-DT-CON-3.                           0021-0006
01805            PERFORM IS-MCE-MAE-DT-CON-8.                           0021-0007
01806            PERFORM IS-MCE-MAE-DT-CON-6.                           0021-0009
01807                MOVE RESULT-VECTOR-1 TO WORK-VECTOR-1              0021-0022
01808                MOVE ZERO TO WV-1(3) WV-1(5)                       0021-0033
01809                   IF WORK-VECTOR-1 = "33020"          ← RULE-20   0021-0037
01810                       MOVE "2" TO RL1                             0021-0039
01811                       MOVE "0" TO RL2                             0021-0042
01812                       ADD   1 TO CNT1                             0021-0046
01813                       ADD   1  TO COUNTER-1                       0021-0051
01814                           GO TO IS-MCE-MAE-DT-END.
    PRT 0154                                                            0021-0052
01815            PERFORM IS-MCE-MAE-DT-CON-5.                           0021-0053
01816                MOVE RESULT-VECTOR-1 TO WORK-VECTOR-1              0021-0056
01817                MOVE ZERO TO WV-1(4)                               0021-0067
01818                    IF WORK-VECTOR-1 = "11101"      ← RULE-01      0021-0071
01819                        MOVE "0" TO RL1                            0021-0073
01820                        MOVE "1" TO RL2                            0021-0075
01821                        ADD   1  TO CNT1                           0021-0080
01822                        ADD   1  TO COUNTER-1                      0021-0084
01823                            GO TO IS-MCE-MAE-DT-END.               0021-0086
01824                MOVE RESULT-VECTOR-1 TO  WORK-VECTOR-1             0021-0088
01825                    IF WORK-VECTOR-1 ="22321"       ← RUEL-03      0021-0092
01901                        MOVE "0" TO RL1                            0021-0094
01902                        MOVE "3" TO RL2                            0021-0097
01903                        ADD   1  TO CNT1                           0021-0101
01904                        ADD   1  TO COUNTER-3                      0021-0106
01905                            GO TO IS-MCE-MAE-DT-END.               0021-0107
01906                MOVE RESULT-VECTOR-1 TO WORK-VECTOR-1              0021-0110
01907                MOVE ZERO TO WV-1(1)                               0021-0120
01908                    IF WORK-VECTOR-1 = "01121"      ← RULE-09      0021-0125
01909                        MOVE "0" TO RL1                            0021-0127
01910                        MOVE "9" TO RL2                            0021-0129
01911                        ADD   1  TO CNT1                           0021-0133
01912                        ADD   1  TO COUNTER-9                      0021-0138
01913                            GO TO IS-MCE-MAE-DT-END.               0021-0139
01914                MOVE RESULT-VECTOR-1 TO WORK-VECTOR-1              0021-0142
01915                MOVE ZERO TO WV-1(2)                               0021-0152
01916                    IF WORK-VECTOR-1 = "30232"      ← RULE-16      0021-0157
01917                        MOVE "1" TO RL1                            0021-0159
01918                        MOVE "6" TO RL2                            0021-0161
01919                        ADD   1  TO CNT1                           0021-0166
01920                        ADD   1  TO COUNTER-16
```

```
                                GO TO IS-MCE-MAE-DT-END.                                0021-0170
01921                       GO TO IS-MCE-MAE-DT-ELSE-RULE.          ← ELSE-RULE         0021-0171
01922
    PRT 0155                                                                            0021-0172
01923     IS-MCE-MAE-DT-GRP-2.                                                          PROCEDURE 0021 SIZE 0172
                                                                                        0022-0001
01924         MOVE ALL ZEROS TO RESULT-VECTOR-1.                                        0022-0003
01925             PERFORM IS-MCE-MAE-DT-CON-2.                                          0022-0004
02001             PERFORM IS-MCE-MAE-DT-CON-8.                                          0022-0006
02002             PERFORM IS-MCE-MAE-DT-CON-6.                                          0022-0007
02003                 IF RESULT-VECTOR-1 = "10031".               ← RULE-18         0022-0011
02004                     MOVE "1" TO RL1                                             0022-0013
02005                     MOVE "8" TO RL2                                             0022-0016
02006                     ADD  1  TO CNT2                                             0022-0021
02007                     ADD  1  TO COUNTER-18                                       0022-0025
02008                         GO TO IS-MCE-MAE-DT-END.                                0022-0026
02009             PERFORM IS-MCE-MAE-DT-CON-3.                                        0022-0028
02010                 MOVE RESULT-VECTOR-1 TO WORK-VECTOR-1                           0022-0052
02011                 MOVE ZERO TO WV-1(3) WV-1(4) WV-1(5)                            0022-0063
02012                     IF WORK-VECTOR-1 = "31000"              ← RULE-25         0022-0067
02013                         MOVE "2" TO RL1                                         0022-0069
02014                         MOVE "5" TO RL2                                         0022-0071
02015                         ADD  1  TO CNT2                                         0022-0076
02016                         ADD  1  TO COUNTER-25                                   0022-0080
02017                             GO TO IS-MCE-MAE-DT-END.                            0022-0082
02018                 MOVE RESULT-VECTOR-1 TO WORK-VECTOR-1                           0022-0095
02019                 MOVE ZERO TO WV-1(3) WV-1(5)                                    0022-0106
02020                     IF WORK-VECTOR-1 = "24020"              ← RULE-02         0022-0110
02021                         MOVE "0" TO RL1                                         0022-0112
02022                         MOVE "2" TO RL2                                         0022-0114
02023                         ADD  1  TO CNT2                                         0022-0119
02024                         ADD  1  TO COUNTER-2                                    0022-0124
02025                             GO TO IS-MCE-MAE-DT-END.                            0022-0125
02101                 MOVE RESULT-VECTOR-1 TO WORK-VECTOR-1                           0022-0149
02102                 MOVE ZERO TO WV-1(3) WV-1(4) WV-1(5)                            0022-0160
02103                     IF WORK-VECTOR-1 = "22000"              ← RULE-30         0022-0164
02104                         MOVE "3" TO RL1                                         0022-0166
02105                         MOVE "0" TO RL2                                         0022-0168
02106                         ADD  1  TO CNT2                                         0022-0173
02107                         ADD  1  TO COUNTER-30                                   0022-0178
02108                             GO TO IS-MCE-MAE-DT-END.                            0022-0179
02109                     IF RESULT-VECTOR-1 = "23043"            ← RULE-07         0022-0183
02110                         MOVE "0" TO RL1                                         0022-0185
02111                         MOVE "7" TO RL2                                         0022-0187
02112                         ADD  1  TO CNT2                                         0022-0192
02113                         ADD  1  TO COUNTER-7                                    0022-0197
02114                             GO TO IS-MCE-MAE-DT-END.                            0022-0198
02115             PERFORM IS-MCE-MAE-DT-CON-5.                                        0022-0199
02116                 MOVE RESULT-VECTOR-1 TO WORK-VECTOR-1                           0022-0202
02117                 MOVE ZERO TO WV-1(4)                                            0022-0213
02118                     IF WORK-VECTOR-1 = "12201"              ← RULE-24         0022-0217
02119                         MOVE "2" TO RL1                                         0022-0219
02120                         MOVE "4" TO RL2                                         0022-0221
02121                         ADD  1  TO CNT1                                         0022-0226
02122                         ADD  1  TO COUNTER-24                                   0022-0230
02123                             GO TO IS-MCE-MAE-DT-END.                            0022-0232
02124                 MOVE RESULT-VECTOR-1 TO WORK-VECTOR-1                           0022-0234
02125                 MOVE ZERO TO WV-1(1)                                            0022-0245
02201                     IF WORK-VECTOR-1 = "03231"              ← RULE-13         0022-0249
02202                         MOVE "1" TO RL1                                         0022-0251
02203                         MOVE "3" TO RL2                                         0022-0254
02204                         ADD  1  TO CNT2
```

```
02205                       ADD   1  TO COUNTER-13                                  0022-0258
02206                          GO TO IS-MCE-MAE-DT-END.                            0022-0263
02207                    GO TO IS-MCE-MAE-DT-ELSE-RULE.                            0022-0264
02208  IS-MCE-MAE-DT-GRP-3.                                                        0022-0265
                                                                                   PROCEDURE 0022 SIZE 0265
02209      MOVE ALL ZEROS TO RESULT-VECTOR-1.                                      0023-0001
02210          PERFORM IS-MCE-MAE-DT-CON-2.                                        0023-0003
02211          PERFORM IS-MCE-MAE-DT-CON-5.                                        0023-0004
02212          PERFORM IS-MCE-MAE-DT-CON-6.                                        0023-0006
02213                  IF RESULT-VECTOR-1 = "30120"          ← RULE-06             0023-0007
02214                     MOVE "0" TO RL1                                          0023-0011
02215                     MOVE "6" TO RL2                                          0023-0013
02216                     ADD   1  TO CNT3                                         0023-0016
02217                     ADD   1  TO COUNTER-6                                    0023-0020
02218                        GO TO IS-MCE-MAE-DT-END.                              0023-0025
02219          PERFORM IS-MCE-MAE-DT-CON-3.                                        0023-0026
02220          PERFORM IS-MCE-MAE-DT-CON-4.                                        0023-0027
02221              MOVE RESULT-VECTOR-1 TO WORK-VECTOR-1                           0023-0029
02222              MOVE ZERO TO WV-1(4)                                            0023-0031
02223                  IF WORK-VECTOR-1 = "14302"            ← RULE-23             0023-0042
02224                     MOVE "2" TO RL1                                          0023-0046
02225                     MOVE "3" TO RL2                                          0023-0048
02301                     ADD   1  TO CNT3                                         0023-0051
02302                     ADD   1  TO COUNTER-23                                   0023-0055
02303                        GO TO IS-MCE-MAE-DT-END.                              0023-0060
02304                  IF RESULT-VECTOR-1 = "21231"          ← RULE-19             0023-0061
02305                     MOVE "1" TO RL1                                          0023-0065
02306                     MOVE "9" TO RL2                                          0023-0067
02307                     ADD   1  TO CNT3                                         0023-0069
02308                     ADD   1  TO COUNTER-19                                   0023-0074
02309                        GO TO IS-MCE-MAE-DT-END.                              0023-0078
02310              MOVE RESULT-VECTOR-1 TO WORK-VECTOR-1                           0023-0080
02311              MOVE ZERO TO WV-1(4) WV-1(5)                                    0023-0093
02312                  IF WORK-VECTOR-1 = "33100"            ← RULE-14             0023-0104
02313                     MOVE "1" TO RL1                                          0023-0108
02314                     MOVE "4" TO RL2                                          0023-0110
02315                     ADD   1  TO CNT3                                         0023-0112
02316                     ADD   1  TO COUNTER-14                                   0023-0117
02317                        GO TO IS-MCE-MAE-DT-END.                              0023-0121
02318              MOVE RESULT-VECTOR-1 TO WORK-VECTOR-1                           0023-0123
02319              MOVE ZERO TO WV-1(4)                                            0023-0125
02320                  IF WORK-VECTOR-1 = "21201"            ← RULE-26             0023-0136
02321                     MOVE "2" TO RL1                                          0023-0140
02322                     MOVE "6" TO RL2                                          0023-0142
02323                     ADD   1  TO CNT3                                         0023-0145
02324                     ADD   1  TO COUNTER-26                                   0023-0149
02325                        GO TO IS-MCE-MAE-DT-END.                              0023-0154
02401              MOVE RESULT-VECTOR-1 TO WORK-VECTOR-1                           0023-0155
02402              MOVE ZERO TO WV-1(4) WV-1(5)                                    0023-0168
02403                  IF WORK-VECTOR-1 = "12100"            ← RULE-29             0023-0179
02404                     MOVE "2" TO RL1                                          0023-0183
02405                     MOVE "9" TO RL2                                          0023-0185
02406                     ADD   1  TO CNT3                                         0023-0188
02407                     ADD   1  TO COUNTER-29                                   0023-0192
02408                        GO TO IS-MCE-MAE-DT-END.                              0023-0197
02409              MOVE RESULT-VECTOR-1 TO WORK-VECTOR-1                           0023-0198
02410              MOVE ZERO TO WV-1(2) WV-1(3) WV-1(4) WV-1(5)                    0023-0233
02411                  IF WORK-VECTOR-1 = "10000"            ← RULE-03             0023-0243
02412                     MOVE "0" TO RL1                                          0023-0248
02413                     MOVE "3" TO RL2                                          0023-0250
02414                     ADD   1  TO CNT3                                         0023-0252
```

```
02415                       ADD   1  TO COUNTER-3.                                  0023-0257
02416                           GO TO IS-MCE-MAE-DT-END.                           0023-0261
02417                   GO TO IS-MCE-MAE-DT-ELSE-RULE.          ← ELSE-RULE        0023-0262
02418   IS-MCE-MAE-DT-GRP-4.                                                      0023-0264
                                                                                   PROCEDURE 0023 SIZE 0263
02419       MOVE ALL ZEROS TO RESULT-VECTOR-1.                                     0024-0001
02420           PERFORM IS-MCE-MAE-DT-CON-5.                                       0024-0003
02421           PERFORM IS-MCE-MAE-DT-CON-8.                                       0024-0004
02422                   IF RESULT-VECTOR-1 ="00201"             ← RULE-15          0024-0006
02423                       MOVE "1" TO RL1                                        0024-0010
02424                       MOVE "5" TO RL2                                        0024-0012
02425                       ADD   1  TO CNT4                                       0024-0014
02501                       ADD   1  TO COUNTER-15                                 0024-0019
02502                           GO TO IS-MCE-MAE-DT-END.                           0024-0023
02503           PERFORM IS-MCE-MAE-DT-CON-6.                                       0024-0025
02504                   IF RESULT-VECTOR-1 = "00222"            ← RULE-11          0024-0026
02505                       MOVE "1" TO RL1                                        0024-0030
02506                       MOVE "1" TO RL2                                        0024-0032
02507                       ADD   1  TO CNT4                                       0024-0035
02508                       ADD   1  TO COUNTER-11                                 0024-0039
02509                           GO TO IS-MCE-MAE-DT-END.                           0024-0044
02510               MOVE ZERO TO RV-1(4)                                           0024-0045
02511                   IF RESULT-VECTOR-1 = "00103"            ← RULE-27          0024-0056
02512                       MOVE "2" TO RL1                                        0024-0060
02513                       MOVE "7" TO RL2                                        0024-0062
02514                       ADD   1  TO CNT4                                       0024-0064
02515                       ADD   1  TO COUNTER-27                                 0024-0069
02516                           GO TO IS-MCE-MAE-DT-END.                           0024-0073
02517                   GO TO IS-MCE-MAE-DT-ELSE-RULE.          ← ELSE-RULE        0024-0075
02518   IS-MCE-MAE-DT-GRP-5.                                                       0024-0076
                                                                                   PROCEDURE 0024 SIZE 0076
02519       MOVE ALL ZEROS TO RESULT-VECTOR-1.                                     0025-0001
02520           PERFORM IS-MCE-MAE-DT-CON-2.                                       0025-0003
02521           PERFORM IS-MCE-MAE-DT-CON-3.                                       0025-0004
02522           PERFORM IS-MCE-MAE-DT-CON-6.                                       0025-0006
02523           PERFORM IS-MCE-MAE-DT-CON-8.                                       0025-0007
02524                   IF RESULT-VECTOR-1 = "31022"            ← RULE-12          0025-0009
02525                       MOVE "1" TO RL1                                        0025-0013
02601                       MOVE "2" TO RL2                                        0025-0015
02602                       ADD   1  TO CNT5                                       0025-0017
02603                       ADD   1  TO COUNTER-12                                 0025-0022
02604                           GO TO IS-MCE-MAE-DT-END.                           0025-0026
02605                   IF RESULT-VECTOR-1 = "23033"            ← RULE-16          0025-0027
02606                       MOVE "1" TO RL1                                        0025-0031
02607                       MOVE "6" TO RL2                                        0025-0033
02608                       ADD   1  TO CNT5                                       0025-0036
02609                       ADD   1  TO COUNTER-16                                 0025-0040
02610                           GO TO IS-MCE-MAE-DT-END.                           0025-0045
02611                   IF  RESULT-VECTOR-1 = "13012"           ← RULE-08          0025-0046
02612                       MOVE "0" TO RL1                                        0025-0050
02613                       MOVE "8" TO RL2                                        0025-0052
02614                       ADD   1  TO CNT5                                       0025-0055
02615                       ADD   1  TO COUNTER-8                                  0025-0059
02616                           GO TO IS-MCE-MAE-DT-END.                           0025-0064
02617                   IF RESULT-VECTOR-1 = "22041"            ← RULE-10          0025-0065
02618                       MOVE "1" TO RL1                                        0025-0069
02619                       MOVE "0" TO RL2                                        0025-0071
02620                       ADD   1  TO CNT5                                       0025-0073
02621                       ADD   1  TO COUNTER-10                                 0025-0078
02622                           GO TO IS-MCE-MAE-DT-END.                           0025-0082
02623               MOVE RESULT-VECTOR-1 TO WORK-VECTOR-1.                         0025-0084
```

```
02624                   MOVE ZERO TO WV-1(1)
02625                       IF WORK-VECTOR-1 = "02032"          ← RULE-22
02701                           MOVE "2" TO RL1
02702                           MOVE "2" TO RL2
02703                           MOVE   1  TO CNT5
02704                           ADD    1  TO COUNTER-22
02705                               GO TO IS-MCE-MAE-DT-END.
02706                   MOVE RESULT-VECTOR-1 TO WORK-VECTOR-1.
02707                   MOVE ZERO TO WV-1(2)
02708                       IF WORK-VECTOR-1 = "10022"          ← RULE-04
02709                           MOVE "0" TO RL1
02710                           MOVE "4" TO RL2
02711                           ADD    1  TO CNT5
02712                           ADD    1  TO COUNTER-4
02713                               GO TO IS-MCE-MAE-DT-END.
02714                   IF RESULT-VECTOR-1 = "14033"            ← RULE-21
02715                           MOVE "2" TO RL1
02716                           MOVE "1" TO RL2
02717                           ADD    1  TO CNT5
02718                           ADD    1  TO COUNTER-21
02719                               GO TO IS-MCE-MAE-DT-END.
02720                   MOVE ZERO TO RV-1(1)
02721                       IF RESULT-VECTOR-1 = "01031"        ← RULE-28
02722                           MOVE "2" TO RL1
02723                           MOVE "8" TO RL2
02724                           ADD    1  TO CNT5
02725                           ADD    1  COUNTER-28
02801                               GO TO IS-MCE-MAE-DT-END.
02802                   GO TO IS-MCE-MAE-DT-ELSE-RULE.          ← ELSE-RULE
02803       IS-MCE-MAE-DT-ELSE-RULE.

02804           MOVE "ELSE" TO RULE-NO.
02805           ADD 1 TO CNT6.
028051          GO TO IS-MCE-MAE-DT-END.
02806       IS-MCE-MAE-DT-END.
02807           EXIT.

    PRT 0156
02808       REPORTEX SECTION.
02809       HEADINGS.

02810           WRITE PRINTER-LINE FROM HDR1 AFTER ADVANCING TO CHANNEL 1.
02811           WRITE PRINTER-LINE FROM HDR2 AFTER ADVANCING 2 LINES.
02812       WRITE-LISTING.

02813           MOVE INPUT-CARD TO WS-CARD-IMAGE.
02814           WRITE PRINTER-LINE FROM PRINT-LINE AFTER ADVANCING 1 LINES.
028141          MOVE SPACES TO RULE-NO.
02815       LAST.

02816           WRITE PRINTER-LINE FROM GROUP-CNT-LINE    AFTER ADVANCING 2.
02817           WRITE PRINTER-LINE FROM RULE-COUNT-LINE-1 AFTER ADVANCING 2.
02818           WRITE PRINTER-LINE FROM RULE-COUNT-LINE-2 AFTER ADVANCING 2.
02819           WRITE PRINTER-LINE FROM RULE-COUNT-LINE-3 AFTER ADVANCING 2.
02820           WRITE PRINTER-LINE FROM RULE-COUNT-LINE-4 AFTER ADVANCING 2.
02821           WRITE PRINTER-LINE FROM RULE-COUNT-LINE-5 AFTER ADVANCING 2.
02822           CLOSE CARDIN LISTING WITH RELEASE.
02823           STOP RUN.

    PRT 0156
```

```
0025-0086
0025-0097
0025-0101
0025-0103
0025-0105
0025-0108
0025-0113
0025-0114
0025-0116
0025-0127
0025-0131
0025-0133
0025-0136
0025-0140
0025-0145
0025-0146
0025-0150
0025-0152
0025-0154
0025-0159
0025-0163
0025-0165
0025-0175
0025-0179
0025-0181
0025-0184
0025-0188
0025-0194
0025-0195
0025-0196
PROCEDURE 0025 SIZE 0196
0026-0001
0026-0003
0026-0008
0026-0009
0026-0009
PROCEDURE 0026 SIZE 0009

0027-0001
0027-0001
PROCEDURE 0027 SIZE 0003
0028-0003
0028-0008
0028-0009
PROCEDURE 0028 SIZE 0012
0029-0001
0029-0007
0029-0008
0029-0010
PROCEDURE 0029 SIZE 0013
0030-0003
0030-0007
0030-0012
0030-0016
0030-0020
0030-0024
0030-0028
0030-0030
PROCEDURE 0030 SIZE 0031
PROCEDURE 0031 SIZE 0005
```

```
                                                    PROCEDURE 0008 SIZE 0036
        PRT 0156
                                                    PROCEDURE 0032 SIZE 0002
        PRT 0157
                                                    PROCEDURE 0033 SIZE 0020
        PRT 0160
                                                    PROCEDURE 0034 SIZE 0020
        PRT 0161
                                                    PROCEDURE 0035 SIZE 0020
        PRT 0162
                                                    PROCEDURE 0001 SIZE 0079
                                                    0000-0000
    999999   END-OF-JOB.
COMPILE O K . B-5500 FEB 1968
PRT SIZE 0115
NO. SEGS. 044
COMPILE TIME 00050 SECS.
TOTAL SEG. SIZE 02204
DISK SIZE 03090
MEMORY SIZE 02115
******* .*******
CARDS   00652
```

VI-1-4 TEST DATA LISTING.

The test data was designed to illustrate later in the execution of the program ambiguous rules. This was done by specifying on every record the intended rule.

4	4	20	PINK	77	GREEN	17	AA	BCA	RULE = 19
5	5	20	PINK	73	GREEN	76	AA	QTP	RULE = 19
9	9	20	PINK	31	GREEN	63	AA	LSD	RULE = 19
3	3	20	PINK	05		H1	AA	ABC	RULE = 26
3	7	35	PINK	35		J3	AA	A35	RULE = 23
2	5	35	PINK	47		J2	AA	B77	RULE = 23
1	8	35	PINK	93		J5	AA	CA3	RULE = 23
5	7	35	PINK	85		J4	AA	P3Q	RULE = 23
6	8	35	PINK	33		J1	AA	3A3	RULE = 23
4	4	20	PINK	08		33	AA	QTP	RULE = 26
9	9	20	PINK	74		35	AA	LSD	RULE = 26
2	1	15	PINK	63			AA	375	RULE = 29
			GOKO	10	RED	J5	AA	QTP	RULE = 11
			GOKO	10	RED	J4	AA	LST	RULE = 11
			GOKO	10	RED	J3	AA	SSD	RULE = 11
			GOKO	10	RED	J9	AA	DPQ	RULE = 11
			GOKO	10	RED	J3	AA	TRS	RULE = 11
			PUKO	10	RED	J1	AA	PTR	RULE = 11
			PUKO	10	RED	J7	AA	ATQ	RULE = 11
			PUKO	10	RED	37	AA	RST	RULE = 15
			PUKO	10	RED	H1	AA	UVW	RULE = 15
			MOKO	10	RED	H2	AA	WWW	RULE = 15
			MOKO	10	RED	37	AA	BCQ	RULE = 15
			MOKO	10	RED	33	AA	TRQ	RULE = 15
			MOKO	10	RED	78	AA	EFP	RULE = 15
			MOKO	10	RED	83	AA	RST	RULE = 15
			MOKO	10	RED	95	AA	QRU	RULE = 15
M	3	20		10			BC		RULE = 25
8	2	20		10			DA		RULE = 25
9	1	20		10					RULE = 25
3	2	20		10			PP		RULE = 25
5	2	20		10			QT		RULE = 25
3	3	15		10					RULE = 30
9	9	15		10					RULE = 30
7	7	15		10					RULE = 30
6	6	15		10					RULE = 30
1	1	15		10			TT		RULE = 30
3	9		PINK	13			AA		RULE = 03
2	1		PINK	14	RED			035	RULE = 06
7	3	30	PINK	15	RED			756	RULE = 06
1	0		PINK	16	RED			770	RULE = 06
9	5		PINK	17	RED			077	RULE = 06
7	4		PINK	18	RED			133	RULE = 06
9	1		PINK	19	RED			331	RULE = 06
8	2		PINK	20	RED			551	RULE = 06
3	1		PINK	21	RED			197	RULE = 06
2	0		PINK	22	RED			354	RULE = 06
7	0	30	PINK	23	RED			256	RULE = 06
6	5		PINK	25	RED			128	RULE = 06
3	1		PINK	26	RED		AA	064	RULE = 06
3	2	30	PINK	17				356	RULE = 14
5	3	30	PINK	99				625	RULE = 14
7	1	30	PINK	37				024	RULE = 14
3	3	20	PINK	14	GREEN	33		ABC	RULE = 19
2 &&	3	20	PINK	10	ORNGE	23	AA	333	RULE = 01
3	4	20	PINK	10	RED	01	AA	001	RULE = 09
2	2	35	PINK	10	RED	J1	BB	A3B	RULE = 02
3	3	35	RED	10	RED	H2	BB	305	RULE = 02
4	4	35	RED	10	RED	H3	CA		RULE = 02
5	5	35	BLUE	10	RED	05	BA	653	RULE = 02
6	6	35	PINK	10	RED	77	AB		RULE = 02

7	7	35	BLUE	10	RED	13	CC	BAB	RULE = 02
7	7	30	BLUE	10	WHITE	00	A5	BAB	RULE = 07
3	3	30	BLUE	10	WHITE	00	77	BAB	RULE = 07
2	2	30	BLUE	10	WHITE	00	BC	BAB	RULE = 07
9	9	30	BLUE	10	WHITE	00	DA	BAB	RULE = 07
2	3	30	PINK	10	GREEN	H1	CC	AAA	RULE = 13
3	4	30	----	10	GREEN	H2	/U	BBB	RULE = 13
2	3	30	BLUE	10	GREEN	33			RULE = 18
2	3	30		10	GREEN	H1			RULE = 18
8	9		****	10	GREEN	H2			RULE = 18
3	4	30	RED	10	GREEN	H3			RULE = 18
5	6		RED	10	GREEN	H4			RULE = 18
3	4	30		10	GREEN	H5			RULE = 18
7	8	30		10	GREEN	34			RULE = 18
7	5	30	PINK	10	GREEN	37			RULE = 18
2	3	15		10		35		ABC	RULE = 24
4	5	15		10		36		DEF	RULE = 24
6	7	15		10		37		GHI	RULE = 24
4	3	20		10					RULE = 25
5	2	20		10					RULE = 25
			BLUE	10		00	AA	375	RULE = 27
			BLUE	10		00	AA	001	RULE = 27
				10		00	AA	100	RULE = 27
			RED	10		00	AA	103	RULE = 27
			NONO	10		00	AA	781	RULE = 27
1	9	17	POPO	33	RED	J2	BA		RULE = 04
2	3	30	GOKO	07	BLUE	J3	MA		RULE = 08
7	9	30	SOKO	13	BLUE	J4	PH		RULE = 08
3	3	15	ROKO	17	WHITE	35	DR		RULE = 10
7	7	15	BOKO	23	WHITE	77	KT		RULE = 10
3	1	20	TORO	75	RED	J7	LM		RULE = 12
6	5	20	KORO	81	RED	J8	DP		RULE = 12
3	2	20	TAKA	31	RED	J3	RS		RULE = 12
5	5	30	KAKA	46	GREEN	00	TQ		RULE = 16
6	6	30	SAKA	64	GREEN	00	PP		RULE = 16
9	9	30	DORO	61	GREEN	00	KK		RULE = 16
1	7	35	FORD	73	GREEN	00	RR		RULE = 21
		15	BLUE	56	GREEN	J1	TT		RULE = 22
		15	RAKA	51	GREEN	J2	TT		RULE = 22
		20	FORO	99	GREEN	37	AC		RULE = 28
5	5	77	ELSE	77	PINK	00	**		ELSE RULE
5	5	77	ELSE	81	PINK	00	**		ELSE RULE
5	5	77	ELSE	99	PINK	00	**		ELSE RULE
5	5	77	ELSE	31	PINK	00	**		ELSE RULE
5	5	77	ELSE	77	PINK	00	**		ELSE RULE
5	5	77	ELSE	66	PINK	00	**		ELSE RULE
5	5	77	ELSE	55	PINK	00	**		ELSE RULE

VI-1-5 THE RUN.

The program output was designed to list every input record together with the rule number it satisfies making possible to check if the data satisfied the intended rule.

RULE SATISFIED	A	B	E	G	X	COULOR	NUMBER	VALUE-1	C	INTENDED RULE
										INPUT-CARD
SATISFIES RULE-19	4	4	20	PINK	77	GREEN	17	AA	BCA	RULE = 19
SATISFIES RULE-19	5	5	20	PINK	73	GREEN	76	AA	QTP	RULE = 19
SATISFIES RULE-19	9	9	20	PINK	31	GREEN	63	AA	LSD	RULE = 19
SATISFIES RULE-26	3	3	20	PINK	05		H1	AA	ABC	RULE = 26
SATISFIES RULE-23	3	7	35	PINK	35		J3	AA	A35	RULE = 23
SATISFIES RULE-23	2	5	35	PINK	47		J2	AA	B77	RULE = 23
SATISFIES RULE-23	1	8	35	PINK	93		J5	AA	CA3	RULE = 23
SATISFIES RULE-23	5	7	35	PINK	85		J4	AA	P30	RULE = 23
SATISFIES RULE-23	6	8	35	PINK	33		J1	AA	3A3	RULE = 23
SATISFIES RULE-26	4	4	20	PINK	08		33	AA	QTP	RULE = 26
SATISFIES RULE-26	9	9	20	PINK	74		35	AA	LSD	RULE = 26
SATISFIES RULE-ELSE	2	1	15	PINK	63			AA	375	RULE = 29
SATISFIES RULE-11				GOKU	10	RED	J5	AA	QTP	RULE = 11
SATISFIES RULE-11				GOKU	10	RED	J4	AA	LST	RULE = 11
SATISFIES RULE-11				GOKU	10	RED	J3	AA	SSD	RULE = 11
SATISFIES RULE-11				GOKU	10	RED	J9	AA	DPQ	RULE = 11
SATISFIES RULE-11				GOKU	10	RED	J3	AA	TRS	RULE = 11
SATISFIES RULE-11				PUKU	10	RED	J1	AA	PTR	RULE = 11
SATISFIES RULE-11				PUKU	10	RED	J7	AA	ATQ	RULE = 11
SATISFIES RULE-15				PUKU	10	RED	37	AA	RST	RULE = 15
SATISFIES RULE-15				PUKU	10	RED	H1	AA	UVW	RULE = 15
SATISFIES RULE-15				MOKU	10	RED	H2	AA	WWW	RULE = 15
SATISFIES RULE-15				MOKU	10	RED	37	AA	BCQ	RULE = 15
SATISFIES RULE-15				MOKU	10	RED	33	AA	TRQ	RULE = 15
SATISFIES RULE-15				MOKU	10	RED	78	AA	EFP	RULE = 15
SATISFIES RULE-15				MOKU	10	RED	83	AA	RST	RULE = 15
SATISFIES RULE-15				MOKU	10	RED	95	AA	QRU	RULE = 15
SATISFIES RULE-25	M	3	20		10			BC		RULE = 25
SATISFIES RULE-25	8	2	20		10			DA		RULE = 25
SATISFIES RULE-25	9	1	20		10					RULE = 25
SATISFIES RULE-25	3	2	20		10			PP		RULE = 25
SATISFIES RULE-25	5	2	20		10			QT		RULE = 25
SATISFIES RULE-30	3	3	15		10					RULE = 30
SATISFIES RULE-30	9	9	15		10					RULE = 30
SATISFIES RULE-30	7	7	15		10					RULE = 30
SATISFIES RULE-30	6	6	15		10					RULE = 30
SATISFIES RULE-30	1	1	15		10			TT		RULE = 30
SATISFIES RULE-03	3	9		PINK	13			AA		RULE = 03
SATISFIES RULE-06	2	1		PINK	14	RED			035	RULE = 06
SATISFIES RULE-06	7	3	30	PINK	15	RED			756	RULE = 06
SATISFIES RULE-06	1	0		PINK	16	RED			770	RULE = 06
SATISFIES RULE-06	9	5		PINK	17	RED			077	RULE = 06
SATISFIES RULE-06	7	4		PINK	18	RED			133	RULE = 06
SATISFIES RULE-06	9	1		PINK	19	RED			331	RULE = 06
SATISFIES RULE-06	8	2		PINK	20	RED			551	RULE = 06
SATISFIES RULE-06	3	1		PINK	21	RED			197	RULE = 06
SATISFIES RULE-06	2	0		PINK	22	RED			354	RULE = 06
SATISFIES RULE-06	7	0	30	PINK	23	RED			256	RULE = 06
SATISFIES RULE-06	6	5		PINK	25	RED			128	RULE = 06
SATISFIES RULE-06	3	1		PINK	26	RED		AA	064	RULE = 06
SATISFIES RULE-14	3	2	30	PINK	17				356	RULE = 14
SATISFIES RULE-14	5	3	30	PINK	99				625	RULE = 14
SATISFIES RULE-14	7	1	30	PINK	37				024	RULE = 14
SATISFIES RULE-19	3	3	20	PINK	14	GREEN	33		ABC	RULE = 19
SATISFIES RULE-01	2 &&	3	20	PINK	10	ORNGE	23	AA	333	RULE = 01
SATISFIES RULE-01	3	4	20	PINK	10	RED	01	AA	001	RULE = 09
SATISFIES RULE-02	2	2	35	PINK	10	RED	J1	BB	A38	RULE = 02
SATISFIES RULE-02	3	3	35	RED	10	RED	H2	BB	305	RULE = 02

SATISFIES RULE=02	4	4	35	RED	10	RED	H3	CA		RULE = 02
SATISFIES RULE=02	5	5	35	BLUE	10	RED	05	BA	653	RULE = 02
SATISFIES RULE=02	6	6	35	PINK	10	RED	77	AB		RULE = 02
SATISFIES RULE=02	7	7	35	BLUE	10	RED	13	CC	BAB	RULE = 07
SATISFIES RULE=07	7	7	30	BLUE	10	WHITE	00	A5	BAB	RULE = 07
SATISFIES RULE=07	3	3	30	BLUE	10	WHITE	00	77	BAB	RULE = 07
SATISFIES RULE=07	2	2	30	BLUE	10	WHITE	00	BC	BAB	RULE = 07
SATISFIES RULE=07	9	9	30	BLUE	10	WHITE	00	DA	BAB	RULE = 13
SATISFIES RULE=18	2	3	30	PINK	10	GREEN	H1	CC	AAA	RULE = 13
SATISFIES RULE=18	3	4	30	----	10	GREEN	H2	/U	BBB	RULE = 18
SATISFIES RULE=18	2	3	30	BLUE	10	GREEN	33			RULE = 18
SATISFIES RULE=18	2	3	30		10	GREEN	H1			RULE = 18
SATISFIES RULE=18	8	9		****	10	GREEN	H2			RULE = 18
SATISFIES RULE=18	3	4	30	RED	10	GREEN	H3			RULE = 18
SATISFIES RULE=18	5	6		RED	10	GREEN	H4			RULE = 18
SATISFIES RULE=18	3	4	30		10	GREEN	H5			RULE = 18
SATISFIES RULE=18	7	8	30		10	GREEN	34			RULE = 18
SATISFIES RULE=13	7	5	30	PINK	10	GREEN	37			RULE = 18
SATISFIES RULE=24	2	3	15		10		35		ABC	RULE = 24
SATISFIES RULE=24	4	5	15		10		36		DEF	RULE = 24
SATISFIES RULE=24	6	7	15		10		37		GHI	RULE = 24
SATISFIES RULE=25	4	3	20		10					RULE = 25
SATISFIES RULE=25	5	2	20		10					RULE = 25
SATISFIES RULE=27				BLUE	10		00	AA	375	RULE = 27
SATISFIES RULE=27				BLUE	10		00	AA	001	RULE = 27
SATISFIES RULE=27					10		00	AA	100	RULE = 27
SATISFIES RULE=27					10		00	AA	103	RULE = 27
SATISFIES RULE=27				RED	10		00	AA	781	RULE = 27
SATISFIES RULE=27				NONO	10					RULE = 04
SATISFIES RULE=04	1	9	17	POPO	33	RED	J2	BA		RULE = 08
SATISFIES RULE=08	2	3	30	GOKU	07	BLUE	J3	MA		RULE = 08
SATISFIES RULE=08	7	9	30	SOKO	13	BLUE	J4	PH		RULE = 10
SATISFIES RULE=10	3	3	15	ROKU	17	WHITE	35	DR		RULE = 10
SATISFIES RULE=10	7	7	15	BOKU	23	WHITE	77	KT		RULE = 12
SATISFIES RULE=12	3	1	20	TORO	75	RED	J7	LM		RULE = 12
SATISFIES RULE=12	6	5	20	KORO	81	RED	J8	DP		RULE = 12
SATISFIES RULE=12	3	2	20	TAKA	31	RED	J3	RS		RULE = 16
SATISFIES RULE=16	5	5	30	KAKA	46	GREEN	00	TQ		RULE = 16
SATISFIES RULE=16	6	6	30	SAKA	64	GREEN	00	FP		RULE = 16
SATISFIES RULE=16	9	9	30	DORO	61	GREEN	00	KK		RULE = 21
SATISFIES RULE=21	1	7	35	FORU	73	GREEN	00	RR		RULE = 22
SATISFIES RULE=22			15	BLUE	56	GREEN	J1	TT		RULE = 22
SATISFIES RULE=22			15	RAKA	51	GREEN	J2	TT		RULE = 28
SATISFIES RULE=28			20	FORO	99	GREEN	37	AC		ELSE RULE
SATISFIES RULE=ELSE	5	5	77	ELSE	77	PINK	00	**		ELSE RULE
SATISFIES RULE=ELSE	5	5	77	ELSE	81	PINK	00	**		ELSE RULE
SATISFIES RULE=ELSE	5	5	77	ELSE	99	PINK	00	**		ELSE RULE
SATISFIES RULE=ELSE	5	5	77	ELSE	31	PINK	00	**		ELSE RULE
SATISFIES RULE=ELSE	5	5	77	ELSE	77	PINK	00	**		ELSE RULE
SATISFIES RULE=ELSE	5	5	77	ELSE	66	PINK	00	**		ELSE RULE
SATISFIES RULE=ELSE	5	5	77	ELSE	55	PINK	00	**		ELSE RULE

CNT1 = 005 CNT2 = 032 CNT3 = 028 CNT4 = 020 CNT5 = 002 CNT6 = 008

COUNTER-1 = 002 COUNTER-2 = 006 COUNTER-3 = 001 COUNTER-4 = 001 COUNTER-5 = 000 COUNTER-6 = 012

COUNTER-7 = 004 COUNTER-8 = 002 COUNTER-9 = 000 COUNTER-10 = 002 COUNTER-11 = 007 COUNTER-12 = 003

COUNTER-13 = 001 COUNTER-14 = 003 COUNTER-15 = 008 COUNTER-16 = 003 COUNTER-17 = 000 COUNTER-18 = 009

COUNTER-19 = 004 COUNTER-20 = 000 COUNTER-21 = 001 COUNTER-22 = 002 COUNTER-23 = 005 COUNTER-24 = 003

COUNTER-25 = 007 COUNTER-26 = 003 COUNTER-27 = 005 COUNTER-28 = 001 COUNTER-29 = 000 COUNTER-30 = 005

1982

CHAPTER VII: CONCLUSIONS AND SUGGESTIONS

In this thesis a general mathematical description of decision tables was introduced for the first time. This mathematical formalism served as a basis of the over all investigation and allowed for the first time a precise treatment of the logical basis of decision tables as well as the conversion problem.

Chapter II presents the matrix description of decision tables with detailed classification. Based upon this, section II-2 introduces basic rule operations such as "simple rule splitting", "complete rule splitting" and rule intersection needed in the following discussion of the characteristics of decision tables given in II-2-2, and II-2-3. In the case of decision tables with unrelated conditions a formal theorem (II-2-1) on ambiguity was given. A precise definition of data dependent inconsistancy was also included and demonstrated. Based upon previous definition a clear definition in mathematical terms for completeness is given. In II-2-3 the characteristics of decision tables with related and unrelated conditions are analyzed and given in terms of possible ambiguity. A definition of completeness for this case is also given. To this field of decision tables with related conditions further investigation can be applied by the inclusion of any particular programming language.

Chapter III deals with a more practical aspect of decision tables, namely, manipulations of decision tables and relations

between them. In many practical cases in which the use of decision tables is applied the actual decision tables can become too large and lose clarity and effectiveness. In these cases splitting techniques should be applied. The splitting techniques developed in III-1-1 are based upon the definition of different linkages between decision tables. Three different splitting procedures were developed; splitting by rules, splitting by conditions and splitting by branching. Some of them were used later in different conversion techniques. The problem of merging decision tables was analyzed as well. Unfortunately, the only conclusion reached was that even under severe restriction there is no simple generalized procedure for merging; on which will not produce, in general, ambiguities or other logical errors. Further investigations into this subject without making use of more intelligent procedures for logical rather than structured analysis is impossible. Section III-2 deals with relations between decision tables. Few definitions for isomorphism and equivalence, based upon the previously defined rule operations and the rule combination defined in definition III-2-3 are given. The purpose of these definitions was again to supply guiding rules in the practical use of decision tables creating a way of deciding whether two tables are logically equivalent or not. Under relatively severe restriction automated procedures were developed for checking simple equivalence. This subject can be further developed and analyzed. Nevertheless, this was used in III-2-2 for development of decision tables minimization procedures.

Chapter IV includes an in depth. study and analysis of the advanced conversion methods published in existing literature. The necessity of this study lies in the fact that in implementing a decision table processor, one must be very careful in choosing the conversion procedures. In this chapter every one of the different existing techniques was analysed, leading to a conclusion that not one of these procedures can be used separately for every case.

Chapter V is devoted to testing and diagnostic methods. This chapter is divided into three sections. The first section delt with testing and diagnostic procedures at conversion time. The first set of theorems (for LCE decision tables) is based upon Chapter III and the paper published by P.H.J. King [B4] which was the first to give serious consideration to this subject. Theorems V-1-5 through V-1-8 (for MCE decision tables) were originated for the first time here. In the second section testing and diagnostic procedures at run time are presented (originally published by C.R. Muthukrishnan and V. Rajaraman [B7]), together with some extensions for conversion time. In the third section, two algorithms for LCE and MCE decision tables (with and without related conditions) for conversion and testing in conversion time and run time, are developed. These algorithms give a combined and efficient way of debugging.

In Chapter VI an attempt was made to suggest briefly a method for writing a flexible decision table COBOL pre-compiler. With respect to this Chapter it should be noted that an actual pre-compiler

package has not been produced Also, due to lack of funds, no real pre-compiler was written and the conversion work of the colla borated example included, was done manually. The goal of designing a decision table processor which can serve as a form of direct communication language between the systems analyst and the computer was not accomplished. Most of the work was done with respect to the conversion and testing procedures. Except for suggesting the option of library and data base interface no attempt was made to suggest ways for data and errors description. This part is left open for further investigation in the subject which might lead eventually to the complete elimination of the programmer.

The purpose of this thesis was to provide a general tool and therefore it is expected that an additional number of results could have been obtained for special types of decision tables or for decision tables in special applications.

It is believed that this treatment of decision tables will provide the required tools for the potential decision table user, or the implementer of a decision table translator.

B I B L I O G R A P H Y

The references marked with "*" were not available to me.

A. BIBLIOGRAPHY ON TREE THECHNIQUES.

[A1] Conversion of Limited-Entry Decision Tables to Computer Programs.
by Solomon L. Pollack (The Rand Corporation. Santa Monica, Calif.)
Communications of the ACM Volume 8/Number 11/November 1963 P.P667-62.

[A2] A Procedure for Converting Logic Table Conditions into an Efficiant Sequence of Test Instractions.
by J.F. Egler (McDonnell Aircraft Corp., St. Louis, Mo.)
Communications of the ACM Volume 6/Number 8/September 1963
P.P 510-514.

[A3] Egler Procedure Refuted
by Michael Montalbano (IBM Advanced Systems Development Div., San Jose, Calif.)
Communications of the ACM Volume 7/Number 1/January 1964 P.P 1.

[A4] An Efficient Algorithm for Finding Certain Minimum Cost Procedures for Making Binary Decisions.
by James R. Slage (Lawrence Radiation Laboratory, University of Calif., Livermore, Calif.)
Journal of the ACM. Volume 11, No. 3 (July 1964) P.P 253-264.

[A5] Conversion of Decision Tables to Computer Programs.
by Laurence I. Press (San Fernando Valley State College. Northridge, California)
Communications of the ACM. Volume 8/Number 6/June 1965 P.P. 385-390.

[A6] Conversion of Limited-Entry Decision Tables to Optimal Computer Program I; Minimum Average Processing Time.
by Lewis T. Reinwald and Richard M. Soland (Research Analysis Corp., McLean, Virginia)
Journal of the ACM Volume 13, No. 3 (July 1966) P.P. 339-358.

[A7] Conversion of Limited-Entry Decision Tables to Optimal Computer Program II; Minimum Storage Requirement.
by Lewis T. Reinwald and Richard M. Soland.
Journal of the ACM, Volume 14, No. 4 (October 67) P.P. 742-755.

[A8] Description of the Basic Algorithem in
by Michael D. Callahan and Hanson E. Chapman (System Development

DETAB/65 Preprocessor. Corporation, Santa Monica, Calif.) Communications of the ACM Volume 10/Number 7/July 1967.

[A9] Algorithm 394 Decision Table Translation [H] by Robert B. Dial Communications of the ACM Volume 13/Number 9/September 1970

[A10]*Analysis of Decision Rules in Decision tables by Pollack S.L. Memo RM-3669-PR, Rand Corp., Santa Monica, Calif., 1963.

[A11]*Tables Flow Charts and Program Logic. by Montalbano M. IBM Sys J. (Sept. 1962) P.P.51

B. BIBLIOGRAPHY ON MASK TECHNIQUES.

[B1] Use of Decision Tables in Computer Programing. by H.W. Kirk (Radio Corp., of America Harrison, New Jersey) Communication of the ACM Volume 8/Number 1/January 1965. P.P. 41-43.

[B2] Conversion of Decision Tables to Computer Programs by Rule Mask Techniques. by P.J.H. King (University Colege of Wales, Aberystwyth, U.K. Communications of the ACM, Volume 9/Number 11/November 1966. P.P 796-801.

[B3] Decision Tables. by P.J.H. King The Computer Journal, Volume 10/No. 2/August 1967 P.P. 135-142.

[B4] Ambiguity in Limited Entry Decision Tables. by P.J.H. King. Comunications of the ACM, Volume 11/Number 10/October 1968 P.P 680-84.

[B5] System Analysis: A Diagnose Aproach. by Van Court Hare Jr. Harcout, Brace & World, Inc. 1969

[B6] Decision Tables Translation. by R. Peel. The Computer Bulletin December 69. P.P. 419-421

[B7] On the Conversion of Decision Tables to Computer Programs by C.R. Muthukrishnan and V. Rajaraman (India Institute of Technology, Kanpur, India) Communications of the ACM, Volume 13/Number 6/June 1970 p.p. 347-351.

C. BIBLIOGRAPHY ON OTHER CONVERSION TECHNIQES.

[C1] Programming Decision Tables in FORTRAN, COBOL or ALGOL.
by Cyril G. Veinott (Reliance Electric & Engineering Conpany Cleveland, Ohio)
Communication of the ACM, Volume 9/Number 1/January 1966. P.P 31-35.

D. GENERAL BIBLIOGRAPHY ON DECISION LOGIC AND RELATED TOPICS.

[D1] The Problem of Simplifying Truth Functions.
by W.V. Quine (Harward University) American Mathemathical, Monthly Volume 59 (1952).

[D2] A Way to Simplify Truth Functions.
by W.V. Quine (Harward University) American Mathematical Monthly Volume 62 (1955).

[D3] Encoding of Incompletely Specified Boolean Matrices.
by T.A. Dolotta and E.J. McLuskey Jr. (Princeton University Princeton, New Jersey)

[D4] Simplification of Class of Boolean Functions.
by Edwin Hirshhorn (Northrop Aircraft Inc., Hawthorne, Calif.)
Journal of the ACM, Volume 5 (1958) P.P. 67-75.

[D4] An Algorithm for the Traveling Salesman Problem.
by John D.C. Little (Massachusetts Institute of Technology) Katta G. Murty (Indian Statistical Institute) Dura W. Sweeney (IBM Corporation) Caroline Karel (case Institute of Technology).
Operation Research II(1963) P.P 972
P.P. 972-989.

[D5] Systematics-a non-Programming Language for Designing and specifying Commercial systems for Computers.
by C.B.B. Grindley
The Computer Journal August 1966
Volume Nine Number Two P.P 124-128.

[D6] COBOL and Compatibility
DATAMATION, February 61, P.P. 30-34.

E. BIBLIOGRAPHY ON USAGE OF DECISION TABLES.

[E1] Decision Tables as a Tool. — by Daniel F. Langenwalter

[E2] Logic-Structur Tables — by H.N. Cantrell, J. King and F.E. H. King (General Electric Corp. Schenectary, N.Y) Communications of the ACM, Volume 4/Number 6/June 1961 P.P. 272-275.

[E3] Tabular Form in Decision Logic. — by Burton Grad (IBM Corp.) Thomas J. Watson (Research Center Yorktown Heights, N.Y) Datamation, July 1961 P.P. 22-26.

[E4] An Engineering Application of Logic-Structure Tables. — by R.C. Nickerson (Numerical Methods Engineer, GE Comp., Schenecetady, New York) Communications of the ACM, Volume 11/Number 4/November 1961 P.P. 516-520.

[E5] Decision Tables: A System Analysis and Documentation Techniques. — IBM, F20-8102, 1962.

[E6] Special Report, Decision Tables Symposium. — by Paul Dixon (Technical Staff AUERBACH Corp.) 1962

[E7] Using Decision Structure Tables: Part One: Principals and Preparation. — by D.T. Schmidt and T.F. KAVANAGH DATAMATION FEBRUARY 1964 P.P. 42-52

[E8] Using Decision Structure Tables: Part Two: Manufacturing Applications — by D.T. Schmidt and T.F. KAVANAGH DATAMATION MARCH 1964. P.P 48-54

[E9] Decision Tables and Their Applications. — by Paul Dixon (AUERBACH Corp. Philadelphia PA.) Computers and Automation April 1964 P.P. 14-19.

[E10] Data Documentation and Decision Tables. — by D.L. Fisher (IBM Corp., Los Gatos California) Communications of the ACM, Volume 9/Number 1/January 1966. P.P. 26-31

[E11] The System Structure Design Method. — by P.J. Vincent, 1967

[E12] Decision Tables — by Martin L. Huges C.P.A. Richard M. Shank and Elinor Svendsen Stein M.A. MDI Publications 1968.

[E13] Life with Decision Tables at Manufacturers Life. — by J.R. Campbell Canadian Datasystems April 1970 P.P. 46-48, 87-88.

[E14] Decision Tables a Technique for Minimizing routine, repetitive design — by David Holstein (IBM Corp) August 2, 1962. P.P. 76-79.

[E15] Decision Tables in Systems Design — by B. Grad (IBM Corp. White Plains N.Y.) 1962 ACM National Conference. P.P. 76-77.

[E16] Applications of Decision Tables. — by Herman McDaniel. Editor Brandon/Systems Press Inc. (A Collection of Articles)

[E17] Decision Table Software a Handbook. — by Herman McDanial. Brandon/Systems Press Inc.

[E18]*Decision Table Tutorial Using DETAB-X. — by Instruction Task Force, CODASYL Systems Development Group. 1962.

[E19]*Advanced Analysis Method for Integrated Electronic Data — by Evans D.Y. (1961) IBM General Information Manual F20-8047.

[E20]*Tabular Form of Decision Logic. — by Grad B. Datamation July 1961 P.P. 22

[E21]*Analysis of Decision Rules in Decision Tables. — by Pollack S.L. Memo RM-3669-PR, Rand Corp.,Santa Monica Calif. May 1963.

[E22]*Decision Tables — by Morgan J.J. Manage. Serv. 2(1965) P.P. 13-18

[E23]*Tabular Descriptive Language. — by lans T.B., and Grad B. IBM Tech. Rep. No. 2A5 Jan. 1962.

[E24]*Decision Table Experience on a File Maintenance System. — by Brown L.M. Proc. Decision Tables Symposcum, NYC: 75-80 (Sept. 1962)

[E25]*Place of Decision Tables and DETAB-X. — by Calkins L.W., Proc. Decision Tables Sy posium, NYC: 9-12 (Sept 1962)

[E26]*Commercial and Engineering Applications of decision Tables. — by Cantrell H.W. Proc. Decision Tables Symposium NYC: 55-61 (Sept 1962)

[E27)*Decision Tables Symposium — by Cunnigham, J. Proc. Decision Tables Symposium NYC: 7-8 (Sept 1962)

[E28]*Decision Tables: A Preliminary Reference Manual. — by Evans, O.Y. IBM General Information Manual F20-8047

[E29]*Structure and Concept of Decision Tables — by Grad, B. Proc. Decision Tables Symposium NYC: 19-28 (Sept 1962)

[E30]*The Need for Precize Problem Definition — by Hawes, M.K. Proc. Decision Tables Symposium NYC: 13-18 (Sept 1962)

[E31]*Manufacturing Applications of Decision structure Tables. — by Kavanagh, T.F. Proc. Decision Tables Symposium NYC: 89-97 (Sept 1962)

[E32]*Application of Deci-Tables to Management Information Systems. — by Naramore, F. Proc. Decision Tables Symposium NYC: 63-74 (Sept 1962)

F. DECISION TABLES ORIENTED LANGUAGES.

[F1] TABSOL: A Fundamental Concept for System Oriented Languages. — by T.F. Kavanagh (Manufacturing Services, GE, New-York, N.Y.)

[F2] TABSOL: The Languageof Decision Making. — by T.F. Kavanagh (Production Control Service, GE, New-York N.Y.) Sept. 1961.

[F3] TABSOL: The Decision Language of GE-225. — General Electric.

[F4] FORTAB: A Decision Table Language For Scientific Computing Applications. — by G.W. Armerding Memorandum RM-3306-pr, September 1962 The Rand Corporation.

[F5] GETRAN: GECOM to COBOL Translator. — General Electric, 1966

[F6] DETAB-65: Decision Tables Directly into Programs. — by S.L. Pollack The ACM 1966

[F7] DETAB-65: User's Manual. by Working Group 2 on Decision Tables of the Special Interest Group on Programming Languages (SIGPLAN) of the Los Angeles Chapter of the ACM.

[F8] DETRAN: User's Manual (COBOL Version) by R.L. Martino & Com. and I.I.I. Information Industries Inc. 1968.

[F9] DETAP: Compact User's Manual (COBOL). Information Management Inc. 1969.

[F10] NAYTABTRANS: User's Guide (COBOL) Navy Table Translator. by LTJG H.B. Towne U.S. Navy NOV. 1968.

[F11] GECOM II: GE-200 Series General Electric Sept. 1965 (Rev. Feb. 1969)

[F12] DETAB-X: Preliminary Specifications for a Decision Table Structured Language. Prepared and Published by Data Description and Transformation logic Task Forces, CODASYL. Systems Development Group Sept. 62.

[F13]*GE 225 TABSOL Manual by GE Computer Dept. Arisona No. CPB-147 (5M 3-61)

[F14]*DETAB-X: An Improved Business-Oriented Computer Language. by Pollack S.L. Rand Corp. No. RM-3273-pr (Aug. 1962)

[F15]*What is DETAB-X by Pollack S.L. Proc. Decision Tables Symposium NYC: 29-39 (Sept. 1962)

[F16]*TABSOL: Application Introduction to by GE Computer Dept. Arizona No. CPB-147A (5M.6-61)

[F17]*Approch to Decision Table Processor by Wright K.R. Proc. Decision Tables Symposium NYC: 41-44(Sept. 1962)

G. PREPROCESSORS LISTINGS.

[G1] DETAB/65 COBOL PREPROCESSOR LISTING by Working Group 2 on Decision Tables of the Special Interest Group on Programming Languages (SIGPLAN) of the Los Angeles Chapter of the ACM.

[G2] NAVTABTRANS-C by LTJG H.B. Towne U.S. Navy Nov. 68

VITAE

NAME: Albert Pinhas

BORN: Sofia, Bulgaria (Nov. 4, 1938)

SCHOOL: University of Tel-Aviv, Israel B. Sc. (1967)
in Applied Mathematics