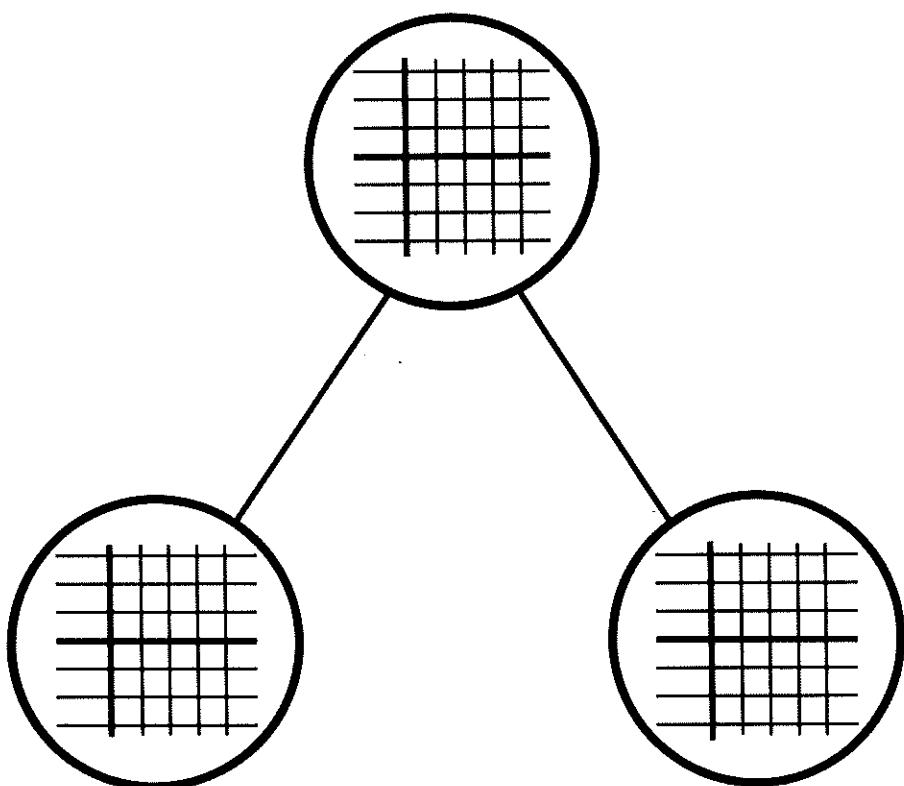


A MODERN APPRAISAL OF
**DECISION
TABLES**



**A
CODASYL
REPORT**

Association for Computing Machinery
11 West 42nd Street
New York, NY 10036

Reproducing Policy

Any organization interested in reproducing a CODASYL report of specification in whole or in part, using ideas from a report for any purpose, is free to do so. However, all such organizations are requested to reproduce the following paragraph in its entirety as part of the preface to any such publication. Any organization using a short passage from a CODASYL document, such as in a book review, is requested to mention "CODASYL" in an acknowledgement of the source, but need not quote the acknowledgement.

"A product of the CODASYL is not the property of any individual, company or organization or of any group of individuals, companies or organizations. No warranty, expressed or implied, is made by any contributor or by any CODASYL Committee as to the accuracy and functioning of any system developed as a result of this product. Moreover, no responsibility is assumed by any contributor or by any committee in connection therewith. This product neither states the opinion of, nor implies support of, the sponsoring institutions."

Price: ACM Members: \$21.00

Nonmembers: \$27.00

Copies may be ordered, prepaid, from:

ACM Order Department
P.O. Box 64145
Baltimore, MD 21264

Order No. 202821

CONFERENCE ON DATA SYSTEMS LANGUAGES
(CODASYL)

DECISION TABLE TASK GROUP

REPORT

AUTHORS

E. Henry Beitz	Consultant
Neil H. Buck	Air Force Accounting and Finance Center
Paul C. Jorgensen	GTE Automatic Electric Labs
Leon Larson	Western Electric Company, Inc.
Rik Maes	University of Amsterdam, The Netherlands
Nicholas L. Marselos	Western Electric Company, Inc.
Charles Muntz	Massachusetts Computer Associates, Inc.
Jonas Rabin	American Bell
Lewis T. Reinwald	U.S. Bureau of Census
Horst Strunz	Mathematischer Beratungs und Programmierungsdienst GmbH
Maurice Verhelst	Leuven University, Belgium

This work was done under the auspices of
the CONFERENCE ON DATA SYSTEMS LANGUAGES
(CODASYL) Systems Committee.

Acknowledgements

We would like to express our sincerest appreciation to all who have assisted us throughout the course of our work. Many people have assisted in keeping us moving toward our goal. Their support and encouragement fueled our effort and inspired us. A special thanks to

Melvin B. Bishop (U.S. General Services Administration)
Raymond F. Brogan (NASA HQ/DS Washington)
Harvey Cohn (American Hospital Supply)
Rene Le Blanc (GTE Automatic Electric)

who put in a significant amount of effort.

Our thanks also to William H. Stieger, past Chairman of the CODASYL Systems Committee. Bill's guidance and comments on the drafts of this report aided in significantly improving its quality.

We extend a special thanks to Rik Maes and the Onderzoeksfonds of the Catholic University of Leuven (Belgium) for permission to include the Annotated Bibliography in our work.

Foreword

The Decision Table Task Group was formed in November 1973 through the efforts of Jonas Rabin. CODASYL's System Committee under the chairmanship of Edgar H. Sibley chartered Mr. Rabin to determine why such a valuable technique as the decision table was not in greater use, and what could be done to improve its usability.

The success Mr. Rabin had in forming the Decision Table Task Group was evident by the level of international participation and a total membership that numbered 33 at its high point. Mr. Rabin was elected the first chairman of the task group and served in that role until January 1975. During that time the task group collected the actual experiences of decision table users, developers, and theoreticians from a worldwide audience. The task group used this information to begin defining the components of a decision table methodology.

From January 1975 to July 1978, Paul Jorgensen served as chairman of the task group. During that time this report was formulated and the preliminary drafts composed.

Nick Marselos became chairman in July 1978 and saw the report through to completion serving also as its editor.

The Decision Table Task Group has explored the past and present use of decision tables in an attempt to understand why the decision table has not gained more than a tenable place both within and external to the data processing community. The task group found that the current status of the decision table has been dictated by several key circumstances in its history:

The early decision table processors and preprocessors were quite inefficient at a time when minimizing the use of computer memory was important.

Powerful decision table constructs have not been incorporated into commercial compilers which limited their availability to the general programmer.

An assumption has been proliferated that decision tables are easy to use, when in fact this is true only after someone has been trained in their effective use.

Few effective programs have been developed for training people in the effective use of decision tables. Consequently students often receive insufficient education in the depth and scope of using single decision tables and multiple table systems.

The objective of this report is to give a new perspective on the use of decision tables. To this end Lewis Reinwald developed Chapter 3 to give a theoretical bases and Chapter 4 to review optimization techniques for decision tables. The task group developed Chapter 6 to assist educators in teaching decision tables to various audiences. Chapter 5 was written for systems developers to describe a process for using decision tables as a system tool. The remainder of the report is meant to aid those in understanding and using decision tables.

The authors hope that this text promotes new interest in decision tables. It is a time to reevaluate this important tool in light of the current needs and trends of technology both in the data processing community and outside of it. The authors feel that the potential benefits and power of the decision table have not been fully explored. They hope that this text will encourage further interest in discovering this unexplored potential.

Table of Contents

- 1. INTRODUCTION**
- 2. OVERVIEW**
- 3. THEORY**
- 4. CONVERSION**
- 5. DEVELOPMENT METHODOLOGY**
- 6. EDUCATION**
- 7. GLOSSARY**
- 8. ANNOTATED BIBLIOGRAPHY**

1. INTRODUCTION

1.1	THIS REPORT	1- 1
1.2	DESCRIPTION OF CHAPTERS	1- 2
1.2.1	Chapter 2 OVERVIEW	1- 2
1.2.2	Chapter 3 THEORY	1- 2
1.2.3	Chapter 4 CONVERSION ALGORITHMS	1- 2
1.2.4	Chapter 5 DEVELOPMENT METHODOLOGY	1- 2
1.2.5	Chapter 6 EDUCATION	1- 3
1.2.6	Chapter 7 GLOSSARY	1- 3
1.2.7	Chapter 8 ANNOTATED BIBLIOGRAPHY	1- 3
1.3	SUGGESTIONS FOR READING THIS REPORT	1- 3

2. OVERVIEW

2.1	INTRODUCTION	2- 1
2.2	A STARTING POINT	2- 1
2.3	BASIC DEFINITIONS	2- 1
2.3.1	Decision Table Format	2- 1
2.3.2	Decision Table Form	2- 3
2.4	EXAMPLES	2- 4
2.4.1	Example 1	2- 4
2.4.2	Example 2	2- 5
2.4.2.1	Expansion	2- 5
2.4.2.2	Solution	2- 6
2.4.3	Example 3	2- 6
2.4.3.1	Expansion	2- 7
2.4.3.2	Solution	2- 7
2.5	COMMENTARY	2- 11

3. THEORY

3.1	INTRODUCTION	3- 1
3.1.1	Objectives	3- 1
3.1.2	Notation conventions	3- 1
3.2	SYNTAX I: DECISION TABLES AS ALGEBRAIC STRUCTURES . . .	3- 2
3.2.1	Formal definition	3- 2
3.2.2	Conditions	3- 2
3.2.2.1	Condition set <u>C</u>	3- 2
3.2.2.2	Condition space <u>C̄</u>	3- 2
3.2.3.	Action	3- 2
3.2.3.1	Action set <u>A</u>	3- 2
3.2.3.2	Action space <u>Ā</u>	3- 3
3.2.4	Relation <u>R</u>	3- 3
3.2.5	Tabular representations	3- 4
3.2.6	Images, domains, and ranges	3- 5
3.2.7	Functions	3- 5
3.2.8	Restrictions	3- 6
3.2.9	Theorem 1	3- 6
3.2.10	Subspaces and decompositions	3- 6
3.2.11	Occurrences	3- 7
3.2.12	Rule classes - equivalence class partitions of <u>C</u> . . .	3- 7
3.3	SEMANTICS	3- 7
3.3.1	Definition	3- 7
3.3.2	Transactions, outputs, and transaction processing . . .	3- 8
3.3.3	Rule satisfaction, supported transactions	3- 9
3.3.4	Programming language representation of decision tables	3- 9
3.3.5	Semantic functions	3-10
3.3.6	Weak constraints for a semantic function	3-11
3.3.7	Semantic restrictions	3-12
3.3.8	Theorem 2	3-12
3.3.9	T-SETS, rule overlap	3-13
3.3.10	Theorem 3	3-13
3.3.11	Semantic decomposition	3-13
3.4	ENTRY AND TABLE TYPES	3-14
3.4.1	Generalized alternatives	3-14
3.4.2	Entry types	3-15
3.4.3	Table types	3-16
3.4.4	Limited entries - a programming language representation	3-16
3.4.5	Implied limited entries	3-17

3.5	SYNTAX II: DECISION TABLES AS LINGUISTIC STRUCTURES	3-18
3.5.1	Tables and quadrants	3-18
3.5.2	Rules	3-19
3.5.3	Stubs and entries	3-19
3.5.4	Relevancy	3-20
3.5.5	Subtables and proper subtables	3-21
3.5.6	Theorem 4	3-22
3.5.7	Inpaths, outpaths, outspans	3-22
3.5.8	Theorem 5	3-22
3.5.9	Cononical Form of decision tables	3-24
3.6	SYSTEMS OF DECISION TABLES	3-26
3.6.1	Linkage via invocation	3-26
3.6.2	Expansion	3-26
3.6.3	Comparison with traditional decision table theory	3-27
3.6.4	Tables invoked by conditions	3-28
3.6.5	Sequences of tables	3-30
3.6.6	Factoring decision tables, linkage via invocation	3-31
3.6.7	Factoring decision tables, linkages via transfer of control	3-32
3.6.8	Total factoring	3-33
3.6.9	Paths and rules	3-34
3.6.10	Total factoring and table completeness and consistency	3-35
3.7	THE EVALUATION CONSTRAINTS	3-36
3.7.1	Introduction	3-36
3.7.2	Output generating clauses (OGC)	3-37
3.7.3	Evaluation conventions for consistent tables	3-37
3.7.4	Basis for consistency criteria for tables with OGC intermixed with conditions	3-38
3.7.5	Consistency with regard to OGC	3-39
3.7.6	Evaluation of inconsistent tables; precedence convention	3-39
3.7.7	Precedence relationships between rules	3-41
3.7.8	ELSE rule	3-41
3.7.9	Multiple hit tables	3-41
3.8	ANALYSIS AND MANIPULATION	3-43
3.8.1	Introduction to checking for completeness and consistency	3-43
3.8.1.1	Motivation	3-43
3.8.1.2	Structural checking by inspection	3-43
3.8.1.3	Semantic checks	3-43
3.8.1.4	Automated structural checks	3-44
3.8.1.5	Automated semantic checks	3-44
3.8.1.6	Impossible rules	3-44
3.8.1.7	Execution time validation	3-45
3.8.1.8	Checking based upon simple rule counts	3-46
3.8.1.9	Theorem 6	3-48

3.8.2	Reordering clauses within a decision table	3-48
3.8.3	Reordering rules	3-48
3.8.4	Resolution of inconsistencies using the precedence convention	3-49
3.8.5	Combining rules	3-50
3.8.6	Converting from extended to limited entries	3-51
3.8.7	Completeness and consistency in converted tables	3-53
3.8.8	Elimination of redundant conditions	3-55
3.8.9	Expansion -- combining tables	3-55
3.8.10	Equivalent decision tables, transformations theorem 7	3-57
3.9	DECISION TABLES AND ITERATIONS	3-58
3.9.1	Motivation	3-58
3.9.2	Simple and complex iteration	3-58
3.9.3	Implementing iterations	3-59
3.9.4	Decision table entry points	3-60
3.9.5	Explicit subtables	3-61
3.9.6	Subtable and "good" programming practices and iterations	3-64
3.9.7	The WHILE condition	3-66

4. CONVERSION ALGORITHMS

4.1	INTRODUCTION	4- 1
4.1.1	Background	4- 1
4.1.2	Parsing techniques	4- 2
4.1.3	The remainder of this chapter	4- 4
4.2	TREE GENERATION I: ORDERED PARSES	4- 4
4.2.1	The mechanics	4- 4
4.2.2	Ordered parses	4- 7
4.2.2.1	Condition order	4- 7
4.2.2.2	Rule order	4- 7
4.2.2.3	"Forgetful" parses - parses with limited look-back	4- 7
4.2.3	Sequencing restrictions	4-10
4.3	TREE GENERATION II: MINIMIZING AVERAGE PROCESSING COSTS	4-14
4.3.1	Goal and cost mechanism	4-14
4.3.2	Verification and search costs	4-17
4.3.3	A branch and bound algorithm	4-19
4.3.4	Extension to handle implied limited entries	4-21
4.3.5	Mutually exclusive conditions which exhaust a set of states	4-21
4.3.6	Generalization to rule classes	4-26
4.3.7	More about action set selection	4-34
4.3.8	A single-alternative algorithm	4-37
4.3.9	Rules that are not mutually exclusive	4-37
4.3.10	Search-free tables	4-44
4.4	TREE GENERATION III: MINIMIZING TOTAL STORAGE COSTS	4-47
4.4.1	Goal and costing mechanism	4-47
4.4.2	Comparison with minimizing average processing cost	4-47
4.4.3	Mergeable entries	4-49
4.4.4	Search entries for optimizing storage	4-49
4.4.5	Search-free conditions	4-49
4.4.6	Storage costs for a branch and bound algorithm	4-52
4.4.7	Hoisting actions	4-54
4.5.	INTERPRETIVE MASKING	4-56
4.5.1	The method	4-56
4.5.2	Processing and storage costs for rule masking	4-59

5. DEVELOPMENT METHODOLOGY

5.1	THE ROLE OF DECISION TABLES IN SYSTEM DESIGN	5- 1
5.1.1	Overview of this chapter	5- 1
5.1.2	Approaching a system problem with decision tables . . .	5- 2
5.2	DECISION TABLE COMPOSITION	5- 2
5.2.1	Criteria for a good table	5- 3
5.2.1.1	Table completeness	5- 3
5.2.1.2	Table size	5- 4
5.2.1.3	Impossible rules	5- 5
5.2.1.4	Consistency	5- 6
5.2.1.5	Improving decision table readability	5- 7
5.2.2	Composition techniques	5- 9
5.2.2.1	The search mode (progressive rule development) . . .	5- 9
5.2.2.2	The direct mode (exhaustive enumeration)	5-11
5.3	SYSTEM DEVELOPMENT	5-16
5.3.1	Systems specification phase	5-16
5.3.1.1	User to producer contract	5-17
5.3.1.2	Significance of the phase	5-17
5.3.1.3	Methodology	5-18
5.3.1.3.1	Understand the problem	5-18
5.3.1.3.2	Structure the problem	5-19
5.3.1.3.3	Elaborate the lower level tables	5-19
5.3.1.3.4	Illustration of the system specifications process .	5-19
5.3.1.3.4.1	The problem as presented	5-20
5.3.1.3.4.2	Problem interpreted by producer	5-21
5.3.1.3.4.3	Restructuring the problem	5-26
5.3.1.3.4.3.1	Initial attempt	5-26
5.3.1.3.4.3.1.1	List all known facts	5-27
5.3.1.3.4.3.1.2	Construct an overall table	5-28
5.3.1.3.4.3.1.3	Elaborate conditions and actions	5-30
5.3.1.3.4.3.2	Final attempt	5-32
5.3.1.3.4.3.3	The final solution	5-37
5.3.1.3.4.3.4	Conclusions	5-40
5.3.2	System design phase	5-41
5.3.2.1	Product of this phase	5-41
5.3.2.2	Method used in this phase	5-42
5.3.2.2.1	Analyze the system specifications for completeness	5-42
5.3.2.2.2	Add the movement of data	5-44
5.3.2.2.3	Add iterations	5-47
5.3.2.2.4	Define the system architecture	5-49
5.3.2.2.5	Add data access and control functions	5-50
5.3.2.2.6	Partition into programmable modules	5-51
5.3.3	Implementation phase	5-53
5.3.3.1	Implementation using a decision table translator . .	5-53
5.3.3.2	Manual translation	5-54

5.3.4	Operation	5-57
5.3.4.1	The problem	5-57
5.3.4.2	The role of decision tables	5-58
5.3.4.3	The format for decision tables	5-58
5.3.4.4	An example	5-58
5.3.5	User acceptance	5-60
5.3.5.1	Objective	5-60
5.3.5.2	Procedure	5-60
5.3.6	Maintenance and extensions	5-60
5.3.6.1	Introduction	5-60
5.3.6.2	Maintenance	5-60
5.3.6.2.1	Need for maintenance	5-60
5.3.6.2.2	Finding the cause	5-65
5.3.6.2.3	Fixing the cause	5-66
5.3.6.3	Extensions and improvements	5-66
5.3.6.4	Analysis and documentation	5-67
5.3.6.4.1	Record all possible paths	5-68
5.3.6.4.2	Study for correctness	5-73
5.3.6.4.3	Rewrite	5-73

6. EDUCATION

6.1	INTRODUCTION	6- 1
6.2	THE SUBJECT MATTER MODULES	6- 1
6.3	COURSE OUTLINE	6- 3
6.4	MODULE DESCRIPTION	6- 4
6.4.1	Introduction	6- 4
6.4.2	Motivation	6- 6
6.4.3	Interpretation	6- 6
6.4.4	Structure	6- 7
6.4.5	Analysis	6- 8
6.4.6	Construction	6- 9
6.4.7	Manipulation	6-10
6.4.8	Multi-Table Systems	6-11
6.4.9	Human Factors	6-12
6.4.10	Translation	6-13
6.4.11	Decision Table Processors	6-14
6.4.12	Testing	6-15
6.4.13	Software Engineering	6-16
6.5	SAMPLE PROBLEMS	6-18
6.5.1	Credit Approval	6-18
6.5.2	Airline Reservation System	6-18
6.5.3	Filling Orders	6-18
6.5.4	Obtaining a Secret Clearance	6-18
6.5.5	Current or Expired Year Appropriations	6-19
6.5.6	To Quit or not to Quit	6-20
6.5.7	Stock Purchase	6-20
6.5.8	Craps	6-20
6.5.9	Inventory File Update	6-20

THE

CONFERENCE ON DATA SYSTEMS LANGUAGES
(CODASYL)

DECISION TABLE TASK GROUP

REPORT

1. INTRODUCTION

1.1	THIS REPORT	1- 1
1.2	DESCRIPTION OF CHAPTERS	1- 2
1.2.1	Chapter 2 OVERVIEW	1- 2
1.2.2	Chapter 3 THEORY	1- 2
1.2.3	Chapter 4 CONVERSION ALGORITHMS	1- 2
1.2.4	Chapter 5 DEVELOPMENT METHODOLOGY	1- 2
1.2.5	Chapter 6 EDUCATION	1- 2
1.2.6	Chapter 7 GLOSSARY	1- 3
1.2.7	Chapter 8 ANNOTATED BIBLIOGRAPHY	1- 3
1.3	SUGGESTIONS FOR READING THIS REPORT	1- 3

1. INTRODUCTION

1.1 THIS REPORT

The purpose of this report is to define decision tables, to explain much of the underlying theory, and to recommend a development methodology that centers on decision tables. The conclusion of this report is that, when properly used, decision tables are an excellent tool for

- . problem analysis
- . specification
- . design
- . coding
- . documentation

This versatility is a consequence of the succinct way in which decision tables can be used as a medium of communication.

The enlightened use of decision tables throughout the full software development process is completely compatible with the use of top-down development with levels of abstraction, program design languages, and structured programming.

The effective use of decision tables requires a proper understanding of their properties and capabilities. This takes an investment in learning time and good source material from which to learn. The purpose of this report is to supply the latter.

1.2 DESCRIPTION OF CHAPTERS

1.2.1 Chapter 2 Overview

Chapter 2 introduces a variety of decision table topics and illustrates some of the advantages of the use of decision tables. A novice will gain a general knowledge of the basic decision table mechanism and will see a vivid demonstration of how decision tables can be used to determine how well a problem is specified.

1.2.2. Chapter 3 THEORY

A theoretical foundation for decision tables based on set theory and modern algebra is given as a basis for the discussion of conversion algorithms in Chapter 4. This formulation is a significant departure from the conventional definitional approach to decision tables, and contains several novel results. This description provides an excellent starting point for serious research in decision table theory and also clarifies various aspects of decision table usage that are otherwise difficult to describe.

1.2.3 Chapter 4 CONVERSION ALGORITHMS

This chapter is a review of decision table optimization techniques. The algebraic formulation of decision tables given in Chapter 3 is the basis for an extensive discussion of issues related to decision table processors. Two types of tree generation are discussed, with various parsing techniques applicable to each. This chapter is replete with brief examples to illustrate the concepts, techniques, and algorithms being defined.

1.2.4 Chapter 5 DEVELOPMENT METHODOLOGY

The recommended use of decision tables in each phase of the software development process, from specification to acceptance and maintenance is described in this chapter. The wide variety of possible ways to use decision tables described in Chapter 3 is filtered through current software engineering constraints, resulting in techniques for using decision tables to best advantage through the full software development process. One of the interesting instances of the use of decision tables is an iterative process between the problem poser and the problem solver to produce a mutually understandable (and acceptable) statement of the problem to be solved. The chapter includes a non-trivial example carried through this process; many of the salient advantages of the use of decision tables are illustrated.

1.2.5 Chapter 6 EDUCATION

A plan for teaching decision table technology to various categories of users is presented in this chapter. A series of thirteen subject matter modules is identified. The content of each module is defined so that it can be taught at an overview level for a general audience or at a detailed level for a sophisticated audience. Training sequences are then derived from these basic thirteen modules. A set of sample problems useful in teaching various decision table concepts is given at the end of the chapter.

1.2.6 Chapter 7 GLOSSARY

The glossary contains a brief definition of many terms used in the report.

1.2.7 Chapter 8 ANNOTATED BIBLIOGRAPHY

The Catholic University of Leuven, Belgium has published an annotated bibliography on decision table publications. This bibliography has been included here with permission of the author.

1.3 SUGGESTIONS FOR READING THIS REPORT

This report has been written with audiences in mind ranging from those who wish only a general introduction to decision tables to the computer science researcher. The sequence in which the chapters are read should be varied by the readers' interests and background. Chapter 2, BASIC NOTIONS, and a light reading of Chapter 5, DEVELOPMENT METHODOLOGY, will provide a general level introduction into the use and benefits of decision tables. The educator can continue on to Chapter 6, EDUCATION, and a serious review of the bibliography in Chapter 8. Finally, anyone interested in serious research in decision tables should read Chapters 3, THEORY, and Chapter 4, CONVERSION ALGORITHMS.

2. OVERVIEW

2.1	INTRODUCTION	2- 1
2.2	A STARTING POINT	2- 1
2.3	BASIC DEFINITIONS	2- 1
2.3.1	Decision Table Format	2- 1
2.3.2	Decision Table Form	2- 3
2.4	EXAMPLES	2- 4
2.4.1	Example 1	2- 4
2.4.2	Example 2	2- 5
2.4.2.1	Expansion	2- 5
2.4.2.2	Solution	2- 6
2.4.3	Example 3	2- 6
2.4.3.1	Expansion	2- 7
2.4.3.2	Solution	2- 7
2.5	COMMENTARY	2- 11

2. OVERVIEW

2.1 INTRODUCTION

This chapter is intended for readers having little or no familiarity with decision tables. The basic concepts and principles are defined so that the inexperienced reader will be able to appreciate the lucidity and flexibility of decision tables and understand the way in which decision tables are applied in the solution of procedural decision situations.

2.2 A STARTING POINT

Decision tables are used to describe and analyze problems that contain procedural decision situations that are characterized by one or more conditions the state of which determines the execution of a set of actions. Such situations may be too complex to be described simply with IF-THEN-ELSE and DO-CASE structures.

The process of describing a procedural decision situation by a decision table involves: (1) identifying all the conditions and actions associated with the situation, and (2) indicating which actions must be executed for various combinations of conditions. The decision table itself is simply a concise notational device for listing these conditions and actions as decision rules.

2.3 BASIC DEFINITIONS

2.3.1 Decision table format

A decision rule is the basic ingredient of the decision table. A rule describes a set of condition alternatives and a series of actions to be performed.

A decision table is a structure for describing a set of related decision rules. The basic parts of a decision table are shown in Figure 2-1. The upper left portion of the format is called the condition stub quadrant; it contains statements of the conditions. Similarly, the lower left portion is called the action stub quadrant; it contains statements of the actions. The condition entry and action entry quadrants appear in the upper right and lower right portions of the format, respectively. Each column in the entry portions (condition and action) forms a decision rule.

Decision tables are named by the content in the entry portion of the table:

Limited-entry tables
 Extended-entry tables
 Mixed-entry tables

The decision table in Figure 2-1 is an example of a limited-entry table. This table contains only Y, N or - in the condition entry area and X or - in the action entry area.

		Decision Rules					
		1	2	3	4	5	6
Condition Stub		Condition Entries					
:	Condition 1	: Y	Y	Y	N	N	N
:	Condition 2	: Y	Y	N	Y	N	N
:	Condition 3	: Y	N	-	-	Y	N
Action Stub		Action Entries					
:	Action 1	: X	-	X	X	X	X
:	Action 2	: -	X	X	-	X	-
:	Action 3	: X	-	-	X	X	-
:	Action 4	: -	X	-	X	-	X

Where:

Y = Yes	= Condition is true
N = No	= Condition is false
- = Irrelevant	= Condition doesn't matter
- or blank	= Action is not performed
X	= Action is performed

Figure 2-1
 Decision Table Format

In an extended-entry decision table, a portion of the condition [action] appears in the entry position of the table. An example of this extension is shown in Figure 2-2. A mixed-entry table is one which contains both limited and extended entries.

Decision Rules						
	1	2	3	4	5	
Condition Stub						
Variable 1	<7	<7	=7	=7	>7	
Variable 2	=1	Not = 1	-	-	-	
Variable 3	=5	-	<5	>5	-	
Action Stub	Action Entries					
Execute Procedure	105	71	47	36	81	

Figure 2-2
Extended-entry Decision Table

2.3.2 Decision table form

The general form for a procedural decision situation is the "If Condition 1 and Condition 2 and Condition 3 and ... then Action 1 and Action 2 and ...". Interpreting the first two decision rules in Figure 2-2 gives -- If Variable 1 is less than 7 and Variable 2 is equal to 1 and Variable 3 is equal to 5, then Execute Procedure 105, or If Variable 1 is less than 7 and Variable 2 is not equal to 1, then Execute Procedure 71.

2.4 EXAMPLES

Three examples are presented here. The first is a brief example to illustrate the basic decision table definitions discussed above, while the second the third are more extensive and show the way in which decision tables can be used to analyze as well as describe a procedural decision situation.

2.4.1 Example 1

Utility Rates: The local electric company charges residential customers a basic charge of \$1.00 and a use charge of \$0.03 per kilowatt hour (KWH) during the peak period (9 a.m. to 6 p.m.) and \$0.01 per KWH from 6 p.m. till 9 a.m.; commercial customers are charged a base of \$5.00 and a rate of \$0.015 during the peak period and \$0.005 per KWH other times; industrial customers pay a \$10.00 base and a \$0.009 rate.

Expressing the company's rate structure as a decision table requires the identification of conditions, actions and decision rules. The conditions are: type of customer and peak period. There are two actions: basic charge and use charge. Displaying these conditions and actions in decision table form results in Figure 2-3.

							Decision Rules			
		1	2	3	4	5				
:	Customer is	Residential	Residential	Commercial	Commercial	Industrial				
:	Peak Period?	Y	N	Y	N	-				
:										
:	Basic Charge	\$1.00	\$1.00	\$5.00	\$5.00	\$10.00				
:	Use Charge	\$0.03	\$0.01	\$0.015	\$0.005	\$0.009				

Figure 2-3
Electric Rates

Although this example is very simple, comparing the decision table of Figure 2-3 with the corresponding text, entitled "Utility Rates", immediately shows that decision tables can more clearly specify a decision situation.

2.4.2 Example 2

Traffic Light: There is a slightly intelligent traffic light at the intersection of Otis St. and Lowell Avenue. There is much more traffic on Lowell Avenue than there is on Otis Street, and so the light is normally green for Lowell Avenue and normally red for Otis Street. But there are two treadles on the roadway of Otis Street, for cars approaching on Otis Street from either direction. If a car on Otis Street drives up to the intersection, then ordinarily, and irrespective of any traffic on Lowell Avenue, the light will turn green practically at once for the Otis Street driver, and he will be able to go through the intersection with a green light and hardly any delay. But that green light for Otis Street lasts only long enough for four or five cars to drive through the intersection, and then it once more turns red. Now if another car approaches on Otis Street, that light stays red a long time, irrespective of any cars on Lowell Avenue to use it. It lasts two minutes, and then once more it will turn green for the Otis Street driver.*

The procedural decision situation presented above is basically good, but some clarification is necessary and additional information required to improve the situation. Such words as "ordinarily", "practically", and "hardly any delay" can be very misleading.

2.4.2.1 Expansion

Apparently the light is controlled by a timer whose behavior is not described in the quotation. Also, the duration of the green light for Otis Street is described as "only long enough for four or five cars". The original quotation can be augmented to include the following statements:

The duration of a green light for Otis Street is 15 seconds.

There is a minimum duration of 2 minutes for a green light on Lowell Avenue.

The timer operates in intervals of 0.01 seconds.

Every hundredth of a second the status of the light, of the treadle, and of the time are interrogated. If this interrogation results in a change in the status of the light, the timer is reset to zero; otherwise, it is incremented by 0.01.

* Example from Edmund C. Berkely

2.4.2.2. Solution

A decision table describing the behavior of the light at Otis Street and Lowell Avenue is shown in Figure 2-4.

	:	1	2	3	4	5	6	:	
:	:	
:	Is Lowell Avenue Light Green?	:	Y	Y	Y	Y	N	N	:
:	Is Otis Street Treadle OFF?	:	Y	Y	N	N	-	-	:
:	Is Time?	:	≤ 120	> 120	> 120	≤ 120	> 15	≤ 15	:
:	Change Light Direction	:	-	-	X	-	X	-	:
:	Set Timer to zero	:	-	-	X	-	X	-	:
:	Set Treadle OFF	:	-	-	-	-	X	-	:
:	Increment Timer by 0.01 sec	:	X	-	-	X	-	X	:
:	Repeat Table Logic	:	X	X	X	X	X	X	:
:	:	

Figure 2-4
Traffic Light

2.4.3 Example 3

This example illustrates the way in which decision tables can be used to resolve ambiguities and identify incomplete problem descriptions.

Examination Regulations: The student who was not successful in the first examination session of the academic year belongs to one of the following categories: failed, rejected, equivalent to rejected. If a student failed the first examination, he is admitted to the second examination session of the academic year. A student is considered "rejected" only because of cheating during the exams. A rejected student cannot participate in the second examination session. A student is considered "equivalent to rejected" if he does not participate in the first examination session (unless there is a serious reason for missing the exam). A student who is "equivalent to rejected" may participate in the second examination only by special permission of his department.

2.4.3.1 Expansion

These regulations are described by the decision table in Figure 2-5.

	Decision Rules															
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
: Student successful in first exam	Y	Y	Y	Y	Y	Y	Y	Y	Y	N	N	N	N	N	N	N
: Student caught cheating	Y	Y	Y	Y	N	N	N	N	Y	Y	Y	Y	N	N	N	N
: Student took first exam	Y	Y	N	N	Y	Y	N	N	Y	Y	N	N	Y	Y	N	N
: Student has serious reason for not taking first exam	Y	N	Y	N	Y	N	Y	N	Y	N	Y	N	Y	N	Y	N
: Failed	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
: Rejected	-	-	-	-	-	-	-	-	X	X	X	X	-	-	-	-
: Equivalent to Rejected	-	-	-	-	-	-	-	-	-	-	X	-	-	X	-	-

Figure 2-5
Examination Regulations

The decision table in Figure 2-5 exactly records the verbal statement of the examination regulations. Nothing has been added or deleted. Notice the following aspects of the regulations:

- (1) No actions are indicated for the first eight rules. The regulations only seem to be concerned with students who were not successful in the first examination session.
- (2) The "failed" action is never executed. The regulations do not explicitly state the rules under which a student fails.

2.3.4.2 Solution

In this section, various properties of decision tables (which are explained in Chapters 3 and 5) are used to analyze the stated requirements and improve the problem description.

Completeness

If the decision table in Figure 2-5 is complete, then there are no rules that are missing. An alternate way of saying this is that actions are given for all possible circumstances of the conditions. The decision table in Figure 2-5 is, in fact, a complete decision table. Every possible combination of conditions is stated and no rules are redundant.

Consistency

Given a complete decision table, the first step is to determine whether or not it is consistent. This involves identifying whether or not certain combinations of conditions are "impossible" in terms of the problem. The second step is to resolve missing actions for "possible" rules.

Referring to the decision table in Figure 2-5, it is clear that Rules 3, 4, 7, and 8 are impossible because they refer to a student who was successful on the first exam and did not take the first exam. Similarly, Rules 11 and 12 are impossible because they refer to a student who was caught cheating and did not take the first exam. Rule 12 as originally stated contains a second error: the student is considered to be both "rejected" and "equivalent to rejected."

At this point, the remaining rules (1, 2, 5, 6, 13, 14, 15) would be returned to the problem poser (a faculty committee) for further definition. For the purpose of this example, suppose the response is as follows:

Rules 1 and 2: Refer to a faculty committee to choose between "pass" and "reject" depending on the degree of cheating involved.

Rules 5 and 6: Student is considered "passed."

Rules 13, 14, 15: Student is considered "failed."

The decision table in Figure 2-6 contains the first revision of the examination policy. Notice that two new actions (passed and impossible) have been added.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
: Successful?	Y	Y	Y	Y	Y	Y	Y	Y	N	N	N	N	N	N	N	N
: Cheating?	Y	Y	Y	Y	N	N	N	N	Y	Y	Y	Y	N	N	N	N
: Took first exam?	Y	Y	N	N	Y	Y	N	N	Y	Y	N	N	Y	Y	N	N
: Serious reason for : missing first exam?	Y	N	Y	N	Y	N	Y	N	Y	N	Y	N	Y	N	Y	N
: Impossible	-	-	X	X	-	-	X	X	-	-	X	X	-	-	-	-
: Passed	-	-	-	-	X	X	-	-	-	-	-	-	-	-	-	-
: Failed	-	-	-	-	-	-	-	-	-	-	-	X	X	X	X	-
: Rejected	-	-	-	-	-	-	-	-	X	X	-	-	-	-	-	-
: Equivalent to Rejected:	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	X
: Refer to committee	X	X	-	-	-	-	-	-	-	-	-	-	-	-	-	-

Figure 2-6
Revised Examination Regulations

Finally, it is necessary to consolidate decision rules where possible. The result of rule consolidation is a decision table that can be more easily read by people. This result is shown in Figure 2-7. The rule numbers refer to the rules in the decision table in Figure 2-6.

	(Original Rule Numbers)								
	1,2	3,4,7,8	5,6	9,10	11,12	13,1		16	
: Successful?	:	Y	Y	Y	N	N	N	N	N
: Cheating?	:	Y	-	N	Y	Y	N	N	N
: Took first exam?	:	Y	N	Y	Y	N	Y	N	N
: Serious reason for : missing first exam?	:	-	-	-	-	-	-	Y	N
: Impossible	:	-	X	-	-	X	-	-	-
: Passed	:	-	-	X	-	-	-	-	-
: Failed	:	-	-	-	-	-	X	X	-
: Rejected	:	-	-	-	X	-	-	-	-
: Equivalent to Rejected:	-	-	-	-	-	-	-	-	X
: Refer to committee	:	X	-	-	-	-	-	-	-

Figure 2-7
Condensed Examination Regulations

2.5 COMMENTARY

Based on the foregoing examples, it should be clear that decision tables are a concise, accurate, and flexible format to represent procedural decision situations. The material in Chapter 5 (Development Methodology) describes the various ways that decision tables can be used in systems development. Such situations are partitioned by the desired end result: either a computer program or a set of decision procedures to be manually executed. For the former case, various computer and translator-related issues become important; these are discussed in Chapter 4 (Conversion Algorithms). Generally speaking, if a decision table is to be "executed" manually, it can have a less restrictive (more human-oriented) format, limited only by the person who will "execute" the decision table.

Perhaps the greatest advantage of using decision tables in designing systems is that they introduce a discipline. The designer must begin with a notion of how the system will work. However, because of the exhaustive combinatorial power of decision tables and their inherent logical testing possibilities, system failures will be brought to the designer's attention during the design stage rather than in the middle of a crucial 'live' situation. Because of their simple syntactic structure, decision tables can be read and stored very easily by a computer. They can therefore form the basis for a machine-readable representation of the design.

Decision tables have a simple and human-oriented structure. Therefore, they are an ideal medium for documentation and communication. Indeed, decision tables can be manipulated and understood very easily by the user. This gives to the user some control over systems development and will increase the user's confidence in the system.

3. THEORY

3.1	INTRODUCTION	3- 1
3.1.1	Objectives	3- 1
3.1.2	Notation conventions	3- 1
3.2	SYNTAX I: DECISION TABLES AS ALGEBRAIC STRUCTURES	3- 2
3.2.1	Formal definition	3- 2
3.2.2	Conditions	3- 2
3.2.2.1	Condition set <u>C</u>	3- 2
3.2.2.2	Condition space <u>C̄</u>	3- 2
3.2.3.	Action	3- 2
3.2.3.1	Action set <u>A</u>	3- 2
3.2.3.2	Action space <u>Ā</u>	3- 3
3.2.4	Relation <u>R</u>	3- 3
3.2.5	Tabular representations	3- 4
3.2.6	Images, domains, and ranges	3- 5
3.2.7	Functions	3- 5
3.2.8	Restrictions	3- 6
3.2.9	Theorem 1	3- 6
3.2.10	Subspaces and decompositions	3- 6
3.2.11	Occurrences	3- 7
3.2.12	Rule classes - equivalence class partitions of <u>C</u>	3- 7
3.3	SEMANTICS	3- 7
3.3.1	Definition	3- 7
3.3.2	Transactions, outputs, and transaction processing	3- 8
3.3.3	Rule satisfaction, supported transactions	3- 9
3.3.4	Programming language representation of decision tables	3- 9
3.3.5	Semantic functions	3-10
3.3.6	Weak constraints for a semantic function	3-11
3.3.7	Semantic restrictions	3-12
3.3.8	Theorem 2	3-12
3.3.9	T-SETS, rule overlap	3-13
3.3.10	Theorem 3	3-13
3.3.11	Semantic decomposition	3-13
3.4	ENTRY AND TABLE TYPES	3-14
3.4.1	Generalized alternatives	3-14
3.4.2	Entry types	3-15
3.4.3	Table types	3-16
3.4.4	Limited entries - a programming language representation	3-16
3.4.5	Implied limited entries	3-17

3.5	SYNTAX II: DECISION TABLES AS LINGUISTIC STRUCTURES	3-18
3.5.1	Tables and quadrants	3-18
3.5.2	Rules	3-19
3.5.3	Stubs and entries	3-19
3.5.4	Relevancy	3-20
3.5.5	Subtables and proper subtables	3-21
3.5.6	Theorem 4	3-22
3.5.7	Inpaths, outpaths, outspans	3-22
3.5.8	Theorem 5	3-24
3.5.9	Cononical Form of decision tables	3-24
3.6	SYSTEMS OF DECISION TABLES	3-26
3.6.1	Linkage via invocation	3-26
3.6.2	Expansion	3-26
3.6.3	Comparison with traditional decision table theory	3-27
3.6.4	Tables invoked by conditions	3-28
3.6.5	Sequences of tables	3-30
3.6.6	Factoring decision tables, linkage via invocation	3-31
3.6.7	Factoring decision tables, linkages via transfer of control	3-32
3.6.8	Total factoring	3-33
3.6.9	Paths and rules	3-34
3.6.10	Total factoring and table completeness and consistency . .	3-35
3.7	THE EVALUATION CONSTRAINTS	3-36
3.7.1	Introduction	3-36
3.7.2	Output generating clauses (OGC)	3-37
3.7.3	Evaluation conventions for consistent tables	3-37
3.7.4	Basis for consistency criteria for tables with OGC intermixed with conditions	3-38
3.7.5	Consistency with regard to OGC	3-39
3.7.6	Evaluation of inconsistent tables; precedence convention .	3-39
3.7.7	Precedence relationships between rules	3-41
3.7.8	ELSE rule	3-41
3.7.9	Multiple hit tables	3-41
3.8	ANALYSIS AND MANIPULATION	3-43
3.8.1	Introduction to checking for completeness and consistency .	3-43
3.8.1.1	Motivation	3-43
3.8.1.2	Structural checking by inspection	3-43
3.8.1.3	Semantic checks	3-44
3.8.1.4	Automated structural checks	3-44
3.8.1.5	Automated semantic checks	3-44
3.8.1.6	Impossible rules	3-45
3.8.1.7	Execution time validation	3-46
3.8.1.8	Checking based upon simple rule counts	3-46
3.8.1.9	Theorem 6	3-48

3.8.2	Reordering clauses within a decision table	3-48
3.8.3	Reordering rules	3-48
3.8.4	Resolution of inconsistencies using the precedence convention	3-49
3.8.5	Combining rules	3-50
3.8.6	Converting from extended to limited entries	3-51
3.8.7	Completeness and consistency in converted tables	3-53
3.8.8	Elimination of redundant conditions	3-55
3.8.9	Expansion -- combining tables	3-55
3.8.10	Equivalent decision tables, transformations theorem 7 . . .	3-57
3.9	DECISION TABLES AND ITERATIONS	3-58
3.9.1	Motivation	3-58
3.9.2	Simple and complex iteration	3-58
3.9.3	Implementing iterations	3-59
3.9.4	Decision table entry points	3-60
3.9.5	Explicit subtables	3-61
3.9.6	Subtable and "good" programming practices and iterations . .	3-64
3.9.7	The WHILE condition	3-66

3. THEORY

3.1 INTRODUCTION

3.1.1 Objectives

This section endeavors to establish a rigorous theoretical foundation for decision tables and upon this foundation to systematically build techniques for analyzing and manipulating tables and systems of tables. Hopefully such an approach will align decision tables more closely with mathematics and logic, and strengthen their bond to current thought in computer science and theory of computation. The foundation is also intended to provide a conceptual framework for the automatic conversion of decision tables to computer programs. Finally, the section extends the concepts and vocabulary associated with decision tables.

Subsection 3.2 immediately following treats a decision table as an algebraic mapping from points in a condition space to points in an action space. This is followed by subsection 3.3 which views decision tables as a machine or algorithm which transforms inputs into outputs. Subsection 3.4 reviews the popular classification of decision tables into extended, limited, and mixed entry type tables. Then subsection 3.5 provides a third view of decision tables; in this case as a specialized format for depicting relationships. Subsection 3.6 examines systems of decision table and forms the basis for 3.7, which provides a definitive statement of what it means to evaluate a table. The next subsection, 3.8, discusses decision table manipulation and analysis subject to the constraints of the evaluation scheme developed in 3.7. The last subsection, 3.9, addresses the problem of iteratively repeating tables and subtables.

3.1.2 Notation conventions

Brackets, [], are used to denote an unordered set.

Parentheses, (), are used to denote ordered sets including subscripts.

--: are used to denote implies, or is mapped into.

Numerical suffixes and subscripts are used solely as identifiers and have no implication with regard to ordering. Likewise the ordinals (first, second, third, etc.), and cardinals, such as rule 3 and condition 2, are usually used to identify and do not imply order.

3.2 SYNTAX I: DECISION TABLES AS ALGEBRAIC STRUCTURES

3.2.1 Formal definition

A decision table can be defined as a triple $D = (C, A, R)$ where C is a set of conditions, A is a set of actions, and R is a set of rules, i.e., R is a relation which maps C into A so that $A = R(C)$. Examining C , A , and R more closely, we have:

3.2.2 Conditions

3.2.2.1 Condition set C

$C = [C(1), C(2), C(3), \dots]$ = a set of conditions $C(1), C(2), C(3), \dots$. Each condition $C(i)$ consists of a subject $CS(i)$ and a set of alternatives $[CA(i,j), j = 1, 2, \dots, m(i)]$ and is written as:
 $C(i) : CS(i), [CA(i,1), CA(i,2), \dots].$

3.2.2.2 Condition space \bar{C}

$\bar{C} = C(1) \times C(2) \times C(3) \dots C(n)$ where \times denotes cartesian product, so that C is a collection of points each of which is defined as an ordered n -tuple

$(CA(1,K1), CA(2,K2), \dots)$ where $CA(1,K1) \in [CA(1,1), CA(1,2), \dots]$ and $CA(2,K2) \in [CA(2,1), CA(2,2), \dots]$ etc.

EXAMPLE

Let C_1 = a set of two conditions

$C_1(1) : SEX, [male, female]$
 $C_1(2) : AGE, [under 20, 20-60, over 60]$

then $\bar{C}_1 = [(C_1A(1,1), C_1A(2,1)), (C_1A(1,1), C_1A(2,2)), (C_1A(1,1), C_1A(2,3))$
 $\quad (C_1A(1,2), C_1A(2,1)), (C_1A(1,2), C_1A(2,2)), (C_1A(1,2), C_1A(2,3))]$
 $= [(male, under 20), (male, 20-60), (male, over 60),$
 $\quad (female, under 20), (female, 20-60), (female, over 60)]$

3.2.3 Actions

3.2.3.1 Action set A

$A = [A(1), A(2), A(3), \dots]$

Each action $A(i)$ consists of a subject $AS(i)$ and a set of alternatives $[AA(i,j), j = 1, 2, \dots, m(i)]$ written as:
 $A(i) : AS(i), [AA(i,1), AA(i,2), \dots]$

3.2.3.2 Action space \bar{A}

$\bar{A} = A(1) \times A(2) \times A(3) \dots \times A(n)$ where \times denotes cartesian product so that \bar{A} is a collection of points each of which is an ordered n-tuple $(AA(1,K1), AA(2,K2)\dots)$ where $AA(1,K1) \in [AA(1,1), AA(1,2)\dots]$, $AA(2,K2) \in [AA(2,1), AA(2,2)\dots]$ etc.

EXAMPLE

Let action set $A1$ consist of three actions:

```
A1(1): COUNT-1, [incre-by-1,do-not-change]
A1(2): set Z to, [1,3,0]
A1(3): TALLY-REC, [do,do-not-do]
```

Then

$$\begin{aligned}\bar{A1} &= [(A1A(1,1), A1A(2,1), A1A(3,1)), (A1A(1,1), A1A(2,1), A1A(3,2)), \\ &\quad ((A1A(1,1), A1A(2,2), A1A(3,1)), (A1A(1,1), A1A(2,2), A1A(3,2)), \\ &\quad ((A1A(1,1), A1A(2,3), A1A(3,1)), (A1A(1,1), A1A(2,3), A1A(3,2)), \\ &\quad ((A1A(1,2), A1A(2,1), A1A(3,1)), (A1A(1,2), A1A(2,3), A1A(3,2)), \\ &\quad \dots] \\ &= [(incre-by-1,1,do), (incre-by-1,1,do-not-do), \\ &\quad (incre-by-1,3,do), (incre-by-1,3,do-not-do), \\ &\quad (incre-by-1,0,do), (incre-by-1,0,do-not-do), \\ &\quad (do-not-change,1,do), (do-not-change,1,do-not-do), \\ &\quad (do-not-change,3,do), (do-not-change,3,do-not-do), \\ &\quad (do-not-change,0,do), (do-not-change,0,do-not-do)],\end{aligned}$$

3.2.4 Relation R

Relation R maps points of \bar{C} into points of \bar{A} . It consists of a series of correspondences or rules each of which associates a point in \bar{C} with a point in \bar{A} .

EXAMPLE D1

Given $C1$ and $A1$ as previously defined we can define a relation $R1$ as follows:

```
[(male,under 20) --: (incre-by-1,1,do),
 (male,20-60) --: (do-not-change,1,do),
 (male,over 60) --: (do-not-change,0,do-not-do),
 (female,under 20) --: (incre-by-1,1,do),
 (female,20-60) --: (incre-by-1,1,do),
 (female,over 60) --: (do-not-change,3,do)]
```

3.2.5 Tabular representations

Linguistically a decision table can be looked upon as a stylized format for representing the relation A=R(C). In such tables: 1) the subjects, CS(i) and AS(i) generally become column headings or, more likely row stubs, 2) the alternatives CA(i,j), and AA(i,j), become entries, and 3) each rule becomes a row or, more frequently, a column of entries.

EXAMPLE (rules written as rows)

```
.....:.....:.....:.....:.....:.....:  
: : : : : :  
: SEX= AGE= : COUNT-1 Set Z to TALLY-REC :  
:.....:.....:.....:.....:.....:  
: male under 20 : incre-by-1 1 do :  
: : : : :  
: male 20-60 : do-not-change 1 do :  
: : : : :  
: male over 60 : do-not-change 0 do-not-do :  
: : : : :  
: female under 20 : incre-by-1 1 do :  
: : : : :  
: female 20-60 : incre-by-1 1 do :  
: : : : :  
: female over 60 : do-not-change 3 do :  
:.....:.....:.....:.....:.....:  
.....:.....:.....:.....:.....:.....:  
: : : : : :  
:SEX : male male male female female female :  
: : : : : :  
:AGE : under 20 20-60 over 60 under 20 20-60 over 60 :  
:.....:.....:.....:.....:.....:.....:  
:COUNT-1 : incre-by-1 do-not- do-not- incre-by-1 incre-by-1 do-not- :  
: : change change :  
: : : : : :  
:set-Z to : 1 1 0 1 1 3 :  
: : : : : :  
:TALLY-REC : do do do-not-do do do do :  
: : : : : :  
:.....:.....:.....:.....:.....:  
.....:.....:.....:.....:.....:
```

EXAMPLE (rules written as columns)

```
.....:.....:.....:.....:.....:.....:  
: : : : : :  
:SEX : male male male female female female :  
: : : : : :  
:AGE : under 20 20-60 over 60 under 20 20-60 over 60 :  
:.....:.....:.....:.....:.....:.....:  
:COUNT-1 : incre-by-1 do-not- do-not- incre-by-1 incre-by-1 do-not- :  
: : change change :  
: : : : : :  
:set-Z to : 1 1 0 1 1 3 :  
: : : : : :  
:TALLY-REC : do do do-not-do do do do :  
: : : : : :  
:.....:.....:.....:.....:.....:  
.....:.....:.....:.....:.....:
```

3.2.6 Images, domains, and ranges

If c is a point of \bar{C} , a is a point in \bar{A} , and $R(c) = a$, then a is said to be the image of c under R . The domain of R is defined as the collection of points in \bar{C} for which R yields an image in \bar{A} . The range of R is the collection of points in \bar{A} which are the image under R of some point in \bar{C} .

EXAMPLE

The domain of R_1 as specified previously (section 3.2.5) consists of all points in C_1 viz:

```
[(male, under 20), (male, 20-60), (male, over 60),
 (female, under 20), (female, 20-60), (female, over 60)]
```

The range of R is a subset of A_1 , viz:

```
[(incre-by-1, 1, do), (do-not-change, 1, do), (do-not-change, 0,
 do-not-do), (do-not-change, 3, do)]
```

3.2.7 Functions

R is said to be a function from C to A if

- 1) The domain of R is \bar{C} , i.e., every point c in \bar{C} has an image $R(c) = a$ in \bar{A}
- 2) The image of c under R is unique; if $R(c_1) = a_1 \neq a_2 = R(c_2)$ then $c_1 \neq c_2$.

These requirements can be restated in terms of R 's decision table representation. First, the table must be complete, i.e., have a rule for every combination of condition alternatives. And it must be consistent, i.e., the rules must not overlap; they must be pairwise mutually exclusive so that there is only one rule for each combination of conditions.

EXAMPLE

The relation R_1 displayed in 3.2.5 above is a function. However, the following relation on C_1 and A_1 as previously defined fails both criteria and is not a function:

```
[(male, under 20) --: (incre-by-1, 1, do)
 (male, under 20) --: (incre-by-1, 3, do-not-do)
 (female, 20-60) --: (do-not-change, 2, do)]
```

3.2.8 Restrictions

Let R be a relation from C to A and let \bar{D} be subset of points in \bar{C} . The relation Q is said to be a restriction of R to D , if D is the domain of Q , and $Q(d) = R(d)$ for every d in D .

3.2.9 Theorem 1

R is a function if and only if every restriction of R is a function.

Later on in our treatment of decision tables, this seemingly trivial theorem is important, because it allows us to determine a table's completeness and consistency by breaking it into smaller and smaller tables. It also is the foundation for correctness proofs for algorithms which parse decision tables into program trees.

3.2.10 Subspaces and decompositions

\bar{D} is said to be a subspace of \bar{C} , if it is a subset of \bar{C} obtained by fixing the alternatives for m conditions associated with C . In this circumstance \bar{D} is said to be selected by the m alternatives. The relation Q is a decomposition of R , if Q is a restriction of R to E and E is a subspace of the domain of C . As a matter of convenience, the fixed alternatives may be omitted.

EXAMPLE

Let C_1 be as previously defined viz.:

Condition $C_1(1)$: SEX, [male, female]

$C_1(2)$: AGE, [under 20, 20-60, over 60]

Then $\bar{C}_1 = \{(male, under 20), (male, 20-60), (male, over 60), (female, under 20), (female, 20-60), (female, over 60)\}$

The subspace of \bar{C}_1 selected by the alternative, age 20-60 is:

$\{(male, 20-60), (female, 20-60)\}$

If R_1 is as defined as in 3.2.5, the decomposition of R_1 selected by AGES = 20-60 is:

$\{(male, 20-60) --: (do-not-change, 1, do), (female, 20-60) --: (incre-by-1, 1, do)\}$

Or with the fixed alternative dropped:

$\{(male) --: (do-not-change, 1, do), (female) --: (incre-by-1, 1, do)\}$

3.2.11 Occurrences

Sometimes it is of interest to consider a decomposition involving only one condition, say $C(k)$, in a table. Such a decomposition is defined as an occurrence of $C(k)$ and is selected by assigning fixed alternatives to the $n-1$ conditions $C(j) \neq k$.

3.2.12 Rule classes - equivalence class partition of C

If R is a function from \bar{C} to \bar{A} , R can be used to partition \bar{C} into a collection of equivalence classes or rule classes. A rule class is a set of points in C each of which is mapped by R into a given point of A .

Example: considered R_1 as previously defined; R_1 partitions \bar{C}_1 into 4 rule sets, namely:

$\bar{R}C_1 = \{(male, under 20), (female, under 20), (female, 20-60)\}$
are mapped into (incre-by-1,1,do)

$\bar{R}C_2 = \{(male, 20-60)\}$ is mapped into (do-not-change,1,do)

$\bar{R}C_3 = \{(male, over 60)\}$ is mapped into (do-not-change,0,do-not-do)

$\bar{R}C_4 = \{(female, over 60)\}$ is mapped into (do-not-change,3,do)

3.3 Semantics

3.3.1 Definition

So far we have been concerned solely with the structure of decision tables and have not dealt with the meaning or semantics of a table. Semantically, we define a decision table as a mechanism which transforms a set of inputs $V = [V_1, V_2, \dots]$ into a set of outputs $W = [W_1, W_2, \dots]$. The inputs to a decision table comprise variables directly tested by the table's conditions or utilized by its actions, and indirectly, those used as input to procedures invoked by the table. We shall call those variables tested by a table's conditions, selector inputs. Outputs comprise variables directly set or altered by one of the table's actions and indirectly, by procedures invoked by the table.

EXAMPLE

Inputs to D_1 , as previously defined, are SEX, AGE, COUNT-1 and, indirectly, variables used by TALLY-REC. Outputs comprise COUNT-1, Z, and, indirectly, any variables set or altered by TALLY-REC.

3.3.2 Transactions, outputs, and transaction processing

A unit of input to a decision table will be called a transaction, t . A transaction contains values for the variables $[V_1, V_2, \dots]$ which are input to the table. Similarly, we define s as the set of outputs (W_1, W_2, \dots) generated by processing t against decision table D , and express this relationship by writing $s = D(t)$. Transaction t is processed against a table as follows:

1. Using values supplied by t , each of the table's conditions is evaluated to find the alternative appropriate to t .
2. The alternatives chosen in step 1 are combined to determine a point, c , in C .
3. Point a , the image of c in \bar{A} , is established.
4. The actions called for by a are executed.
5. Processing of the transaction is now complete and control "exits" from the table.

Example: Consider decision table $D_2 = (C_2, A_2, R_2)$

Condition set $C_2 =$

```
C2(1): COLOR, [red, blue]
C2(2): LENGTH, [20 or less, over 20]
```

Action set $A_2 =$

```
A2(1): set PRICE to, [R, S + 5, S * P]
A2(2): Increase LENGTH by, [1, 4, 10, 0]
```

Relation $R_2 =$

```
[(red, 20 or less) --: (R,1),
 (red, over 20) --: (S + 5, 10),
 (blue, 20 or less) --: (S + 5, 4)
 (blue, over 20) --: (S * P, 0)]
```

D_2 has five inputs: COLOR, LENGTH, R, S, P
and two outputs: PRICE, LENGTH

One possible input to D_2 is the transaction $t = \{\text{COLOR} = \text{red}, \text{LENGTH} = 28, R = 20, S = 35, P = 0.9\}$

Transaction t is processed as follows:

Step 1. t is converted to alternate red for condition 1 and over 20 for condition 2.

- Step 2. point (red, over 20) in $\overline{C_2}$ is selected.
 Step 3. the image in $\overline{A_2}$ of (red, over 20) is (S + 5, 10)

Step 4. PRICE is set to $S + 5 = 35 + 5 = 40$ and LENGTH is set to
 $LENGTH + 10 = 28 + 10 = 38$

Step 5. Processing of t terminates

3.3.3 Rule satisfaction, supported transactions

Transaction t satisfies rule r if its values meet all the condition alternatives specified by rule r, (i.e., the t's values "select" the point c in C which corresponds to r). We can represent this as a predicate, SATISFY (t,r) which is true, if and only if, t satisfies rule r. Transaction t is said to be supported by a table if the table contains a rule r such that SATISFY (t,r) is true.

Example

In 2.2 we found transaction $t = \{\text{color} = \text{red}, \text{length} = 28, R = 20, S = 35, P = 0.9\}$ satisfies rule 2 of table D2; thus SATISFY ($t, 2$) is true, while SATISFY ($t, 4$) is false.

3.3.4 Programming language representation of decision tables

The mode of transaction processing just described implies a mapping of a decision table into a series of IF-THEN statements linked together via an ELSE connector. This represents a 'first cut' at defining the way in which a decision table is executed and is completely adequate as long as the conditions have no side effects and can be tested as many times as desired for any individual transaction. Later we shall remove these restrictions in a more general interpretation of the execution of a decision table; meanwhile the following representation will be useful in manipulating tables and in proving some theorems.

For example, table D2 is represented by:

```

IF (COLOR is red AND LENGTH is 20 or less)
    THEN (set PRICE to R, increase LENGTH by 1)

ELSE IF (COLOR is red AND LENGTH is over 20)
    THEN (set PRICE to S + 5, increase LENGTH by 10)

ELSE IF (COLOR is blue AND LENGTH is 20 or less)
    THEN (set PRICE to S + 5, increase LENGTH by 4)

ELSE IF (COLOR is blue AND LENGTH is over 20)
    THEN (set PRICE to S * P, increase length by 0)
  
```

An alternative could be to dispense with the ELSE's and express a table as a series of simple IF... THEN's with no connectives, as follows:

```

IF (COLOR is red AND LENGTH is 20 or less)
    THEN (set PRICE to R, increase LENGTH by 1)

IF (COLOR is red AND LENGTH is over 20)
    THEN (set PRICE to S + 5, increase LENGTH by 10)

IF (COLOR is blue AND LENGTH is 20 or less)
    THEN (set price to S + 5, increase LENGTH by 4)

IF (COLOR is blue AND LENGTH is over 20)
    THEN (set PRICE to S*P, increase LENGTH by 0)

```

This second approach is inappropriate; its use may result in a transaction satisfying more than one rule and causing execution of the actions of the several different rules. This can happen even if the original rules are mutually exclusive. Consider our example for a transaction having COLOR = blue and LENGTH = 17. Rule 3 will be satisfied by this transaction and length will be increased by 4 to 21. Without the ELSE connective, the transaction is next processed against rule 4 which it now satisfies because color is blue and length is over 20.

3.3.5 Semantic functions

To be a function in a structural sense, we required that each point c in C contains a unique image $a = R(c)$ where a is a point in A . To be a function in the semantic sense each possible transaction (i.e., set of inputs) must be mapped into a unique set of outputs. One way of satisfying these criteria is to require that R be a structural function and, in addition, require that every possible transaction be mapped into a unique point of C .

The limitation that each t be mapped into a unique c in \bar{C} places the requirement of completeness and consistency on the conditions comprising C . To be complete, condition $C(i)$ must have an alternative $CA(i,j)$ for each possible value of its subject $CS(i)$. To be consistent there must be only one such alternative for any value of its subject.

Example

- (1) Consider condition $C1(2)$ in example D1.

```

C1(2): AGE, [under 20, 20-60, over 60]
C1(2) is complete and consistent given that:
AGE means age at last birthday.
Under 20 means <20
20-60 means ≥20 and ≤60
and over 60 means >60

```

(2) Consider C1(2): AGE, [20-30, 31-40, 41-65, over 60]

C1(2) is not complete because it fails to provide an alternate for AGE less than 20. It is not consistent because it provides two alternatives for AGE 61 to 65.

Completeness and consistency are relative to the universe, (i.e., range of possible values) to which a condition applies. In the last example, the conclusion that C1(2) is incomplete and inconsistent involved the tacit assumption that C1(2) applies to the general population. But, if we restrict C1(2) to a universe of persons 20 to 59 years of age, then C1(2) is complete and consistent.

Suppose CS(i) has n logically possible alternatives CA(i,j), but only m (where m < n) are explicitly stated. The entries CA(i,j) j = m + 1, ..., n can be combined into an "else" condition alternative depicted by "OTHER", to guarantee completeness.

EXAMPLE

Consider C1(2)': AGE, [20-30, 31-40, 41-65, .OTHER.]. C1(2)' is now complete because the "else" condition alternative OTHER provides an alternative for AGE <20 and AGE >65.

3.3.6 Weak constraints for a semantic function

We have just stated two requirements for a decision table to be a function in a semantic sense: (1) the table must be a function structurally, and (2) each condition must be complete and consistent. We can relax these requirements somewhat. One can achieve this by working not with the whole condition space C but just a part of it--relation R is restricted to subset C* of C. We require only that C* contain a unique point for every possible transaction t. In this case an individual condition's alternatives need not be mutually exclusive; instead only the combinations of alternatives comprising C* need to be mutually exclusive.

Example decision table D3 = (C3, A3, R3)

Condition set C3
 C3(1): XBAR, [GE- ∞ , GE 0, GE 100]
 C3(2): XBAR, [LS 0, LS 100, LE ∞]

where

GE means greater than or equal to
 LS means less than
 LE means less than or equal to
 ∞ means infinity
 EQ (or =) means equal

Action set A3 =

A3(1) = CODE :=, [1,2]

Relations R3 =

[(GE ∞ , LS 0) --: (1),
 (GE 0, LS 100) --: (2),
 (GE 100, LE ∞) --: (1)]

Structurally relation R3 is incomplete and is not a function. However, semantically relation R3 is a function from C3 to A3; this holds even though the domain of R3 is not C3. Some of the points in C3 for which R3 is not defined are impossible; for example (GE 100, LS 0) is logically impossible because it requires that XBAR be both less than 0 and greater than or equal to 100. Other points missing from the domain of R3 are "covered" by points for which R3 is defined, e.g., the point (GE ∞ , LS 100) has no image under R because it is "covered" by the points GE ∞ , LS 0) and (GE 0, LS 100).

3.3.7 Semantic restrictions

Let D be a decision table and T be the set of all possible transactions input to D. Table D* is said to be a restriction of D to U, if U is a subset of T, $U \subseteq T$; and $D^*(u) = D(u)$ for all $u \in U$.

Example: Consider D4 = (C4, A4, R4)

Condition set C4

C4(1): = XBAR, [GE 50, GE 100]
 C4(2): = XBAR, [LS 100, LE ∞]

where GE, LS, LE are as previously defined

Action set A4

A4(1): CODE:=,[1,2]

relation R4

[(GE 50, LS 100)--: (2),
 (GE 100, LE ∞)--: (1)]

Now D4 is a restriction of D3 to values of 50 or over for XBAR.

3.3.8 Theorem 2

Decision table D represents a semantic function if and only if every semantic restriction of D is a semantic function.

3.3.9 T-SETS, rule overlap

At times it is useful to consider semantic restrictions related to a table's structure.

We can associate with each rule r a T-SET of transactions which satisfy rule r ; viz:

$t \in T\text{-SET}(r)$ if and only if $\text{SATISFY } (t, r)$ is true

Example

In D3 the T-SET for rule 3 comprises all transactions having an XBAR of 100 or more.

The overlap of rules r and s is defined as the set of transactions which satisfies both rule r and rule s and can be expressed as $T\text{-SET}(r) \cap T\text{-SET}(s)$ where \cap means intersection. If r and s are mutually exclusive $T\text{-SET}(r)$ and $T\text{-SET}(s)$ are disjoint and the intersection is empty.

Sometimes it will be useful to generalize the concept of T-SETS and apply them to a partial rule which involves only a subset of the table's conditions.

Example

In D3 the T-SET for the first condition of rule 2 comprises all transactions with XBAR equal to or greater than zero.

3.3.10 Theorem 3

Decision table D is a semantic function on transaction set T if and only if the T-SETS associated with the rules of D partition T into a set of blocks such that each $t \in T$ belongs to one and only one block.

3.3.11 Semantic decomposition

Let D be a decision table with relation R and transaction set T . Table D' is said to be a semantic decomposition of D , if U is a subset of T obtained by assigning values independently to one or more selected variables of $t \in T$ and $D'(u) = D(u)$ for all $u \in U$. Later we shall see how semantic decompositions are useful in identifying and selecting meaningful subtables.

Example

- 1) Let decision table $D5 = (C5, A5, R5)$ where:

```
Condition Set C5 =
C5(1): A =, [1, 2, 3]
C5(2): B =, [10, 30]
C5(3): C =, [LS 50, GE 50]
```

```
Action Set A5 =
A5(1): SEX X = [1, 2, 3, 4, 5, 6]
```

```
R5 = [ (1,10, LS 50) --: (1),      (1,10,GE 50) --: (2),
       (1,30, LS 50) --: (2),      (1,30,GE 50) --: (3),
       (2,10, LS 50) --: (3),      (2,10,GE 50) --: (5),
       (2,30, LS 50) --: (4),      (2,30,GE 50) --: (4),
       (3,10, LS 50) --: (1),      (3,10,GE 50) --: (4),
       (3,30, LS 50) --: (6),      (3,30,GE 50) --: (6) ]
```

2) Consider D5' defined on C5 and A5 and having relation

```
R5' = [ (1,10,GE 70) --: (2),      (1,30,GE 70) --: (3)
        (3,10,GE 70) --: (4),      (3,30,GE 70) --: (6) ]
```

D5' is a semantic decomposition of D5 selected by the values 1 and 3 for A and GE 70 for C.

3) Consider D5" defined on C5 and A5 and having

```
R5" = [ (1,10,GE 50) --: (2),      (2,30,GE 50) --:(5) ]
```

D5" is not a semantic decomposition of D5 because the values for A and B are not independent -- when A is 1, B must be 10 and when A is 2, B must be 30.

A decomposition is considered proper, if all of the selection variables remain fixed in the reduced table -- none of them need to be tested in selecting a rule in the restricted table. (For the sake of simplicity, we will assume that each condition tests a distinct set of variables including necessary "dummy" duplication of variables tested in other conditions.)

EXAMPLE

Decomposition D5' is not a proper decomposition of D5 because selection variable A is not fixed in R5'. The following relation yields a proper decomposition of D5 using A and B as selection variables.

```
R5" = [ (2,30, LS 50) --: (3),      (2,30, GE 50) --: (5) ]
```

3.4 ENTRY AND TABLE TYPES

3.4.1 Generalized alternatives

There are a few alternatives which are general in nature, and have a meaning which is independent of the subject being considered.

For conditions we have:

- Y Yes, the subject must be true
- N No, the subject must be false
- Irrelevant, indifferent, ignore, the subject can have any alternative.
- # Undefined (condition cannot occur); do not test.

For actions we have:

- X Execute, do the action specified by the subject
- Ignore, do not execute, do not do the action specified by the subject, do not apply.

We shall consider the entry '-' as being appropriate for any condition or action. In the case of a condition, '-' implies that any of the condition's alternatives may be true. For actions, '-' means apply none of the subject's alternatives.

Example, consider D6

Previously we wrote condition C1(1) = SEX is, [male, female] now we can write it as C6(1) = SEX is male, [Y,N]

And action A1(1) which was COUNT-1, [inc-by-1,do-not-change] becomes A6(1) = incr-COUNT-1 by 1, [X]

And action A1(3) which was TALLY-REC, [do, do-not-do] can be expressed as A6(1) = TALLY-REC, [X]

Retaining C1(2) and A1(2) as previously expressed, i.e.,

C6(2) = C1(2) = AGE, [under 20, 20-60, over 60]
A6(2) = A1(2) = SET Z to, [1,3,0]

Given these redefinitions relation R1 is rewritten as R6 as follows:

$(Y, \text{under } 20) \rightarrow (X, 1, X)$	$(Y, 20-60) \rightarrow (-, 1, X)$
$(Y, \text{over } 60) \rightarrow (-, 0, -)$	$(N, \text{under } 20) \rightarrow (X, 1, X)$
$(N, 20-60) \rightarrow (X, 1, X)$	$(N, \text{over } 60) \rightarrow (-, 3, X)$

3.4.2 Entry types

These generalized alternatives form the set of "limited" entries which have played such a prominent role in the development of decision table methodology. One can only speculate about the (unfortunate ?) preoccupation with this type of entry; one possible reason for this preoccupation is the similarity to the yes/no logic employed in computer logic, programming and flow-charts.

By way of contrast, the previously used, non-generalized alternatives are considered "extended" entries. They can be thought of as being extended because they "extend" the condition's or action's subject and also because they can take on any of an extended range of values.

3.4.3 Table types

Traditionally, a table in which all entries are limited (Y,N,X, or -) is called a "limited entry decision table" (LEDT). An "extended entry decision table" (EEDT) is a table having only extended entries (including "--" and "#") and a MEDT (mixed entry decision table) has both limited and extended entries. Decision tables can also be categorized in terms of their stubs; certain applications make "condition-only" and "action-only" tables desirable.

3.4.4 Limited entries - a programming language representation

In 3.3.4 we examined a programming language representation of decision tables; here we extend the representation to include limited entries.

For condition C:

"Y" is represented by C itself,

"N" is represented by negation of C, i.e., by NOT (C),

"-" is represented by the logical constant TRUE (or by omitting the condition)--conditions are represented by an AND sequence; the introduction of "AND TRUE" as a component of this sequence, says "always consider this component as sustaining the truth of the sequence."

For actions:

"X" is represented by the action itself

"_" is represented by SKIP.

Example decision Table D7

with condition set C7 =

C7(1): A = 0, [Y,N]

C7(2): B = , [1,2,3]

With action set A7 =

A7(1): S:=, [1,12,20]

A7(2): T:= 0, [X]

and with relation set R7 =

```
[(Y,1) --: (1,-), (Y,2) --: (1,-), (Y,3) --: (1,-)
 (N,1) --: (12,X), (N,2) --: (20,X) (N,3) --: (20,-)]
```

is represented by:

```
IF (A=0 AND B=1) THEN (S:=1, SKIP)
ELSE IF (A=0 AND B=2) THEN (S:= 1, SKIP)
ELSE IF (A=0 AND B=3) THEN (S:= 1,SKIP)
ELSE IF (NOT (A=0) AND B = 1) THEN (S:= 12, T:=0)
ELSE IF (NOT (A=0) AND B = 2) THEN (S:= 20, T:=0)
ELSE IF (NOT (A=0) AND B = 3) THEN (S: = 20, SKIP)
```

3.4.5 Implied limited entries

There are cases in which there is a logical relation among a table's conditions so that one entry in a rule is derivable from other entries in the rule. Three limited entries have been suggested to cover such situations: Y! meaning true by implication, N! meaning false by implication, and # meaning undefined and cannot be evaluated. These entries for instance come into prominence when one converts extended into limited entries as we shall do in 3.8 following.

Example

Consider decision table D8 with:

Condition Set D8 =

```
C8(1): marital status = , [husband, wife, single]
C8(2): Sex = male, [Y,N]
```

Action set A8 =

```
A8(1): set code = [1,2,3,4]
```

Rule set R8

```
[(husband,Y!) --: (1),
 (wife, N!) --: (2),
 (single, Y) --: (3),
 (single, N) --: (4)]
```

It is not clear as to the extent to which implied limited entries ($Y!, N!, \#$) should be used; they play a very significant role in the conversion of decision tables (see Chapter 4, Conversion Algorithms). This is particularly true where one is using a computer to process decision tables and, as part of the processing, has converted an extended entry table to a limited entry table. But in an overwhelming majority of cases it may be better to make the table easier to understand by employing a (-) and assuming that the table's human (or non-human) reader can draw the inferences among a rule's entries.

Example

An alternate version of rule set R8 is R8'

```
((husband, -) --: (1)
(wife, -)    --: (2)
(single, Y)  --: (3)
(single, N)  --: (4))
```

This version is probably more readable than the original R8 since most people would realize that if one is a husband his sex need not be tested (i.e., is not relevant) because it is always male. Hopefully, advances in computer science may allow computers to store information from which it can draw such inferences (discussed further in 3.8).

3.5 SYNTAX II: DECISION TABLES AS LINGUISTIC STRUCTURES

3.5.1 Tables and quadrants

We shall now return to a topic which we touched on briefly before, namely, the representation of decision tables in a tabular format.

Syntactically, a decision table consists of four parts or quadrants:

A condition stub, CS, which holds condition subjects.

An action stub, AS, which holds action subjects.

A set of condition entries, CE, which holds condition alternatives.

A set of actions entries, AE, which holds action alternatives.

These four parts divide a decision table into four quadrants as in the following figure:

```
.....:.....:.....:.....:  
: CS      :       CE :  
.....:.....:.....:  
: AS      :       AE :  
.....:.....:.....:
```

3.5.2 Rules

In our examination of decision tables, we have defined a relation as a series of mappings, each of which is a rule that associates a point in condition-space, C, with its image in action-space, A. In a tabular format each rule is represented by a series of condition entries (i.e., alternatives) followed by a series of action entries placed in the same table column (or row).

3.5.3 Stubs and entries

It is convenient to treat CS, CE, AS and AE as arrays:

CS is a one-dimensional array of stubs with CS(ic) holding the stub (subject) for condition ic, ic = 1,2...nc = the number of conditions.

CE is two-dimensional array of entries with CE(r,ic) containing condition ic's alternative for rule r, r = 1...nr = number of rules.

AS is a one-dimensional array of stubs with AS (ia) holding the stub (subject) for action ia = 1, ..., na, na = number of actions.

AE is a two-dimensional array of entries with AE (r,ia) holding action ia's alternative for rule r.

Sometimes we shall generalize these concepts and talk about:

C, a clause -- a condition or action

S, a single dimensional array with S(i) containing the stub (i.e., subject) for clause i = 1,2...nc, nc + 1, nc + 2,...nc + na

E, a two-dimensional array with E (r,i) containing the entry clause i of rule r.

Example D9

```
.....:  
: Customer is : Residential Residential Commercial Industrial :  
: :  
: Season is :  
: Summer : Y N - - :  
.....:  
: Set base to : 3.00 3.00 20.00 100.00 :  
: :  
: Set rate to : .005 .003 .002 .001 :  
.....:
```

Is a tabular representation of D9 with:

Condition Set C9 =

C9(1): customer is, [residential, commercial, industrial]
C9(2): season is summer, [Y, N]

Action Set A9 =

A9(1): set base to, [3.00, 20.00, 100.00]
A9(2): set rate to [.005, .003, .002, .001]

and rules components R9 =

```
((residential, Y) --: (3.00, .005),  
 (residential, N) --: (3.00, .003),  
 (commercial, -) --: (20.00, .002),  
 (industrial, -) --: (100.00, .001))
```

Selected elements of this table include:

```
S(2) = CS(2) = "Season is Summer"  
E(3,2) = CE(3,2) = "_"  
S(3) = AS(1) = "set base to"  
S(4,4) = AE(4,2) = ".001"
```

3.5.4 Relevancy

Clause c is relevant for rule r, if $E(r,c)$ contains a value other than '-' (i.e., irrelevant) or '#' (i.e., undefined). Sometimes it is useful to represent relevance by a mask. One way is to have a mask for each clause, the mask carrying a '1' in positions which correspond to rules for which the clause is relevant.

Example

Decision Table							Relevancy Mask				
:	:	:	:	:	:	:	:	:	:	:	:
:	C1 A = :	1	1	1	2	:	:	1	1	1	1
:	C2	:	Y	N	N	-	:	1	1	1	0
:	C3	:	-	Y	N	-	:	0	1	1	0
:	A1	:	X	X	-	-	:	1	1	0	0
:	A2	:	-	X	-	-	:	0	1	0	0
:	A3	:	X	-	X	-	:	1	0	1	0

3.5.5 Subtables and proper subtables

A selection of rows and columns from decision table D comprises a subtable of D. We shall call a subtable S of D a proper subtable of D, if S corresponds to a semantic decomposition of D (as defined in Section 3.3.11; a semantic decomposition of D is isolated by fixing independently the values of one or more selector inputs).

Example

		R1	R2	R3	R4	R5	R6	R7	
:		:							:
:	C1	A =	:	1	1	1	1	2	2
:			:						:
:	C2	B =	:	1	2	2	2	-	-
:			:						:
:	C3		:	-	Y	Y	N	Y	N
:			:						:
:	C4		:	-	Y	N	-	-	Y
:			:						:
:	A1	X:=	:	1	1	2	6	5	3
:			:						:

One possible subtable is:

		R2	R3	R4	
:		:			:
:	C3	:	Y	Y	N
:			:		:
:	C4	:	Y	N	-
:			:		:
:	A1	X:=	:	1	2
:			:		:

This is a proper subtable resulting from the semantic decomposition obtained by restricting variable A to the value 1 and variable B to the value 2.

Another subtable is:

		R3	R4	R5	
:		:			:
:	C2	B =	:	2	2
:			:	-	:
:	C4		:	N	-
:			:	-	:
:	A1	X:=	:	2	6
:			:	5	:

This subtable does not correspond to a semantic decomposition and is not a proper subtable.

3.5.6 Theorem 4

Also in Section 3.3.11 we described a semantic decomposition as complete, if all of the selection variables remain fixed in the resulting decomposed table. Bearing this in mind we have the following theorem:

Given D, a semantically complete and consistent decision table and S, a proper subtable of D, then S is semantically complete and consistent if S corresponds to a complete semantic decomposition of D.

3.5.7 Inpaths, outpaths, and outspans

Let $E(r,c)$ be the entry for clause c of rule r, then the inpath to $E(r,c)$ -- $IP(r,c)$ -- is the set of entries which precedes clause c in rule r. The outpath from $E(r,c)$ -- $OP(r,c)$ -- consists of $E(r,c)$ itself, and those entries which follow it in rule r.

Note that $OP(r,c)$ includes $E(r,c)$ and can be looked upon as the outpath "produced" by or consequent to inpath $IP(r,c)$; rule r consists of $IP(r,c)$ followed by $OP(r,c)$. We shall have many more occasions to relate an outpath to an inpath than to an entry, and shall generally associate outpaths with inpaths.

Example

	R1	R2	R3	R4	R5	
:						:
:	C1 A = :	1	1	1	2	3 :
:						:
:	C2 :	Y	N	N	-	- :
:						:
:	C3 :	-	Y	N	-	- :
:						:
:	A1 :	X	X	-	-	- :
:						:
:	A2 :	-	-	X	X	- :
:						:

$IP(2,4) =$ the inpath to entry 4 of rule 2

= (1,N,Y)

$OP(2,4) =$ the outpath from entry 4 of rule 2

= (X,-)

If clause c is a condition, $IP(r,c)$ may not have a single, unique outpath. There may be several outpath reflecting the multiple alternatives for c and for any conditions which may follow c in the table. The subtable composed of all outpaths which are the consequence of a given inpath is called the inpath's outspan. Outspans are defined in terms of their defining inpaths: $OS(e_1, e_2, \dots)$ is the outpath consequent to the inpath having e_1 as its first entry, e_2 as its second entry, etc.

Example

In the last table the inpath to $E(3,3)$ is $(1,N)$. There are two outpaths for this inpath -- one, or rule 2, is $(Y,X-)$; the other, rule 3, is $(N, -, X)$. Thus the OS $(1,N)$ is:

	R2	R3	
.....		
: C3 :	Y	N	:
.....
:	:		:
: A1 :	X	-	:
:	:		:
: A2 :	-	X	:
.....

Formally, an outspan is defined in terms of an inpath's T-SET. If ip is an inpath, the T-SET associated with ip is $T\text{-SET}(ip)$ and consists of all transactions t which satisfy the condition alternatives implied by ip . The outspan for ip , $OS(ip)$, consists of those rules which are satisfied by some $t \in T\text{-SET}(ip)$ with the clauses belonging to ip suppressed.

Example

	R1	R2	R3	R4	R5	R6	R7	R8
:
: C1	:	Y	Y	Y	N	N	N	N
:	:							
: C2 W	:	-	-	-	<1	<1	<0	>0 and <1
:	:							
: C3	:	Y	Y	Y	Y	Y	N	N
:	:							
: C4	:	Y	Y	N	Y	N	-	-
:	:							
: C5	:	Y	N	-	-	-	-	-
:
: A1 A:=	:	1	2	3	4	5	6	7
:	:							
: A1 A:=	:	8						

$IP(1,4)$ consists of $(Y, -, Y)$; its T-SET comprises all transactions for which C_1 and C_3 are true. The rules satisfied by the transactions are R_1 , R_2 , and R_3 . Suppressing clauses C_1 , C_2 , and C_3 we have $OS(Y, -, Y) =$

	R1	R2	R3	
:	:			:
:	C_4	Y	Y	N
:				:
:	C_5	Y	N	-
:				:
:	$A_1 A_2 :=$	1	2	3
:				:

$IP(4,3)$ consists of $(N, <1)$; some transactions which satisfy it may also satisfy R_5 , R_6 , and R_7 ; and the outspan for $(N, <1)$ is:

	R4	R5	R6	R7	
:	:				:
:	C_3	Y	Y	N	N
:					:
:	C_4	Y	N	-	-
:					:
:	C_5	-	-	-	-
:					:
:	$A_1 A_2 :=$	4	5	6	7
:					:

An inpath, ip , is said to be regular if every rule in its outspan has an inpath equal to ip . In the preceding example $IP(1,4) = (Y, -, Y)$ is regular but $IP(4,3) = (N, <1)$ is not because the portion of $OS(N,1)$ corresponding to R_6 in the original table is reached via $(N, <0) \neq (N, <1)$.

3.5.8 Theorem 5

Given D , a semantically complete and consistent decision table, and ip , an inpath to a clause of D . Then inpath ip is regular, if and only if $OS(ip)$ corresponds to a complete decomposition of D .

3.5.9 Canonical form of decision tables

Every decision table can be written in a canonical form. This process follows immediately from the Cartesian product

$$C = C(1) \times C(2) \times \dots \times C(n)$$

3.6 SYSTEMS OF DECISION TABLES

3.6.1 Linkage via invocation

One decision table invokes another table by DOing it, CALLing it, PERFORMing it, using it as a FUNCTION reference, etc. Generally, communication between the two tables is by means of shared, common "GLOBAL" data structures and/or a formal argument list. We shall limit our attention to the first of these methods although extension of our presentation to cover both cases is not particularly difficult.

Example

Table 1

:	:	:
:	C11	: Y N N :
:	C12	: - Y N :
:	A11	: X X - :
:	A12	: - X X :
:	A13 Do Table 2	: X X - :
:	A14	: - X X :
:		:

Table 2

:	:	:
:	C21	: Y N :
:	A21	: - X :
:	A22	: X - :
:	A23	: X - :
:		:

Table 2 is invoked by Action A13 of Table 1.

3.6.2 Expansion

One can eliminate the linkage between tables by replacing each reference to the invoked table by the invoked table itself. We shall call such a process "expansion".

Example

In the previous example Table 1 contains two invocations of Table 2. Expansion of these invocations yields the following table:

:	:							:
:	C11	:	Y	Y	N	N	N	:
:	:							:
:	C12	:	-	-	Y	Y	N	:
:	:							:
:	A11	:	X	X	X	X	-	:
:	:							:
:	A12	:	-	-	X	X	X	:
:	:							:
:	C21	:	Y	N	Y	N	-	:
.....
:	:							:
:	A21	:	-	X	-	X	-	:
:	:							:
:	A22	:	X	-	X	-	-	:
:	:							:
:	A23	:	X	-	X	-	-	:
:	:							:
:	A14	:	-	-	X	X	X	:
.....

3.6.3 Comparison with traditional decision table theory

Expansion of an invoked table may result in the interleaving of actions between conditions as is the case of A11 and A12 in the preceding example. Some decision table purists reject such an approach and require that all conditions be tested before any actions are executed. More recently some investigators have suggested the intermixing of conditions and actions through the use of 'supporting' or 'bound' actions.

One suggested alternative to the use of intermixed conditions and actions is the use of a 'repeat table command' in conjunction with a control mechanism for restricting the portion of the table to be executed.

EXAMPLE: Compare with the examples in 3.6.2; (note: CONTROL is assumed to be automatically zero upon entry to the table).

:		:							:	
:	CONTROL =	:	0	0	0	1	1	2	2	:
:	C11	:	Y	N	N	-	-	-	-	:
:	C12	:	-	Y	N	-	-	-	-	:
:	C21	:	-	-	-	Y	N	Y	N	:
:		:								:
:	A11	:	X	X	-	-	-	-	-	:
:	A12	:	-	X	X	-	-	-	-	:
:	A21	:	-	-	-	-	X	-	X	:
:	A22	:	-	-	-	X	-	X	-	:
:	A23	:	-	-	-	X	-	X	-	:
:	A14	:	-	-	X	-	-	X	X	:
:	CONTROL : =	:	1	2	-	-	-	-	-	:
:	REPEAT TABLE	:	X	X	-	-	-	-	-	:
:		:								:

The present paper's approach is to allow conditions and actions to be intermixed and to view such intermixing as the result of an arbitrary choice of expanding a subordinate table 'inline', rather than invoking it as a separate table or reaching it via a repeat-table scheme. Regardless of the structure selected there is intermixing in D7 because actions A11 and A12 are done before C21 is tested.

3.6.4 Tables invoked by conditions

So far we have considered the case of tables invoked from the actions of another. Sometimes it is convenient to have a condition in one table invoke another table and to have the invoked table 'return' a value which is tested by the invoking condition.

Example

Table 3

```
.....:  
:  
: C31 : Y Y Y Y N N :  
:  
: C32 Table 4 returns : 1 2 2 3 - - :  
:  
: C33 : - Y N - Y N :  
.....:  
: A31 X; = : A B C C - D :  
:  
: A32 : X - - - X X :  
.....:
```

Table 4

```
.....:  
:  
: C41 : Y Y N N :  
:  
: C42 : Y N Y N :  
.....:  
: A41 : X - X X :  
:  
: A42 : X X - X :  
:  
: A43 Return: = : 1 1 2 3 :  
.....:
```

The expansion of Table 4 as condition C32 yields the following table (note the "absorption" of A43 and C32).

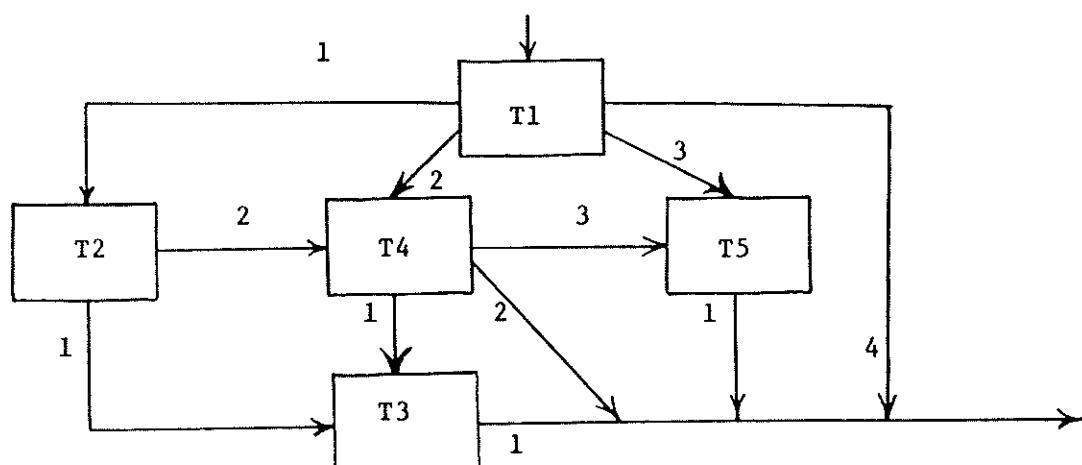
Composite of 3 and 4

.....
:	C31	:	Y	Y	Y	Y	Y	N	N
:	C41	:	Y	Y	N	N	N	-	-
:	C42	:	Y	N	Y	Y	N	-	-
:	A41	:	X	-	X	X	X	-	-
:	A42	:	X	X	-	-	X	-	-
:	C33	:	-	-	Y	N	-	Y	N
:	A31 X: =	:	A	A	B	C	C	-	D
:	A32	:	X	X	-	-	-	X	X
.....

3.6.5 Sequences of tables

Another way tables can be linked is by transfer of control either by "falling-thru" to another table or by doing a 'GO TO' to another table. The advent of structured programming and the controversy over GO-TO-LESS programming have diminished the attractiveness of this form of linkage. One can transform a network of tables linked by transfer of control to a structure consisting of a master table which contains conditions that invoke the other tables, and use the values returned to select the next table to be invoked.

Example: (In the flowchart the labels on the links between boxes correspond to values returned in the tables invoked from the master table):



This flowchart can be represented by:

```
.....:.....:.....:.....:.....:.....:.....:.....:.....:.....:  
: : T1 returns : 1 1 1 1 2 2 2 3 4 :  
: : : : : : : : : : :  
: : T2 returns : 1 2 2 2 - - - - - :  
: : : : : : : : : : :  
: : T4 returns : - 1 2 3 1 2 3 - - - :  
: : : : : : : : : : :  
: : T3 returns : 1 1 - - 1 - - - - - :  
: : : : : : : : : : :  
: : T5 returns : - - - 1 - - 1 1 - - :  
:.....:.....:.....:.....:.....:.....:  
: Exit : X X X X X X X X X X :  
:.....:.....:.....:.....:.....:.....:
```

3.6.6 Factoring decision tables, linkage via invocation

We shall call the opposite of expansion, "factoring". One factors a decision table by making selected segments separate tables which are invoked from the original table. The segments selected for representation as invoked tables must be proper subtables of the original table (see 3.5.5, preceding).

Example

```
.....:.....:.....:.....:  
: : : : : : :  
: C51 : Y Y Y N :  
: : : : : : :  
: C52 : Y Y N - :  
: : : : : : :  
: C53 : Y N - - :  
:.....:.....:.....:  
: A51 : X - - - :  
: : : : : : :  
: A52 : - X - - :  
: : : : : : :  
: A53 : X X X - :  
:.....:.....:.....:
```

can be factored into

Table 1

```
.....:.....:.....:.....:  
: : : : : : :  
: C51 : Y Y N :  
: : : : : : :  
: C52 : Y N - :  
:.....:.....:.....:  
: A50 Do Table 2 : X - - - :  
: : : : : : :  
: A53 : X X - :  
:.....:.....:.....:
```

Table 2

```
.....:.....:.....:  
: : : : : : :  
: C53 : Y N :  
:.....:.....:.....:  
: A51 : X - :  
: : : : : : :  
: A52 : - X :  
:.....:.....:.....:
```

3.6.7 Factoring decision tables, linkage via transfer of control

One can also factor tables into smaller tables which are linked by transfer of control.

Example

```
.....:::  
: C1 : Y Y Y N ::  
: C2 : Y Y N - ::  
: C3 : Y N - - ::  
.....:::  
: A1 : X X - - ::  
: A2 : - X X - ::  
: A3 : - - - X ::  
: A4 Exit: X X X X ::  
.....:::
```

Can be factored into

Table 1

```
.....:::  
: C1 : Y N ::  
.....:::  
: A3 : - X ::  
: Go to Table : 2 - ::  
: A4 Exit : - X ::  
.....:::
```

Table 2

```
.....:::  
: C2 : Y Y N ::  
: C3 : Y N - ::  
.....:::  
: A1 : X X - ::  
: A2 : - X X ::  
: A4 Exit: X X X ::  
.....:::
```

3.6.8 Total Factoring

As we have seen, factoring breaks one decision table into several smaller tables. One can in turn factor these tables until one arrives at a series of tables each containing only a single condition or action plus transfers to other tables. At this stage we have a structure which is equivalent to the conventional flow chart.

Example

Table 1

```
.....:.....:.....:  
: C1 : Y N N :  
: : : :  
: C2 : - Y N :  
.....:.....:.....:  
: A1 : X - - :  
: : :  
: A2 : - X - :  
: : :  
: A3 Exit: X X X :  
.....:.....:.....:
```

can be factored into:

Table 2

```
.....:.....:.....:  
: : : :  
: C1 : Y N :  
.....:.....:.....:  
: Go to table : 4 3 :  
.....:.....:.....:
```

Table 3

```
.....:.....:.....:  
: : : :  
: C2 : Y N :  
.....:.....:.....:  
: Go to table : 5 6 :  
.....:.....:.....:
```

Table 4

```
.....:.....:.....:  
: A1 : X :  
: : :  
: Go to table 6 : X :  
.....:.....:.....:
```

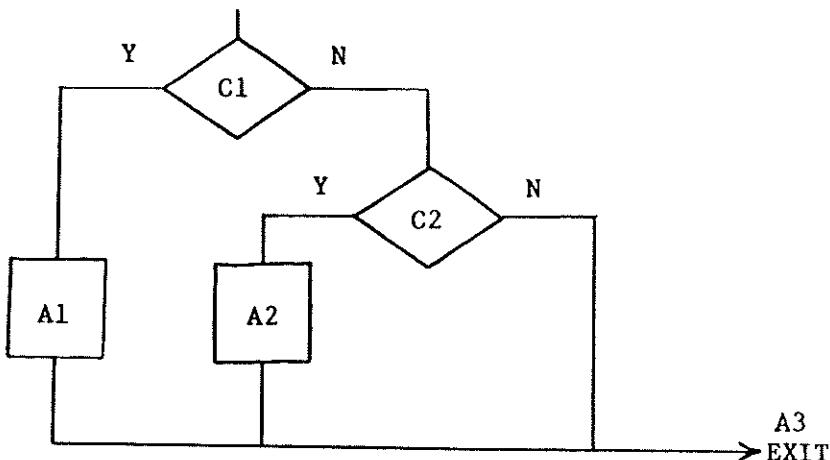
Table 5

```
.....:.....:.....:  
: : :  
: A2 : X :  
: : :  
: Go to table : X :  
.....:.....:.....:
```

Table 6

```
.....:.....:.....:  
: : :  
: A3 Exit : X :  
: : :  
.....:.....:.....:
```

The flowchart for which is:



3.6.9 Paths and rules

A path through a flowchart is a complete transversal of the chart from its entry point to its exit. Each rule in a decision table appears as one or more paths in the table's flowchart. The execution of one of a table's rules is equivalent to "following" a path through a flowchart.

Example

In the last example , the leftmost path in the flowchart corresponds to the first rule in Table 1.

3.6.10 Total factoring and table completeness and consistency

Total factoring provides an elegant and efficient way of validating a decision table for structural completeness and consistency. Further, it enables one to identify those transactions that are not supported by a table and those that satisfy two or more rules of a table.

Example

	R1	R2	R3	R4	
:	:				:
: C1	:	Y	Y	Y	N
:	:				:
: C2	:	Y	Y	N	Y
:	:				:
: C3	:	Y	-	-	Y
.....
: A1 Rule := :	1	2	3	4	:
.....

factors into:

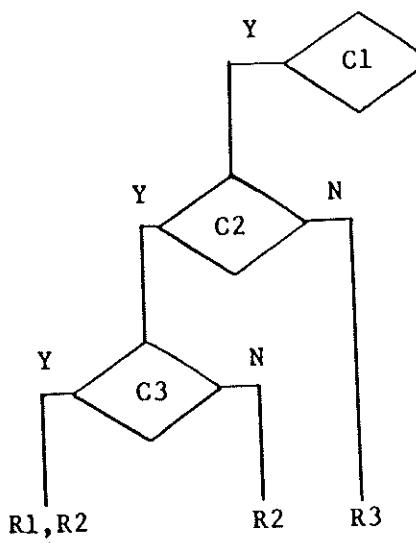


Table is inconsistent.
(Y,Y,Y) satisfies
both rule 1 and rule 2.

Table is incomplete.
(N,Y,N) is not
supported.

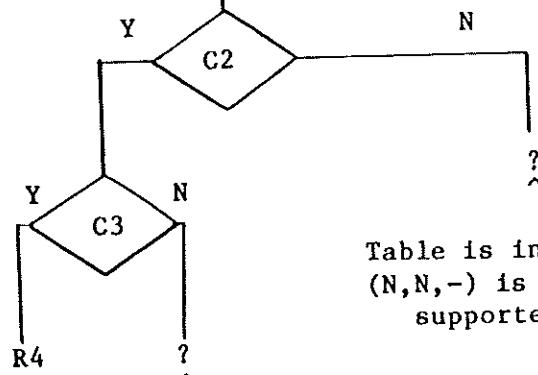


Table is incomplete.
(N,N,-) is not
supported.

3.7 THE EVALUATION CONSTRAINTS

3.7.1 Introduction

Thus far we have glossed over the details of executing a decision table -- details which become significant when we address the problem of mechanically interpreting a table manually by a clerk or by a computer program hand-coded or automatically generated from a table. We have said that a transaction is evaluated against a table's conditions, a rule is selected, and the selected rule's actions are executed. This description leaves many questions unanswered such as: In what order are the conditions tested? Must the actions be done in the order written? Can the testing of a condition be skipped, if it has no effect on the actions performed? Can the same condition be tested twice?

These questions intensify when, as in the preceding section, one permits actions to be interleaved with conditions and allows conditions to invoke other tables which may contain actions. This interleaving produces problems as illustrated by the following example:

Table 1

	R1	R2	R3	R4	R5	R6	R7
:	:						:
: C11	:	Y	Y	Y	Y	N	N
:	:						:
: C12	:	Y	Y	N	N	-	-
:	:						:
: A11 P:= P + 1	:	X	-	-	-	-	-
:	:						:
: C13 Table 2 returns 1	:	-	-	Y	N	Y	N
:	:						:
: C14 P>W	:	Y	N	Y	Y	-	-
:	:						:
: A12 BNUM:=	:	1	2	3	4	-	W
:	:						:

Table 2

	R1	R2	R3	R4
:	:			:
: C21	:	Y	Y	N
:	:			:
: C22	:	Y	N	Y
:	:			:
: A21 W:=	:	1	5	3
:	:			:
: A22 Return	:	1	2	3
:	:			:

- 1) In Table 1 rules R1 and R2 are not consistent. Rule R1 says (A11) increment P by 1 and compare it with W; while Rule R2 directs that P be left unchanged when compared with W. The only conditional difference between R1 and R2 is C14 which involves P. Should P be increased before or after testing condition C14?
- 2) There is a similar problem with rules R3 thru R5 of Table 1. By invoking Table 2 in C13, rules R3 and R4 modify W before comparing it with P in C14. R5 does not invoke Table 2 before C14. Condition C14 is the sole discrimination between R3-R4 and R5 -- should C14 be tested before or after C13 is tested?
- 3) Rules R6 and R7 in Table 1 imply the same action, namely the assigning of W to BNUM, and might be combined (Sec. 3.8.5) by using an indifference (-) for C13. But this cannot be done because C13 invokes Table 2, which alters W, the variable assigned to BNUM in action A7.

3.7.2 Output generating clauses (OGC)

An output generating clause (OGC) is an action or condition which produces an output directly or indirectly. In general it is useful to enlarge the concept of "output" to include the setting of various switches or interrupts such as would happen when trying to divide by zero.

3.7.3 Evaluation conventions for consistent tables -- tables in which each transaction t satisfies one and only one rule.

- 1) Each output-generating clause is executed once and only once for each rule in which it is relevant.
- 2) If an output-generating clause is not relevant for rule r, it must not be executed for rule r.
- 3) If C1 is an output-generating clause, and clause C2 follows C1 in a table and uses C1's outputs as inputs, then C1 must be executed before C2 for any rule in which both C1 and C2 are relevant.
- 4) If C2 is an output-generating clause and if C1 precedes C2 in a table and uses some of C2's outputs as inputs, then C1 must be executed before C2 for any rule in which both C1 and C2 are relevant.
- 5) If C1 precedes C2 and produces some of the same outputs as C2, then C1 must be executed before C2 for any rule in which both C1 and C2 are relevant.
- 6) The "Table X returns" type of condition is interpreted as the action, RETURN:= Table X (Table X is invoked as a function) followed by the condition, "RETURN = -".
- 7) Any nonoutput-generating clause can be executed zero, one, or as many times as desired.

3.7.4 Basis for consistency criteria for table with OGC intermixed with conditions

The development of consistency requirements for this circumstance is straightforward once one realizes that the interleaving of conditions and actions (and output-generating conditions) arises from the elaboration of invoked tables. First factor out the invoked table starting with the first conditions after one or more actions or after an occurrence of an output-generating condition; then the resulting tables are checked for consistency.

Table 3

		R1	R2	R3	R4	
:	:					:
:	C31	:	Y	Y	Y	N
:	:					:
:	C32	:	Y	Y	N	-
:	:					:
:	A31	:	X	X	-	X
:	:					:
:	A32	:	X	-	-	-
:	:					:
:	C33	:	Y	N	-	-
.....
:	A35	:	-	X	-	-
:	:					:
:	A36	:	-	-	X	X
.....

Unfactored table

Table 30

		R1	R2	R3	R4	
:	:					:
:	C31	:	Y	Y	Y	N
:	:					:
:	C32	:	Y	Y	N	-
.....
:	A31	:	X	X	-	X
:	:					:
:	A32	:	X	-	-	-
:	:					:
:	A33 Do Table 31	:	X	X	-	-
:	:					:
:	A36	:	-	-	X	X
.....

Table 31

		R1	R2
:	:	:	:
:	C33	:	Y
:	:		N
.....
:	A35	:	-
:	:		X
.....

Factored Table 30 displays inconsistency between rule R1 and rule R2.

3.7.5 Consistency with regard to OGC

An OGC, $k(i)$, is assumed to be a consequent of the clauses which precede $k(i)$ in the table (i.e., $k(i)$'s antecedent). The table is consistent, if $k(i)$ is always a consequent of the same antecedents.

A more formal definition of consistency follows:

Let $k(i)$ be an OGC in decision table D, and $E(r,i)$ be an entry which calls for the execution of $k(i)$.

Consider $P = IP(i,r) =$ the inpath to $E(r,i)$ and
 $TP = T-SET(P) =$ the transaction set for inpath P.

Then D is consistent with respect to $k(i)$, if $k(i)$ is executed with the same inputs for every transaction $t \in TP$.

The inpath to the first clause of a table is considered to have a null inpath and is satisfied by any transaction processed against the table.

Example

		R1	R2	R3	R4	
:	C1 Sex = male	:	Y	Y	Y	N
:		:				:
:	C2 Age > 20	:	Y	Y	N	-
:		:				:
:	A1	:	-	X	-	-
:		:				:
:	C3 Income 10,000	:	Y	N	-	-
:		:				:
:	A4 Score:=	:	1	2	3	4
:		:				:

This table is inconsistent with respect to OGC action A1. Entry E(2,3) specifies the execution of A1. The inpath to E(2,3) is (C1 = Y, C2 = Y) which has a T-SET comprised of all transactions for males over 20 years of age; this set of transactions also satisfies rule R1 which says that A1 is not to be executed. Hence the table is inconsistent.

3.7.6 Evaluation of inconsistent tables; precedence convention

In section 3.6.3 we enunciated six conventions for the evaluation of consistent decision tables. We now present a seventh convention to cover inconsistent tables; but first it is necessary to introduce the concepts of inpath satisfaction and inconsistency relative to transaction t.

Consider entry $E(r,c)$ in decision table D and let $p = IP(E(r,c))$ = the inpath to $E(r,c)$. Transaction t is said to satisfy (or select) p if it satisfies all of the conditions specified by p. Let $K(i)$ be a OGC; $K(i)$ is consistent relative to transaction t if 1) $K(i)$ is relevant for all inpaths to clause i satisfied by t or 2) $K(i)$ is irrelevant for all inpaths to clause i satisfied by t. $K(i)$ is inconsistent relative to t, if it is relevant for some inpaths to clause i satisfied by t and is not relevant for other inpaths to i satisfied by t.

Now for convention 7:

If OGC $K(k)$ is inconsistent relative to transaction t, then K is executed for transaction t if and only if it is relevant for the first inpath to $K(i)$ selected by t. We shall call this practice the precedence convention, and say that the selected rule preempts transactions from the other inpaths satisfied by t.

Example (1)

	R1	R2	R3
.....			
: C1 A<	:	10	20
		30	:
.....			
: A1	:	X	-
	:	-	-
: A2	:	-	X
	:	-	:
: A3	:	X	-
	:	-	:
.....			

This table is inconsistent for transactions having $A=5$; for such transactions the precedence convention causes rule 1 to be satisfied with the result that actions A1 and A3 are executed and action A2 is not executed.

Example (2)

	R1	R2	R3	R4
.....				
: C1 A=1	:	Y	Y	-
	:			N
.....				:
: C2 B>5	:	Y	N	N
	:			Y
: A1 X:=3	:	-	X	-
	:		-	-
: C3 X=C	:	-	Y	N
	:			-
.....				:
: A2 D:=6	:	-	-	X
	:			X
.....				:

This table is inconsistent for the transactions for which C1 is true and C2 is false. Action A1 is to be performed for such transactions because rule R2 contains the first inpath to A1 satisfied by the transaction and R2 calls for the execution of A1.

3.7.7 The precedence relationships between rules

The precedence convention approach to inconsistent tables is compatible with section 3.3.4 previous in which an ELSE relation between the rules of a table was established with the result that the selection of one rule precludes the selection of any additional rules.

3.7.8 ELSE rule

The precedence convention also supports the concept of a ELSE rule -- the last rule in a table becomes a "catch-all" to be selected when all other rules fail. One can generalize the ELSE rule and use our precedence convention to reduce the size of a table by combining rules which imply the same actions:

Example

	R1	R2	R3	R4	R5	
: C1 :	Y	Y	Y	Y	-	:
: C2 :	Y	Y	Y	N	-	:
: C3 :	Y	N	-	N	-	:
: C4 :	-	Y	-	-	-	:
: C5 :	Y	-	-	Y	-	:
<hr/>						
: A1 :	X	-	X	-	X	:
: A2 :	-	X	-	-	X	:
: A3 :	-	-	X	-	X	:
<hr/>						

Under the precedence convention rule R5 in the example functions as a traditional ELSE rule -- it is selected by transactions which do not satisfy any other rule. Rule R3 functions as a "local" ELSE rule for transactions for which both C1 and C2 are true.

3.7.9 Multiple hit tables

Some have suggested an alternative interpretation for inconsistent tables. One can imagine applications in which it is desirable to have more than one rule apply to a transaction. Such tables are termed "Multiple Hit Tables" as opposed to the more common tables, which would be termed "Single Hit Tables" and direct that every rule appropriate to a transaction be executed.

A Multiple Hit Table allows an inclusive-or relationship to relate the rules of a table to a particular transaction. Because of this, Multiple Hit Tables are inconsistent when interpreted under the exclusive-or (single-hit) assumption. One can always rework a multiple hit table as a series of single hit tables.

3.8 ANALYSIS AND MANIPULATION

3.8.1 Introduction to checking for completeness and consistency

3.8.1.1 Motivation

One major claim supporting the use of decision tables is that they can be checked for completeness and consistency. We have defined completeness and consistency on two levels. First, on a structural level, completeness and consistency are defined in terms of combinations of condition alternatives, and it is relatively easy to check for them and to automate the checking process. But for the second case, on the semantic level, the problem is more difficult. Completeness and consistency are defined in terms of the universe of transactions which are to be processed against the table and one may need considerable knowledge about this universe before he can be certain that the table is complete and consistent. A few years ago, incorporation of a substantial portion of this "semantic" knowledge into a computer automated procedure for checking decision tables appeared to be almost impossible; now it is well within the state of the art in computer science. This will be discussed further in 3.8.1.5.

3.8.1.2 Structural checking by inspection

The best way to check a decision table for completeness and consistency is to step systematically through all possible combinations of alternatives and to ascertain, by inspection, that each combination satisfies one and only one rule. This is easier if the table is structured in a systematic manner.

EXAMPLE

```
.....  
: Season :Winter Spring Spring Summer Summer Fall Fall :  
:  
:  
:  
: Attendance :> 0.9: - Y N Y N Y N :  
: Enrollment :  
:  
:.....:  
: Code2:= : 7 5 6 1 2 3 4 :  
:.....:
```

is easier to check than:

```
.....  
: Season = : Fall Spring Winter Summer Spring Summer Fall :  
:  
:  
:  
: Attendance :> 0.9: N Y - N N Y Y :  
: Enrollment :  
:  
:.....:  
: Code2:= : 4 5 7 2 6 1 3 :  
:.....:
```

3.8.1.3 Semantic checks

Semantic checks often require knowledge of the environment in which a decision table is to be used. Such is the case with the table in the last example where the variable SEASON is assumed to have one of the four values 'WINTER', 'SPRING', 'SUMMER', or 'FALL' and the table makes no provisions for other values. If the table is to be used in a computer program to process unvalidated inputs, the table is incomplete, as there is a good chance that someone will misspell a season's name or use 'AUTUMN', in place of 'FALL'. On the other hand, the table is complete with respect to SEASON, if a 'forgiving' human is to process the inputs or if the program has validated SEASON prior to entering the table and one assumes that the value of SEASON is not unintentionally altered between the time it is validated and the time it is accessed from the table.

Even limited entry conditions may be incomplete; if both 'attendance' and 'enrollment' are zero, the ratio attendances/today's enrollment is undefined and is neither ($>.9$) nor (not $>.9$).

The consistency of the table in the last example is also a matter of interpretation. It depends upon vulnerability of the literals referenced by the table; the table in the last example is inconsistent, if there is a chance that both 'SPRING' and 'SUMMER' will be overwritten by a zero.

3.8.1.4 Automated structural checks

The best approach is to combine validation with completeness and consistency thru conversion of the decision table to a tree as described earlier in 3.6.10.

3.8.1.5 Automated semantic checks

Little has been done in this area. One reason may be that it appears to be impossible to achieve a one hundred percent accurate semantic verification for an individual condition's completeness and consistency. Nevertheless, it appears feasible to have a computer verify a very substantial proportion of the possible cases and to warn of cases which it cannot verify.

The key to semantic checks lies in the ability to draw inferences between conditions; for example: $A = 0$ true implies $A > 0$ is false. Sometimes such inferences are difficult to draw, e.g., $X=3$ implies $X^2-6X+9=0$. Nevertheless a great many inferences can be drawn from analysis of the following:

1. The (syntactical) equality of two expressions -- expression e_1 and e_2 are syntactically equal if they parse into identical sequences of syntactical elements. This involves a way of ignoring interspersed blanks and other extraneous punctuation.

2. Implications between conditions which compare pairs of equal expressions, e.g., between $X < a + b$ and $X = a + b$.
3. Implications between conditions which compare equal expressions with literals -- e.g., between ORDERNBR= 20 and ORDERNBR=100.
4. Association of a range of permissible values with variables to facilitate checking for completeness

Example

```
.....:.....:.....:  
: : :  
: C1 AGE < 20 : Y - N :  
: : :  
: C2 AGE > 60 : - Y N :  
.....:.....:  
: A1 RECODE:= : 1 2 3 :  
.....:.....:
```

This table appears to be inconsistent because the transaction (Y,Y) satisfies both R1 and R2. However, the procedure just outlined can be applied to draw an inference between C1 and C2, yielding a consistent table (Note: as defined in 3.4.5 "N!" means no by implication).

```
.....:.....:.....:  
: : :  
: C1 AGE < 20 : Y N! N :  
: : :  
: C2 AGE > 60 : N! Y N :  
.....:.....:  
: A1 RECODE:= : 1 2 3 :  
.....:.....:
```

3.8.1.6 Impossible rules

The semantic analysis described in 3.8.1.5 may reveal impossible rules -- rules such that one condition's entry implies that another condition's entry cannot be satisfied. One can remove impossible rules from a table since they will never be executed; however, one should report their presence as part of completeness and consistency checking.

Example

	R1	R2	R3	R4
.....:.....:.....:				
: : :				
: X \geq 10 : Y Y N N :				
: : :				
: X \geq 30 : Y N Y N :				
.....:.....:.....:				
: A: = : 1 2 3 4 :				
.....:.....:.....:				

Table rule R3 is headed by an i to indicate impossible -- it requires both $X < 10$ and $X \geq 30$.

	R1	R2	R3	R4
:			i	:
: $X \geq 10$:	Y!	Y	N
:				N :
: $X \geq 30$:	Y	N	Y
:				N! :
:.....:
:A:=	:	1	2	3
				4 :
			

3.8.1.7 Execution time validation

Our previous discussion has concentrated on validating a table when it is initially examined by a computer at compile time. One can do execution time validation by converting an incomplete table into a program which performs some sort of error procedure when an unsupported transaction is processed. Analogously an inconsistent table can be mapped into a program which takes some sort of error action when a transaction processed against a table satisfies more than one of the table's rules. Pragmatically, one might consider a table semantically complete and consistent, if these error procedures are never executed during the table's operational lifetime.

3.8.1.8 Checking based upon simple rule counts

A frequently cited method for checking a decision table's completeness is to count the simple rules it accommodates, and compare this count, NSR, with NISR, the number of simple rules implied by the table's conditions. Such a comparison can tell one that a table is not syntactically complete and consistent, but unfortunately it cannot establish that a table is complete and consistent nor can it identify which transactions cause the problems.

A simple rule corresponds to an individual point in \bar{C} , a decision table's condition space, and consists of exactly one alternative for each of the table's conditions. The number of simple rules implied by a table's conditions equals the product of the number of alternatives for the table's conditions. Letting NISR = the number of simple rules implied by a table's conditions and NCA(c) = the number of alternatives for condition c, we have:

NISR=NCA(1)*NCA(2)*NCA(3)*...

Each rule within a decision table corresponds to one or more simple rules. If we define the power of a rule, $PW(r)$, as the number of simple rules represented by rule r ; $NA(r,c)$ as the number of alternatives for condition c associated with entry c of rule r ; and SPW as $PW(r)$ summed over all rules, we have:

$$PW(r) = NA(r, 1) * NA(r, 2) * \dots$$

and $SPW = PW(1) + PW(2) \dots$
 Note if $E(r,c) = " - "$ the $NA(r,c) = NCA(c)$.

Example

Consider D11
 with condition set C11=
 C11(1): A=0, [Y,N]
 C11(2): B=, [1, 2, 3, 4]
 C11(3): W=0, [Y,N]
 and the following table

	R1	R2	R3	R4	R5
: C11(1) A=0 :	Y	Y	Y	Y	N
:	:				:
: C11(2) B= :	1	1	2 or 3	4	-
:	:				:
: C11(3) W=0 :	Y	N	-	-	Y
.....:	:
: A11(1) :	X	-	X	-	X
:	:				:
: A11(2) :	X	X	-	-	X
.....:	:

We have

```

NISR = NAC(1) * NAC(2) * NAC(3)
      = 2 * 4 * 2 = 16
NA(1,1) = 1 -- one alternative, Y, for C11(1)
NA(3,2) = 2 -- two alternatives, 2 and 3, for C11(2)
NA(3,5) = 4 -- when the entry is '-', use NA(C) because all of c's
               alternatives are permissible.
PW(1) = 1 * 1 * 1 = 1
PW(2) = 1 * 1 * 1 = 1
PW(3) = 1 * 2 * 2 = 4
PW(4) = 1 * 1 * 2 = 2
PW(5) = 1 * 4 * 1 = 4
SPW = 1 + 1 + 4 + 2 + 4 = 12

```

A comparison of NISR and SPW can reveal a defective table. If the table D contains no implied entries, and SPW is less than NISR, then D is incomplete. If SPW is greater than NISR then D is inconsistent. In the last example SPW is 12 which is less than NISR = 16; inspection of the table reveals that it is incomplete and the transactions satisfying the rule N, -,N are not supported. When SPW=NISR and the situation is not straightforward and we have:

3.8.1.9 Theorem 6

If table D is complete and consistent and contains no implied entries, then SPW = NISR. If SPW = NISR and table D is consistent and contains no implied entries, then D is complete. If SPW = NISR and table D is complete and contains no implied entries, then D is consistent. Thus SPW is only a necessary condition for D to be complete and consistent, and is not a sufficient condition.

3.8.2 Reordering clauses within a decision table

Two clauses within a decision table are disjoint, if there is no rule within the table for which both clauses are relevant. The order of two adjacent disjoint clauses can be reversed since neither affects the other.

Two adjacent non-disjoint clauses may be reordered if one clause does not output a variable used as an input by the other, they do not output the same variables and the reorder does not result in an OGC (output generating clause) being executed for a rule for which it is not relevant. (See 3.7.3) Non-adjacent clauses may be reordered by repeated reordering involving adjacent clauses.

3.8.3 Reordering rules

Two adjacent rules may be reordered, if they are mutually exclusive or if they imply the generation of identical output.

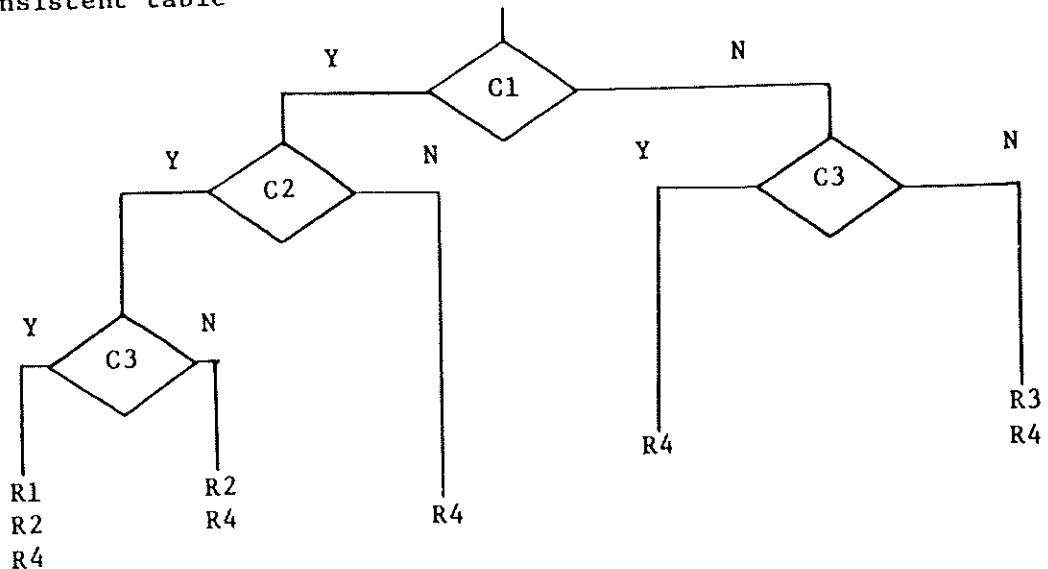
3.8.4 Resolution of inconsistencies using the precedence convention

In Sections 3.7.6 through 3.7.8 we discussed the evaluation of inconsistent tables such as those which have an ELSE rule and formulated the precedence convention as a way of resolving the table's inconsistencies. Sometimes one may wish to display the convention's effect by applying it to convert an inconsistent table to a consistent table. An effective way of doing this is to convert the inconsistent table to a tree and then convert the tree back to a consistent table. Conversion of a table to a tree reveals inconsistencies by containing situations where there are no more conditions to test but where two or more rules apply. In such situations, the precedence conventions direct that the rule which occurs earliest in the table be selected. This approach is especially effective when one is using a computer to process decision tables. While generating such a tree, one usually employs a stack to record his position within the tree. Production of the consistent table requires only that the stack be stored as a rule in the "new" table whenever a rule is selected within the tree.

Example

	R1	R2	R3	R4
.....				
:	:			:
: C1	:	Y	Y	N
:				-
:				:
: C2	:	Y	Y	-
:			-	-
:				:
: C3	:	Y	-	N
:				-
.....
: A1 X;= :	1	2	3	4
.....

a) inconsistent table



b) tree

	R1	R2'	R4'	R4''	R3	
:	:	:	:	:	:	:
: C1	:	Y	Y	Y	N	N
:	:	:	:	:	:	:
: C2	:	Y	Y	N	-	-
:	:	:	:	:	:	:
: C3	:	Y	N	-	Y	N
:	:	:	:	:	:	:
: A1 X:=	:	1	2	4	4	3
:	:	:	:	:	:	:

- c) Consistent table -- primes indicate rules altered in converting the inconsistent table to a consistent table.

3.8.5 Combining rules

Sometimes it is possible and desirable to combine rules which imply the same sequence of actions (i.e., belong to the same rule class as defined in 3.2.12). Rules can be combined only if they are mutually exclusive with respect to any intervening rules. A decision table is said to be consolidated if all possibilities to combine rules are accomplished. It is instructive to show the mechanics of this combining by referring to a table's programming language representation as developed in 3.3, since the latter is more amenable to manipulation using a formal logic calculus; consider the following decision table:

Example D7

.....
: C1	A=0	:	Y	Y	Y	N
:	:	:	:	:	:	N
: C2	B =	:	1	2	3	1
:	:	:	2	3	:	2
:	:	:	3	:	:	3
: A1 S:=	:	1	1	1	12	20
:	:	1	1	1	12	20
: A2 T:=0	:	-	-	-	X	X
:	:	-	-	-	X	X
:	:	-	-	-	-	-

Expressing this in the format of subsection 3.3.4, we have:

```

IF (A=0 AND B = 1) THEN (S:= 1,SKIP)
ELSE IF (A=0 AND B = 2) THEN (S:=1,SKIP) ELSE....
=> IF (A=0 AND B = 1) OR (A=0 AND B = 2) THEN (S:=1,SKIP) ELSE....
=> IF (A=0 AND (B=1 OR B=2)) THEN (S:=1,SKIP) ELSE....
=> IF (A=0 AND B= (1 OR 2)) THEN (S:=1,SKIP) ELSE....

```

Now we can repeat the process and combine the newly derived rule with the third rule of the original table to obtain:

```
IF (A=0 AND B = (1 OR 2 OR 3)) THEN (S:=1,SKIP)
ELSE IF (NOT(A=0) AND B = 1) THEN (S:= 12, T:=0)
ELSE IF (NOT(A=0) AND B=2) THEN (S:= 20, T:=0)
ELSE IF (NOT(A=0) AND B=3) THEN (S:=20,SKIP)
```

If B has only the values 1, 2, and 3, the condition $B = (1 \text{ OR } 2 \text{ OR } 3)$ exhausts all alternatives associated with B, is always true, and can be replaced by the logical constant TRUE yielding:

```
IF (A=0 AND TRUE) THEN (S:=1,SKIP)
ELSE IF (NOT (A=0) AND B =1) THEN (S:=12, T:=0)
ELSE IF (NOT (A=0) AND B =2) THEN (S:=20, T:=0)
ELSE IF (NOT (A=0) AND B =3) THEN (S:=20,SKIP)
```

Representing:

.....
:	:	:	:
: C1	A=0	: Y N N N :
:	
: C2	B=	: - 1 2 3 :
.....
: A1	S:=	: 1 12 20 20 :
:	
: A2	T:=0	: - X X - :
.....

3.8.6 Converting from extended to limited entries

It is often desirable to transform an extended entry table into a limited entry table. Such a transform is required for implementation on a computer, because the limited entry form is compatible with the computer's binary branching capabilities while the extended entry format is not compatible with binary branching. (The 'GO TO Depending On' case is so seldom applicable that it can be ignored.) The automated computer processing of an extended entry table either interpretively or by execution of a program compiled from the table involves conversion of extended to limited entries. Sometimes, the conversion is done implicitly, leading to the misleading claim that "conversion to limited entry format is avoided".

We shall illustrate the conversion in terms of decision table D7:

.....
:	:	:	:
: C1 A=0	: Y Y Y N N N :
:	
: C2 B=	: 1 2 3 1 2 3 :
.....
: A1 S:=	: 1 1 1 12 20 20 :
:	
: A2 T:=0	: - - - - X X - :
.....

which has the following programming language representation:

```
IF (A=0 AND B = 1) THEN (S:=1,SKIP)
ELSE IF (A=0 AND B = 2) THEN (S:=1,SKIP)
ELSE IF (A=0 AND B = 3) THEN (S:=1,SKIP)
ELSE IF (NOT (A=0) AND B=1) THEN (S:=12,T:=0)
ELSE IF (NOT (A=0) AND B=2) THEN (S:=20,T:=0)
ELSE IF (NOT (A=0) AND B=3) THEN (S:=20,SKIP)
```

We start the development of a conversion technique by observing that the term AND TRUE can be inserted into any AND sequence of logical terms -- AND TRUE is always true and does not affect the sequence's truth value. Applying this to the programming representation of D7, we have

```
IF (A=0 AND B = 1 AND TRUE) THEN (S:=1,SKIP)
ELSE IF (A=0 AND B = 2 AND TRUE) THEN (S:=1,SKIP)
ELSE IF (A=0 AND B = 3 AND TRUE) THEN (S:=1,SKIP)
ELSE IF (NOT (A=0) AND B=1 AND TRUE) THEN (S:=12,T:=0)
ELSE IF (NOT (A=1) AND B=2 AND TRUE) THEN (S:=12,T:=0)
ELSE IF (NOT (A=0) AND B=3 AND TRUE) THEN (S:=20,SKIP)
```

Rearranging the terms in lines 2,3,5, and 6, we have

```
IF (A=0 AND B=1 AND TRUE) THEN (A:=1,SKIP)
ELSE IF (A=0 AND TRUE AND B=2) THEN (S:=1,SKIP)
ELSE IF (A=0 AND TRUE AND B=3) THEN (S:=1,SKIP)
ELSE IF (NOT (A=0) AND B=1 AND TRUE) THEN (S:=12,T:=0)
ELSE IF (NOT (A=0) AND TRUE AND B=2) THEN (S:=20,T:=20)
ELSE IF (NOT (A=0) AND TRUE AND B=3) THEN (S:=20,SKIP)
```

These steps yield a new version of D7 in which condition 2 has been split into two parts: one, a limited entry test for equality of B and the value 1; the second, taking care of the cases in which B is 2 or 3.

```
.....: C1 A=0 : Y Y Y N N N : :
: C2 B=1 : Y - - Y - - : :
: C3 B= : - 2 3 - 2 3 : :
.....: A1 S:= : 1 1 1 12 12 20 : :
: A2 T:=0 : - - - X X - : :
.....: .....
```

One can apply the procedure again to the comparisons of B with 2 and of B with 3 to obtain D11 in which all entries are limited:

```
D11
.....
: C1: A=0 : Y  Y  Y  N  N  N :
:           :
: C2: B=1 : Y  -  -  Y  -  - :
:           :
: C3: B=2 : -  Y  -  -  Y  - :
:           :
: C4: B=3 : -  -  Y  -  -  Y :
:.....:.....:.....:
: A1: X:=1 : X  X  X  -  -  - :
:           :
: A2: X:=12: -  -  -  X  -  - :
:           :
: A3: X:=20: -  -  -  -  X  X :
:           :
: A4: T:=0 : -  -  -  X  X  - :
:.....:.....:.....:
```

3.8.7 Completeness and consistency in converted tables

Table D11 is a limited entry decision table derived from table D7 (3.8.6). Structurally D11 is neither complete nor consistent. It is not complete, because it fails to provide a rule for point (N,N,N,N) in the condition space of D11 and it is inconsistent because point (Y,Y,Y,Y) satisfies the first three rules. These structural difficulties are a direct consequence of semantic problems in the original table, D7. Semantically, D7 provides for B having the values 1, 2, and 3 only, and does not provide for B's assuming values outside this range. Again in the semantics of D7, the values "1", "2", and "3" are considered mutually exclusive while in the limited entry format, they are not so considered. One cannot cope with these problems syntactically; they must be resolved semantically.

Inconsistencies:

If the semantics tell us that a set of conditions is mutually exclusive, one can get rid of inconsistencies by using the implied limited entries described in section 3.4.5.

Example

In our current table the B=1, B=2, and B=3 are mutually exclusive, and we can write the table as:

:	A=0	:	Y	Y	Y	N	N	N	:
:		:							:
:	B=1	:	Y	N!	N!	Y	N!	N!	:
:		:							:
:	B=2	:	N!	Y	N!	N!	Y	N!	:
:		:							:
:	B=3	:	N!	N!	Y	N!	N!	Y	:
:		:							:
:	S:=1	:	X	X	X	-	-	-	:
:		:							:
:	S:=12	:	-	-	-	X	-	-	:
:		:							:
:	S:=2	:	-	-	-	-	X	X	:
:		:							:
:	T:=0	:	-	-	-	-	X	-	:
:		:							:

Incompleteness: If the conditions are semantically exhaustive, one can make the table complete by writing an ELSE rule designated as being impossible or by employing additional implied limited entries.

Example

If in D7, B can assume only the values 1, 2, and 3, we can write:

:	A=0	:	Y	Y	Y	N	N	N	:
:		:							:
:	B=1	:	Y!	N!	N!	Y!	N!	N!	:
:		:							:
:	B=2	:	N!	Y!	N!	N!	Y!	N!	:
:		:							:
:	B=3	:	N!	N!	Y!	N!	N!	Y!	:
:		:							:
:	S:=1	:	X	X	X	-	-	-	:
:		:							:
:	S:=12	:	-	-	-	X	-	-	:
:		:							:
:	S:=20	:	-	-	-	-	X	-	:
:		:							:
:	T:=0	:	-	-	-	X	X	-	:
:		:							:

In each rule the implied entries for the conditions involving B (i.e., conditions 2, 3, and 4) say that if one knows two entries, he can deduce the third. For rule 1, if B is not 2 and is not 3, then it must be 1 (the only choice left). Hence the implied YES, i.e., "Y!".

3.8.8 Elimination of redundant conditions

A condition is redundant if it carries no relevant entries -- i.e., all its entries are '`-`', '`#`', '`Y!`', or '`N!`'

A redundant condition can be removed provided it is not an OGC (output generating clause). If a redundant condition is removed, every implied entry, e, for the remaining conditions must be reviewed because entries in the removed condition may have been involved in determining that e is implied.

Example

The last table in subsection 3.8.6 contains three redundant clauses, C2, C3, and C4. We can remove one of these, say C2, and obtain the following reduced table:

:	C1	A = 0	:	Y	Y	Y	N	N	N	:
:			:							:
:	C3	B = 2	:	N	Y	N!	N	Y	N!	:
:			:							:
:	C4	B = 3	:	N	N!	Y	N	N!	Y	:
.....
:	A1	S: = 1	:	Y	X	X	-	-	-	:
:			:							:
:	A2	S: = 12	:	-	-	-	X	-	-	:
:			:							:
:	A3	S: = 20	:	-	-	X	-	X	-	:
:			:							:
:	A4	T: = 20	:	-	-	.	X	X	-	:
.....

Note that many of the entries for C3 and C4 no longer are implied. Thus C3 and C4 have '`N!`'s for R1 in the original table. In the latter these were implied false because C2 required $B = 1$ for R1 and $B = 1$ implies not $B = 2$ and not $B = 3$. When C2 is removed, we no longer have $B = 1$ and cannot use this to infer not $B = 2$ and not $B = 3$. This reason prevents one from removing all of the last 3 conditions of the last table in 3.8.7.

3.8.9 Expansion -- combining tables

We have already discussed (section 3.6.1) the combining of tables accessed from one another table via an invocation or a GO TO. We return to this subject because the combined tables may contain logically related clauses and it may be possible to simplify the combined table by applying the semantic checks of section 3.8.6.

Example (Note: condition C2 is common to both tables.)

Table 1

Table 2

.....	
: C1	: Y Y Y N :	: C4 : Y Y N :
:	:	:
: C2	: Y Y N - :	: C2 : Y N - :
:	:	:
: C3	: Y N - - :	: A3 : X X - :
.....
: A1	: - X - - :	: A4 : X - X :
:	:	:
: A2 Do Table 2	: X - X - :
.....

The following is the combined table with results of semantic analysis but with redundant conditions and impossible rules retained. Rules R2 and R5 are tagged as impossible because they require that condition C2 be both true and false. After the removal of R2 and R5, the second occurrence of C2 contains only Y! and N! entries and can be dropped.

	R1	R2	R3	R4	R5	R6	R7	R8
	i	i			i			
:	C1	:	Y	Y	Y	Y	Y	N
:	:							
:	C2	:	Y	Y	Y	Y	N	N
:	:						-	-
:	C3	:	Y	Y	Y	N	-	-
:	:					-	-	-
:	A1	:	-	-	-	X	-	-
:	:					-	-	-
:	C4	:	Y	Y	N	-	Y	N
:	:					-	-	-
:	C2	:	Y!	N	Y!	-	Y	N!
:	:					-	-	-
:	A3	:	X	X	-	-	X	-
:	:					-	-	-
:	A4	:	X	-	X	-	X	-
:	:					-	-	-

Combined table after simplification

	R1	R3	R4	R6	R7	R8	
: C1 :	Y	Y	Y	Y	Y	N	:
: :							:
: C2 :	Y	Y	Y	N	N	-	:
: :							:
: C3 :	Y	Y	N	-	-	-	:
: :							:
: A1 :	-	-	X	-	-	-	:
: :							:
: C4 :	Y	N	-	Y	N	-	:
.....							
: A3 :	X	-	-	X	-	-	:
: :							:
: A4 :	X	X	-	-	X	-	:
.....							

3.8.10 Equivalent decision tables, transformations theorem 7

Two decision tables, D and D' are equivalent under evaluation conventions, E, if they are defined over the same set of transactions, T and if for every transaction t in T, $D(t)=D'(t)$ when evaluated using evaluation convention E. The manipulations on individual tables described in the present subsections, 3.8, and those on systems of tables described in 3.6 can be thought of as transformations which change one decision table, D into another table D'. We have the following theorem which says that all of the manipulations that we have described are valid -- each transforms a table D into an equivalent table D'.

Theorem 4: Decision tables D and D' are equivalent under the seven evaluation conventions given in 3.7.3 and 3.7.6, if they apply to the same set of transactions and if D' consists of one or more tables, derived from D, by one of the following transformations:

1. Reordering of clauses as defined in 3.8.2
2. Reordering of rules as defined in 3.8.3
3. Combining of rules as defined in 3.8.5
4. Conversion of extended into limited entries as defined in 3.8.6
5. Conversion of entries into implied entries as described in 3.8.1.5
6. Elimination of impossible rules as defined in 3.8.1.6
7. Removal of redundant condition as defined in 3.8.
8. Expansion as defined in 3.6.2 and 3.8.8
9. Factoring as defined in 3.6.6
10. Elimination of inconsistencies as defined in 3.8.7

3.9 DECISION TABLES AND ITERATIONS

3.9.1 Motivation

So far we have avoided one very important aspect of procedures and computer programs; namely, iterations or "looping" -- the repeating of one or more statements in accordance with some sort of control mechanism.

3.9.2 Simple and complex iteration

Initially it is desirable to distinguish the types of iteration; simple and complex. This distinction is based on the number of statements which they encompass, and their effect upon a program's overall structure. A simple iteration involves only one, two, or a very few statements; usually should be written as a single decision table action, and is 'local' in that it affects only these few instructions.

Example

```
.....  
: C1 : Y Y N :  
:  
: C2 : Y N - :  
.....  
: SBT(I):=SBT(I)+ITEM(I),I:=1,N: - X - :  
:  
: [(TOT(1)=TOT(1)+SBT(I), :  
: SBT(I):=0],I:=1,10 : X - - :  
:  
: DISPLAY TOT(I), I:=1,N : - - X :  
.....
```

By way of contrast, complex iterations involve a number of statements, and usually have global effect -- upon their completion, the program takes a fundamentally different path. As such complex iterations represent significant program decisions, they should be integrated into the program's decision table structure.

Example

```
.....  
: More states to process : Y Y N :  
: More counties in current state : Y N - :  
.....  
: Roll state subtotal to US total : - X X :  
: Tally into state subtotal : X - - :  
: Display state subtotal total : - X X :  
: Display US total : - - X :  
: Advance to next county : X - - :  
: Advance to next state : - X - :  
: Start with first county in state: - X - :  
: Zero out state subtotal : - X - :  
: Repeat table : X X - :  
.....
```

3.9.3 Implementing iterations

The writing of minor iterations as individual actions within a decision table is straightforward and adequate, particularly, if one extends the notion of conditions to "ANY" or "ALL" constructions, e.g.,

```
ANY A(I)=0 I:=1,10
ALL B(K)+C(J,K) GTR 25,J:=1,N,2 K:=1,M
```

The case of major iterations is not so simple. One approach is to use a repeat table command as in the last example. With this approach, one cannot place any initialization actions in the iterated table but must isolate them in a separate table.

Example

1. Initialize

```
.....  
: I:=1 : X :  
: Sum1:=0 : X :  
: Sum2:=0 : X :  
.....:....:
```

2. Iterate

```
.....  
: I<=N : Y Y Y Y N :  
: SEX(I)=Male : Y Y Y N - :  
: AGE(I) < 18 : Y Y N - - :  
: INCOME(I) > 15,000 : Y N - - - :  
.....:.....:  
:  
: SUM1:=SUM1 + W(I) : X - - - - :  
: SUM2:=SUM2 + W(I) : - X - - - - :  
: I:=I+1 : X X X X - :  
: Repeat table : X X X X - :  
.....:.....:
```

Satisfactory handling of major iterations in a single decision table appears to imply the ability to re-enter a decision table at a point other than its initial entry point. So before proceeding with iterations we shall take a look at decision table entry points and subtables.

3.9.4 Decision table entry points

Naively, one might think that one "enters" a decision table at the first clause of the table's first rule. This is not right. When one executes a decision table one goes through a selection process to determine which rule to execute; initially all rules are candidates for this selection; hence, it is appropriate to think of entering the table at the first clause of every rule in the table.

There is a similar problem with respect to a command such as GO TO clause c of rule r of table D (either directly or indirectly via a GO TO L where L is a label associated with entry r,c.) Presumably the command directs one to enter some subtable of D having r as its initial rule and c as its initial clause. It thus establishes r and c as the first rule and clause of this subtable but tells nothing about the other rules and clauses in the subtable. One can reasonably assume that clauses which follow c in the original comprise the remaining clauses in the subtables. By analogy one could reason that subtable rules comprise rule r and the rules which follow it in the table; however, the rules may yield a "bad" subtable -- one which is inconsistent or incomplete. The inconsistency problem can be resolved by applying the precedence convention described in 3.7.6 but there is no easy way of making the subtable complete. As we shall see, the somewhat unorthodox approach of using multiple labels is better.

Example (The Label stub signifies a table row which may contain a label associated with the next clause in the table -- the "L1" in line 3 designated L1 as the label for clause C3 of rule R5)

```

R1   R2   R3   R4   R5   R6   R7   R8
: C1      : Y    N    N    N    N    N    N    N    N    ;
: C2      : -    Y    Y    N    Y    N    N    N    N    ;
:
:
: A1 LABEL :                   L1
:
:
: C3      : -    -    Y    Y    N    N    Y    N    ;
: C4      : -    N    Y    Y    Y    N    N    Y    ;
:
:
: A2 X:=   : 1    1    2    3    -    4    5    2    ;
: A3 GO TO L1 : -    X    -    X    -    -    -    X    ;
:
:
```

One can reason that the subtable implied by the GO TO L1 consists of C3 and the clauses which follow it in the table and of rule R5 and those that follow it viz:

	R5	R6	R7	R8	
.....
: C3	:	N	N	Y	N
: C4	:	Y	N	N	Y
.....
: A2 X:=	:	-	4	5	2
: A3 GO TO L1	:	-	-	-	X
.....

This table is neither complete nor consistent. By applying the precedence convention one can identify the following consistent subtable, but the subtable is still incomplete.

.....
:	:	:	:	:	:
: C3	:	N	N	Y	:
: C4	:	Y	N	N	:
.....
: A2 X:=	:	-	4	5	:
: A3 GO TO L1	:	-	-	-	:
.....

3.9.5 Explicit subtables

By "GOING TO" an entry or attaching a label to an entry, one is really defining a subtable implicitly. An alternative is to define a subtable explicitly by allowing "SUBTABLE" as a specialized, "declarative" decision table action, and having it carry a label which functions as the subtable's identifier. The subtable's rules comprise just those rules for which the subtable action is relevant. The subtable's clauses are those clauses which 1) follow the subtable action, and 2) are relevant for at least one of the subtable's rules. The order of the rules and columns in the original table is preserved in the subtable; the subtable does not have to be a proper subtable as defined in 3.5.5.

Example

	R1	R2	R3	R4	R5	R6	R7	R8
: C1	:	Y	Y	N	N	N	N	N
:	:							
: A1 subtable L1	:	X	-	-	-	-	X	X
:	:							
: C2	:	Y	N	Y	Y	Y	N	N
:	:							
: C3 AMAX=	:	-	-	1	2	3	1	2
:	:							
: A2	:	X	X	-	-	X	X	-
:	:							
: A3	:	-	X	-	X	-	-	-
:	:							

defines decision table L1 as:

	R1	R6	R7	R8
: C2	:	Y	N	N
:	:			
: C3 AMAX	:	-	1	2
:	:			
: A2	:	X	X	-
:	:			
: A3	:	-	-	-
:	:			

If we allow the SUBTABLE action to support extended entries, we can define several subtables with a single SUBTABLE action. Further, if SUBTABLE actions are excluded from the selection of clauses elicited by a subtable command, the same clause can participate as the initial clause in several subtables.

Example

	R1	R2	R3	R4	R5	R6	R7	R8	R9
: C1	:	Y	Y	N	N	N	N	N	N
:	:								
: C2	:	-	-	Y	Y	Y	N	N	N
:	:								
: A1 Subtable	:	L1	L1	-	-	-	L2	L2	L2
:	:								
: A2 Subtable	:	L3	-	-	-	L3	-	L3	L3
:	:								-
: C3	:	Y	N	Y	N	N	Y	N	N
:	:								
: C4	:	-	-	-	Y	N	-	Y	Y
:	:								N
: C5	:	-	-	-	-	-	-	Y	N
:	:							-	
: A3 WGM:=	:	1	2	3	4	5	6	7	8
									9

Defines three subtables

	R1	R2		R6	R7	R8	R9	
L1 : C3	:	Y	N	L2 : C3	:	Y	N	N
:	:		:	:				
: A3 WGM:=	:	1	2	: C4	:	-	Y	Y
:	:		:	:			N	
				: C5	:	-	Y	N
				:			-	
				: A3 WGM:=	:	6	7	9

	R1	R5	R7	R8
L3 : C3	:	Y	N	N
:	:			
: C4	:	-	N	Y
:	:			Y
: C5	:	-	-	Y
:	:			N
: A3 WGM:=	:	1	5	7
				8
				:

3.9.6 Subtables, "good" programming practices and iterations

A fundamental reason for using decision tables is to make procedures, algorithms and programs easier to read and understand. Use of subtables reduces this ease of comprehension, particularly when accompanied, as is usual, with the use of GO TO's as a means of linking to subtables. Extensive utilization of subtables and GO TO's spawn all the difficulties observed in unstructured programs by proponents of GO TO-less, structured programming. Still it is felt that subtables used judiciously can enhance usefulness of decision tables by providing a means of iterating the part of a table. This can be accomplished by adhering to the following guidelines:

1. Subtables should be complete and consistent.
2. A subtable should be the consequent of a single inpath.
3. A subtable can be accessed only by an inpath to the table or by a REPEAT action.
4. Any rule which calls for the repetition of subtable S must be relevant for the action which defines S -- the command to repeat S must be nested within S.

Example 1

```
.....:  

: N>0 : Y Y Y Y Y Y N :  

: : : : : : : :  

: I:= 1 : X X X X X - :  

: : : : : : : :  

: SUM1:= 0 : X X X X X - :  

: : : : : : : :  

: SUM2:= 0 : X X X X X - :  

: : : : : : : :  

: SUBTABLE : L1 L1 L1 L1 L1 - :  

: : : : : : : :  

: I<N : Y Y Y Y N - :  

: : : : : : : :  

: A(I) : >0 =0 -0 <0 - - :  

: : : : : : : :  

: B(I) = 0 : - Y N - - - :  

: : : : : : : :  

: SUM1:= SUM1 + 1 : - X - - - - :  

: : : : : : : :  

: SUM2:= SUM2 + 1 : - - X - - - :  

: : : : : : : :  

: I:= I + 1 : X X X - - - :  

: : : : : : : :  

: Display SUM1, SUM2 : - - - X X - :  

: : : : : : : :  

: Display "BAD A(I)" : - - - X - - :  

: : : : : : : :  

: REPEAT L1 : X X X - - - :  

: : : : : : : :  

:.....:
```

Example 2

```

.....:.....:.....:.....:.....:.....:.....:.....:.....:.....:.....:
: S = 0      : X   X   X   X   X   X   X   X   X   :
:           :   :
: T = 0      : X   X   X   X   X   X   X   X   X   :
:           :   :
: I = 1      : X   X   X   X   X   X   X   X   X   :
:           :   :
: Subtable   : L2  L2  L2  L2  L2  L2  L2  L2  L2  :
:           :   :
: I>N       : Y   Y   Y   Y   Y   Y   Y   N   :
:           :   :
: J_ = 1      : X   X   X   X   X   X   X   -   :
:           :   :
: Subtable   : L3  L3  L3  L3  L3  L3  L3  -   :
:           :   :
: J<M       : Y   Y   Y   Y   Y   N   -   :
:           :   :
: A(I,J)<  : 80  70  90  100 200 -   -   :
:           :   :
: B(I,J)>=5: Y   N   -   -   -   -   -   :
:           :   :
.....:.....:.....:.....:.....:.....:.....:.....:.....:.....:.....:
: S:= S + 1  : X   -   -   -   -   -   -   -   :
:           :   :
: T:= T + 1  : -   X   -   -   -   -   -   -   :
:           :   :
: J:= J + 1  : X   X   X   -   -   -   -   -   :
:           :   :
: I:= I + 1  : -   -   -   X   -   X   -   :
:           :   :
: REPEAT     : L3  L3  L3  L2  -   L2  -   :
:           :   :
.....:.....:.....:.....:.....:.....:.....:.....:.....:.....:.....:
```

3.9.7 The WHILE condition

Although they provide a method of iterating a portion of a decision table, the SUBTABLE and REPEAT constructs described in the last section are cumbersome to use; for example, it requires three clauses to implement an iteration. The WHILE condition is proposed here as an instrument to overcome this inefficiency.

The WHILE condition is similar to the LABEL and SUBTABLE constructs discussed earlier, in that it establishes a subtable within its parent decision table. The subtable's clauses comprise the WHILE condition itself and all clauses which follow it in the original table. The subtable's rules comprise those rules for which the WHILE condition is relevant. The WHILE's subtable is executed either by reaching it from its parent table's initial clause or by repeating the WHILE. In either circumstance execution of the subtable begins at its first clause (i.e., the WHILE clause) with all of the subtable's rules as candidates for selection and proceeds through the subtable's conditions and actions as dictated by the mechanics of table evaluation and rule selection. The table is not repeated until all actions in the selected rule have been executed.

The WHILE condition comprises a subject and a set of limited entry alternatives. The subject is written in the stub portion of a decision table and comprises the word WHILE, optionally followed by a conditional expression or iteration control expression.

The optional conditional expression is a "normal" condition such as:

AGE > 21

QUANTITY-ON-HAND < QUANTITY-ORDERED

A + B = C + D

A = 0 OR A > 19

An iteration control expression provides a way of controlling the number of times the WHILE is repeated and is analogous to the indexing portion of a FORTRAN or ALGOL DO-statement. It has a form such as:

`index:=initial-value,terminal-value, increment`

and implies roughly that the subtable elicited by the WHILE is to be executed first with `index=initial-value`, then with `index=initial-value + increment`, then with `index=initial-value + 2* increment`, and so forth, until the index exceeds the terminal-value. If omitted, the increment is assumed to be 1. A control expression is said to hold or to be true, if the index is less than an equal to the terminal-value. It fails or is false when the index exceeds the terminal-value.

Examples

```
I:=1,NCASES
J:=1,100,5
LEVEL:=MIN,MAX
```

Alternatives (i.e., rule entries) for a WHILE condition play the dual role of specifying criteria for rule selection and of governing repetition of the WHILE and are restricted to four values. Let clause c be a WHILE condition, then permissible values for entry E(r,c) are:

- "R"--Repeat. The WHILE's conditional or control expression must be true for rule r to be selected. The WHILE's subtable is repeated, if rule r is selected.
- "Q"--Quit. The WHILE's conditional or control expression must be true for rule r to be selected. The WHILE's subtable is not repeated, if rule r is selected.
- "N"--No. The WHILE's conditional or control expression must be false for rule r to be selected. The WHILE's subtable is not repeated, if rule r is selected.
- "--" Ignore. The WHILE's conditional or control expression is irrelevant for rule r. The WHILE's subtable is not repeated, if rule r is selected.

Example

		R1	R2	R3
:		:		:
: A1	NFAILURES:=0	:	X X X	:
:		:		:
: C1	WHILE I:=1,NSTUDENTS	:	R R N	:
:		:		:
: C2	AVERAGE(I) < 70	:	Y N -	:
:		:		:
: A2	NFAILURES:=NFAILURES +1	:	X - -	:
:		:		:
: A3	PRINT NAME (I),GRADE(I),	:	X - -	:
:	AVERAGE (I)	:		:
:		:		:
: A4	PRINT NFAILURES,"FAILURES"	:	- - X	:
:		:		:

Condition C1 in this example's decision table is a WHILE condition. Its associated subtable comprises conditions C1 and C2 and actions A2, A3 and A4, and rules R1 thru R3 viz:

		R1	R2	R3
:		:		:
: C1	WHILE I:=1,NSTUDENTS	: R	R	N
:		:		:
: C2	AVERAGE (I) < 70	: Y	N	-
:		:		:
: A2	NFAILURES:=NFAILURES +1	: X	-	-
:		:		:
: A3	PRINT NAME(2),GRADE(2), AVERAGE(2)	: X	-	-
:		:		:
: A4	PRINT NFAILURES,"FAILURES"	: -	-	X
:		:		:

The WHILE involves the expression, "I:=1,NSTUDENTS" which controls repetition of the subtable. Rules R1 and R2 carry an 'R' for C1 and require that the control expression be true -- that I is between 1 and NSTUDENTS inclusive. The 'R' entry in for C1 also implies that the subtable be repeated if rule R1 or R2 is selected. In the case of rule R1, NFAILURES is incremented by one and the current student's name, grade, and average are printed prior to repeating the subtable.

The WHILE's control expression directs that variable I be increased by one each time the subtable is repeated. Initially I is set to 1, the control expression's control value - then the subtable is executed, and then it becomes 2, then 3, and so forth, until it exceeds NSTUDENTS at which time the control expression fails and rule R3 applies -- C1 carries an 'N' which requires that the WHILE's control expression be false. Selection of rule R3 causes the number of failures to be displayed and control to exit from the table.

The QUIT entry (Q) is applicable when the decision not to repeat the WHILE's subtable is based upon a condition other than that carried by the WHILE itself. Usage of the QUIT entry is illustrated by the following modification of the preceding example in which the student data is read from a file rather than accessed from an internal array.

Example

		R1	R2	R3
:	:	:	:	:
:	A1 :	NFAILURES: = 0	: X X X :	
:	C1 :	WHILE	: R R Q :	
:	A2 :	READ NAME, GRADE, AVERAGE FROM STUDENT-FILE	: X X X :	
:	C2 :	NAME = '99999999'	: N N Y :	
:	C3 :	GRADE < 70	: Y N - :	
:	A3 :	NFAILURES:=NFAILURES + 1	: :	:
:	A4 :	PRINT NAME, GRADE, AVERAGE	: X - - :	
:	A5 :	PRINT NFAILURES, "FAILURES FOUND"	: - - X :	

We mentioned that the WHILE's conditional control expression is optional; the present example illustrates the case of a "null" WHILE -- a WHILE with no condition or control expression. In such instances the WHILE plays no role in rule selection, but its entries do control repetition of the associated subtable. In the present example, condition C2 which tests for a special end-of-file indicator -- NAME equals eight 9s -- determines whether the subtable is repeated.

Rules R1 and R2 carry an "N" for C2 to insure that actual student data be read rather than the special end-of-data sentinel. For these rules, C1, the WHILE condition, has an 'R' entry directing that the subtable be repeated. Rule R3 is selected when the end-of-data signal is read; it carries a "Q" for the WHILE stipulating the subtable is not to be repeated.

WHILE conditions can be "nested" with the interpretation that the subtable associated with the "outer" WHILE is not repeated until all repetitions of the "inner" WHILE's subtable have been completed.

Example

:	:								:
:	A1	:	MAXZEROES:=0		:	X	X	X	X
:	C1	:	WHILE r :=1, NROWS		:	R	R	R	R
:	A2	:	ZCOUNT:=0		:	X	X	X	-
:	C2	:	WHILE c:=1,NCOLS		:	R	R	N	N
:	C3	:	A(r,c)=0		:	Y	N	-	-
:	C4	:	ZCOUNT>MAXZEROES		:	-	-	Y	N
:	A3	:	ZCOUNT:=ZCOUNT+1		:	X	-	-	-
:	A4	:	MAXZEROES:=ZCOUNT		:	-	-	X	-
:	A5	:	MAXZEROROW:=r		:	-	-	X	-
:	:				:				

The table in this example searches through array A of dimension NROWS by NCOLS for the row containing the greatest number of zeroes. The table utilizes two nested WHILE conditions. The 'outer' WHILE comprises condition C1 and is associated with a subtable composed of clauses C1, A2, C2, C3, C4, A3, A4, and A5 and rules R1 through R5. The inner WHILE (condition C2) and its subtable (formed from clauses C2, C3, C4, A3, A4, and A5 and rules R1 through R4) is completely contained in the sub-table for the outer WHILE -- hence the term "nested".

The outer WHILE iterates through the rows of A using the array using r to index the current row. The next clause of the outer WHILE's subtable initializes to zero ZCOUNT, the number of zeroes in the current row. Then comes the second, inner WHILE (C2) which iterates through elements of row r. As long as column index c, is less than or equal to the NCOLS, number of columns, rules R1 and R2 are selected; ZCOUNT is modified if appropriate; and the inner WHILE is repeated. When c exceeds NCOLS the inner WHILE fails and ZCOUNT is compared with MAXZEROS and if it exceeds the latter, MAXZEROES AND MAXZEROROWS are updated. Control then reverts to the outer WHILE -- rules R3 and R4 carry an N for condition C2, the inner WHILE and an R for C1, the outer WHILE.

The subtable associated with the outer WHILE is repeated until r exceeds NROWS at which rule R5 is selected and control exits from the table. Note that condition C2 carries an '-' for rule R5, thereby precluding execution of the inner WHILE.

4. CONVERSION ALGORITHMS

4.1	INTRODUCTION	4- 1
4.1.1	Background	4- 1
4.1.2	Parsing techniques	4- 2
4.1.3	The remainder of this chapter	4- 4
4.2	TREE GENERATION I: ORDERED PARSES	4- 4
4.2.1	The mechanics	4- 4
4.2.2	Ordered parses	4- 7
4.2.2.1	Condition order	4- 7
4.2.2.2	Rule order	4- 7
4.2.2.3	"Forgetful" parses - parses with limited look-back	4- 7
4.2.3	Sequencing restrictions	4-10
4.3	TREE GENERATION II: MINIMIZING AVERAGE PROCESSING COSTS	4-14
4.3.1	Goal and cost mechanism	4-14
4.3.2	Verification and search costs	4-17
4.3.3	A branch and bound algorithm	4-19
4.3.4	Extension to handle implied limited entries	4-21
4.3.5	Mutually exclusive conditions which exhaust a set of states	4-21
4.3.6	Generalization to rule classes	4-26
4.3.7	More about action set selection	4-34
4.3.8	A single-alternative algorithm	4-37
4.3.9	Rules that are not mutually exclusive	4-37
4.3.10	Search-free tables	4-44
4.4	TREE GENERATION III: MINIMIZING TOTAL STORAGE COSTS	4-47
4.4.1	Goal and costing mechanism	4-47
4.4.2	Comparison with minimizing average processing cost	4-47
4.4.3	Mergeable entries	4-49
4.4.4	Search entries for optimizing storage	4-49
4.4.5	Search-free conditions	4-49
4.4.6	Storage costs for a branch and bound algorithm	4-52
4.4.7	Hoisting actions	4-54
4.5.	INTERPRETIVE MASKING	4-56
4.5.1	The method	4-56
4.5.2	Processing and storage costs for rule masking	4-59

4. CONVERSION ALGORITHMS

4.1 INTRODUCTION

4.1.1 Background

Undoubtedly the primary motivation for inventing and developing decision tables was to provide a way of specifying tasks to be performed by an electronic computer. Therefore, it is not surprising that the idea of inputting decision tables directly to a computer was considered almost from the beginning of interest in decision table methodology.

This interest in the automated processing of decision tables has persisted. In 1970 Herman McDaniel published the book, 'Decision Table Software' which lists 30 different decision table processors. Since then, the number of processors has grown considerably.

Most decision table processors convert programs written in a predominantly tabular format to a free-form format which is then passed to a compiler for compilation. Because they manipulate a program before it is processed by a compiler, these decision table processors are usually referred to as preprocessors.

Normally, these preprocessors produce programs in a high level language - COBOL, in most cases, with a few producing FORTRAN, PL/1, or ALGOL. The user writes this higher level language within his tables and the preprocessor's conversion process can be divided into three phases: collection, composition and parsing. The following paragraphs briefly describe these phases as performed by one decision table preprocessor using the following table as an example:

Stub	Rules
I =	1 2 3
AMOUNT:=	20 25 35

Collection:

The condition and action for rule 2 are assembled below:

Stub	+	Entry	=	Clause
I =		2		I = 2
AMOUNT:=		25		AMOUNT:= 25

Composition:

Programming language statements are composed by embedding the clauses within some standard text (those portions of the following which are underlined).

```
L30101 IF NOT (I=2) GO TO L30202;  

AMOUNT:= 25;  

GO TO L30704;
```

Parsing:

A false-branch is determined for each condition as the point to transfer to when that condition is false. The false-branch usually appears as the destination of a GO TO (L30202 in our example).

A true-branch is found for each condition and action. The true-branch designates the point to which control is transferred when a condition is true or when an action has been executed. Often the true-branch goes to the next sentence of the generated program and does not appear explicitly. When it does appear explicitly, the true-branch is often a transfer to the table's exit, i.e., a 'do-nothing' sentence generated to serve as a junction point at the end of the table (L30704 in our example).

4.1.2 Parsing techniques

One "parses" a decision table by mapping it into a control structure compatible with the test-and-branch logic of today's computers. The resulting control structure usually has one of the two forms shown in Figure 4-1. In one case (b in Figure 4-1) the table (a. in Figure 4-1) is mapped into a series of two-way conditional jumps and yields a tree-like structure whose branches terminate in the selection of a single rule from the table. We shall call this approach 'tree generation'. The second approach (c in Figure 4-1) is to map an input transaction into a vector of Y's and N's. This vector is compared rule by rule with the table to select the rule satisfied by the transaction. This second approach is called the rule-mask technique since it is convenient to represent the transaction vector and the table by a series of masks composed of 1's and 0's.

Many people have been challenged and intrigued by the prospect of mapping a decision table into a control structure, which is optimal in some sense. This interest has fostered much research on decision table parsing and the publication of numerous articles.

.....
 : C1 : Y Y N N :
 : C2 : Y N - - :
 : C3 : - - Y N :

 : A1 : X - X - :
 : A2 : - X X - :

a. Sample decision table.

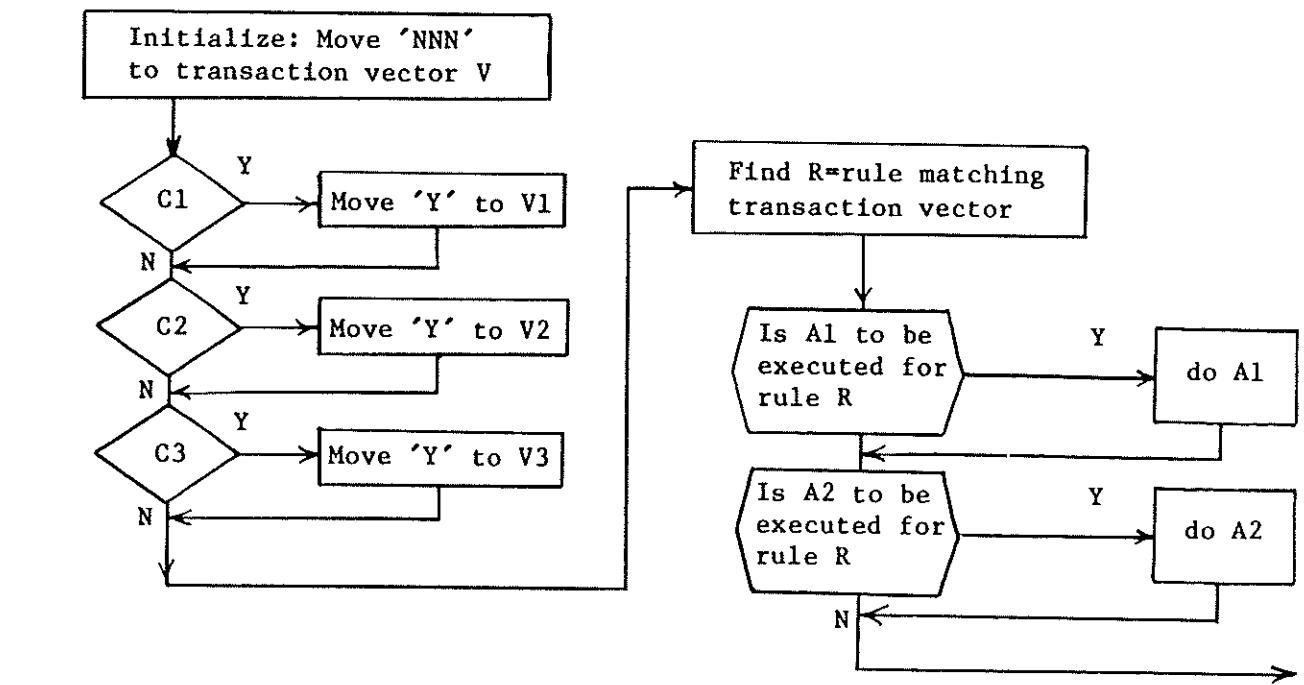
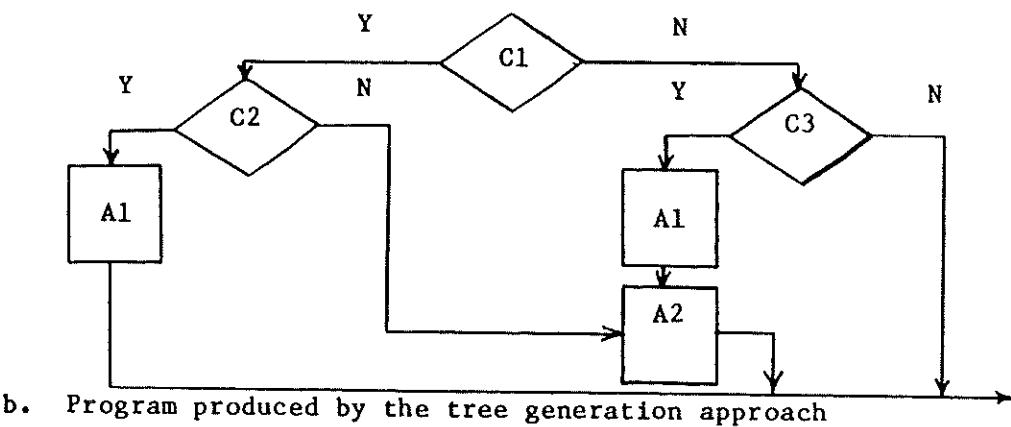


Figure 4-1

4.1.3 The remainder of this chapter

The next three sections of this chapter deal with tree generation. Section 4.2 presents the basic mechanics of tree generation by breaking a decision table into successively smaller tables. It also treats trees which preserve conditions and rule order and deals with clause sequencing requirements. Then follows sections 4.3 on the generation of trees with minimum processing costs, and 4.4 on the generation of trees with minimum storage requirements. The chapter ends with section 4.5 which covers masking techniques.

We shall consider only limited entries and shall assume that extended entries have been converted to limited entries. Initially, we consider only tables which are complete, consistent and contain no implied entries.

4.2 TREE GENERATION I: ORDERED PARSES

4.2.1 The mechanics

Decision tables are not directly compatible with the "test-and-branch" logic of today's sequential computers. What is required is an orderly scheme for converting a decision table into a "program tree", i.e., a series of testing nodes from which "yes" and "no" branches extend to other testing nodes or to a terminus representing the selection of an appropriate rule. Such a scheme shall be called parsing of the table into a program tree.

Basically such a parse proceeds in two steps. First select a condition and test it. Then, divide the original table into two subtables. One subtable will be dependent on the selected condition being true and composed of rules which carry a Y or a - for the selected condition. The other subtable is dependent on the condition being false and contains those rules which carry a N or a - for the selected condition.

As illustrated in Figure 4-2, these two steps are repeated on successive subtables, generating a tree-like structure of tests and subtables. A branch of the tree terminates upon the occurrence of a subtable having only one rule with no relevant conditions -- any condition left in the subtable carries a (-) for the rule.

A branch can be terminated for other reasons; viz., the occurrence of a subtable of one rule but with entries which are not -'s, or the occurrence of a condition subtable with no relevant conditions and more than one rule, e.g., Y, Y, N. Such circumstances indicate that the original table is incomplete and/or that its rules are inconsistent. (Figure 4-3)

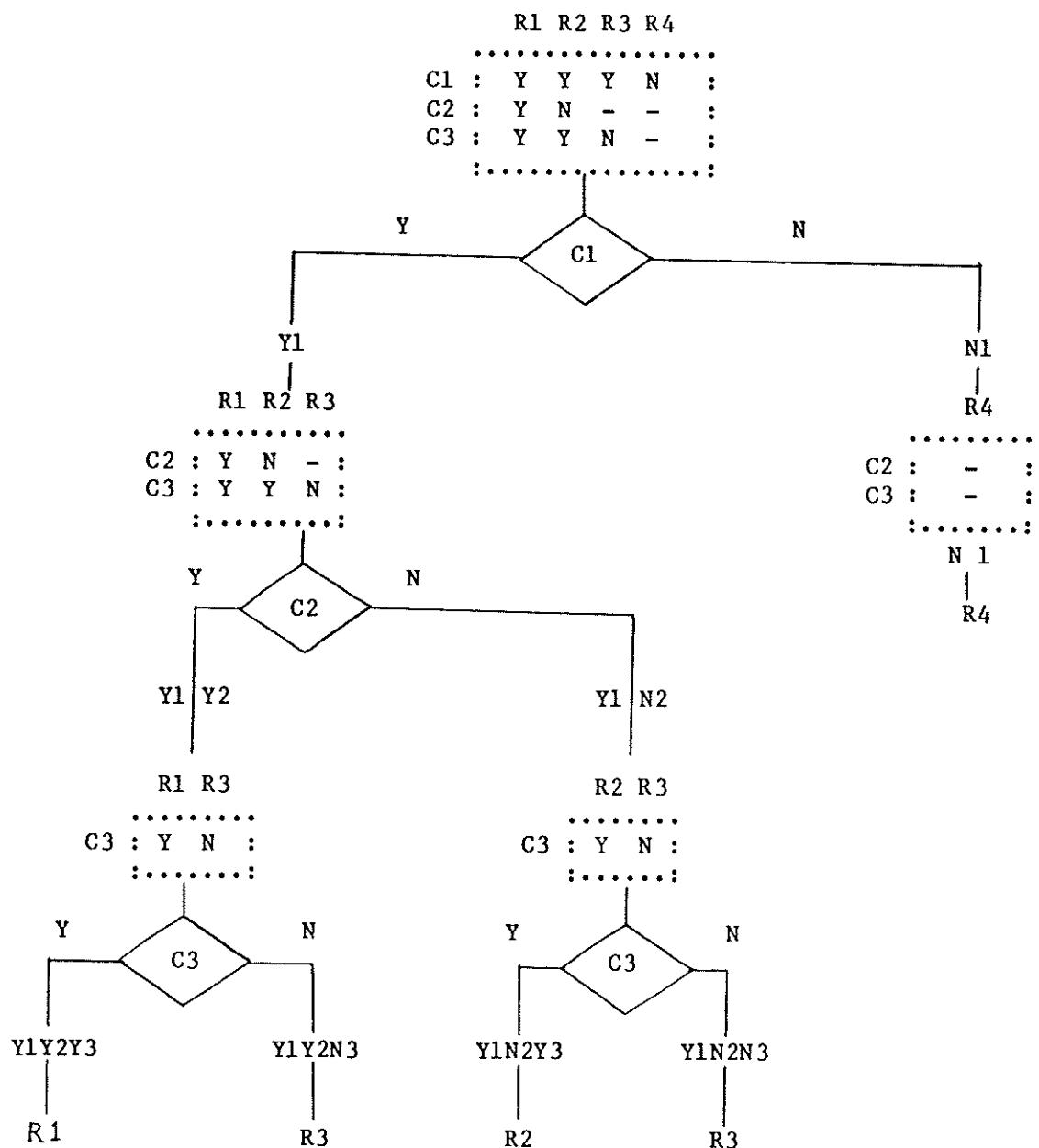


Figure 4-2 Parsing a complete and unambiguous decision table.

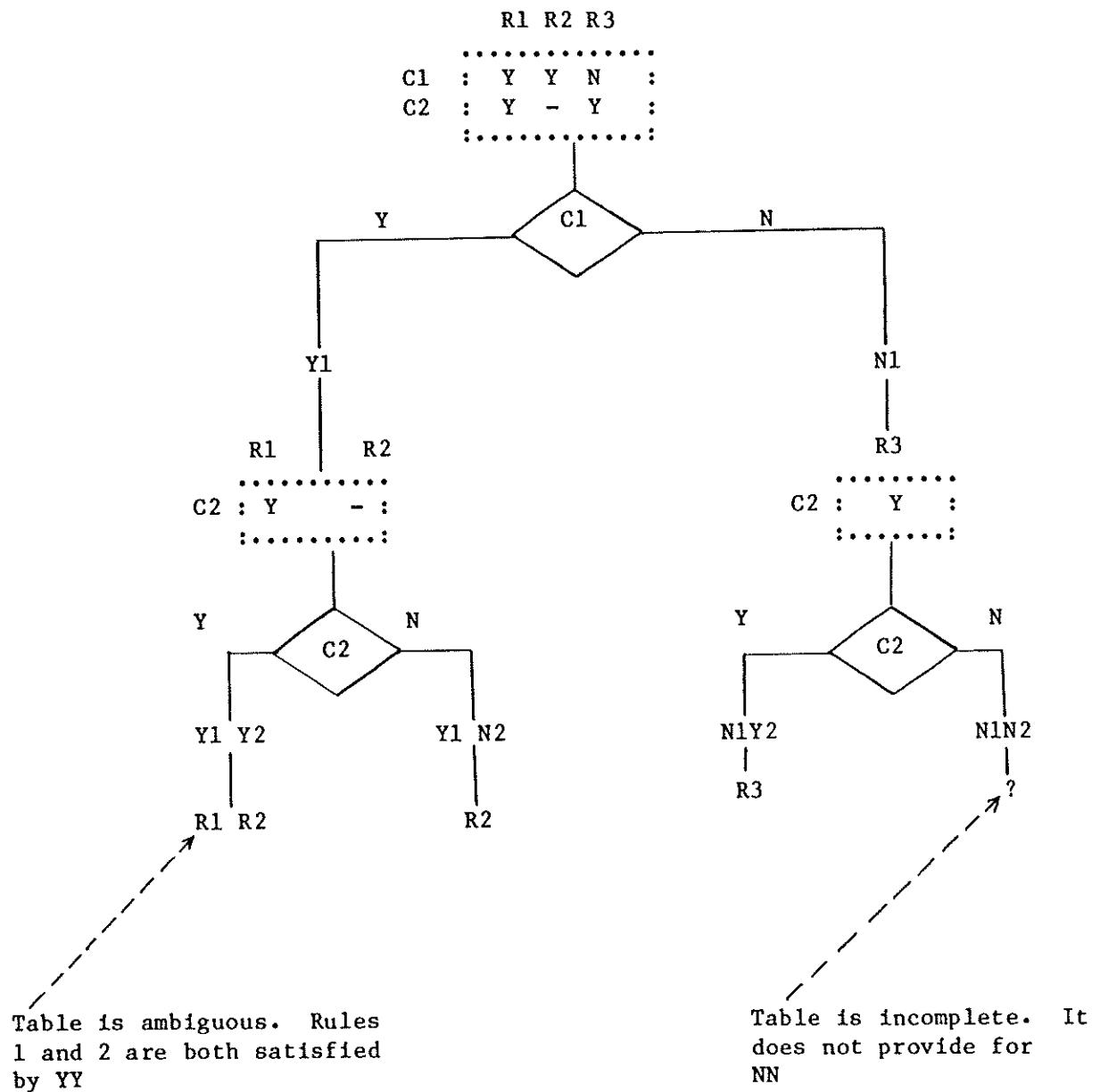


Figure 4-3 Generation of a program tree for a table that is incomplete and contains overlapping rules.

4.2.2 Ordered parses

4.2.2.1 Condition order

Condition order requires that the conditions be tested in the same order in which they appear in the table. Figure 4-4 illustrates a table parse which preserves condition order.

4.2.2.2 Rule order

There is some confusion about what is meant by rule order. A naive interpretation would be "identifying rules in the same order as they appear in the table." However, this is not appropriate, because the order in which rules are identified depends upon the order in which conditions are tested and, even more importantly, upon one's conventions with respect to ordering the branches generated when a condition is tested. For example, the tree in part b of Figure 4-4 has the Y-branch as the left link out of a condition and preserves the rule order in the table in part a of the figure. If one alters the branching conventions and assigns the N-branch as the left link, one reverses the order in which the rules are recognized, as in part c of the figure.

A better definition of rule order is that of testing all conditions that are relevant (i.e., those which do not have not a " - ") for the first rule, then those that are not relevant in the first rule but are relevant in the second rule, then those that are not relevant in the first two rules but are relevant in the third, etc. Figure 4-5 displays a table parsed using this scheme.

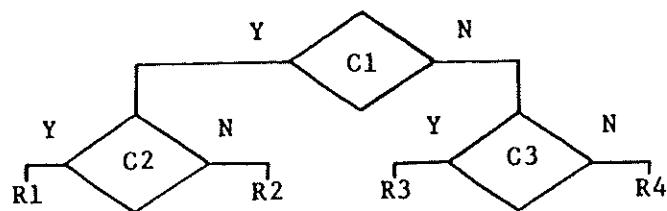
4.2.2.3 "Forgetful" parses -- parses with limited look-back

Most decision table parsing algorithms involve some sort of mechanism for keeping track of (i.e., remembering) those conditions which have already been tested and, thereby, avoid testing them again. "Forgetful" parsing techniques have no such mechanism and frequently produce inefficient parses. Since "forgetful" techniques usually preserve condition and rule order some people have confused the two, and have, as a result, rejected order preserving techniques when they are really objecting to "forgetful" techniques.

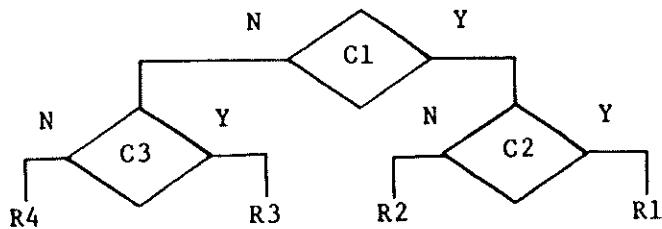
One "forgetful" approach (Figure 4-6) is to make a completely independent series of tests for each rule and to test every condition that is relevant for the rule. A more sophisticated "forgetful" approach tests in rule and condition order and selects a branch based only on the current condition and those which precede it in the current rule. Thus, in Figure 4-7 the first test of C4 is associated with rule R1. When determining which branch to take when C4 is false, one looks for a rule satisfied by Y-YN (i.e., the negation of C4 and the affirmation of C1 and C3 which precede it in R1 and selects R2. Similarly, the first test

	R1	R2	R3	R4
.....
C1 : Y	Y	N	N	:
C2 : Y	N	-	-	:
C3 : -	-	Y	N	:
.....

a. Sample table



b. Condition order parse with YES-branching to the left

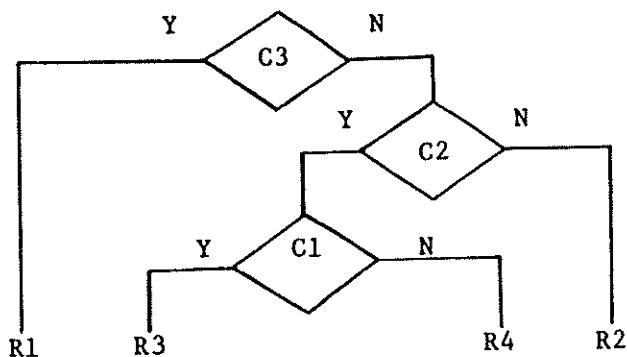


c. Condition order parse with YES-branching to the right

Figure 4-4
Parses which preserve condition order

	R1	R2	R3	R4
.....
C1 :	-	-	Y	N :
C2 :	-	N	Y	Y :
C3 :	Y	N	N	N :
.....

a. Sample table



b. Rule order parse with YES-branching to the left

Figure 4-5
A parse which preserves rule order

of C3 branches to the test of C2 in rule R3 -- there is a match on Y-N. Condition C2 is tested because it was not tested in R1. The test of C3 is then repeated because there is no mechanism for remembering that it was previously tested in R1 and there is nothing preceding C3 in R3 to indicate that it has been tested.

4.2.3 Sequencing restrictions

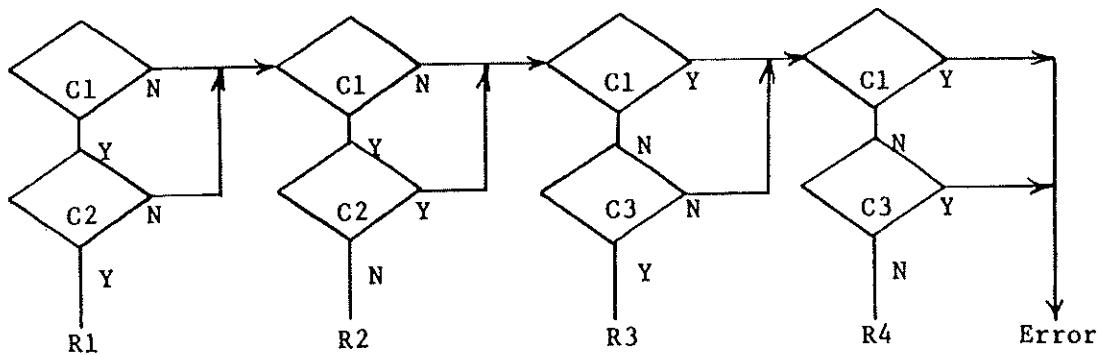
By "sequencing restrictions" we mean requirements that one or more conditions be tested before some other condition is tested. Such restrictions are implicit in the evaluation conventions adopted in 3.7.3. The idea of sequencing restrictions seems inimical to nearly all algorithms which convert decision tables to optimal trees, since the algorithms assume that there is complete freedom with respect to the order in which conditions are tested, and in some cases it is assumed further that actions can be performed in any sequence and that they can be intermixed with conditions.

In general, these assumptions are broader than they need to be. The algorithms look at a decision table (or subtable) and select a condition to be tested. All conditions are considered as potential candidates in this selection procedure. Sequencing restrictions can be accommodated by limiting the selection to candidates which satisfy sequencing requirement. For example, the table in Figure 4-8 implies the restriction that C3 be tested before C4 to avoid a divide error when computing the ratio, pay/hours. Hence the selection of the condition to be tested first is limited to C1, C2 and C3. To be eligible for testing a condition should be movable to the first condition position in the table by applying the clause reordering procedure described in 3.8.2.

In the ensuing discussion, we shall note only those algorithms which can not support a selection procedure to preserve sequencing restrictions. Further, we shall assume that there is complete freedom in the order in which the conditions and actions can be sequenced.

	R1	R2	R3	R4
.....
C1 :	Y :	Y :	N :	N :
C2 :	Y :	N :	- :	- :
C3 :	- :	- :	Y :	N :
.....

a. Sample table

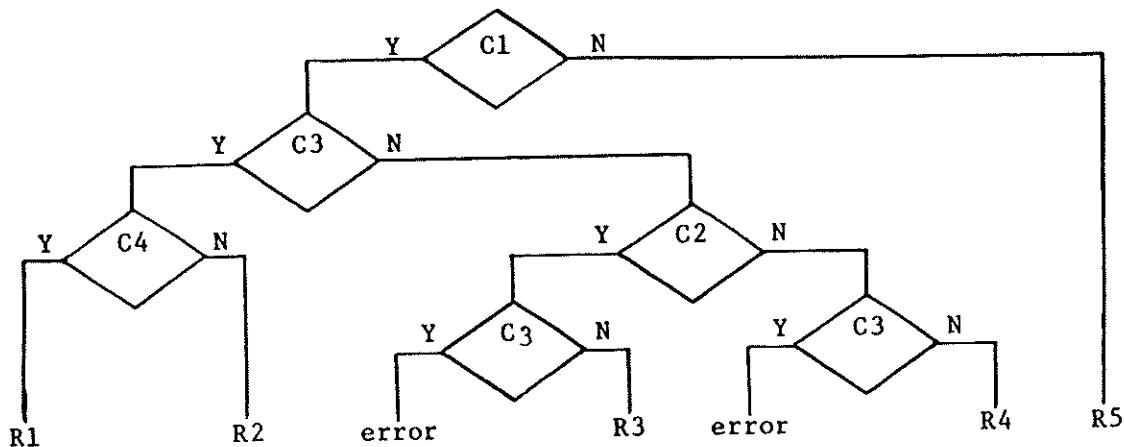


b. "Forgetful" parse

Figure 4-6
A "forgetful" parse which tests each condition for each rule

	R1	R2	R3	R4	R5
C1 :	Y	Y	Y	Y	N :
C2 :	-	-	Y	N	- :
C3 :	Y	Y	N	N	- :
C4 :	Y	N	-	-	- :
	:

a. Sample table



b. "Forgetful" parse"

Figure 4-7
 A "forgetful" parse which uses the current condition and
 those which precede it in the current rule

	R1	R2	R3	R4	R5	
C1 : Pay > 100	:	Y	Y	N	N	N :
C2 : Jobcode = 5	:	-	-	Y	N	- :
C3 : Hours = 0	:	Y	Y	N	N	- :
C4 : Pay/hours > 2	:	Y	N	-	-	- :

Figure 4-8
A sample table containing a clause sequencing restriction

4.3 TREE GENERATION II: MINIMIZING AVERAGE PROCESSING COSTS

4.3.1 Goal and costing mechanism

Minimization of average processing costs entails minimizing the sum $(PrTr)$ where Pr is the probability that rule r will be satisfied by a transaction, and Tr is the time (i.e., cost) required to establish that rule r is satisfied by the transaction, and r ranges over all rules of the table. Implicit in this interest is (1) a probability for each rule reflecting the frequency with which the rule will be satisfied by transactions processed against the table, (2) a testing cost for each condition, and (3) condition probabilities, P_i , which represent the probability that condition C_i is true given a '-' for C_i as a rule entry. (Shortly we shall see how these condition probabilities are used and later we shall discuss a mechanism through which their significance can be diminished and/or completely eliminated.)

As illustrated in Figure 4-9 the rule probabilities are arrayed across the top of the table, while the testing costs and condition probabilities are listed to the left of the table.

The paragraph before last presents a definition of average processing costs as a function of rule probabilities and time requirements for testing conditions. It is often difficult or laborious to obtain these quantities, a circumstance which has led some people to seek a definition of average processing costs which does not involve rule probabilities and/or testing time. It is difficult to see how such a redefinition can be valid since "average" usually connotes mean (or more precisely, arithmetic mean) which, in turn, implies the use of probabilities (expressed perhaps as rule frequencies), and 'processing costs' are usually measured in terms of CPU time. In most cases such 'redefinitions' usually involve some assumption about rule probabilities and/or testing costs. For example, it could be assumed that the rules are equally likely, each having a probability of $1/n$ where n = the number of rules, and that condition testing costs are identical, with each test requiring one unit of CPU time. Another assumption frequently employed is that the conditions are independent and each has a 0.5 probability of being true.

For our purpose it is convenient to carry the condition costs, condition probabilities and rule probabilities along in the subtables produced during the parse (Figure 4-10). With one exception these values are identical to those in the original table. The exception is the probabilities associated with rules which carry a '-' for the selected condition. It is at this point that P_i 's (condition probabilities) come into play. If condition C_i is selected as a test and rule r has a probability Pr and a '-' for C_i , then rule r has the probability PrP_i in the subtable dependent on the C_i being true and $Pr(1-P_i)$ in the subtable dependent on C_i being false. In Figure 4-10, see the subtables resulting from the test of C_2 .

Testing Cost		Condition Probabilities									
		Ti	Pi	.40	.30	.20	.10	:	Rule Probabilities		
				R1	R2	R3	R4	:	Rules		
		C1	: 4	: .8	: Y	: Y	: Y	: N	:		
		Conditions	C2	: 2	: .7	: Y	: N	-	-	:	
			C3	: 3	: .4	: Y	: Y	: N	-	:	

Figure 4-9

A decision table which includes testing costs (Ti) as well as condition probabilities (Pi) and rule probabilities (Pr)

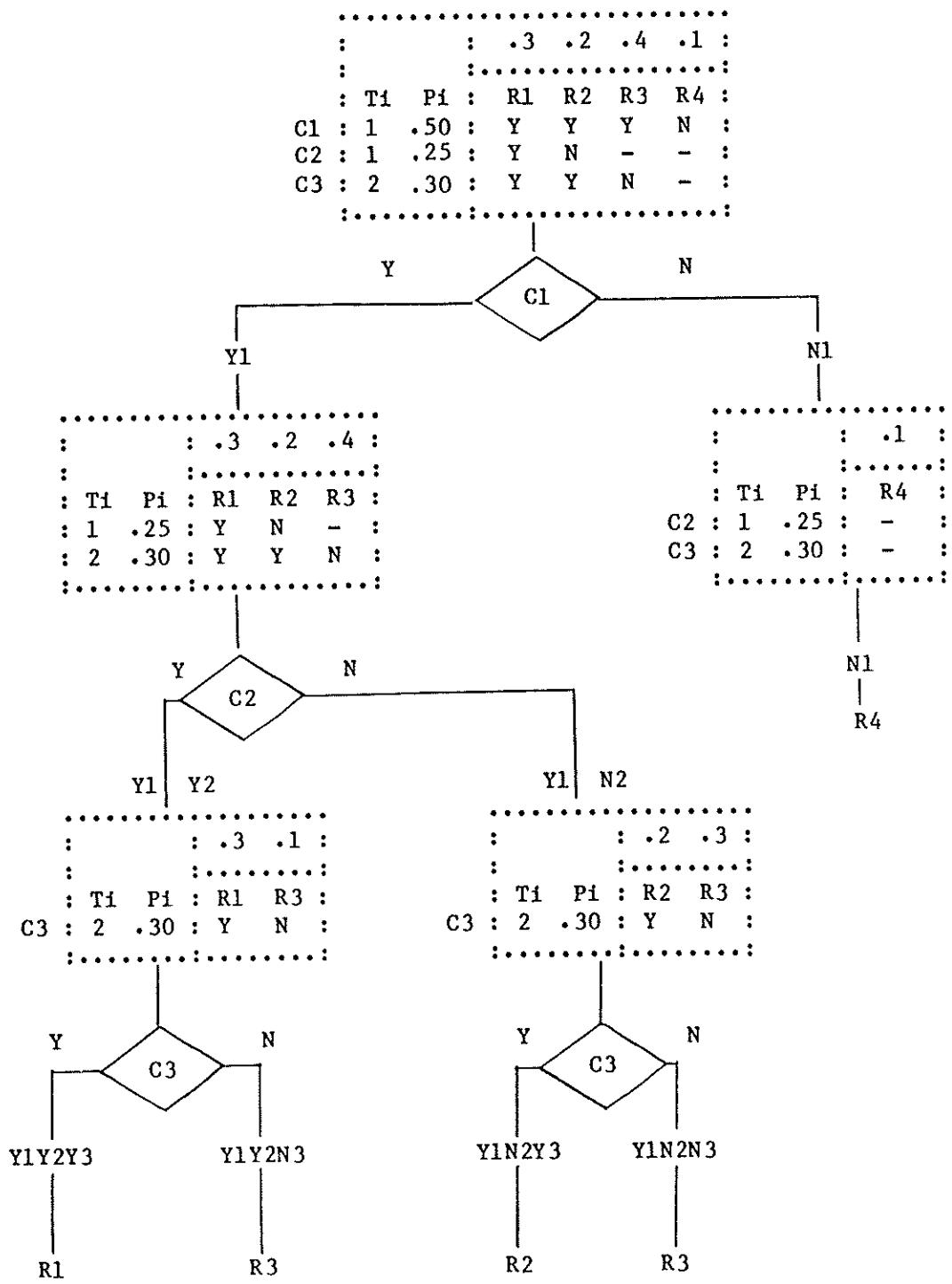


Figure 4-10
Parsing a complete and unambiguous decision table

4.3.2 Verification and search costs

At this point is is convenient to digress briefly and consider a somewhat different problem. Suppose we have a series of records, each of which contains a transaction relative to a decision table and also an identification (i.e. number) of the rule satisfied by the transaction. Further, suppose that these records were generated by another computer installation. Having had some unfortunate experiences previously, we are skeptical and therefore decide to write for each rule a verification program. Our processing scheme is as follows:

1. Input a record.
2. Obtain the rule number from the record.
3. Use the rule number to jump to the appropriate verification program.
4. Do something drastic if the transaction does not satisfy the ascribed rule. (What is done is not germane to the present discussion, since this "should never happen".)

Figure 4-11 shows a sample decision table and the appropriate verification programs (Progs. A, B, C, D, E). The important thing to observe is that the only conditions tested in these programs are those corresponding to Y and N entries in the table.

When one tries to parse the table shown in Figure 4-11 into a program tree, complications arise. Let's assume we decided to test condition 3 first. This test would be applied to all transactions processed against the table; in the case of satisfying rules 2 through 5 there is no question as to the value of testing condition 3, since programs B through E in part c of Figure 4-11 show that it must be applied in verifying these rules. Hence, for such transactions execution of this test can be considered a verification. However, when a transaction satisfying rule 1 is encountered, we are not so fortunate. Test 3 fails to yield any verification information and in this case testing of condition 3 can be considered a search to select the proper rule x.

Clearly, any computer program must execute all verifications present in a table. For example, condition 1 must be tested for all input satisfying rule 1. Consequently the objective in finding an efficient program for a table (i.e., one which minimizes average processing time) is to reduce the time spent in the execution of searches.

a. Decision Table

:	:	.30	.30	.20	.10	.10	:	
:	:	:	
:	T1	P1	R1	R2	R3	R4	R5	:
:	10	.5	Y	N	-	Y	N	:
C1	10	.5	Y	-	N	N	Y	:
C2	10	.5	-	Y	N	Y	N	:
C3	20	.5	-	Y	N	Y	N	:
:	:

B. Task: To write 5 programs where:

- Program A verifies that C1, C2, C3 satisfy Rule Number 1
- Program B verifies that C1, C2, C3 satisfy Rule Number 2
- Program C verifies that C1, C2, C3 satisfy Rule Number 3
- Program D verifies that C1, C2, C3 satisfy Rule Number 4
- Program E verifies that C1, C2, C3 satisfy Rule Number 5

c. Solution: Five Verification Programs

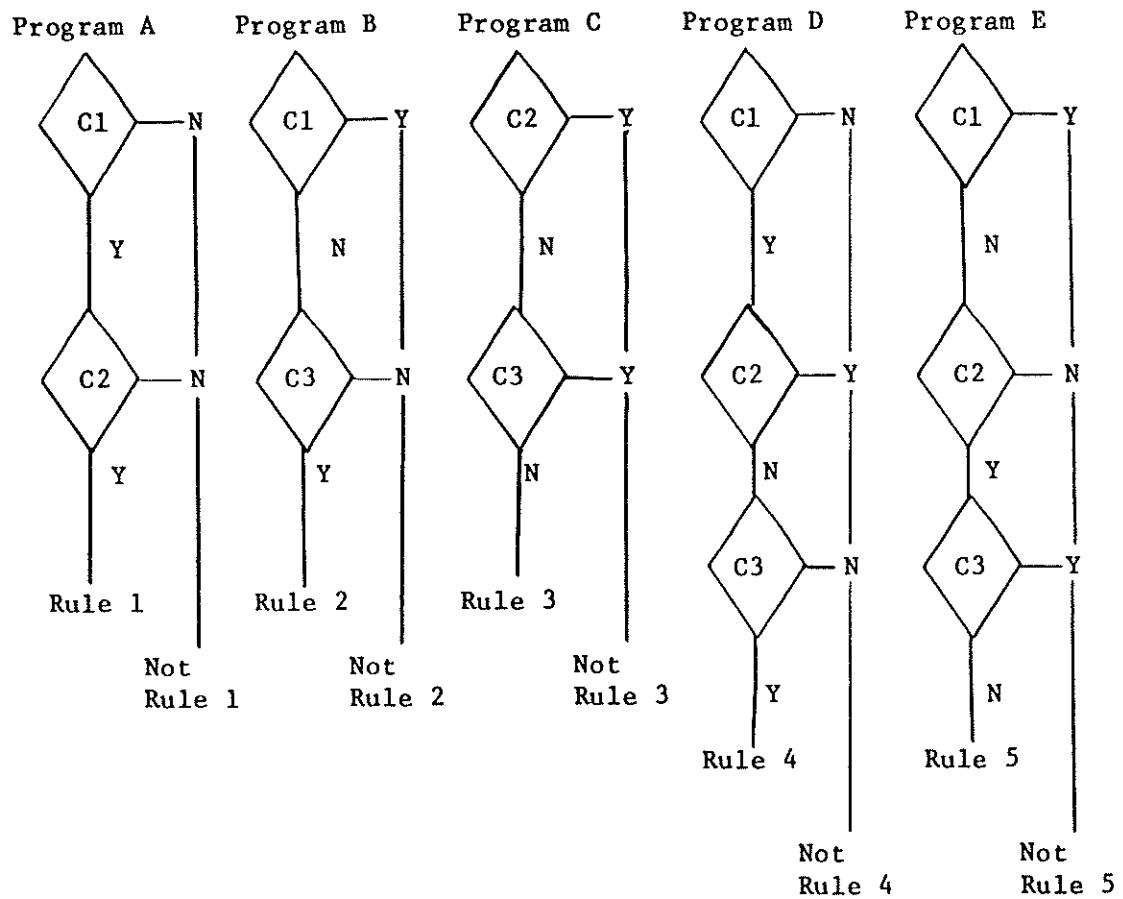


Figure 4-11
Input Verification Problem

Given a cost for testing each condition, and a probability for each rule, we can compute for each condition an (expected) verification cost and an (expected) search cost. As illustrated by part a. in the upper right of Figure 4-11 for condition i:

1. Search cost (S_i) = sum of the (probabilities of rules containing a - for condition i) times the cost of testing condition i, (T_i).
2. Verification cost (V_i) = sum of the (probabilities of rules containing a Y or an N for condition i) times the cost of testing condition i, (T_i).
3. Verification costs (V_i) + search costs (S_i) = cost of testing, (T_i), since the two probabilities sum to 1.0.

4.3.3 A branch and bound algorithm

In selecting C1 as the first condition to be tested, a search cost of 2 is incurred. There is no way of directly recovering this expenditure since the search costs of the other tests are not reduced (note the observation in part b of Figure 4-12 that testing C1 first does not alter the search costs associated with C2 and C3). Thus 2 is a lower bound on the search cost of all computer programs which test C1 first. In our selection process the choice of testing C1 first is a branch which has a bound of 2. Similarly, 3 is a lower bound on the search cost of all programs which start by testing C2 and 6 is a lower bound on all which start with C3.

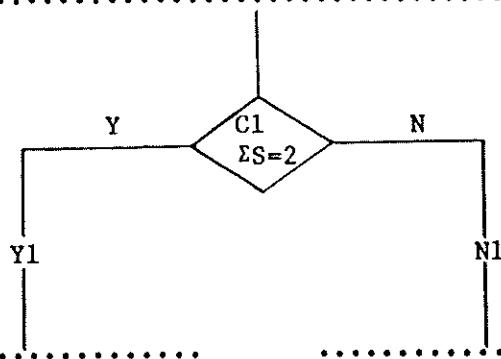
Parsing of the subtables can add additional search costs, thereby increasing the lower bound. The present example is fairly trivial and as shown in part c of Figure 4-12, we are able to complete the parsing without incurring additional search costs.* However, had we chosen to test C3 in the subtable dependent on C1=Y, we would have incurred an additional search cost of 6. Previous testing of C1 involved a cost of 2. Thus there is a lower bound of $2 + 6 = 8$ on all programs which start by testing C1 and then test C3 in the subtable dependent on C1=Y.

The implications of the preceding paragraph are fairly clear when one graphs the lower bound on total search cost against steps in a parsing

* The diamond shaped boxes in Figure 4-12, representing condition tests, include an expression of the form ' 'S=...' near the bottom. This expression is the accumulated search cost for all condition tests through the one represented by the box. For the first condition test in Figure 4-12, we have ' S=2', because a search cost of 2 is associated with the testing of C1. Tests following the first all have a search cost of zero and hence, all remaining boxes carry ' S=2'. The tree in Figure 4-14 illustrates the situation in which the second and succeeding tests have non-zero search costs.

a. Costing Procedure

$: Ti : Pi :$.30	.30	.20	.10	.10	Search Cost (S_i)	Verification Cost (V_i)
$C1 : 10 : .7 : Y$	N	-	Y	N	:	2	8
$C2 : 10 : .5 : Y$	-	N	N	Y	:	3	7
$C3 : 20 : .5 : -$	Y	N	Y	N	:	6	14



b. Search Cost After Testing C1

Observe: Testing C1 first does not alter total search costs associated with C2 and C3.

$: Ti : Pi : .30$.14	.10:	$: Ti : Pi : .30$.06	.10:	(sum of S_i from both subtables)
$C2:10 : .5 : Y$	N	N : 0	$C2:10 : .5 : -$	N	Y : 3	$C2 0+3 = 3$
$C3:20 : .5 : -$	N	Y : 6	$C3:20 : .5 : Y$	N	N : 0	$C3 6+0 = 6$

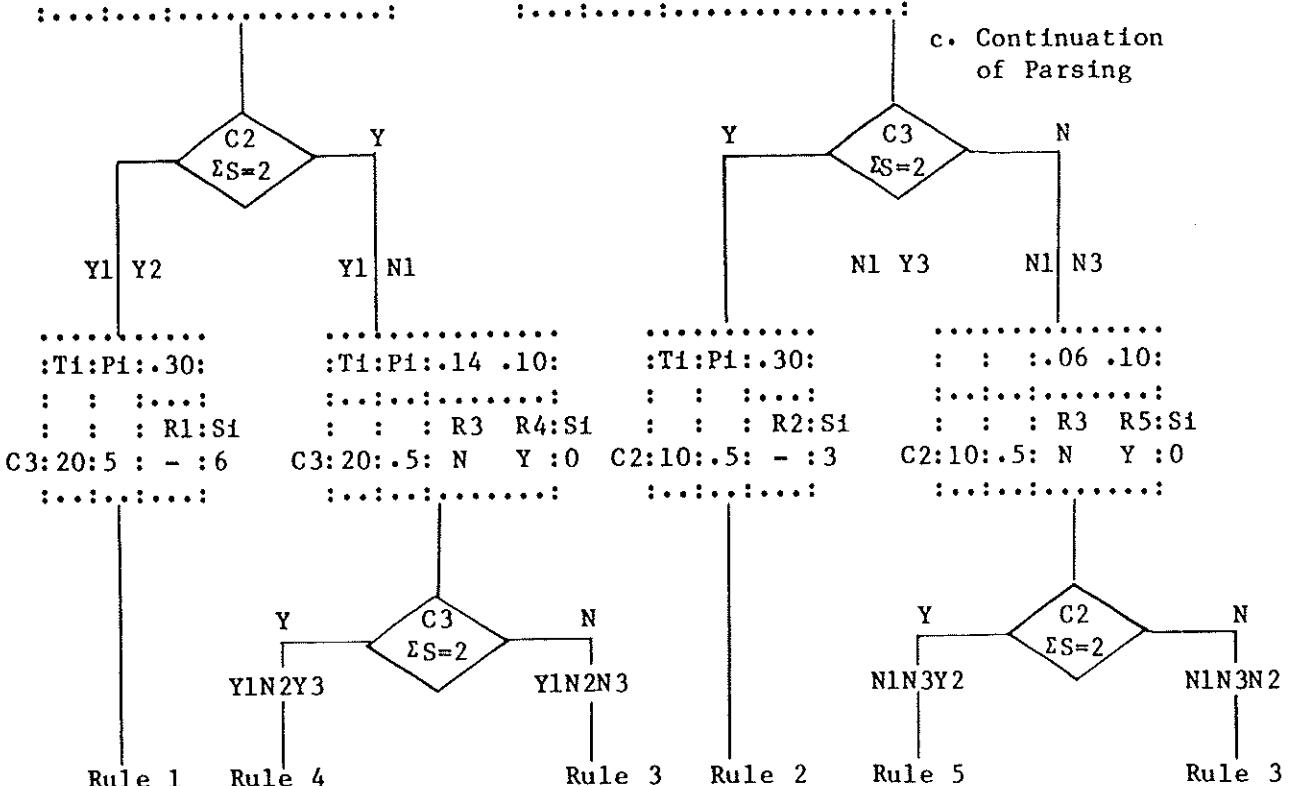


Figure 4-12

scheme. Figure 4-13 shows such a graph for the present example. Initially (step 0) there are three choices: test C1, test C2 or test C3. This takes us to step 1 where the smallest lower bound corresponds to testing C1 first. Looking at the subtable conditional on C1=Y we can test C2 or C3. C2 is best since it has a search cost of zero. For the subtable conditional on C1=N we again have to choose C2 or C3. Here C3, with a search cost of zero, yields the smallest lower bound. Thus a decision table can be parsed into a computer program with the minimum average test cost through the use of such a graph by consistently selecting the best extension from the lowest terminal point on the graph.

Here it is desirable to clarify a point which may have confused some investigators in the field of decision tables. The ordering of rules in a limited entry decision table has no effect on the parsing of that table into a timewise efficient program. The generation of such a program is dependent solely on search costs. A search cost is simply a testing cost multiplied by a sum of rule probabilities and, by the commutative law of addition, this sum is independent of the order of the terms of which it is composed.

4.3.4 Extension to handle implied limited entries

The branch and bound algorithm which has just been described can be extended easily to handle tables with dependent conditions. As illustrated in Figure 4-14, the dependent entries, "Y!" and "N!" are grouped along with the "-" entries as contributing to search costs. However, they behave as ordinary Y and N entries in dividing a table into sub-tables. There is no splitting such as is associated with "-" entries', i.e., rules carrying a 'Y!' are allocated solely to the yes-branch subtable and rules having a "N!" are allocated completely to the no-branch subtable. Note that the decision table shown in Figure 4-14 does not contain a column of condition probabilities since the table contains no "-" entries and these probabilities are not used.

The program generated in Figure 4-14 has minimum average processing cost and is a binary search, just as it should be. Any other program produced from the table will have greater search costs. The presence of dependent condition entries is instrumental in producing this anticipated result. Other versions of the table (Figure 4-15) do not yield a binary search as an optimal tree.

4.3.5 Mutually exclusive conditions which exhaust a set of states

The optimal program tree for the decision table in part a of Figure 4-16 is shown as part b of the Figure. Note that the tree first tests for $X > 0$, a condition which is not present in the original table but which must be true when, as in the table's last rule, both $X < 0$ and $X = 0$ are false. One can make this substitution because the conditions $X < 0$, $X = 0$,

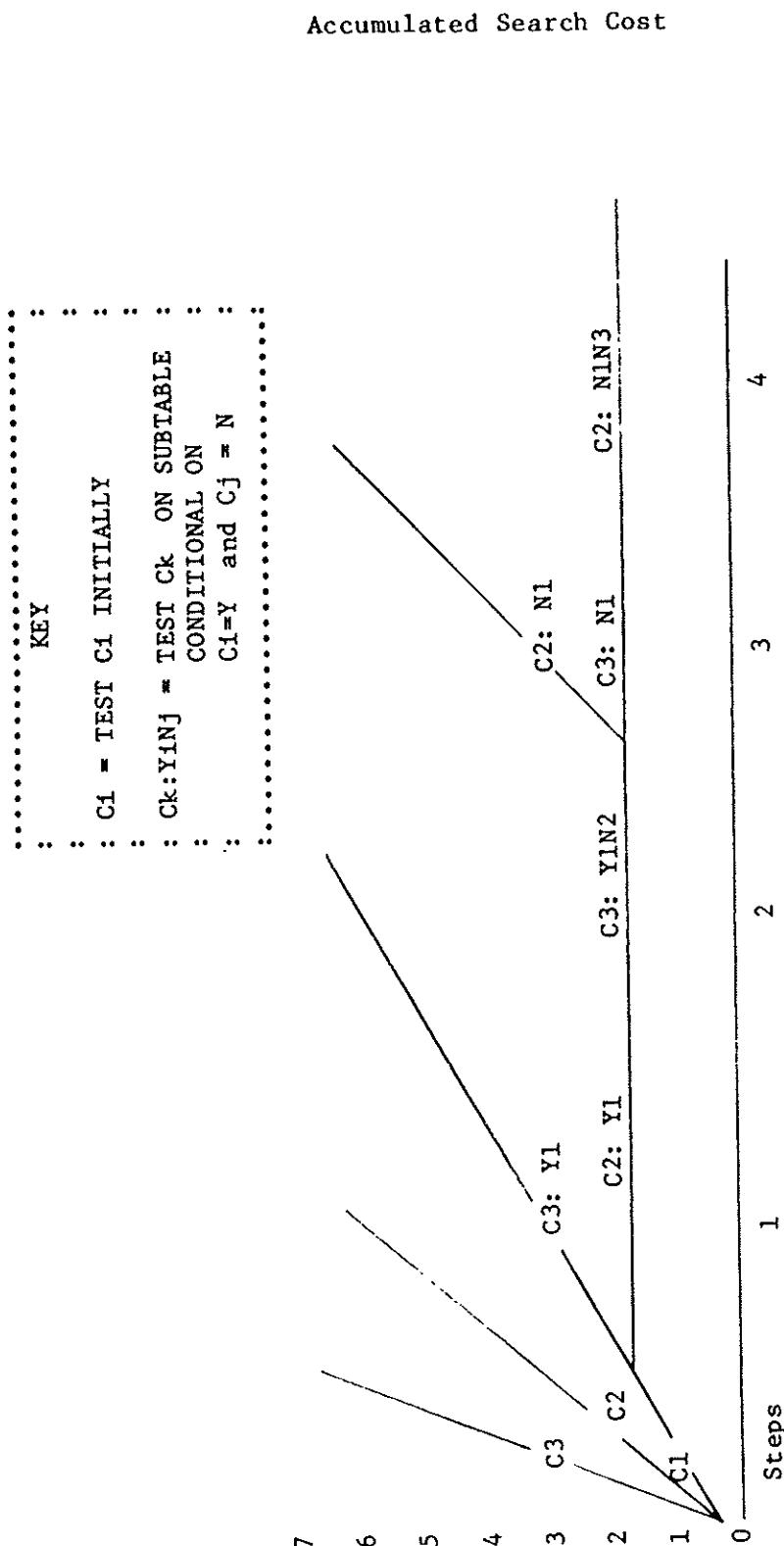


Figure 4-13

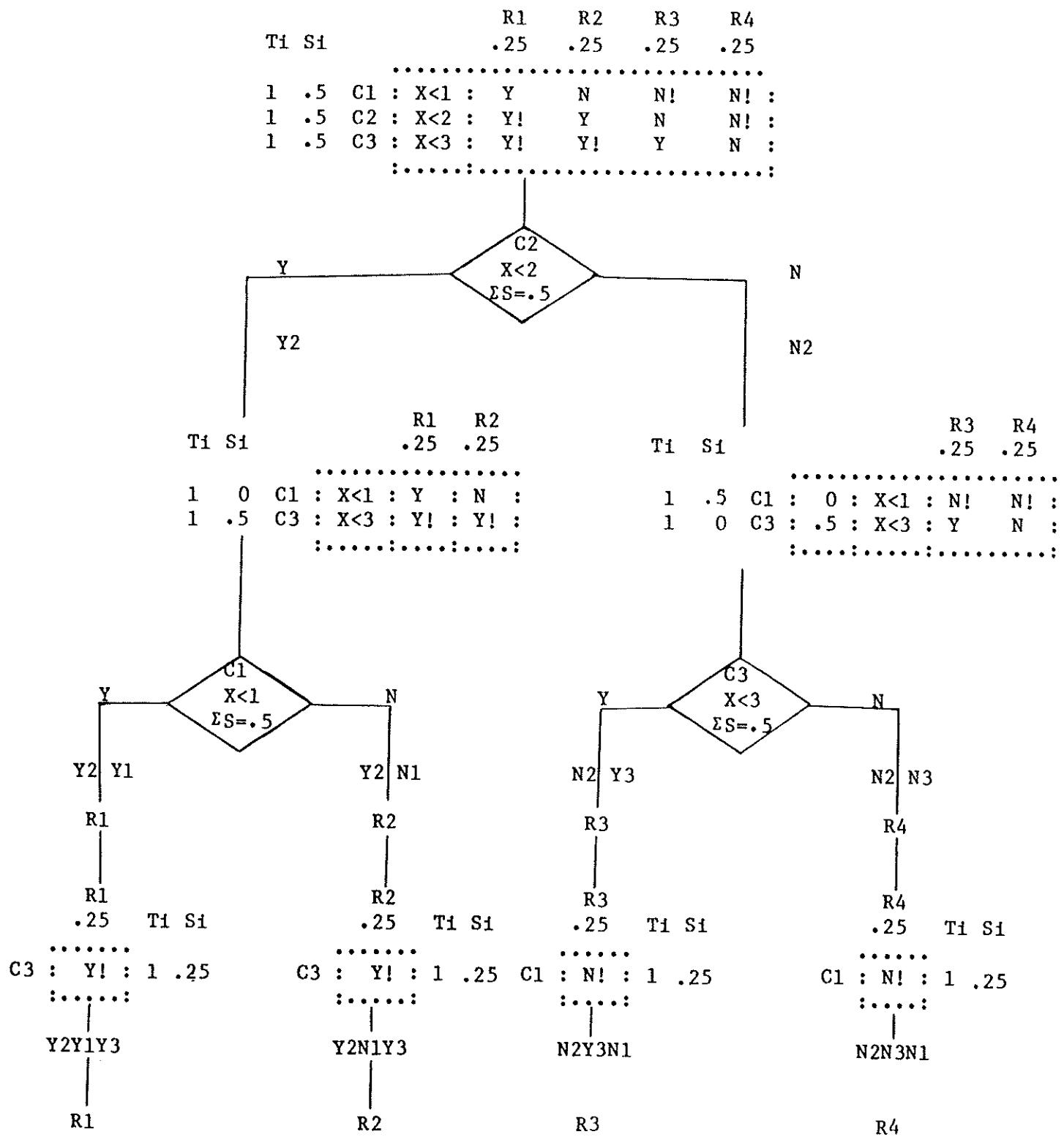


Figure 4-14

a.

Ti	Si		R1 .25	R2 .25	R3 .25	R4 .25
1	.5	C1 : X < 1 : Y	N	N	N	:
1	.5	C2 : X < 2 : -	Y	N	N	:
1	.5	C3 : X < 3 : -	-	Y	N	:
		

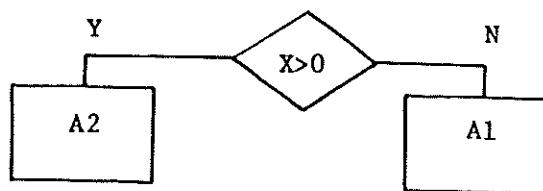
b.

Ti	Si		R1 .25	R2 .25	R3 .25	R4 .25
1	.5	C1 : X < 1 : Y	-	-	-	:
1	.5	C2 : X < 2 : -	Y	-	-	:
1	.5	C3 : X < 3 : -	-	-	Y	-
		

Figure 4-15
Alternate versions of the table shown in Figure 4-14

:	:	:	.1	.2	.3	:	
:	:	:	:	
:	Ti	:		R1	R2	R3	:
C1	: 1	: X < 0	:	Y	N!	N	:
C2	: 1	: X = 0	:	N!	Y	N	:
			:	

a. A decision table



b. An optimal program for the decision table in a.

Figure 4-16

and $X>0$ form an exhaustive set of mutually exclusive states. The set is exhaustive because one of the states $X<0$, $X=0$ and $X>0$ must be true. The states $X<0$, $X=0$ and $X>0$ are mutually exclusive because one and only one of them can be true.

To conform strictly with our branch and bound algorithm, one should represent an exhaustive set of mutually exclusive states by conditions which carry only dependent entries as in the table in part a of Figure 4.17 (note we assumed that by the time the table is entered, it has been determined that Season can have only one of the four legitimate values: 'WINTER', 'SPRING', 'SUMMER' or 'AUTUMN'). The rationale for this is that one can establish the truth of any one of the conditions by showing all the others to be false. The drawback to this approach is that there is no criterion for selecting the best condition to test first, because all four conditions have 1.0 as a search cost, S_i . As shown in part b of Figure 4-17, the natural way to write this table is with a 'Y' instead of a 'Y!' for entries when a condition is required to be true. But this approach violates our earlier notion that verification costs remain constant. Thus, one can avoid the verification associated with the last step of the trees (i.e., the verification Season = 'WINTER'), since the only possible outcome is yes, thereby implying action A4.

It appears that we can tolerate this deviation because testing a member of a set of exhaustive conditions always diminishes the search costs for other members of the set by an amount equal to the verification cost for the condition tested. Thus, in the original table in part b. of Figure 4-17, the second condition Season = 'SUMMER' has a verification cost V_1 of 0.4. This condition is tested in node 1 of the generated tree. The verification cost for this condition equals 0.4 and is associated with the conditions in the yes-branch subtable. This subtable avoids tests of the three remaining conditions because they carry $N!$ entries and are known to be false. Each condition in the yes-branch table carries a search cost of 0.4, which is equal to the verification cost associated with testing Season = 'SUMMER', and can be saved (i.e., avoided) because the conditions need not be tested.

4.3.6 Generalization to rule classes

The purpose of processing a transaction against a decision table and of selecting an appropriate rule is to execute the actions associated with that rule. In practice, the selection of a rule, r , is of secondary importance. The primary concern is that the actions appropriate to rule r be executed. It is not infrequent that two or more rules in a decision table belong to the same rule class because they imply the same actions. In this presentation we are not interested in the specific actions implied by a rule but rather in those rules which imply identical actions, i.e., have the same action set. Consequently, we shall group together those rules implying the same action set and in the last row of the table abstractly indicate the action sets as $A_1, A_2\dots$. The resulting

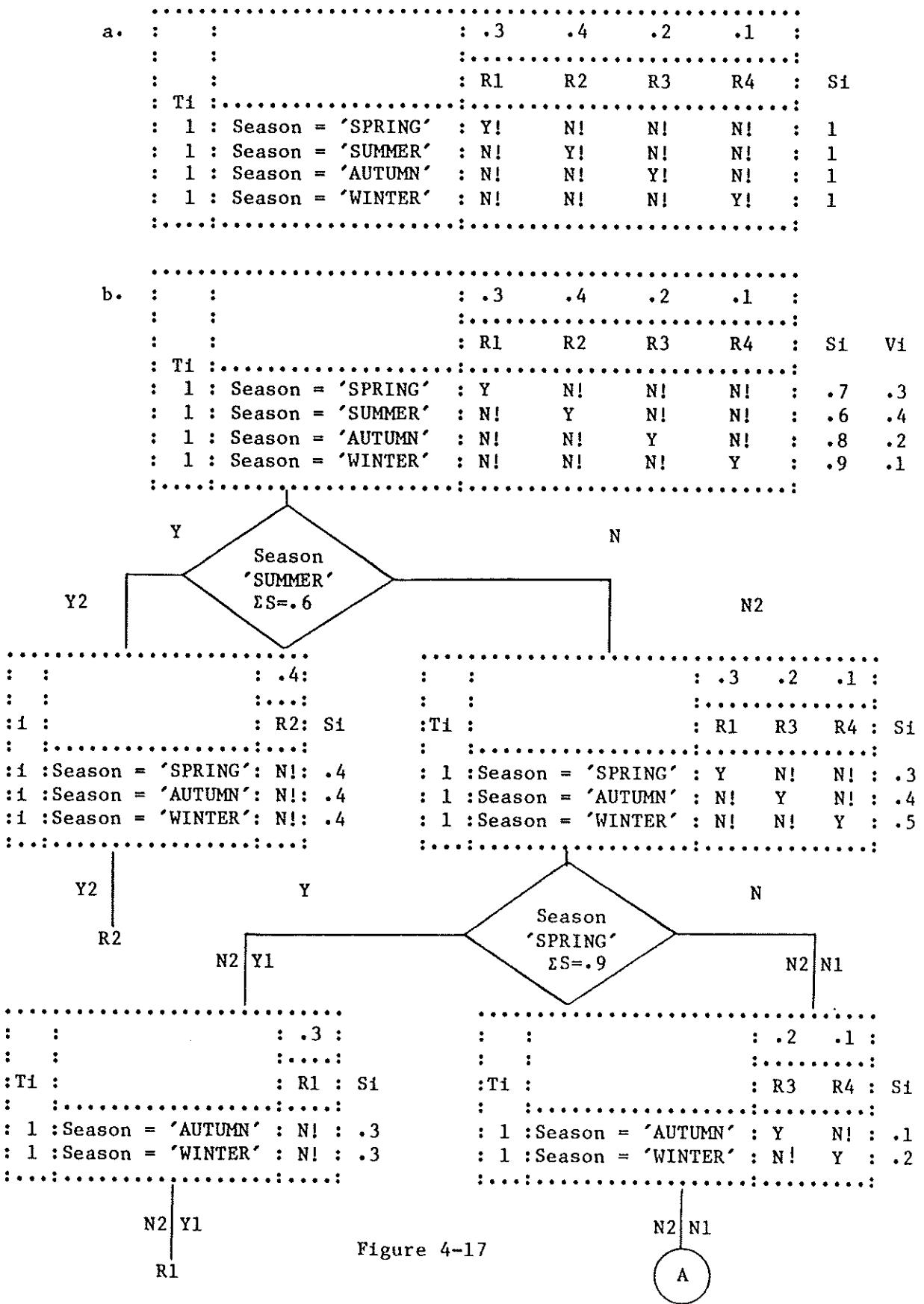


Figure 4-17

Continued on
Next Page

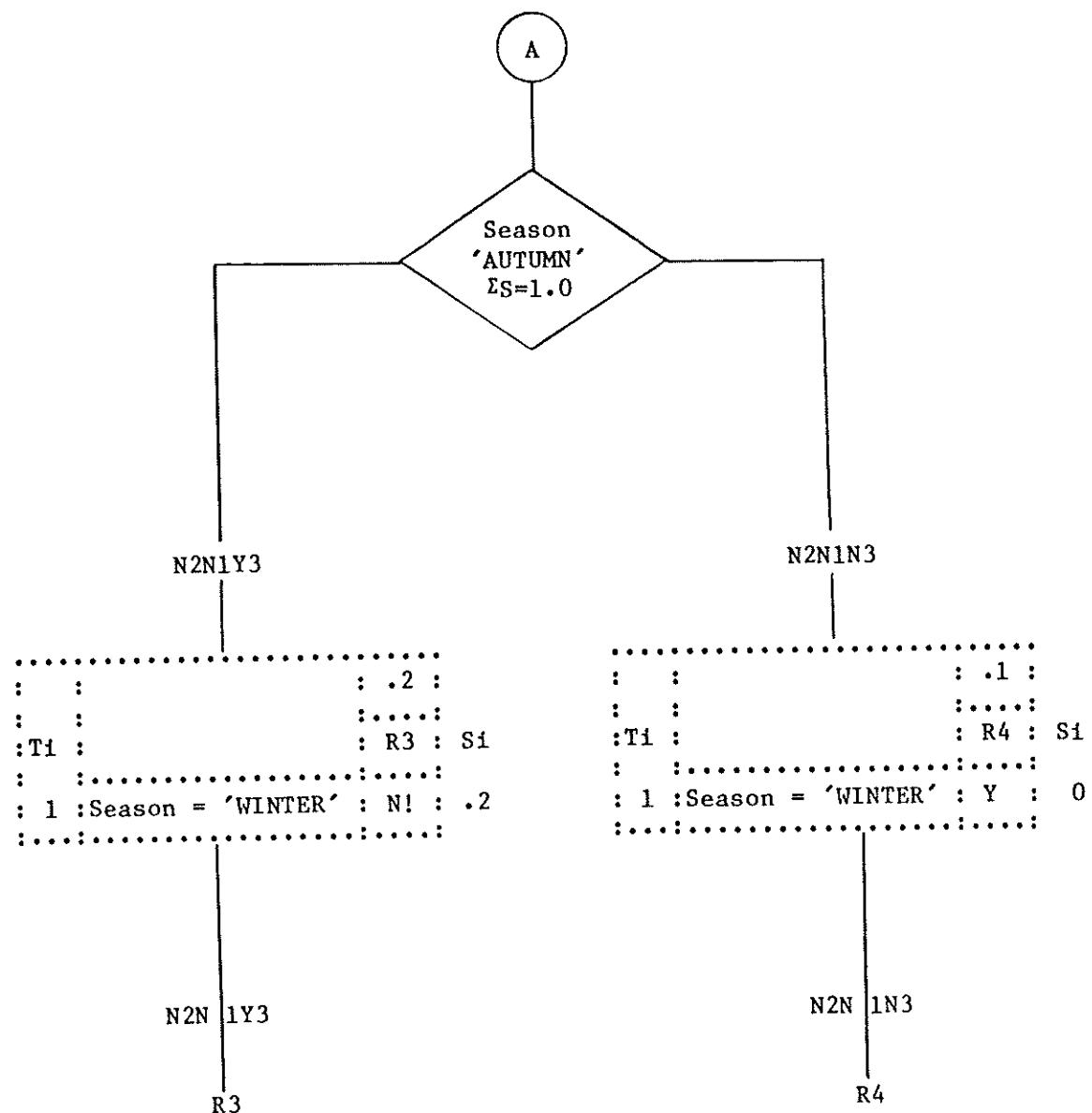


Figure 4-17 (Cont.)

format is illustrated in Table a. of Figure 4-18. Now our goal has changed somewhat. Rather than optimally selecting a rule we now wish to select a rule class optimally.

A given rule class may not have a unique representation in terms of the rules of a decision table. For example, in a. of Figure 4-18, action set A1 may be represented as

(1)	(2)	(3)
C1 Y N	C1 - N Y	C1 Y Y N N
C2 Y -	C2 Y N Y	C2 Y Y Y N
C3 - Y	C3 Y Y N	C3 Y N Y Y

This lack of a unique representation for rule classes precludes the use of -'s as a criterion for identifying search costs. For example, the -'s in (1) above will contribute to (i.e., increment) the search costs ascribed to conditions C2 and C3 but not to C1. In contrast, the -'s in (2) contribute to the search costs of C1, but not to those of C2 and C3; and (3) contains no -'s and hence fails to contribute to the search cost of any condition.

Thus we must rely on something other than -'s in our computation of search costs. Although, as we shall see later (section 4.3.7), it is feasible to compute search costs for the conditions of a table containing -'s and action sets (e.g., a. in Figure 4-18), it is easier from an expository standpoint to consider a table which contains no -'s. Such a table can always be constructed by breaking rules with a - for condition Ci into two rules -- one containing a Y for Ci and the other containing a N for Ci. In Figure 4-18, table b. was derived from table a. through this process.

We shall call such a table a fully expanded or canonical table since each rule corresponds to a point in the table's condition space. As an aside, observe that the conditional probabilities (P_i 's) are deleted from the table since they are used only in connection with the -'s in a table. In general one can dispense with P_i for condition Ci by making all entries for Ci, a Y or N.

As a vehicle for identifying search and verification entries present in a fully expanded table, it is convenient to introduce the concept of the complement of rule r with respect to (condition) Ci. In a fully expanded limited entry table we shall define rule s to be the complement of rule r with respect to Ci if rules r and s:

- a) have opposite entries for Ci
- and b) have identical entries for all conditions other than Ci.

a. Multiple Rule Action Sets

```

.....:..30   .30 : .20   .10 : .10 :
: Ti   P1 :.....:.....:.....:
:       : R1    R2 : R3    R4 : R5 :
C1 : 10  .5 : Y    N : -    Y : N :
C2 : 10  .5 : Y    - : N    N : Y :
C3 : 20  .5 : -    Y : N    Y : N :
.....:.....:.....:.....:.....:
:       : A1    : A2    : A3 :
.....:.....:.....:.....:.....: Action Sets

```

b. Fully Expanded Table

```

.....:..15   .15   .15   .15 : .10   .10   .10 : .10 :
: Ti :.....:.....:.....:.....:.....:.....:.....:
:       : R1    R2    R3    R4: R5    R6    R7 : R8 :
C1 : 10 : Y    Y    N     N : Y    N    Y : N :
C2 : 10 : Y    Y    Y     N : N    N    N : Y :
C3 : 20 : Y    N    Y     Y : N    N    Y : N :
.....:.....:.....:.....:.....:.....:.....:.....:
:       : A1    : A2    : A3 :
.....:.....:.....:.....:.....:

```

c. Costing Procedure

										Verifi-
										cation
:	:.15	.15	.15	.15 :	.10	.10	.10	.10 :	Search	Cost
:	Ti:::::::	Cost	(S _i)
:	:	R1	R2	R3	R4:	R5	R6	R7 :	R8 :	(V _i)
C1	: 10 :	(Y)	Y	(N)	N :	(Y)	(N)	Y :	N :	5
C2	: 10 :	Y	Y	(Y)	(N):	N	N	N :	Y :	3
C3	: 20 :	(Y)	(N)	Y	Y :	(N)	N	(Y) :	N :	10
:::::::::	10
:	:	A1	:	A2	:	A3	::::
::::::::::

Figure 4-18

Entry i of rule r is a search if the complement of rule r with respect to C_i has the same action set as rule r . Entry i of rule r is a verification if the complement of r with respect to C_i has an action set different from that of rule r . For example, in table b. of Figure 4-18 the complement of rule R_1 with respect to C_3 is R_2 . Both R_1 and R_2 imply action set A_1 hence entry 3 of R_1 is a search. We can immediately observe that entry 3 of R_2 is also a search. In contrast, entry 2 of rule R_1 is a verification since the complement of R_1 with respect to C_2 is R_7 and the action set for R_7 is A_2 rather than A_1 .

It should be fairly obvious that search entries come in pairs -- if entry i of rule r is a search, then entry i of the rule that is the complement of rule r with respect to C_i is also a search. These two rules could be combined by making the i th entry a -. Thus our definition of a search is just another way of specifying those entries in a fully expanded table which could "disappear" into -'s by an appropriate merger of the rules comprising an action set.

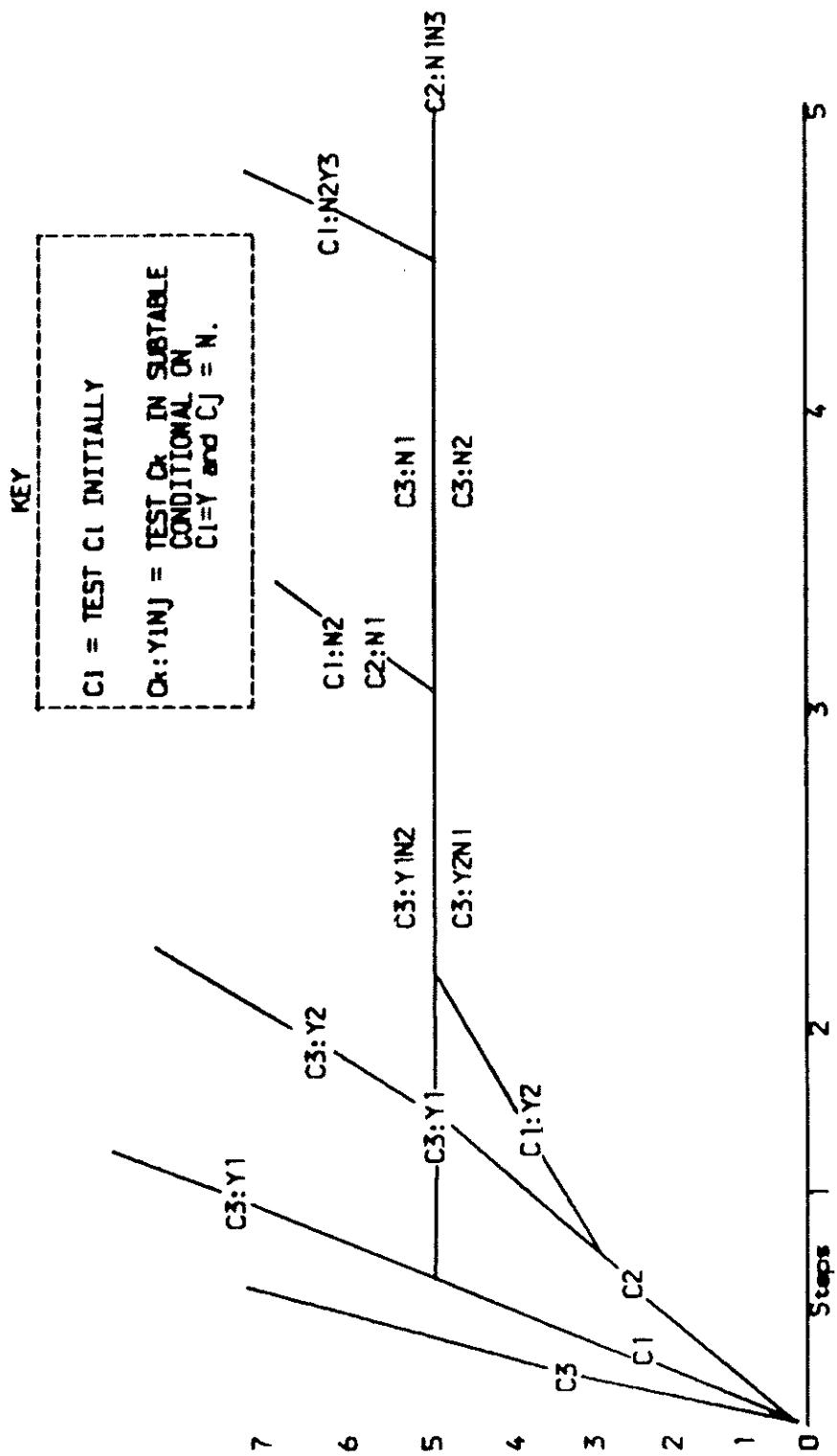
Verification and search costs for the present example are computed in table c. of Figure 4-18. Some entries in the table appear in parentheses to distinguish them as searches. The search cost for condition C_i is the testing cost for C_i multiplied by the sum of the probabilities of those rules for which entry i is a search (i.e., enclosed in parentheses). Analogously, the verification cost for C_i is the testing cost for C_i multiplied by the sum of the probabilities of those rules for which entry i is a verification (i.e., not enclosed in parentheses).

The generalities developed in Section 4.3.3 still hold; specifically:

- a) Testing condition C_i first does not alter the search cost for any other condition.
- b) The search cost for condition C_i is a lower bound on the total search cost of all parsing procedures which start with C_i .

The second of these generalities allows us to apply the procedure of section 4.3.3 and develop Figure 4-19 in which lower bounds are plotted against parsing steps. As may be seen from Figure 4-19, the program having the lowest total search cost begins by testing C_1 first and has a total search cost of 5. The criterion for stopping the parse is no longer the occurrence of a subtable with only one rule but rather the occurrence of a subtable with only one action set. The parse for generating this program is shown in Figure 4-20. The resulting computer program has an average testing cost of 27 since one must augment the total search cost of 5 by the total verification cost of $(5 + 7 + 10) = 22$ obtained from table c. in Figure 4-18.

As a final item, let us compare the current section with section 4.3.3 above. Both sections deal with the same decision table. However, the computer programs parsed from this table must perform different tasks. In section 4.3.3, the program must be able to assign a transaction to individual rule in the table. In the current section, the desired program has a less difficult task. That is, it must be able to assign a



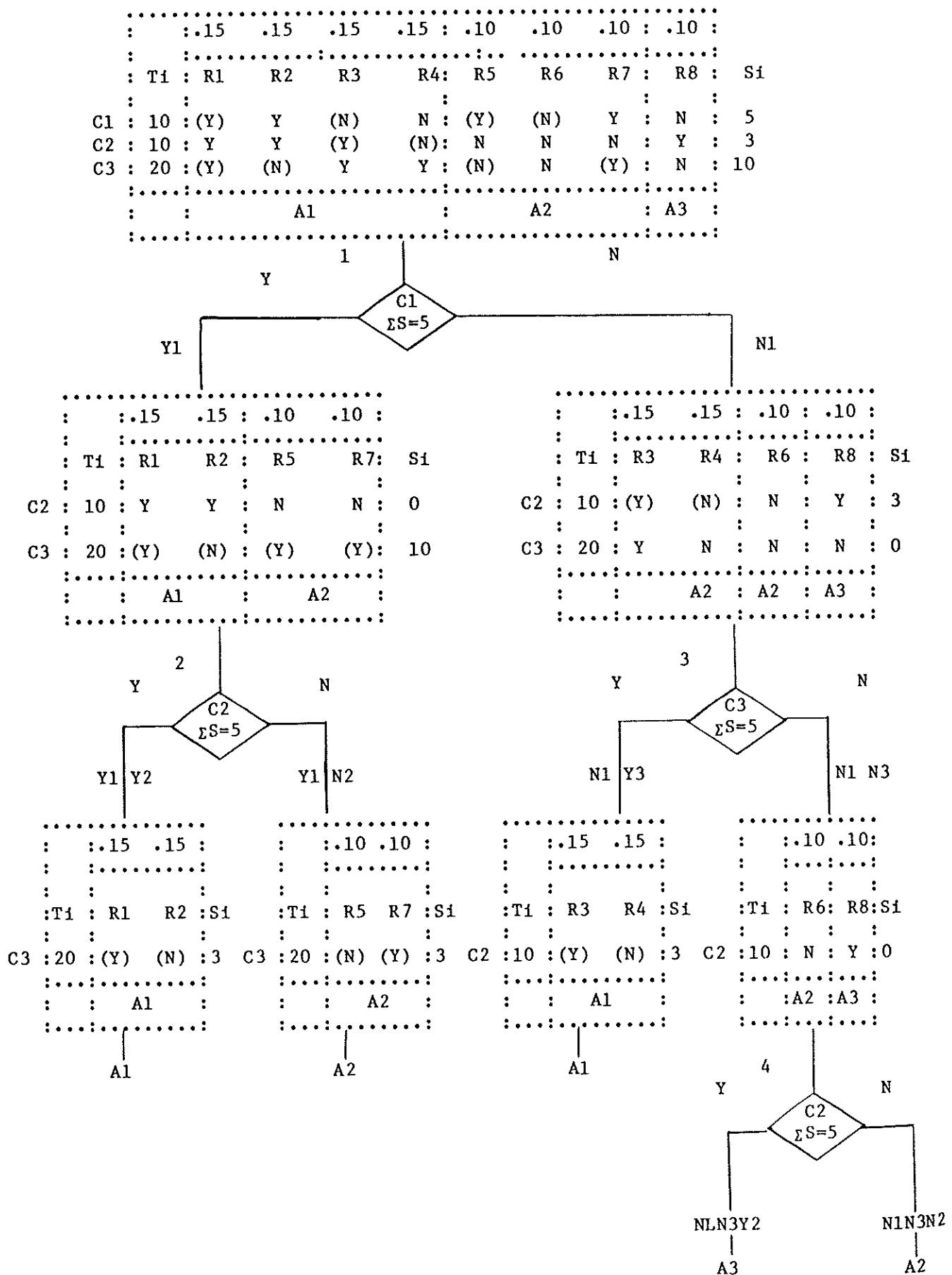


Figure 4-20

transaction to a group of rules implying identical actions, but not to an individual rule within the group. It is reasonable to anticipate that the latter task is less costly to accomplish. The results obtained here support this conclusion. The program constructed in section 4.3.3 to select an individual rule has an expected testing cost of 31, whereas the current section's program to select a group of rules has an expected testing cost of 27.

4.3.7 More about action set selection

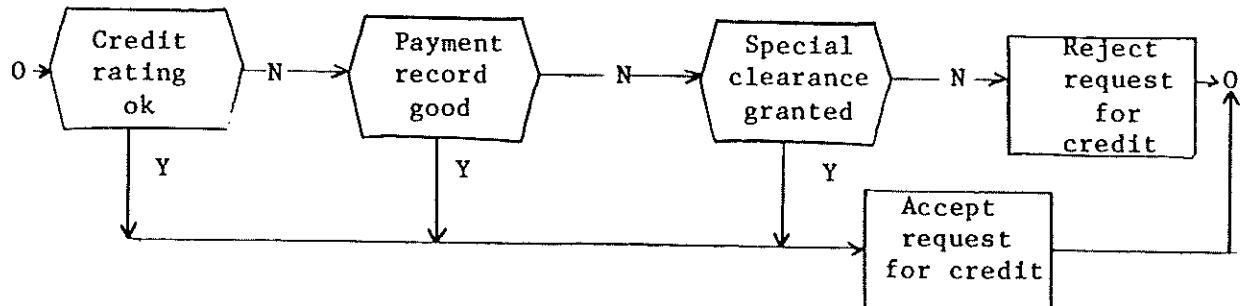
The generation of trees which optimally select action sets rather than rules has received little consideration. Many people think that there are relatively few instances of multiple-rule action sets. It has been found, however, that they do occur fairly often. Figure 4-21 presents a table which has been circulated very widely as a classic example of a decision table. This table contains a multi-rule action set. Specifically, the first three rules of this table all have the same action set which comprises the single action "Accept the request for credit."

The flowchart in part b. of the figure is generally considered to be optimal for the table and the derivation of this flowchart from the table is obvious and poses no problems. But if one looks at the fully expanded form of the table, the selection of the best flowchart is anything but obvious. Each condition has six search entries and selection of a condition to test first must rely on additional information on testing costs and rule probabilities. The flowchart in part b. of the figure is based on the tacit assumption that checking for a good credit rating costs less than checking for a favorable payment record which in turn costs less than obtaining special clearance. One can think of cases in which these relationships do not hold. Take the case of Mom and Pop's Corner Grocery Store. Here obtaining special clearance is the cheapest since it entails asking Pop, who is usually in the back of the store; whereas testing for a favorable payment record takes several minutes, because it involves checking the store's books; and checking for a good credit rating is the most costly, since it requires monetary outlays for a phone call and a rating look-up by the local credit bureau. In this circumstance the flowchart appearing in part d. of Figure 4-21 is best.

When generating trees to select action sets one can use dependent entries in the same fashion as in selecting rules (Figure 4-22). Also one need not work with a fully expanded table but can use a table augmented on the right by pseudo rules, as illustrated in Figure 4-23. These rules are used in costing conditions but are ignored on determining the completion of a branch of a program tree.

.....
 : Is credit rating ok : Y N N N :
 : Is payment record good : - Y N N :
 : Is special clearance granted : - - Y N :
:
 : Accept request for credit : X X X - :
 : Reject request for credit : - - - X :

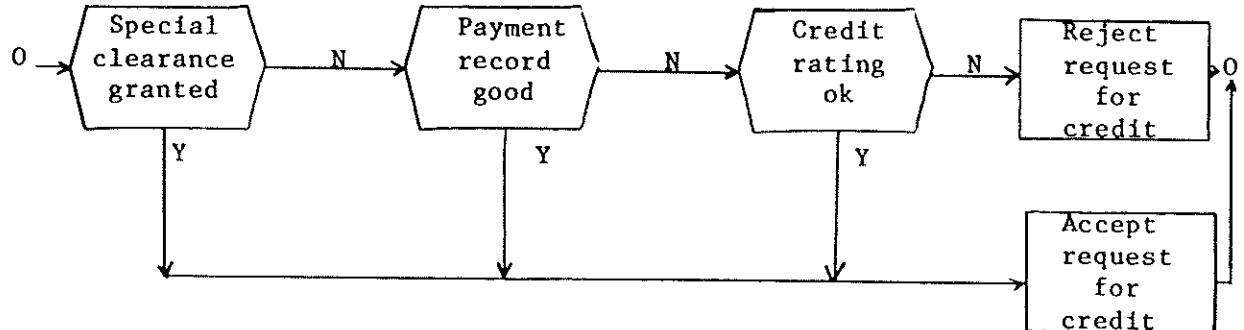
a. A widely used example of a decision table



b. The program tree normally associated with the table in part a.

.....
 : Is credit rating ok : (Y) (Y) (Y) Y (N) (N) (N) N :
 : Is payment record good : (Y) (Y) (N) (N) (Y) Y (N) N :
 : Is special clearance granted : (Y) (N) (Y) (N) (Y) (N) Y N :
:
 : Accepted request for credit : X X X X X X X X - :
 : Reject request for credit : - - - - - - - - X :

c. The true situation



d. Program tree for Mom and Pop's Corner Grocery

Figure 4-21

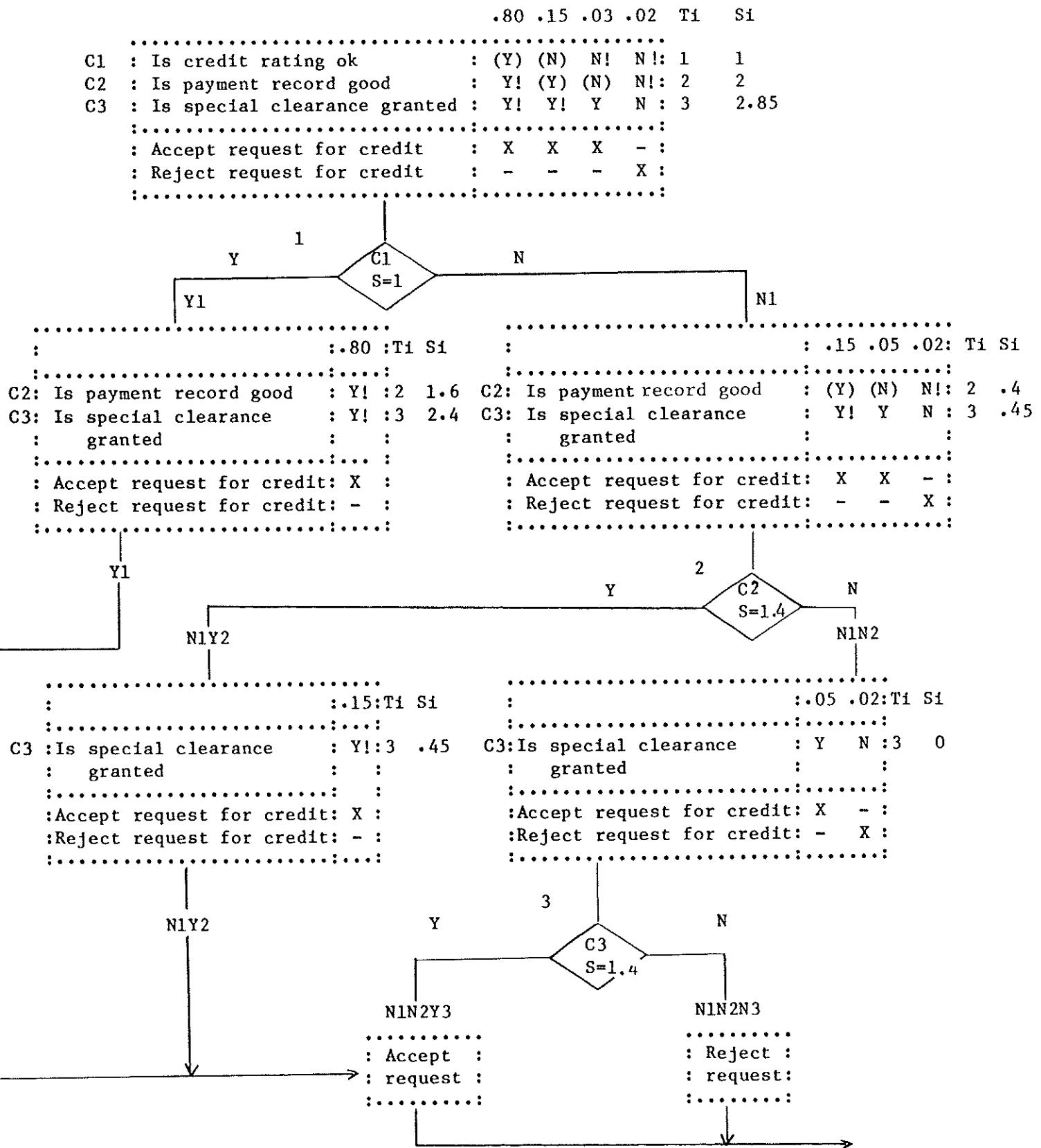


Figure 4-22
 Parsing tables with both action sets and dependent entries

4.3.8 A single-alternative algorithm

There has been some concern about the complexity and speed of the branch and bound algorithm. The branch and bound algorithm appears to have the potential of being slow because it may build and discard a number of trees before selecting the best one. (The number of trees considered is but a very small fraction of that claimed by some of the algorithm's critics.) In the case of the table in Figure 4-23, the most natural approach is to select C2, the condition with the smallest search cost (S_1), and construct a tree which tests it initially. Once having done this, one hypothesizes that the tree which tests C1 first might have a chance of yielding a lower search cost. When such a tree is built one discovers that, indeed, the tree which tests C1 first does have a smaller average processing cost than the best tree in which C2 is initially tested.

One may characterize the branch and bound algorithm as a multiple alternative approach, because more than one tree may be considered at each step in the algorithm. One can speed things up by using a single-alternative approach in which only one tree is considered at each step. The most obvious way of doing this is to always select, as in Figure 4-24, the condition with the smallest search cost (S_1).

Comparison of Figures 4-20 and 4-24, however, shows that our single-alternative algorithm did not find the best tree. The tree generated in Figure 4-24 has a total search cost of 8, sixty percent greater than the total search cost of 5 for the tree in Figure 4-20.

4.3.9 Rules that are not mutually exclusive

The previously mentioned techniques for parsing a decision table have required that the table's rules be mutually exclusive, i.e., any given transaction satisfies only one rule of the table. However, many do provide for a so-called "ELSE" rule which is taken when all other rules fail. Generally, the ELSE rule is the last rule in the table. It has '-'s for all its condition entries. Given such an implementation, the ELSE rule becomes a special case of the subject matter considered here.

The methods described in the preceding section of this chapter are easily extended to tables in which the rules are not mutually exclusive but for which the precedence convention described in section 3.6 is adopted. The precedence convention applies when a transaction satisfies two or more rules and directs us to select the first rule satisfied. In cases when the precedence convention is applied, we shall say that the rule selected preempts some of the transactions satisfying the other rule(s). Additional restrictions placed on the method are as follows:

(1) Interpretation of rule probabilities

The probabilities associated with the rules of a decision table must be considered as joint probabilities. Thus P_r is no longer the probability that a transaction satisfies rule r ,

Pseudo Rules

```
.....: .30 .30 .20 .10 .10 : .30 .20 Si
: Ti Pi .....:
: : R1 R2 R3 R4 R5 :
: 10 .5 : Y N - Y N : - Y 5
: 10 .5 : Y - N N Y : Y N 3
: 20 .5 : - Y N Y N : Y - 10
.....: A1 A2
.....: A1 A2
```

Figure 4-23
Use of pseudo rules to cost a table having action sets.

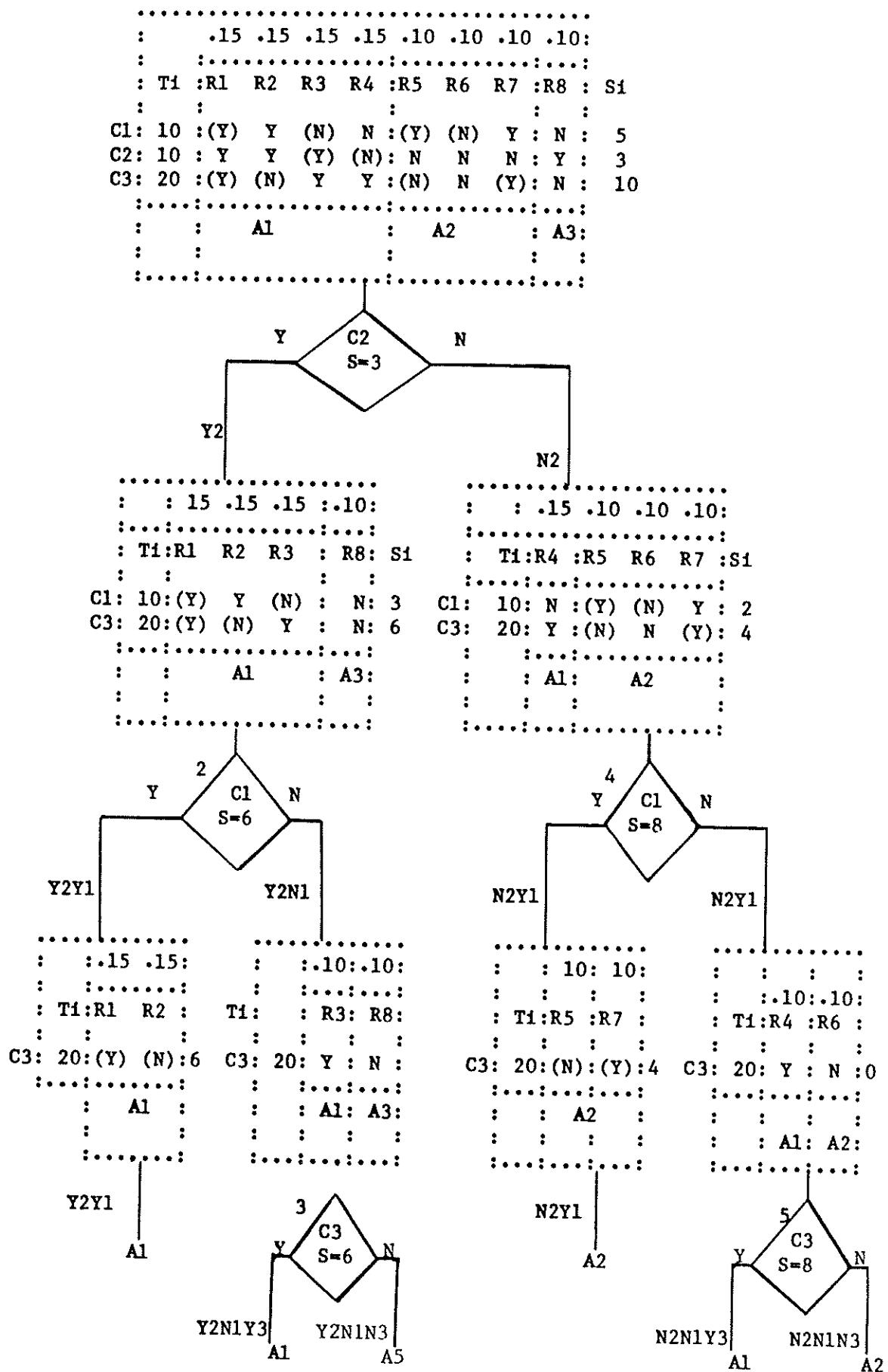


Figure 4-24
 Single - alternative tree.

but rather the probability that a transaction satisfies rule r and that it does not satisfy rules 1, 2, ..., r-1.

(2) Preservation of rule order

We have seen how the parsing operation generates successively smaller subtables, each of which contains a subset of the original table's condition and rules. A given subtable will not contain all the rules in the original table, since some rules would have been eliminated by the conditions upon which the subtable is dependent. The requirement here is that those rules which remain in a subtable be in the same sequence as they appeared in the original table.

(3) Test for first rule dominance

If all entries in the first rule of a subtable are '-'s, y!'s or N!'s, then that rule will be satisfied by all transactions processed against that subtable. Hence, the subtable should not be further parsed since the first rule is the only relevant rule.

(4) Final parsing steps

As we have seen, the steps in a parse often terminate in subtables containing only a single condition. It is at this point that the precedence comes into prominence and resolves obvious ambiguities. For illustration, consider the following portion of a single condition table:

	R1	R2	R4	R6
C4:
	Y	Y	-	N

The fact that rules R1, R2 and R4 are satisfied when C4=Y indicates that the corresponding rules of the original table were not mutually exclusive. The precedence convention directs us to take R1 for C4=Y. Again, rules R4 and R6 in the original table were not mutually exclusive, since in the subtable both are satisfied by C4=N. Here the precedence convention directs that R4 be chosen.

Many of these points are illustrated in Figure 4-25 which presents a parse of a decision table containing an ELSE rule. One can use the resulting tree to determine those transactions which are actually processed by the ELSE rule and are not preempted by other rules in the table. In the case of the table in Figure 4-25, these transactions correspond to the two branches which end in the selection of A5, namely, 'NYY' and 'NNY'.

The table in Figure 4-25 also illustrates the real problem involved in optimally parsing tables containing an ELSE rule or, more generally, containing ambiguous rules. The problem is that the '-' entries may not always identify a search. We found that the ELSE rule in Figure 4-24 is satisfied by two transactions: 'NYY' and 'NNY'. These can be combined into the form 'N-Y' which is the "true" expression for the rule. The ELSE rule (i.e., the last rule in the table in Figure 4-25 contributes to the search costs of all three conditions. This is correct only for condition 2. Condition 1 cannot entail a search cost for the rule because only transactions carrying an N for condition 1 are actually processed by this rule. (All transactions with C1=Y are preempted by other rules.) Likewise, the ELSE rule does not involve a search of condition 3 since only transactions which carry a Y for condition 3 reach the ELSE rule.

Thus, the table in Figure 4-25 is really the table in Figure 4-26. The latter is, in turn, a version of the table shown in Figure 4-12 in which the second rule has been moved to become the last rule in Figure 4-26. Note that two different optimal trees were produced from basically the same table. Compare Figure 4-25 in which C2 is tested first with Figure 4-12 in which C1 is tested first. The tree in Figure 4-12 is correct since the tree in Figure 4-25 is based on the faulty assumption that C1 and C3 contain -'s for the ELSE rule.

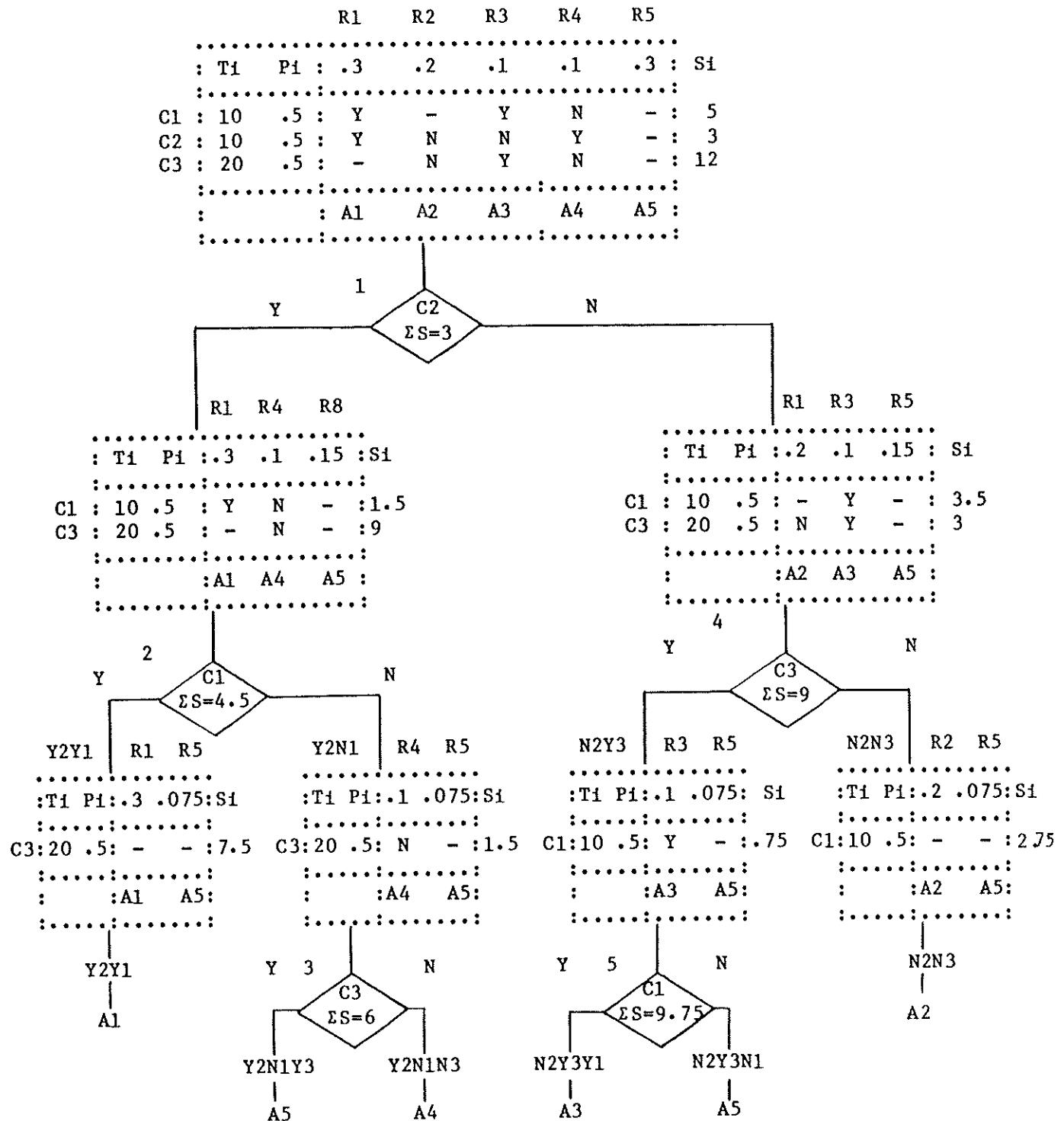


Figure 4-25
Parse of a decision table containing an ELSE rule.

```
.....  
: Ti   Pi : .3   .2   .1   .1   .3 : Si  
:.....:  
C1 : 10   .5 : Y   -   Y   N   N : 2  
C2 : 10   .5 : Y   N   N   Y   - : 3  
C3 : 20   .5 : -   N   Y   N   Y : 6  
:.....:  
:           : A1   A2   A3   A4   A5 :  
:.....:
```

Figure 4-26
The table in Figure 4-25 with the ELSE rule transformed.

4.3.10 Search-Free tables

A condition is search-free if it has a search cost of zero. A search-free decision table is a table that can be mapped into a program tree that tests only search-free conditions. As illustrated in Figure 4-27, this definition does not require that every condition be search-free, but rather that there be some sequence of condition testing which yields a tree with an accumulated search cost of zero.

Search-free tables do, in fact, occur frequently. This circumstance has led at least two investigators to hold erroneously that testing costs and rule probabilities have no significance, since they are cancelled out when multiplied by the zero search cost. This observation is more appropriate for the case in which one is selecting rules and using the occurrence of '-'s as a costing criteria, but it breaks down badly when one considers the generation of trees which optimally select an action set. Recall from sections 4.3.7 and 4.6 that optimal action set selection involves the creation, and addition to the table, of so-called pseudo rules. For example, the table in Figure 4-28 is search-free with respect to rule selection, but is not search-free with regard to action set selection. Specifically, rules R1 and R3 contribute to the creation of the first pseudo rule to the right of the table, while rules R4 and R7 cause the creation of the second pseudo rule for use in optimal action set selection. Once these pseudo rules are added to the table, it is revealed that when C1=Y, a search cost is involved in choosing between C2 and C3 as the next condition to test; and when C1=N and C4=Y, choosing between C2 and C5 also involves a search cost.

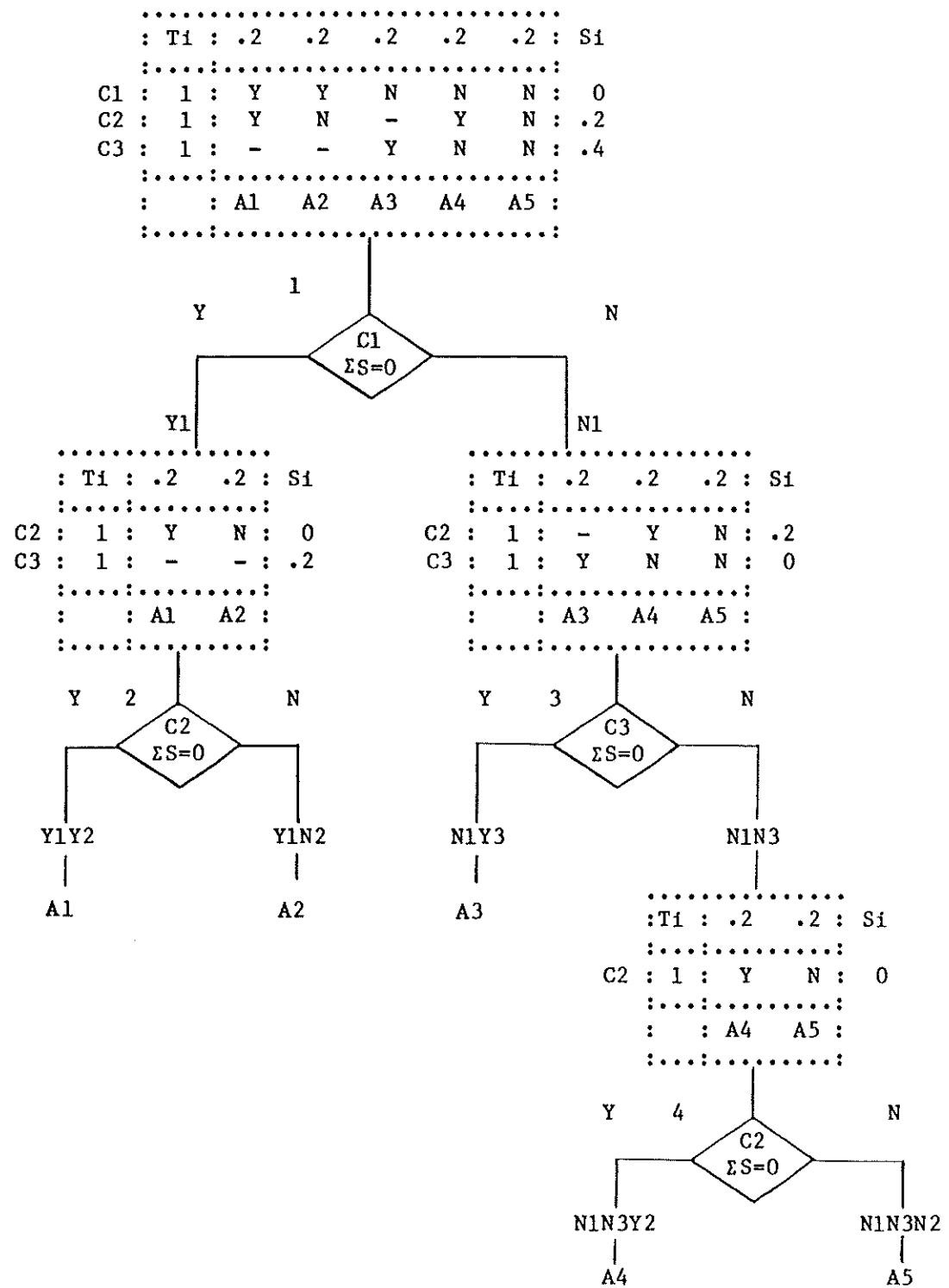


Figure 4-27
 Parsing a search-free table.

	R1	R2	R3	R4	R5	R6	R7	R8	Pseudo Rules
<hr/>									
: C1 :	Y	Y	Y	N	N	N	N	N	Y N
: C2 :	Y	Y	N	Y	Y	Y	N	N	- -
: C3 :	Y	N	-	-	-	-	-	-	Y -
: C4 :	-	-	-	Y	Y	Y	Y	N	- Y
: C5 :	-	-	-	Y	N	N	-	-	- Y
: C6 :	-	-	-	-	Y	N	-	-	- -
<hr/>									
: A1 :	X	-	X	-	-	-	-	-	X -
: A2 :	-	X	-	-	-	X	-	-	- -
: A3 :	-	X	-	-	-	-	-	-	- -
: A4 :	-	-	-	X	X	X	X	-	- X
: A5 :	-	-	-	X	-	-	X	-	- X
: A6 :	-	-	-	X	X	-	X	-	- X
: A7 :	-	-	-	-	-	X	-	-	- -
: A8 :	-	-	-	-	-	X	-	-	- -
: A9 :	-	-	-	-	-	-	-	X	- -
: A10 :	-	-	-	-	-	-	-	X	- -
<hr/>									

Figure 4-28

A table which is search-free with respect to rule selection, but
is not search-free with regard to the action set selection.

4.4 TREE GENERATION III: MINIMIZING TOTAL STORAGE COSTS

4.4.1 Goal and costing mechanism

Minimization of total storage requirements entails minimizing the sum $\sum_i (N_i S_i)$ where N_i is the number of times that condition C_i appears in the tree and S_i is the amount of storage required by one occurrence of C_i .

4.4.2 Comparison with minimizing average processing cost

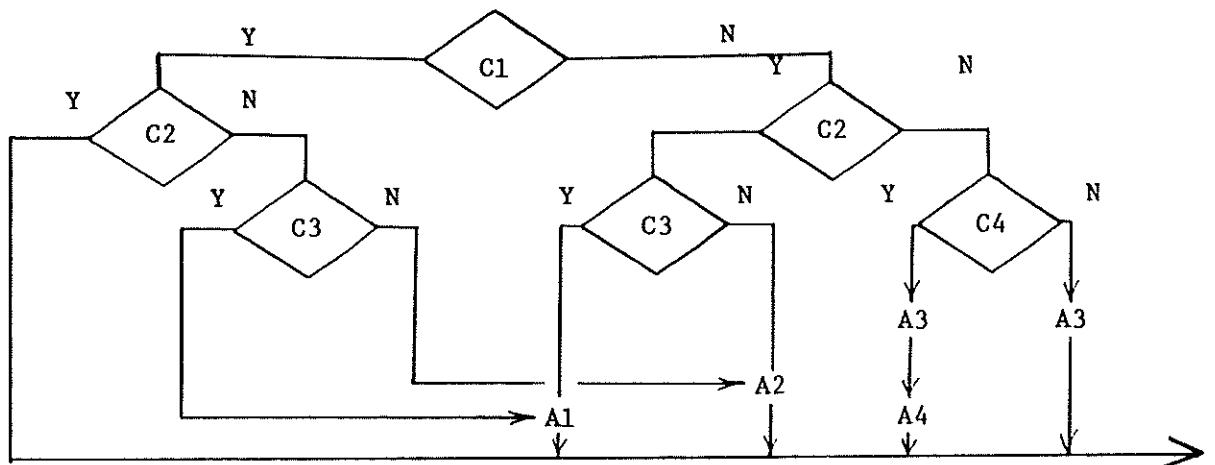
The conversion of a decision table to a program tree which is optimal with respect to total storage requirements is much more difficult than the conversion to a program tree which is optimal with respect to average processing costs. In minimizing processing costs we found that testing condition C_i first did not alter the costs associated with condition C_j . This does not hold when minimizing total storage costs. Instead, testing C_i first frequently doubles the costs associated with condition C_j , because C_j must be tested twice - once in the subtable dependent on C_i being true, and a second time for the subtable dependent on C_i being false.

There are other complexities in minimizing storage requirements. First, there is what will be called 'condition merging', illustrated in Figure 4-29. In the decision table shown in part a. of the figure, condition C_3 is used to select between action sets A_1 and A_2 in rule pair R_2 and R_3 and again in pair R_4 and R_5 . If our only interest is minimizing average processing costs, we can place two tests of C_3 in our processing tree as in part b of the figure. But we can reduce storage requirements by combining the two tests as in part c. (Note there is no difference between b and c with regard to average processing costs; both trees test C_3 once and only once for each transaction which satisfies one of the rules 2 thru 5.)

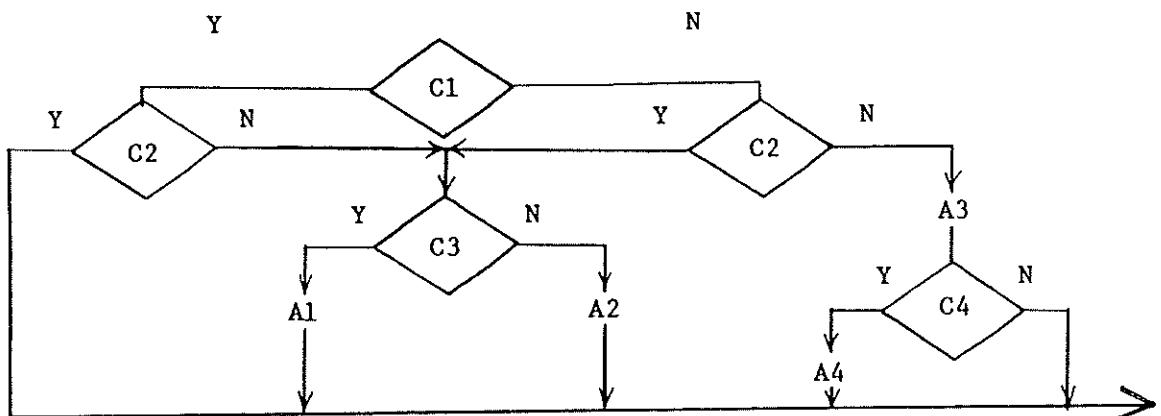
Figure 4-29 also illustrates intermixing conditions and actions as another way of decreasing storage requirements. Action A_3 is to be executed in both rules 6 and rule 7. If as in part b, it is placed after C_4 is tested, two allocations of storage for A_3 are required; one when C_4 is true and the other when C_4 is false. In part c of the figure, the second allocation of space for A_3 is avoided by 'hoisting' A_5 so that it precedes the test of C_4 . Again note there is no difference in processing costs since A_3 is executed with the same frequency in both trees.

	R1	R2	R3	R4	R5	R6	R7
.....							
: C1 :	Y	Y	Y	N	N	N	N
: C2 :	Y	N	N	Y	Y	N	N
: C3 :	-	Y	N	Y	N	-	-
: C4 :	-	-	-	-	Y	N	
.....							
: A1 :	-	X	-	X	-	-	-
: A2 :	-	-	X	-	X	-	-
: A3 :	-	-	-	-	-	X	X
: A4 :	-	-	-	-	-	X	-
.....							

a. Sample decision table.



b. Program tree for the above decision table.



c. Program tree when merging and intermixing permitted.

Figure 4-29

4.4.3 Mergeable entries

Two occurrences (i.e., entries) of the same condition are mergeable, if they discriminate between the same action sets. For a more precise definition of mergeable conditions, let us first review the concept of complementary rules introduced in section 4.3.6. Rule s is said to be the complement of rules r with respect to condition Ci, if rules r and s are identical for all condition entries except for those pertaining to the condition Ci and that for condition Ci, r has a Y as an entry and s has an N or vice versa.

Assume: 1) that rules T1 and T2 both carry a Y for condition Ci; and 2) that F1 and F2 are, respectively, the complements of T1 and T2 with respect to Ci. If 3) T1 and T2 have the same action set and 4) F1 and F2 have the same action set, then 5) the rule pairs (T1, F1) and (T2, F2) are mergeable with respect to Ci.

In Figure 4-30 the complement, with respect to condition C3, of R1 is R2 and the complement of R4 for C3 is R5. Rule pairs (R1, R2) and (R4, R5) are mergeable for C3 because R1 and R4 both imply action set A1 and R2 and R5 both have A2 as an action set. Note that rules R6 and R7 are also complementary with respect to C3 but they are not mergeable with R1 and R2 because rule R2 and R7 have different action sets.

4.4.4 Search entries for optimizing storage

When optimizing storage, an entry is considered a search entry if 1) it can be combined with another entry to yield a '-', 2) it is an implied limited entry; or 3) it is a mergeable entry. We shall continue our practice of placing search entries in parentheses.

4.4.5 Search-free conditions

We previously used the term 'search-free' to describe conditions which can be tested without incurring any search costs. In terms of total storage costs this means 1) there are no rule pairs which can be combined to yield a "-" for the condition, 2) the condition has no implied entries, and 3) there are no groups of rule pairs which are mergeable with respect to the condition.

Let Ci, Cj and Ck be search-free conditions. One can prove that tests of Cj must appear at least twice in any program tree which tests Ci before testing Cj; and that Ck must appear at least four times in any tree which tests both Ci and Cj before testing Ck. The relationships permit one to produce a minimal storage tree for a table containing only search-free conditions by simply testing the conditions in decreasing order of storage requirements. Thus the minimal storage tree for the table shown in Figure 4-31 consists of testing C3 first, then C1 and finally C2; storage required for this tree equal $5 + 2 \times 3 + 4 \times 2 = 19$ words.

	R1	R2	R3	R4	R5	R6	R7	
C1	:	Y	Y	Y	N	N	N	:
	:							:
C2	:	Y	Y	N	Y	Y	N	:
	:							:
C3	:	Y	N	-	Y	N	Y	N
	:							:
	:	A1	A2	A3	A1	A2	A1	A4
	:							:

Figure 4-30

W1

```
.....: C1 : Y   Y   Y   Y   N   N   N   N   : 3
.....: C2 : Y   Y   N   N   Y   Y   N   N   : 2
.....: C3 : Y   N   Y   N   Y   N   Y   N   : 5
.....: : A1  A2  A3  A4  A5  A6  A7  A8 :
.....:
```

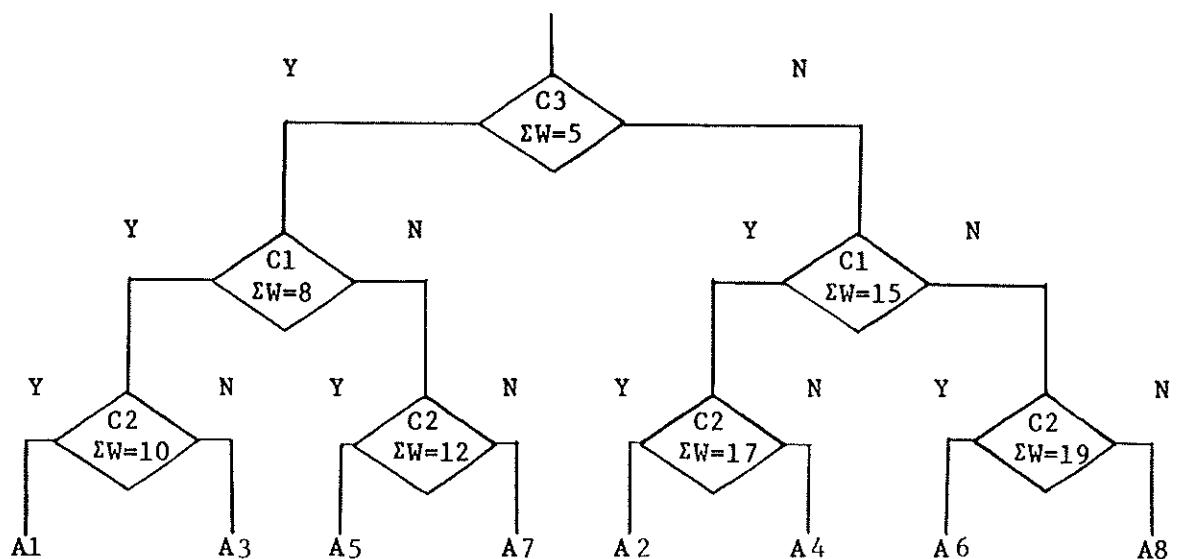


Figure 4-31
Minimizing the storage for a table containing
only "search-free conditions"

4.4.6 Storage costs for a branch and bound algorithm

We are now in a position to develop a branch-and-bound approach to minimizing total storage requirements using the following expression as a costing criterion.

$$ES_i = W_i + ELBY_i + ELBN_i$$

Where ES_i = estimated total storage costs associated with testing C_i first.

W_i = number of words required to test C_i

$ELBY_i$ = estimated lower bound on the storage requirements for the subtable dependent on $C_i = Y$.

$ELBN_i$ = estimated lower bound on the storage requirements for the subtable dependent on $C_i = N$.

The estimated lower bounds for the two subtables are obtained from computations involving three categories of conditions.

- a. Conditions which have no search entries within the subtable: the technique previously described in 4.4.5 is used to form a composite value for these conditions.
- b. Conditions which have both search and verification entries: each of these is counted once.
- c. Conditions which have only search entries (i.e., have no verification entries) are not counted.

Thus in the table shown in Figure 4-32 the estimated storage cost associated with condition C_1 , ES_1 equals the sum of:

- a. 1 - the number of words required to test C_1
- b. $3 + 2 \times 2 = 7$ - the cost of testing C_2 once and C_3 twice (neither C_2 nor C_3 contain any search entries in the subtable dependent on C_1 being true)
- c. 0 - C_2 and C_3 have only search entries in the subtable for $C_1 = N$.

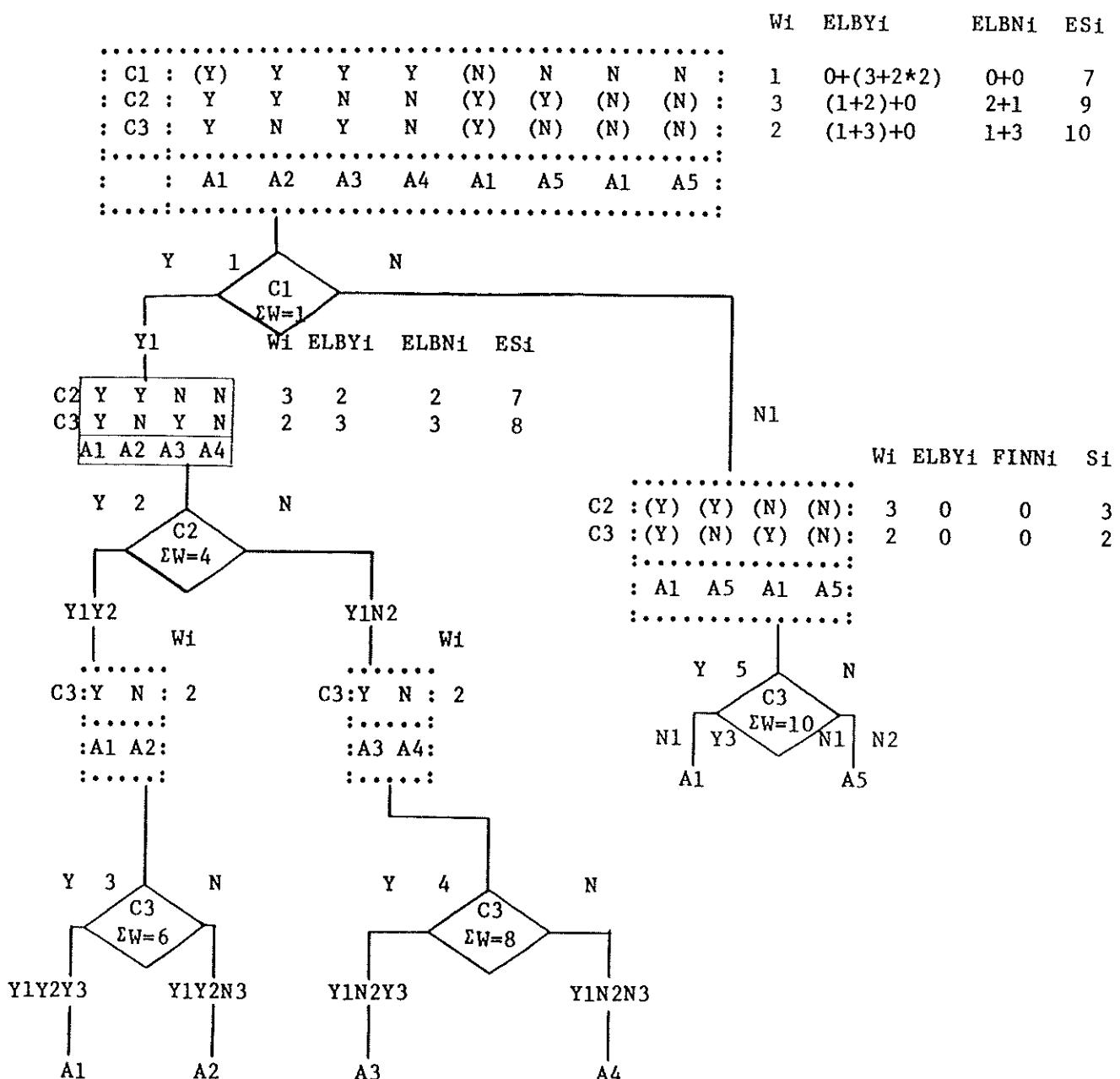


Figure 4-32

Also in the figure, ES2 = the sum of:

- a. $3 =$ the storage required to test C2.
- b. $3 = 1+2 =$ the cost of testing C1 once and C2 once (each condition contains both verification and search costs) in the subtable conditional on CZ = Y.
- c. $3 = 2+1$ from the subtable conditional on C2=N. In this subtable C3 contains both verification and search costs and is counted once. Condition C1 is also counted once because it is the one and only condition having no search costs.

4.4.7 Hoisting actions

Another possible way of reducing the amount of storage required by a program tree is to intermix conditions and actions (here we assume that there is complete freedom in changing the order of conditions and actions). This processing has been spoken of as 'hoisting' an action, in that it is moved upward in a table to precede one or more conditions. At times hoisting an action ahead of a condition may decrease the number of allocations (i.e., copies) required for the action. For example, in part a of Figure 4-33, three copies of A4 are required because there are three ---,x-x,-xx, different outpaths linking A4 to the table's exit, i.e., many,x-x, and -xx. Action A4 is executed, if and only if C1 is Y. By inserting A4 immediately before C2 (as in part b of 4-33) we decrease the number of allocations of A4 from three to one.

The relocation of A4 has several side effects which yield additional savings in total storage. Prior to the repositioning (part a of the figure) the rule pairs (R3,R4) and (R5,R6) imply different action sets, one executing A4 and the other ignoring A4. When A4 is moved, these rule pairs have like action sets and become mergable on C4. Furthermore, the table in part a of figure 4.33 implies two allocations of A3: one for the outpath x--- and the second for ----; with A4 removed there is only the outpath --- and only one allocation of A3 is required. The same thing happens for actions A1 and A2 and has the additional effect of reordering them. With A4 in its original position there is no reason to place A1 after A2 since it must be allocated twice; once when A4 is executed and a second time when A4 is ignored. After A4 has been intermixed with the table's conditions we can reverse the order of A1 and A2 and save one allocation of A1.

Notice that action A7 could be placed before C4 in rules R3 thru R6; but this repositioning will have no effect on storage requirements, since only one allocation of A7 is required in the original table.

a.

	R1	R2	R3	R4	R5	R6	R7	R8	R9	W1
: C1 :	Y	Y	Y	Y	N	N	N	N	N	:
: C2 :	Y	Y	N	N	Y	Y	N	N	N	:
: C3 :	Y	N	-	-	-	-	Y	Y	N	:
: C4 :	-	-	Y	N	Y	N	Y	N	-	:
.....
: A1 :	X	-	-	-	-	-	X	-	-	:
: A2 :	-	-	-	-	-	-	X	-	-	:
: A3 :	-	X	-	-	-	-	-	X	-	:
: A4 :	X	X	X	X	-	-	-	-	-	:
: A5 :	-	-	X	-	X	-	-	-	-	:
: A6 :	-	-	-	X	-	X	-	-	-	:
: A7 :	-	-	X	X	X	X	-	-	-	:
.....

b.

	R1	R2	R3	R4	R5	R6	R7	R8	R9	W1
: C1 :	Y	Y	Y	Y	N	N	N	N	N	:
: A4 :	X	X	X	X	-	-	-	-	-	:
: C2 :	Y	Y	N	N	Y	Y	N	N	N	:
: C3 :	Y	N	-	-	-	-	Y	Y	N	:
: C4 :	-	-	Y	N	Y	N	Y	N	-	:
.....
: A2 :	-	-	-	-	-	-	X	-	-	:
: A1 :	X	-	-	-	-	-	X	-	-	:
: A3 :	-	X	-	-	-	-	-	X	-	:
: A5 :	-	-	X	-	X	-	-	-	-	:
: A6 :	-	-	-	X	-	X	-	-	-	:
: A7 :	-	-	X	X	X	X	-	-	-	:
.....

Figure 4-33
Intermixing conditions and actions

4.5 INTERPRETIVE MASKING

4.5.1 The method

Interpretive masking provides an alternative to the conversion of a decision table into a program tree. Interpretive masking is a technique in which each transaction processed against a table is mapped into a bit mask of one's and zero's which is processed against a set of table masks to find the rule(s) satisfied by the transaction.

There are several ways of representing a table by a series of masks. In this section the most widely used representation, rule masking, will be described.

As shown in Figure 4-34, each rule, $R(j)$, in a table is represented by three masks:

- 1) relevancy mask, $RL(r)$, which carries a one in bit i , if rule r is relevant or dependent, (i.e., is $Y, N, Y!$ or $N!$ for condition i), and a zero, if it is not relevant or dependent (i.e., a $-$) for condition i
- 2) a yes-no mask, $YN(r)$, which carries a one in bit i , if rule r carries a Y or $Y!$ for condition i and is zero otherwise
- 3) An action mask, $AC(r)$, which carries a one in bit i , if action i is to be executed for rule r .

A transaction is processed against the table by mapping it into a transaction mask, TM , with bit i being a one if condition i is true.

TM is then processed against the table's masks to find a rule r such that

$AND (TM, RL(r)) = YN(r)$

where the AND of two masks yields a third mask containing a one for those bits for which both masks contain a one.

Once r has been selected $AC(r)$ (the action mask for rule r) is retrieved and those actions for which it carries a one are executed (see Figure 4-35).

	C1	C2	C3	C4	A1	A2	A3	
R1 :	Y	Y	Y	Y	X	X	X	:
R2 :	Y	Y	Y	N	X	X	-	:
R3 :	Y	Y	N	-	X	X	-	:
R4 :	Y	N	-	-	-	-	X	:
R5 :	N	Y	-	-	-	-	-	:
R6 :	N	N	Y	-	X	-	-	:
R7 :	N	N	N	Y	X	-	X	:
R8 :	N	N	N	N	-	X	X	:

	Relevancy Masks				Yes-no Masks				Action Masks		
RL(1)	1	1	1	1	YN(1)	1	1	1	AC(1)	1	1
RL(2)	1	1	1	1	YN(2)	1	1	0	AC(2)	1	1
RL(3)	1	1	1	0	YN(3)	1	1	0	AC(3)	1	1
RL(4)	1	1	0	0	YN(4)	1	0	0	AC(4)	0	0
RL(5)	1	1	0	0	YN(5)	0	1	0	AC(5)	0	0
RL(6)	1	1	1	0	YN(6)	0	0	1	AC(6)	1	0
RL(7)	1	1	1	1	YN(7)	0	0	0	AC(7)	1	0
RL(8)	1	1	0	1	YN(8)	0	0	0	AC(8)	0	1

Figure 4-34

	Sex	Age	Married	Call	Call	:
	Male	>21		Tally 1	Tally 2	:
Rule 1	:	Y	Y	Y	X	X
Rule 2	:	Y	Y	N	-	X
Rule 3	:	Y	N	-	X	-
Rule 4	:	N	-	-	-	-

RL(1)	1	1	1	YN(1)	1	1	1	AC(1)	1	1
RL(2)	1	1	1	YN(2)	1	1	0	AC(2)	0	1
RL(3)	1	1	0	YN(3)	1	0	0	AC(3)	1	0
RL(4)	1	0	0	YN(4)	0	0	0	AC(4)	0	0

Let transaction T relate to a married male, 20 years of age

Then

$TM = 1 \ 0 \ 1 = Y \ N \ Y$

and rule 1 is not selected because

$AND(TM, RL(1)) = AND(1 \ 0 \ 1, 1 \ 1 \ 1) = 1 \ 0 \ 1 \neq 1 \ 1 \ 1 = \text{mask } YN(1)$

but rule 3 is selected because

$AND(TM, RL(3)) = AND(1 \ 0 \ 1, 1 \ 1 \ 0) = 1 \ 0 \ 0 = \text{mask } YN(3)$

and only TALLY 1 will be called because $AC3 = 1 \ 0$.

Figure 4-35

4.5.2 Processing and storage costs for rule masking

Rule masking is of no advantage with respect to average processing costs since each condition in the table must be tested for every transaction processed against a table. In contrast, tree generation offers the prospect of processing many transactions without testing all conditions. In addition, rule masking imposes overhead costs to initialize and build the transaction mask TM, to select a rule r by comparing TM with the table masks, and to interpret the action mask AC(r) to determine which actions to execute.

Since rule-masking is disadvantageous with regard to processing time any argument supporting its use must be founded upon permitting a reduction in storage costs. As will be shown in the analysis which follows, the case for using rule masking is unclear. Rule-masking is efficient storage-wise for two classes of tables: those with many conditions and few rules (a. in Figure 4-36) and those with many rules (b. in Figure 4-36). Intermediate to those extremes are a substantial majority of tables (c. in Figure 4-36) for which rule masking appears to require more storage than does a generated tree.

Storage requirements for the condition testing portion of a generated rule masking program can be estimated in terms of the following components:

<u>task</u>	<u>hardware - software</u>
1. Initialize--set transaction mask to zero	1 word storage requirements hypothesized on 1981 state of the art in computer
2. Test condition--load index, load value, compute, compare, jump conditionally	5 words per condition
3. Build mask--load mask, OR in bit, store mask	3 words per condition
4. Use mask to select rule--call subroutine to make the selections	4 words
5. Storage for rule masks.	1 word per rule

```
.....: C1 Y Y N : .....: C1 Y N - :  

: C2 Y N N : .....: C2 Y N - :  

: C3 Y Y N - : .....: C3 Y N - :  

: C4 Y N Y - : .....: C4 Y N - :  

: C5 Y N N - : .....:  

: C6 N Y Y - : .....:  

: C7 N Y - - : .....:  

: C8 N - N - : .....:  

.....:  

: A1 A2 A3 A4 : .....:
```

- a. Rule masking is efficient storagewise for tables with many conditions and few rules

C1	Y	Y	Y	Y	Y	Y	Y	N	N	N	N	N	N	N	N	N	N		
C2	Y	Y	Y	Y	Y	Y	N	N	Y	Y	Y	N	N	N	N	N	N		
C3	Y	Y	Y	N	N	N	-	-	-	-	-	Y	Y	N	N	N	N		
C4	Y	Y	N	Y	Y	N	-	-	Y	Y	N	N	Y	N	Y	N	N		
C5	Y	N	Y	N	-	Y	N	Y	N	Y	N	-	-	Y	N	Y	N		
	A1	A2	A3	A4	A5	A6	A7	A8	A9	A10	A11	A12	A13	A14	A15	A16	A17	A18	A19

- b. Rule masking is efficient storagewise for tables with many rules

C1	Y	Y	Y	Y	N		C1	Y	Y	Y	Y	N	N	N				
C2	Y	Y	Y	N	-		C2	Y	Y	Y	N	N	-	-				
C3	Y	N	N	-	-		C3	Y	N	N	-	-	Y	Y	N			
C4	-	Y	N	-	-		C4	-	Y	N	-	-	-	-				
	A1	A2	A3	A4	A5		C5	-	-	Y	N	Y	N	-				
								A1	A2	A3	A4	A5	A6	A7	A8			

- c. Tables for which tree generation is more efficient storagewise than rule masking

Figure 4-36

To estimate the storage required for a testing tree generated for a decision table, we need to estimate the number of nodes (conditions tested in the tree). If we confine ourselves to consistent tables (i.e., tables which contain no overlapping rules and no ELSE rule), the number of nodes is a function of the number of rules in the table; in nearly all cases a table with NR rules can be converted into a tree containing NR-1 nodes. Each node tests a condition and is estimated to require 5 words of storage.

Applying these two estimates one discovers that tree generation is more efficient storagewise for relatively small consistent tables. For instance, the tree for a consistent table having 6 conditions and 7 rules is estimated to take 30 words of storage, one half of the estimated 61 words required by rule masking for the same table. The estimates show also that rule masking is appropriate for tables with many rules; for example, a table with 6 conditions and 20 rules is estimated to require 95 words of storage when converted to a tree, as against 74 for rule masking.

The two estimates can be combined to form the following inequalities which tells the point at which is better storagewise to apply rule masking for a consistent table with NC conditions and NR rules.

$$\begin{aligned} 1 + 8NC + 4 + NR &< 5(NR-1) \\ 5 + 8NC &< -NR + 5NR-5 \\ 8NC + 10 &< 4NR \\ NR &> 2NC + 3 \end{aligned}$$

The last of the inequalities was used to obtain the rightmost column in Figure 4-37. Values in this give RMIN for a fixed number, NC of conditions. For consistent tables having NC conditions, RMIN specifies the smallest number of rules for which rule masking requires less storage than that required by tree generation.

The larger left side of the table in Figure 4-37 relates to tables which have many conditions and few rules with the result that most of the sets of possible inputs to the table are processed by an ELSE rule. The table presents extreme cases most strongly based in favor of rule masking. For example, in the case of tables with three rules we could have the first rule containing all Y's, the second rule having all N's and the third, an ELSE rule with all -'s. (See the table shown on the right in part A of Figure 4-36). Conversion of such a table results in a tree containing $NC + (NC-1)$ nodes which is the maximum number for any tree associated with NC conditions and three rules. For $NC = 5, 5+4 = 9$ nodes are required, and each node is estimated to require 5 words of storage; hence, the whole tree requires $9*5 = 45$ words. For rule masking total storage required is $1 + 5*8 + 4 + 4 + 3 = 48$ words of storage. One can conclude that for all tables of 3 rules and 5 conditions, rule masking requires at least $48-45 = 3$ more words of storage than tree generation (i.e., the 'savings' accrued by rule masking is -3 as is shown in Figure 4-37).

```
.....:  
:      : For tables with an ELSE rule   : For consistent tables: :  
:      : and structured most favorably  : RMIM = the smallest  :  
:      : for rule masking: Words       : number of rules for  :  
:      : saved (+) or lost (-) by using : which rule masking   :  
: Number : rule masking in place of tree : saves storage     :  
: of     : generation                  :                      :  
: condi- :.....:                      :                      :  
: tions  : number of rules           :                      :  
:        : 2    3    4    5    6    :  
.....:  
: 1    : -10   -    -    -    -    :      -    :  
: 2    : -13   -9   -10   -    -    :      -    :  
: 3    : -16   -7   -3    +1   0    :      -    :  
: 4    : -19   -5   -1    +3   +7    :      11   :  
: 5    : -22   -3   +1    +5   +9    :      13   :  
: 6    : -25   -1   +8    +17  +26    :      15   :  
: 7    : -28   -1   +10   +19  +28    :      17   :  
.....:
```

Comparative storage requirements of rule masking and tree generation

Figure 4-37

5. DEVELOPMENT METHODOLOGY

5.1	THE ROLE OF DECISION TABLES IN SYSTEM DESIGN	5- 1
5.1.1	Overview of this chapter	5- 1
5.1.2	Approaching a system problem with decision tables . . .	5- 2
5.2	DECISION TABLE COMPOSITION	5- 2
5.2.1	Criteria for a good table	5- 3
5.2.1.1	Table completeness	5- 3
5.2.1.2	Table size	5- 4
5.2.1.3	Impossible rules	5- 5
5.2.1.4	Consistency	5- 6
5.2.1.5	Improving decision table readability	5- 7
5.2.2	Composition techniques	5- 9
5.2.2.1	The search mode (progressive rule development) . . .	5- 9
5.2.2.2	The direct mode (exhaustive enumeration)	5-11
5.3	SYSTEM DEVELOPMENT	5-16
5.3.1	Systems specification phase	5-16
5.3.1.1	User to producer contract	5-17
5.3.1.2	Significance of the phase	5-17
5.3.1.3	Methodology	5-18
5.3.1.3.1	Understand the problem	5-18
5.3.1.3.2	Structure the problem	5-19
5.3.1.3.3	Elaborate the lower level tables	5-19
5.3.1.3.4	Illustration of the system specifications process .	5-19
5.3.1.3.4.1	The problem as presented	5-20
5.3.1.3.4.2	Problem interpreted by producer	5-21
5.3.1.3.4.3	Restructuring the problem	5-26
5.3.1.3.4.3.1	Initial attempt	5-26
5.3.1.3.4.3.1.1	List all known facts	5-27
5.3.1.3.4.3.1.2	Construct an overall table	5-28
5.3.1.3.4.3.1.3	Elaborate conditions and actions	5-30
5.3.1.3.4.3.2	Final attempt	5-32
5.3.1.3.4.3.3	The final solution	5-37
5.3.1.3.4.3.4	Conclusions	5-40
5.3.2	System design phase	5-41
5.3.2.1	Product of this phase	5-41
5.3.2.2	Method used in this phase	5-42
5.3.2.2.1	Analyze the system specifications for completeness	5-42
5.3.2.2.2	Add the movement of data	5-44
5.3.2.2.3	Add iterations	5-47
5.3.2.2.4	Define the system architecture	5-49
5.3.2.2.5	Add data access and control functions	5-50
5.3.2.2.6	Partition into programmable modules	5-51
5.3.3	Implementation phase	5-53
5.3.3.1	Implementation using a decision table translator .	5-53
5.3.3.2	Manual translation	5-54

5.3.4	Operation	5-57
5.3.4.1	The problem	5-57
5.3.4.2	The role of decision tables	5-58
5.3.4.3	The format for decision tables	5-58
5.3.4.4	An example	5-58
5.3.5	User acceptance	5-60
5.3.5.1	Objective	5-60
5.3.5.2	Procedure	5-60
5.3.6	Maintenance and extensions	5-60
5.3.6.1	Introduction	5-60
5.3.6.2	Maintenance	5-60
5.3.6.2.1	Need for maintenance	5-60
5.3.6.2.2	Finding the cause	5-65
5.3.6.2.3	Fixing the cause	5-66
5.3.6.3	Extensions and improvements	5-66
5.3.6.4	Analysis and documentation	5-67
5.3.6.4.1	Record all possible paths	5-68
5.3.6.4.2	Study for correctness	5-73
5.3.6.4.3	Rewrite	5-73

5.0 DEVELOPMENT METHODOLOGY

5.0 DEVELOPMENT METHODOLOGY

The intention of this chapter is to define various methods and techniques for using decision tables to develop system products. The underlying theme of the chapter is to describe the use of decision tables in the framework of structured software methodologies. This will be done through the use of examples and tutorials showing the use of decision tables in the various stages of a system's life cycle. The emphasis is on software systems, but references are made to non-software uses of decision tables which can benefit equally from their application.

This chapter has taken the broader scope of decision table techniques described in Chapter 3, THEORY, and defined a working subset for the system's developer. The subset has been selected for its ease of application and demonstration. Although the benefits of decision tables are obtained from this subset, a total understanding and use of them is only possible with a thorough review of the THEORY chapter.

5.1 THE ROLE OF DECISION TABLES IN SYSTEM DEVELOPMENT

The strength of the decision table as a mechanism for stating logical relationships has often been highlighted for the small or single table problem. However, the real world is not often so simply capsulized. In today's world the system's approach is necessary to solve the complex problems that arise.

The purpose of this chapter is to provide a set of guidelines for using decision tables effectively for developing systems. The emphasis will be on data processing applications, although the methods employed can easily be applied to non-data processing applications. In this context decision tables can play an important role in the analysis, design, implementation, and operation of a computer application.

Throughout this chapter, two parties are used to distinguish between the various activities. The first is the user, defined as the person (or persons) who wants to have the system implemented. The second is called the producer. This is the person (or persons) who will develop and deliver the completed system. It is possible for the user and producer to be the same person, but in the context here they are separate and conduct their activities through contractual commitments.

5.1.1 Overview of this chapter

Section 5.2 defines guidelines for composing effective decision tables. It points out the effects of size, completeness, inconsistency, and other properties have upon effective decision tables. This section also covers composition techniques for constructing decision tables.

Section 5.3 details the life cycle of a system and describes how decision tables can be used in each of six identified phases:

Phase I	System Specification
Phase II	System Design
Phase III	Implementation
Phase IV	Operation
Phase V	User Acceptance
Phase VI	Maintenance and Extensions

5.1.2 Approaching a system problem with decision tables

Decision tables can be used to effectively analyze and then remove ambiguities from large complex problems. Individual decision tables can be used to partition and clarify a problem into digestible pieces. During the System Specification Phase, (Phase I), the problem is analyzed and all ambiguities are removed. The function of this phase is to obtain a clear statement of the problem, and then develop a hierarchical structure of decision tables that serves as a functional solution to the problem. That is, it states what must be done and gives the procedures necessary to do it.

The System Design Phase, (Phase II), takes the requirements from the system specifications and develops a design for the system. The design defines the system's architecture for a specific operational environment. Thus the design would be different for each specific computer or manual environment for which it was intended.

After the system design phase is completed, the system is constructed in the Implementation Phase, (Phase III). This phase makes use of tools like decision table translators to translate the design to an operational product.

Before turning over the system to the user, the producer develops operating procedures and user documentation in the Operation Phase, (Phase IV). These documents are made more effective by the concise and precise ways decision tables can describe logical relationships.

The completed system is turned over to the user who conducts the User Acceptance Phase, (Phase V), testing. In this way the user verifies that the delivered product meets the requirements of the original system specifications.

The long term benefit of having used decision tables in the system development process comes in the Maintenance and Extensions Phase, (Phase VI). In a system developed with decision tables, it is easier to find and fix errors and easier to extend or enhance the system.

5.2 DECISION TABLE COMPOSITION

The use of decision tables in developing system products requires a firm understanding of how they might best be used. Chapter 3, THEORY, gives a total view of the potential of decision tables. The following section narrows that view to a working subset. The subset of decision table THEORY selected in this section should be considered a general guideline for using decision tables effectively.

Some of the definitions from THEORY have been restated here to make them more understandable to the general reader. Although the subset provides the user and producer with a fairly effective set of procedures for using decision tables, only a complete understanding of the THEORY chapter will give an appreciation for the overall power of decision tables.

5.2.1 Criteria for a good table

For the orderly and effective use of decision tables, it is desirable to adopt some conventions for defining good decision table practices. The following criteria are used as guidelines for this purpose:

- (a) to guarantee completeness in developing the system,
- (b) to minimize side effects,
- (c) to promote easy manipulation of the tables,
- (d) to obtain a clear and homogeneous documentation of the system.

The remainder of this section proposes recommendations for constructing and using decision tables based upon the above criteria.

5.2.1.1 Table completeness

A powerful aspect of decision tables is their ability to represent a complete description of the relationships between conditions. A complete table may be expressed only in simple rules or in combined rules. It is recommended this form be used instead of hiding combinations in the ELSE-rule. The tables below are examples of complete and consistent tables representing an identical situation, however, the one on the left is in simple rule form while the right one uses combined rules:

.....
: AGE : <20 :>20 ≤60: >60 :	: AGE : <20 :>20 ≤60: >60 :
::.....:.....:	::.....:.....:
: SEX :M : F: M : F :M : F :	: SEX : - : M : F: - :
::.....:.....:.....:	::.....:.....:.....:
: ACTION :1 : 1: 2 : 3 :4 : 4 :	: ACTION : 1 : 2 : 3: 4 :
::.....:.....:.....:	::.....:.....:.....:

A table such as this is complete only if all possible states of each condition are included. Since it is assumed that SEX in the table will

be either M (for male) or F (for female), the table is logically complete. However, in the situation that SEX might be incorrectly specified or omitted, the table would have to be expanded to accommodate this:

```
.....  
: SEX : M : F : OTHER :  
.....  
: AGE : >20 : : >20 : :  
: : <20 : : >60 : <20 : : >60 : - :  
: : <60 : : <60 : :  
.....  
: ACTION: 1 : 2 : 4 : 1 : 3 : 4 : ERROR :  
.....
```

5.2.1.2 Table size

The optimal size of a decision table (as measured by the number of conditions, rules, and actions) must be determined by the table's readability. The problem is one of human engineering. Although decision tables make it possible to express very large complex logical relationships, such tables may be very difficult for a person to use effectively. In such cases it is advisable to partition the large decision table into a set of smaller ones.

For example, the table in Figure 5-1 can be factored into three smaller decision tables, Figure 5-2, for improved readability.

```
.....  
: C1 : Y : N :  
.....  
: C2 : Y : N : Y : N :  
.....  
: C3 : Y : N : Y : N : Y : N : Y : N :  
.....  
: C4 : Y : N : - : Y : N : Y : N : Y : N : Y : N : - :  
.....  
: A1 : X : X : : : : : : : : X : X : :  
: : : : : : : : : : : : : :  
: A2 : X : X : : : : : : : : X : X :  
: : : : : : : : : : : : :  
: A3 : : : X : X : X : X : X : : : :  
: : : : : : : : : : : : :  
: A4 : : : X : X : X : X : X : : :  
.....
```

FIGURE 5-1

:	:	:	:	:
:	C1	:	Y	:
:			N	:
:				:
:	C2	:	Y	:
:			N	:
:				:
:		:	:	:
:	Table A	:	X	:
:				:
:	Table B	:	X	:
:			X	:
:				:

Table A

:	:	:	:
:	C3	:	Y
:		:	N
:			:
:	C4	:	Y
:		:	N
:			-
:			:
:	A1	:	X
:		:	X
:			:
:	A2	:	X
:		:	X
:			:

Table B

:	:	:	:
:	C3	:	Y
:		:	N
:			:
:	C4	:	Y
:		:	N
:			:
:	A3	:	X
:		:	X
:			:
:	A4	:	X
:		:	X
:			:

Figure 5-2

A problem in readability may occur when a process is expressed in too many small decision tables. If the tables are not related in a direct and straightforward way, it can be difficult to follow the process they describe. In such situations it is better to synthesize these small tables into larger ones.

5.2.1.3 Impossible rules

A syntactically proper table is defined here as one without "Impossible Rules." However, semantical interrelations between conditions may introduce impossible rules in an otherwise syntactically proper table. Take for example the following table:

	1	2	3	4
:	C1	:	Y	:
:		:	N	:
:		:	N	:
:			N	:
:	C2	:	-	:
:			Y	:
:			N	:
:			N	:
:	C3	:	-	:
:			-	:
:			Y	:
:			N	:
:			N	:

It is syntactically proper, yet when the actual conditions are inserted it reads:

	1	2	3	4
: A>B :	Y	N	N	N
:
: A<B :	-	Y	N	N
:
: A=B :	-	-	Y	N
:

Due to the relationship between conditions the fourth rule is semantically impossible.

The impossibility can be eliminated by removing one of the conditions. Thus the table becomes:

	1	2	3
: A>B :	Y	N	N
:
: A<B :	-	Y	N
:

The third rule now represents the condition A=B (i.e., A NOT >B and A NOT <B).

Rule 4 of the above example was a true impossibility; however when analyzing a decision table for impossible rules care must be taken not to also eliminate rules that should never happen along with those that will never happen. The former is improbable but not impossible, and thus should be kept in the decision table.

Murphy's Law and defensive programming have taught that the unlikely should always be considered a possibility. Therefore, only truly impossible rules should be removed from a decision table.

5.2.1.4 Consistency

It is possible for a decision table to appear to be complete and not contain impossible rules, and yet be inconsistent. Inconsistency here means that a single transaction can satisfy more than one rule of the table. If the actions in the inconsistent rules are the same, it is called redundancy. If the actions are different, it is called a contradiction. In both cases the decision table should be corrected to eliminate these situations.

A decision table written in canonical form cannot have inconsistencies, since the completeness of it precludes their inclusion. However, a table of combined rules that has not been properly factored from its canonical form can easily fall prey to problems.

This can be shown in the following decision table which on the surface seems correct:

	1	2	3	4	
: C1 :	Y	N	N	N	:
: C2 :	-	Y	Y	N	:
: C3 :	-	N	-	-	:
: C4 :	-	-	Y	-	:

The table appears to have 16 rules as is indicated for completeness by the 4 conditions ($2^4 = 16$), and yet, under closer inspection this table is both incomplete and inconsistent. By expanding rules 2 and 3, it is obvious that they both contain the rule NYNY. And if the table is put into full canonical form, rule NYYN will be found to be missing.

Using the canonical form is a way to guard against inconsistent or incomplete tables.

5.2.1.5 Improving decision table readability

It is possible to improve the readability of a decision table in several ways. One of these is to re-order conditions or rules into a more meaningful sequence. For conditions to be more easily interpreted, they should be stated in a natural ordering, progressing predictably. For example, a numeric sequence should proceed from smallest to largest or largest to smallest.

.....
: C1 : >10 : - : <10 : <u><15</u> :	: C1 : <10 : >10 : - : <u>>15</u> : <u><15</u> :
.....

(a)

(b)

In (b) above the natural order is expressed, which makes it easier to interpret than (a).

Another method for improving the readability of decision tables is to avoid double negatives. Conditions expressed as double negatives are often misinterpreted.

```
.....  
: : : :  
: A ≠ B : Y : N :  
.....:.....:.....:
```

(a)

```
.....  
: : : :  
: A = B : N : Y :  
.....:.....:.....:
```

(b)

Avoiding the double negative by reversing the condition entry logic makes (b) simpler to read than (a).

The readability of a decision table with extended entries is superior to one with limited entries. Consider the two tables below as an illustration:

```
.....  
: AGE : <20 : 20≤60 : >60 :  
.....:.....:.....:  
: ACTION : 1 : 2 : 3 :  
.....:.....:.....:  
Extended Entry
```

```
.....  
: AGE <20 : Y : N : N :  
: : : : :  
: AGE ≤60 : - : Y : N :  
.....:.....:.....:  
: ACTION : 1 2 3 :  
.....:.....:
```

Limited Entry

The entry portion of a decision table can be more easily read if the number of symbols used is reduced.

```
.....  
: C1 : Y:Y:Y:Y:N:N:N:N:  
: :.....:.....:  
: C2 : Y:Y:N:N:Y:Y:N:N:  
: :.....:.....:  
: C3 : Y:N:Y:N:Y:N:Y:N:  
.....:.....:.....:
```

(a)

```
.....  
: C1 : Y : N :  
: :.....:.....:  
: C2 : Y : N : Y : N :  
: :.....:.....:  
: C3 : Y:N:Y:N:Y:N:Y:N:  
.....:.....:.....:
```

(b)

In the above example, the entry portion of (b) is more readable than (a). A similar process can be applied to the action entries:

```
.....  
: A1 = : 51:51:51:10:10:10:10:10 :  
: :.....:.....:.....:  
: A2 = : A :B :B :A :B :B :B :  
.....:.....:.....:.....:
```

(a)

```
.....  
: A1 = : 51 : 10 :  
: :.....:.....:  
: A2 = : A : B : A : B :  
.....:.....:.....:
```

(b)

The action entries of (b) are easier to read than (a). However, if many actions exist in the table, grouping them all may make it difficult to follow the rules down without a straight-edge. Discretion must be used in grouping actions -- it should be done only in order to improve readability.

In all cases, the primary consideration should be to use the format that makes it easiest for people to interpret the decision table.

5.2.2 Composition techniques

Composing decision tables can be done more effectively by using one of two techniques: the "search mode" (progressive rule development), and the "direct mode" (exhaustive enumeration).

The search mode is most effective when the problem has not been fully defined. The producer can use the search mode when interviewing the user to obtain a precise statement of the problem.

The direct mode is most effective when the problem has been fully defined in a form other than decision tables. The producer uses the direct mode to translate the existing problem description into the decision table form.

5.2.2.1 The search mode (Progressive rule development)

This mode should be used when the problem is not structured 'a priori'. The essence of this mode is that the logic of the problem has to be discovered by interviewing the user. During the interview, a decision table is made. The interview may be carried out in a number of ways. The following example is one possible way to proceed.

The user is asked to name as many relevant conditions to the decision situation as possible and at least one state that each condition can assume. If there is more than one condition, then the producer submits to the user one combination of states for the named conditions, and asks the user to indicate the actions to be taken when that particular combination occurs. Then a second state combination is presented to the user and he is asked again to fill in the corresponding actions, etc. During this process, conditions and actions will normally be added as a consequence of added insights gained by the user's or the producer's inspection of the decision table being built.

The process can be managed so that it is unnecessary to consider all separate rules of the exhaustive combinatoric decision table. To illustrate this, let us assume that a user is about to deal with rule 5 of a decision table looking at the moment as follows. (In this table EC stand for Equivalence Class -- a set of identical actions for two or more rules.)

```
.....  

: C1 : k : 1 :  

:.....  

: C2 : p : q : t : p : q : t :  

:.....  

: C3 : w : w : w : w : w : w : w : w : w : w : w : w :  

:.....  

: A1 : x : : : x : : : : : : : : : : : : :  

: : : : : : : : : : : : : : : : : : : :  

: A2 : x : x : x : x : : : : : : : : : : : : :  

: : : : : : : : : : : : : : : : : : : :  

: A3 : x : : x : : : : : : : : : : : : : : :  

: : : : : : : : : : : : : : : : : : : :  

: A4 : : x : : x : : : : : : : : : : : : : :  

: : : : : : : : : : : : : : : : : : : :  

: A5 : : : x : : x : : : : : : : : : : : : :  

:.....  

: EC : 1 : 2 : 3 : 2 : 4 : : : : : : : : : :  

:.....
```

Figure 5-3
A decision table in the process of being constructed
by the search mode.

Suppose that the user realizes that whenever condition C3 is w the rule will be in EC2; i.e., actions A2 and A4 will be carried out. This means that in the case of this example, it will only be necessary to deal with half of the remaining rules. The user must indicate that EC2 is dependent on C3 being equal to w and that for all cases where this is true the rule will be a member of EC2. The producer should now check that this is in fact true for all cases where C3 = w in those rules that have already been captured. Should there be any problems, they must be brought to the attention of the user who must try to resolve them. As a result, the definition of EC2 might be extended. It is also possible that the definition of EC2 might be removed.

Once all the rules have been considered, an analysis of the ECs may begin. Assume for example, that EC2 above, with half of the rules as members is considered. The condition stub could be reorganized as follows:

```
.....  

: : : : :  

: C3 : w : w :  

: .....:....:  

: C1 : k : 1 : - :  

: .....:....:  

: : : : : : : : :  

: C2 : p : q : t : p : q : t : - :  

: .....:....:....:
```

Figure 5-4
The condition part of a decision table in which the conditions
have been reordered

If there is a simple, discernable definition for the EC (for instance, if when C3 = w and C2 = t, the identical set of actions is executed), the producer should be able to discover the definition and verify it with the user. If this particular case was true, the condition stub of the original decision table could be made to look as follows:

```
.....:.....:.....:.....:  
: : : : :  
: C3 : w : w :  
: : .....:.....:.....:  
: C2 : p : q : t : - :  
: : .....:.....:.....:  
: : : : : : : :  
: C1 : k : 1 : k : 1 : - : - :  
:.....:.....:.....:.....:
```

Figure 5-5

The condition part of a decision table in which the conditions have been reordered twice.

In this way, the seemingly horrendous task of dealing with exhaustive, combinatorial decision tables can be reduced to a simple process. With the user in the loop the very large number of rules is usually pared down to a manageable set of ECs quite rapidly. The main purpose of this technique is to deal with all the possible alternatives in as simple a way as possible, without ever jeopardizing the completeness of the original exhaustive decision table.

5.2.2.2 The direct mode (exhaustive enumeration)

This mode is chosen when a problem definition already exists in the form of written laws, rules, regulations, procedures or a narrative description. We will call this the 'a priori' problem statement. The following steps are recommended when using the direct mode:

1. List the conditions and actions appearing in the 'a priori' statement.
2. Enumerate exhaustively the set of alternative states for each condition.
3. Complete the stubs of the table.
4. Complete the condition entries with all possible relationships between alternatives expressed.
5. Define logical expressions, including a logical expression for the impossibilities.
6. Complete the action entries, eliminating the impossibilities first.

7. Check the table for omissions, ambiguities, redundancies and contradictions. If there are no problems, then go to step 8; otherwise go to either step 1 or step 5.

8. Simplify the table (if possible).

An example will illustrate this process.

Step 1.

3 conditions (C_1 , C_2 , C_3) and 4 actions (A_1 , A_2 , A_3 , A_4) are identified.

Step 2.

The possible sets of alternative are:

- . For C_1 : $[k, l, m]$
- . For C_2 : $[p, q, t, s]$
- . For C_3 : $[w, \bar{w}]$

Step 3 and Step 4 (see figure 5-6)

:	:	:	:	:	:	:
:C1 :	k	:	l	:	m	:
:	:	:	:	:	:	:
:C2 :	p	:	q	:	t	:
:	:	:	:	:	:	:
:C3 :	w:w	:	w:w	:	w:w	:
:	:	:	:	:	:	:
:	:	:	:	:	:	:
:A1 :	:	:	:	:	:	:
:A2 :	:	:	:	:	:	:
:A3 :	:	:	:	:	:	:
:A4 :	:	:	:	:	:	:
:	:	:	:	:	:	:

Figure 5-6
Illustration of steps 3 and 4.

Step 5.

From the 'a priori' problem statement the following logical expressions are defined:

- If ($C_1=k$) and ($C_2 \neq p$) then A_1
- If ($C_1 \neq m$) and (($C_2=t$) or ($C_2=s$)) then A_2 and A_3
- If ($C_1=m$) and ($C_2=s$) then A_1 and A_3
- If ($C_1=k$) and (($C_2=t$) or (($C_2=q$) and ($C_3=w$))) then A_4

Assume from a semantic analysis of the conditions, the following combinations of condition values are determined to be impossible:

- ($C_1=k$) and ($C_2=s$)
- ($C_1=l$) and ($C_2=p$)

Step 6.

Eliminate the impossible rules first, then enter the actions required for each of the possible cases by considering the logical expressions one at a time. The following result is obtained (Figure 5-7).

	k	l	m	
:C1	:	:	:	
:C2	: p : q : t : s : p : q : t : s : p : q : t : s :			
:C3	: w:̄w :			
:	:	:	:	
:A1	: : x:x : x:x : x:x :			
:A2	: : : : x:x : : : : : : : : : x: x:x : x: : : : : : : : :			
:A3	: : : : : x:x : : : : : : x: x:x : x: : : : : : : : x:x :			
:A4	: : : x: : x:x :			
:	:	:	:	

Figure 5-7
Complete table with impossible rules removed.

Step 7.

This step is the most crucial one. It is meant to check whether the 'a priori' structure of the problem is sound and satisfactory. If it is not found satisfactory, a new iteration has to be made, starting from step 1 or from step 5.

The following two (syntactic) checks are recommended in all situations.

- Rules without actions which are not declared as impossibilities, are cases not explicitly dealt with in the 'a priori' problem statement. They have to be examined one at a time, because it is possible that some of these cases were unintentionally excluded when the 'a priori' structure was specified.
- The nature of the actions has to be examined to discover pairs of actions excluding each other. If such pairs are found, the rules in which both actions are specified are ambiguous (wrong) and this ambiguity has to be examined and resolved; e.g., if A1 and A4 exclude each other, three of the rules appearing in Figure 5-7 are ambiguous.

In addition, an overall semantic check is necessary. Suppose, for example, that each of the actions in Figure 5-7 represents the award of a specific premium to an employee being considered. There are two cases in which an employee will be awarded all four of the premiums. The user may decide that this is an undesired result and change the situation accordingly.

Step 8.

Assuming that no omissions, ambiguities or wrong effects are present in the table of Figure 5-7, this table may now be compressed to the one appearing in Figure 5-8.

:	:		:		:		:
:C1	:	k	:	l	:	m	:
:	:
:	:	:	:	:	:	:	:
:C2	: p	q	: t	q	: \bar{q}	: \bar{s}	: s
:	:
:C3	: -	w	: \bar{w}	-	-	-	-
:	:
:	:	:	:	:	:	:	:
:A1	:	x	: x	x	:	:	x
:	:	:	:	:	:	:	:
:A2	:	:	:	x	:	x	:
:	:	:	:	:	:	:	:
:A3	:	:	:	x	:	x	:
:	:	:	:	:	:	:	:
:A4	:	x	:	x	:	:	:
:	:

Figure 5-8
A compressed decision table where the 'impossibilities'
are eliminated.

When the decision logic is relatively simple, a shorter procedure than the one outlined above may be preferred. This shorter procedure could consist of the following steps.

- Steps 1-3: same as above
- Step 4: omitted
- Step 5: same as above
- Step 6: directly make a table containing complex rules
- Step 7: same as above
- Step 8: omitted.

Step 6 then is performed as follows:

- . for rule 1: . enter state 1 for condition 1
 - . for all other conditions enter a dash if the logic permits this; otherwise enter state 1 for those conditions
 - . enter the actions

- . for all other rules:
 - . look for the lowest condition which, in the preceding rule, is not yet in its last state and which does not carry a dash
 - . change that condition to its next state
 - . copy from the preceding rule the states of the conditions appearing at higher rows
 - . for the condition appearing at lower rows, enter a dash, if the logic permits this; otherwise enter state 1
 - . enter the actions
- . the decision table is combinatorically complete when a decision rule has been developed in which each condition is either in its last state or carries a dash.

Consider the following example as an illustration of the short procedure:

Step 1

4 conditions and 4 actions

Step 2

The possible sets of alternatives are:

- . for C1: [Y , N]
- . for C2: [K , L]
- . for C3: [Y , N]
- . for C4: [A , B , C]

Step 3 (complete stubs)

Not represented

Step 4 (omitted)

Step 5

If (C1 = Y) and (C2 = L) and (C3 = N) then A1
 If (C1 = Y) and (C2 = L) and (C3 = Y) and (C4 = C) then A2
 If (C1 = Y) and (C2 = L) and (C3 = Y) and (C4 = B) then A3
 If (C1 = Y) and (C2 = L) and (C3 = Y) and (C4 = A) then A4

Step 6

```
.....  

:C1  : Y : Y : Y : Y : Y : N :  

:C2  : K : L : L : L : L : - :  

:C3  : - : Y : Y : Y : N : - :  

:C4  : - : A : B : C : - : - :  

:.....:.....:.....:.....:  

:  :  :  :  :  :  :  :  

:A1  :  :  :  :  : x :  :  

:A2  :  :  :  : x :  :  :  

:A3  :  :  : x :  :  :  :  

:A4  :  : x :  :  :  :  :  

:.....:.....:.....:.....:
```

Sometimes, if the logic is relatively simple, even step 5 can be omitted without loss of effectiveness. The objective of the direct mode is to consider and record every combination of the alternatives for every condition to make the table fully complete.

5.3 SYSTEM DEVELOPMENT

For the purposes here the process of systems development has been broken into a number of discrete steps (albeit with blurred boundaries), which will be known as phases.

Phase one involves both the user and the producer and will be known as the System Specification Phase. In this first phase, the user's requirements are determined to a level of detail that will enable the producer to move into the next phase. In the second phase, the Systems Design Phase, the producer designs a systems solution to meet the requirements of the system specifications. During the Implementation Phase the producer constructs the operational system.

The fourth phase, Operation Phase, is the process of producing a set of directives to enable the correct use of the system. Phase five, User Acceptance Phase, is concerned with acceptance by the user of the system delivered by the producer. The object of this phase is for the users to satisfy themselves that the system delivered by the producer does in fact meet all the requirements agreed upon at the close of phase one.

Lastly, phase six, Maintenance and Extensions Phase, deals with the maintenance of the system. Indeed, changes, and extensions in its capabilities are almost always part of the systems life cycle.

5.3.1 System specification phase

In this phase the user's requirements are developed to a level of detail that will enable the producer to move into the system design phase without recontacting the user.

5.3.1.1 User to producer contract

If decision tables are used consistently during the System Specification Phase of a system development cycle, the end product will be a system of hierarchically structured decision tables, where each table at a lower level is nothing more than the elaboration of conditions or actions appearing in a table at a higher level.

Ideally, the tables will be in a context-free format, i.e., will only describe the procedures which the ultimate system has to follow, without making assumptions about the hardware and the software that will be used for implementing the system. This implies that they will not contain any actions or conditions related to transportation of data from and to files, or from and to data bases: the data needed for testing conditions are assumed to be available at all times. How this data is to be obtained efficiently, and how files are handled is a matter of design and this is not the concern of the user.

The system of tables obtained as an output of the System Specification Phase constitutes the contract between the user and the producer and may be signed by the user for agreement. It specifies, in an unequivocal and unambiguous way, the product which the producer should make.

The system of decision tables constituting the output of the System Specification Phase is also meant to be used in a later phase (User Acceptance Phase). Here, the user uses these tables for checking whether the system behaves according to the specifications.

5.3.1.2 Significance of the phase

This phase precedes all other phases and is restarted each time the problem changes.

The input to the phase may be an explicit problem definition that already exists in the form of written laws, rules, regulations or procedures. In other cases, the input may have to be created by a discussion between the user and the producer of the system. This phase may also use the prior version of itself, together with a new requirement as input.

It should be pointed out that in executing the System Specification Phase, the user and the producer need not be concerned with the efficiency of the resulting system. Such considerations are only relevant in performing the following phases.

The objective here is to produce a mutually agreed upon and mutually understood document that describes the user's problem and requirements.

In systems development, the exact definition of the system's requirements is a crucial matter. Life cycles of systems are usually represented as feedback loops in which, after having completed some later state, it may become necessary to return to the Systems Specification Phase. In order to reduce time and effort, the number of such

iterations should be minimized. Decision tables can be helpful in this respect, because their use in the Systems Specification Phase results in a better definition of the problem. This follows from their capability of exhaustively showing all cases, their capability of highlighting the effects of interactions within the system, and their ability to provide unambiguous and efficient means of two-way communication between the user and the producer.

Both the user and the producer take part in developing the system specifications. Depending upon the circumstances, several modes of cooperation may be used. At one end of the continuum, the user and the producer are the same person. At the other end, the producer iteratively develops a statement of requirements by interviewing the user. Between these two extremes, any degree of active participation by the user is conceivable.

In all cases, the user should review the output of this phase when it has been completed. The final decision tables should be a mutually satisfactory problem statement.

5.3.1.3 Methodology

The techniques described in 5.2.2 are used by the producer to obtain the requirements for the system specifications. The application of these techniques produce a set of decision tables that when considered in their entirety specify the problem. The producer must take these tables and analyze them for ambiguities and insure their completeness. The resulting tables are then restructured into a hierarchical tree structured network that makes a clear unified statement of the requirements. This is the final product of the system specifications phase.

The process of developing the system specification is described in the following sections. The process is illustrated further by means of a sample problem in section 5.3.1.3.4.

5.3.1.3.1 Understand the problem

Before the process of structuring the problem can start, the producer must understand the problem. This requires some analysis which will usually identify ambiguities. Very often, a useful approach to such an analysis will be to divide the problem statement into problem segments, each of which is then analyzed separately. These problem segments can be cast into decision tables. Depending upon the type of problem environment, the decision table will be constructed using the direct mode or the search mode as described in 5.2.2, and will be checked, corrected and rechecked with the user, until it satisfies both the user and the producer.

5.3.1.3.2 Structure the problem

At this point, an error-free problem statement acceptable to both user and producer has been achieved and problem restructuring can start. Usually, several attempts will be needed before the final structure is obtained.

The iterative process of producing the final structure of the problem by means of a system of decision tables can be described as follows:

First List all known facts (conditions, actions, sets of alternatives for conditions). These facts are discovered by the producer in the process of problem analysis.

Second Construct an overview table.

The structure of the overview table reflects the way in which the producer has partitioned the problem into manageable, comprehensible segments. Each condition and each action in the overview table is potentially such a segment.

The partitioning is accomplished by looking for those conditions and actions that seem to be of major or high level importance.

5.3.1.3.3 Elaborate the lower level tables

The top level table is usually not in sufficient detail to fully specify the desires of the user. Each condition and each action of the top level table may be elaborated into a separate decision table at a lower level. As needed, the elaboration may be several levels deep; deep enough to state the problem to the satisfaction of the user.

Having arrived at this point, the producer will often feel that the partitioning is not as good as it could be and therefore should be abandoned. He will then continue to reiterate between the previous paragraph and this one until the problem structure is acceptable to the producer and the user. At that point, system specifications are defined in unambiguous terms that the producer can use to begin the System Design Phase.

It should be remembered that in producing the system of decision tables each table can be constructed by using the direct mode technique or the search mode technique. The choice of technique depends upon the problem environment.

5.3.1.3.4 Illustration of the system specification process

To demonstrate the process described as the System Specification Phase, a non-trivial example has been chosen. This problem is stated originally in prose and then, using the direct mode method, translated into decision tables and a set of complete system specifications.

The example has three major parts:

- The problem as presented (paragraph 5.3.1.3.4.1)
- The problem as interpreted by the producer (paragraph 5.3.1.3.4.2)
- Restructuring the problem (paragraph 5.3.1.3.4.3)

It is carried through the System Specification Phase and the System Design Phase. To include the Implementation Phases would require giving the reader much information about a theoretical implementation environment that would overpower and cloud the issues we wish to describe.

5.3.1.3.4.1 The problem as presented

(Paragraph and sentence numbers have been assigned for reference purposes).

- Paragraph I
 1. This is a catalog mail order business in which all of the orders are handled on a charge basis.
 2. There is a file for each customer.
 3. Orders coming in are checked against each customer's account (the orders have been previously put in customer number order).
 4. Customer credit limits are set based on past payment performance and income; the credit limits are:
 - X (trial) with a \$900 limit
 - Y (trial) with a \$200 limit
 - Z (trial) with a \$100 limit
 - B (bad) no credit given
 - E (excellent) with \$2000 limit
 - G (good) with a \$900 limit
 - F (fair) with a \$500 limit
 - P (poor) with a \$100 limit
- Paragraph II
 1. Customer records are only reviewed when an order is received from the customer.
 2. When the account is less than or equal to 6 months old it is a trial account and retains its X, Y or Z rating.
 3. Z is for customers with an income of less than \$800 a month; Y is for customers with an income from \$800 through \$1200 a month; and X is for all others.
 4. A customer's order will be processed if it is within their monthly sales limit.
 5. If it does exceed the monthly sales limit, the order will be rejected on Form BA.

Q. Paragraph III

1. Once a trial account exists over 6 months, it receives a new rating if the customer has paid all bills within 30 days of receipt; if not, the account will receive a P rating.
2. Otherwise, the new ratings which may be assigned are: E for customers with incomes over \$3000 per month, G for families earning \$1000 to \$3000 per month, and F for all families earning less than \$1000 per month.

. Paragraph IV

1. This updating procedure only changes the credit rating of trial accounts.
2. When the account is over a year old and the customer has paid the last six bills within 20 days of receipt, the ceiling will be lifted an additional 30 percent if the credit rating is not P.
3. If the order exceeds the credit limit, it is necessary to reject it on Form BA.
4. A credit rating of B causes all orders to be automatically rejected on Form BD.
5. If no record exists for the customer, the order is rejected on Form BC.

. Paragraph V

1. Accepted orders are added to the current month's totals.
2. Billing and shipping must receive a copy of the order.
3. The customer is sent an acknowledgement of the order.

. Paragraph VI

1. Orders for customers living along the West coast are filled by the Los Angeles warehouse.
2. Customers in states west of the Mississippi have orders filled by the Omaha warehouse.
3. Customers on the East coast have orders filled by the Boston warehouse.
4. All other orders go to the Cleveland warehouse.
5. The exception is that all furniture must be ordered on a blue form (otherwise, the order must be rejected on Form BB) and shipped from Texas.

5.3.1.3.4.2 Problem interpreted by producer

In demonstrating the interpretation, the scope has been limited to the problem as stated. Such things as how a customer record gets on file initially, and how a bad credit rating is ever changed, have intentionally been excluded from the discussion.

Before problem structuring can start, the producer must understand the problem. This requires analysis of the problem and resolution of ambiguities resulting from loose use of terms by the user. Decision tables are helpful in this process. Here we have divided the problem statement into problem segments which are then analyzed separately. In the case at hand, the problem is viewed as consisting of 13 segments. The tables produced in this section were composed using the direct mode.

Problem segment 1 (Paragraph I Sentences 1-4)

This is introductory information with three pieces identified.

- . Customer file (customer number sequence)
- . Customer orders
- . Definition of the various credit ratings and their limits

Problem segment 2 (Paragraph II Sentence 1)

This is an informational summary of the overall function from here through Paragraph IV and establishes the order of reviewing prior to processing.

```
.....:.....:.....:  
: : :  
: C1 Time since application : Y Y N :  
: accepted over 6 months : :  
: CS Trial Account : Y N - :  
.....:.....:.....:  
: : :  
: A1 Keep rating : - X X :  
: A2 Review rating : X - - :  
.....:.....:.....:
```

Problem segment 3 (Paragraph II Sentence 2)

At this point, three questions arise:

- . Is a trial account synonymous with the fact that the application was accepted at most 6 months ago?
- . What is to be done when it has been more than 6 months since the application was accepted?
- . What is to be done with accounts more than 6 months old?

We will assume the answers from the user to these questions are - "As long as it has been 6 months or less since the application was accepted, it is a trial account. It stays a trial account until the first order is received beyond 6 months. Credit ratings are changed only on trial accounts, never for regular accounts."

Problem segment 4 (Paragraph II Sentence 3)

This is information only on some codes and does not add to the problem definition.

Problem segment 5 (Paragraph II Sentences 4 & 5)

At this point, two questions arise:

- . Is monthly sales limit the same as credit limit?
- . What is really meant by credit limit?
 - Amount of individual order exceeds credit limit
- or -
 - Amount of order plus all other orders this month exceeds credit limit
- or -
 - Amount of order plus outstanding balance exceeds credit limit.

Assume the answers are "Yes, the terms are the same and the limit is exceeded when the order plus the outstanding balance exceeds the credit limit for that customer."

```
.....:  

: :  

: Does value of order plus : :  

: outstanding balance exceed : :  

: credit limit : Y N :  

:.....:  

: :  

: Process order : X :  

: Reject on BA : X :  

:.....:
```

Problem segment 6 (Paragraph III Sentences 1 & 2)

At this point, two questions arise:

- . To what does "otherwise" refer?
- . Is this possibly a repeat of part or all of problem segment 3?

Assume the answers to be: "The otherwise refers to the "paid all bills within 30 days" and "This" is really details of the review called for in problem segment 3."

```
.....:  

: : : : :  

: C1 : Have all bills been paid : Y : Y : Y : N :  

: : in less than 30 days? : : : : :  

: : : Less : $ 900 to : Over : - :  

: C2 : What does customer earn? : $900 : $3000 : $3000 : :  

:.....:  

: : : : :  

: A1 : Give rating of : F : G : E : P :  

:.....:
```

Problem segment 7 (Paragraph IV Sentence 1)

The "THIS" obviously refers back to paragraph III in its entirety.

Problem segment 8 (Paragraph IV Sentence 2)

```
.....:  

: RATING : NOT=P : NOT=P : NOT=P : =P :  

: : : : :  

: ACCOUNT : OVER : OVER : LESS : - :  

: AGE : 1 YR : 1 YR : 1 YR : :  

: : : : :  

: LAST 6 : PAID : PAID : : :  

: BILLS : LESS OR: OVER : - : - :  

: : = 20 DA: 20 DA: : :  

:.....:  

: : : : :  

: ADD : : : :  

: 30% TO : : : :  

: CEILING : YES : NO : NO : NO :  

:.....:
```

At this point three questions arise:

- . Is this the definition of special cases?
- . Is the 30 percent for this month only, or is it a permanent lifting of the ceiling?
- . Is "ceiling" the same as "credit limit"?

Assume the answers to be "Yes, this defines "special cases", the 20 percent is temporary (checked for each order) and the "ceiling" refers to the actual value for the credit limit."

Problem segment 9 (Paragraph IV Sentence 3)

This ties together the order of the overall review:

- . Trial account review
- . Normal credit limit review
- . Special case review

It specifies the rejection on Form BA when all reviews fail.

Problem segment 10 (Paragraph IV Sentences 4 & 5)

	R1	R2	R3
.....			
: DOES CUST :	N	Y	Y :
: HAVE RECORD :			:
: IS CREDIT :			:
: :			:
: RATING B :	-	Y	N :
.....:			:
: :			:
: REJECT :	BC	BD	NO :
.....:			:

Problem segment 11 (Paragraph V Sentences 1-3)

It is assumed that this is what is meant by "process the order" in earlier paragraphs.

.....				
: ADD ORDER TO OUTSTANDING BAL. :	X	:		
: ONE COPY TO BILLING :	X	:		
: ONE COPY TO SHIPPING :	X	:		
: ACKNOWLEDGE TO CUSTOMER :	X	:		
.....:				:

Problem segment 12 (Paragraph VI Sentences 1-4)

.....					
:CUSTOMER :	ALONG	ELSEWHERE	NEW YORK	ELSEWHERE :	
:LIVES :	PACIFIC	WEST MISS.	OR N.ENG.	EAST MISS.:	
.....					
:SEND :	LOS	OMAHA	BOSTON	CLEVELAND :	
:ORDER :	ANGELES				:
:TO :					:
.....:					:

At this point two questions arise:

- . Is this the detail disposition of the "shipping copy" in Problem segment 11?
- . Can there be any orders from outside continental United States?

Assume the answer to be "Yes, this is the shipping copy" and "No orders will be received from outside the continental United States."

Problem segment 13 (Paragraph VI Sentence 5)

At this point, a question arises: Are blue forms only for ordering appliances? Assume the user's answer to be "Yes".

```
.....  
: ORDER IS FOR APPLIANCE : Y Y N N :  
: ORDER IS ON BLUE FORM : Y N Y N :  
:.....:  
:  
: SEND ORDER TO TEXAS : X - - - :  
: REJECT ON FORM BB : - X X - :  
: PROCESS NORMALLY : - - - - X :  
:.....:
```

5.3.1.3.4.3 Restructuring the problem

This section demonstrates the process of synthesizing a definitive statement of the problem. The process used is restructure the problem components into a hierarchical structured network. This is usually an iterative procedure in real life, and so it is shown here as a multiple attempt process leading to the final solution.

5.3.1.3.4.3.1 Initial attempt

The initial attempt at restructuring the problem consists of a series of steps described in 5.3.1.3.1. In brief they are:

- . List all known facts (conditions, actions, condition alternatives)
- . Construct overall table
- . Elaborate conditions and actions
- . Abandon fruitless tables and redo them.

5.3.1.3.4.3.1.1 List all known facts

The analysis described in 5.3.1.3.4.2 provides all the known facts.
List separately all of the pertinent:

- Conditions and the set of alternatives of each
- Actions

These are:

- List of conditions and alternatives

<u>Condition</u>	<u>Alternative Values</u>
1 - Does customer account exist?	(Yes, No)
2 - What is the age of account?	(6 mo. or less, over a year, over 6 mo. thru a year)
3 - How does balance plus order compare to credit limit?	(greater than or not greater than)
4 - Is the trial account to become normal?	(Yes, No)
5 - What is the family income?	(over \$3000, \$1000 or less, over \$1000 thru \$3000)
6 - Is it a special case?	(Yes, No)
7 - Is credit level bad?	(Yes, No)
8 - Is order accepted?	(Yes, No)
9 - Where does customer live?	(along Pacific (P), elsewhere west of Miss. (W), NY or NE (C), elsewhere east or Miss. (E))
10 - What is the order for?	(appliance, other)
11 - What color is order form?	(blue, other)
12 - Credit level is?	(X, Y, Z, B, E, G, F, P)

- List of actions

- 1.- Review customer record
- 2 - Keep rating
- 3 - Reject order (various forms)
- 4 - Assign new credit limit
- 5 - Raise ceiling 30 percent
- 6 - Add amount of order to balance
- 7 - Send copy of order to shipping
- 8 - Send copy of order to billing
- 9 - Send acknowledgement to customer
- 10 - Send copy of order to Los Angeles
- 11 - Send copy of order to Omaha
- 12 - Send copy of order to Boston
- 13 - Send copy of order to Cleveland
- 14 - Send copy of order to Texas
- 15 - Add order to months total ordered

5.3.1.3.4.3.1.2 Construct an overall table

The producer must now synthesize the conditions and actions into a related set of decision tables. The first assumption is to construct a single decision table out of this problem. However, simple calculation reveals that there are 73,728 possible combinations for the condition values listed. This would result in a decision table of 73,728 rules. Obviously, that would be a table beyond human abilities to comprehend it.

The producer is faced with partitioning the conditions and actions into separate tables to solve the problem. This is a subjective process that depends on the experience of the producer. The proof of the process is the users acceptance of the result.

The producer begins the partitioning by selecting the condition and action that seem to have the greatest significance to the whole problem. Next find the conditions and actions that most directly relate to the selected ones and group them together into a single overall table. The yardstick for significance is level of control, that is finding the conditions that make the principal decisions of the system. This is often an iterative process in which the first choice will be wrong but leads to a better choice. The key consideration is to partition the problem into manageable comprehensible segments. Using this process on the example it appears that:

- Condition 8 (is order accepted) seems to be a higher level condition to which conditions 1, 3, 6, 7, 10, 11 and 12 contribute the answer.
- Action 1 (review customer record) seems to be a higher level action to which conditions 1, 2, 4, 5 and actions A2 and A4 contribute the details.

With no other groupings immediately apparent, list condition 8 and the other conditions not a part of condition 8 or action 1. Include condition 1 because it contributes to both condition 8 and action 1. These then, are conditions 1, 8 and 9. Similarly, we arrive at actions 1, 3 and 5 to 15. This reduces the number of rules to 16 which is more manageable. Using these, start to construct a decision table as described in 5.2.2.2 (Direct Mode).

```
.....  
: OVERALL TABLE : :  
:.....:  
: C1 DOES ACCOUNT EXIST : Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y Y :  
: C8 IS ORDER ACCEPTED : Y Y Y Y N N N N Y Y Y Y N N N N :  
: C9 CUSTOMER LOCATION : P W C E P W C E P W C E P W C E :  
:.....:  
: A1 REVIEW CUSTOMER RCD : X X X X :  
: A2 REJECT ORDER : X X X X X X X X X X X X :  
: A5 RAISE CEILING 30% : :  
: A6 ADD ORDER TO BAL. : :  
: . : :  
: . : :  
: . : :  
:.....:
```

At this point, the producer sees that action 5 also contributes to condition 8 and that there is an additional action 16 (process the order) to which condition 9 and actions 6 through 15 contribute. Therefore, we abandon this table. This gives conditions 1 and 8 and actions 1, 3 and 5 with which to build an overall table using the same technique (Direct Mode).

```
.....  
: OVERALL TABLE : :  
:.....:  
: C1 DOES ACCOUNT EXIST : Y Y N N :  
: C8 IS ORDER ACCEPTED : Y N Y N :  
:.....:  
: A1 REVIEW CUSTOMER RECORD : X :  
: A2 REJECT ORDER : X X X :  
: A16 PROCESS THE ORDER : X :  
:.....:
```

This is then compressed to the final decision table form.

```
.....  
: OVERALL TABLE : :  
:.....:  
: C1 DOES ACCOUNT EXIST : Y Y N :  
: C2 IS ORDER ACCEPTED : Y N - :  
:.....:  
: A1 REVIEW CUSTOMER RECORD : X :  
: A2 REJECT ORDER : X X :  
: A3 PROCESS THE ORDER : X :  
:.....:
```

5.3.1.3.4.3.1.3 Elaborate conditions and actions

The overall table produced in the preceding section is obviously not in sufficient detail to fully specify the desires of the user. The next process then is to elaborate each condition and action in more detail.

- Condition C1 of overall table. Remember from the previous partitioning that none of the other listed conditions or actions contribute to this condition, thus it becomes a simple one condition table.

.....
: DOES ACCOUNT EXIST	:	:
.....
:	:	:
: C1 DOES THE CUSTOMER NUMBER	:	:
: MATCH A RECORD ON FILE	: Y N	:
.....
: A1 ACCOUNT EXISTS	: X -	:
: A2 ACCOUNT DOES NOT EXIST	: - X	:
: A3 REJECTION FORM IS BC	: - X	:
.....

- Condition C2 of overall table. Remember from the partitioning developed in the overall table that the listed conditions 3, 6, 7, 10, 11 and 13 and action 5 contribute to determining this condition. From this we compose the following table.

.....
: IS ORDER ACCEPTED	:	:
.....
: C1 IS ORDER PLUS BALANCE	: Y Y Y Y Y Y Y Y Y Y Y Y Y Y	:
: GREATER THAN CREDIT LEVEL	:	:
: C2 IS IT A SPECIAL CASE	: Y Y Y Y Y Y Y Y N N N N	:
: C3 IS CREDIT LEVEL BAD	: Y Y Y Y N N N N Y Y Y Y	:
: C4 IS ORDER FOR FURNITURE	: Y Y N N Y Y N N Y Y N N	:
: C5 IS ORDER FORM BLUE	: Y N Y N Y N Y N Y N Y N	:
.....
: A1 RAISE CEILING 30%	: - - - X X X X - - -	:
: A2 REJECTION FORM IS BD	: X X X X - - - X X X X	:
: A3 REJECTION FORM IS BB	: - X X - - X X - - X X -	:
: A4 ORDER IS ACCEPTED	: - - - X - - X - - -	:
.....

At this point, the producer feels that this approach to "Is the order accepted" should be abandoned, because further study shows that conditions 2 and 4 and action 4 also seem to be related to this question, and, therefore, we have conditions 2, 3, 4, 6, 7, 10 and 11 and actions 5 and 4 to consider. Additionally, it seems that a further partitioning may give an improved table. Analysis leads to seeing conditions 2, 7, 10 and 8 as being of a higher level of significance than the others. So building the table with them as a basis results in:

```
.....: IS THE ORDER ACCEPTED :  

.....:  

: C1 Is credit level bad : Y N N N N N N N N :  

: C2 Is order for furniture : - Y Y Y Y N N N N :  

: C3 Is order form blue : - Y Y Y N Y N N N :  

: C4 Is account over 6 months : - Y Y N - - Y Y N :  

: C5 Is account over 1 year : - Y N - - - Y N - :  

.....:  

: A1 Rejection form is BD : X - - - - - - - - :  

: A2 Rejection form is BB : - - - - X X - - - - :  

: A3 Do trial account test : - - - X - - - - X :  

: A4 Do regular account test : - - X - - - - X - :  

: A5 Do special case test : - X - - - X - - - :  

.....:
```

All the conditions are sufficiently detailed; however, actions A3, A4 and A5 need elaboration. At this lower level of detail we find some detail conditions not listed in the first step.

- Elaboration of action A3 in "Is the order accepted". Focusing attention on trial accounts, we find the credit levels that apply are Z or Y or X. This is really a detail of the condition 3 listed along with the values they represent. Build the table using these conditions:

```
.....:  

: TRIAL ACCOUNT TEST :  

.....:  

: C1 : Rating is : Z : Z : Y : Y : X : X : Other :  

: C2 : Balance plus : over : Not : Over : Not : Over : Not :  

: : order is : 100 : over : 200 : over : 900 : over : - :  

: : : 100 : : 200 : : 900 : :  

.....:  

: A1 : Accept order : - : X : - : X : - : X : - :  

: A2 : Rejection : X : - : X : - : X : - : - :  

: : form is BA : : : : : : : : : :  

: A3 : Erroneous : - : - : - : - : - : - : - : X :  

: : rating : : : : : : : : : :  

: : SYS ERR : : : : : : : : : :  

.....:
```

Notice we have uncovered a possible system error. The handling of it has been deferred to a later phase.

- Elaboration of action A4 in "Is the order accepted". As was pointed out in the previous table, we are working with details of condition 3 for established accounts for which the credit levels can be P or F or G or E.

In this attempt to build the table, the producer feels that the partitioning being developed is not as good as it could be and should be abandoned in total. A fresh start on the whole problem should be made. The table should not be destroyed, since the details shown are useful for reference. Also, and probably more important, it represents discarded ideas that might be repeated if not recorded.

5.3.1.3.4.3.2 Final Attempt

In the previous section, we saw several cases of abandoning individual tables. This is the result of recognizing that the partitioning done to arrive at the highest level table probably has some major faults that lead to difficulties in the lower level tables.

As a result of going through such iterations, the facts and intricacies of the problem are more thoroughly understood. This provides the producer with the background information needed to achieve insight into a final solution. This is a process that often happens in real life situations. It is a necessary approach, because it gives the producer multiple ways of seeing the problem and thus look for the best solution.

The producer once again reviews the problem to identify the major tasks to be performed. In this way the producer identifies six possible dispositions of the orders:

- Process the order
- Update the customer record
- Reject the order on Form BA
- Reject the order on Form BB
- Reject the order on Form BC
- Reject the order on Form BD

The first two have many details stated in the problem and will require much elaboration while the last four are fairly straightforward.

Similarly, the producer has identified the four high level conditions necessary to select the actions:

- Does customer account exist
- Is the credit rating good
- Is the order in acceptable format
- Is the order within credit limit

The first two require almost no elaboration, the third some, while the fourth is a major part of the problem. all four have only two alternatives each (Y or N). This gives an overall table of sixteen simple rules.

With this reanalysis of the partitioning, the producer proceeds to construct the overall table and expand it through the elaboration process:

- Overall table.
More familiarity and experience have shown that certain conditions usually override others, therefore, these conditions have been sequenced. The table has been built using the direct mode.

```
.....: OVERALL TABLE :  
.....:  
: C1 Does customer account exist : Y Y Y Y N N :  
: C2 Is credit rating good : Y Y Y N - - :  
: C3 Is order in acceptable format : Y Y N - Y N :  
: C4 Is order within credit limit : Y N - - - - :  
.....:  
: A1 Process the order : X - - - - :  
: A2 Update customer record : X - - - - :  
: A3 Reject on Form BA : - X - - - - :  
: A4 Reject of Form BB : - - X - - X :  
: A5 Reject on Form BC : - - - - X X :  
: A6 Reject on Form BD : - - - X - - :  
.....:
```

This table was built in the first attempt, and actually results in a superior partitioning. Also note that the sixth rule calls for rejection on 2 forms. This possible contradiction has not been evident through the analysis of the initial solution attempt and shows how tables uncover such things. A review with the user is appropriate to resolve such items.

- Elaboration of condition C1 in the "overall table". There is one simple condition which elaborates this into a single condition table.

```
.....:  
: Does customer account exist :  
.....:  
: Is there a customer record :  
: with the same account : Y N :  
: number as on the order? :  
.....:  
: Customer account exists : X :  
: Customer account does not exist : X :  
.....:
```

- Elaboration of condition C2 in the "overall table". Again we have a single condition table.

```
.....  
: Is credit rating good :  
:.....  
: Is credit level = B : Y N :  
:.....  
: Credit rating is good : X :  
: Credit rating is not good : X :  
:.....
```

- Elaboration of condition C3 in the "overall table". There are two conditions on which to base this decision.

```
.....  
: Is the order in acceptable format:  
:.....  
: The order form is : BLUE BLUE OTHER OTHER :  
: The order is for : FURNITURE OTHER FURNITURE OTHER :  
:.....  
:  
: The format is acceptable : X X :  
: The format is not acceptable : X X :  
:.....
```

- Elaboration of condition C4 in the "overall table". This can be clearly represented as a sequence of three actions. Thus, we build the action only table:

```
.....  
: Is order within credit limit :  
:.....  
: A1 Establish credit rating : X :  
: A2 Look up credit limit value : X :  
: A3 Test account balance : X :  
:.....
```

This last table is the only one of the four which does not go at least to the level of detail given in the statement of the problem. These actions require elaboration.

- Elaboration of A1. The function of this table is to determine if the credit level should be changed, and if so, to what. All conditions affecting this are used to build the table.

```
.....  
: Establish credit level :  
.....  
: Is account over 6 months : Y Y Y Y Y N :  
: Is credit level = (X or Y or Z) : Y Y Y Y N - :  
: All bills paid in 30 days : Y Y Y N - - :  
: Customer earns : 1000 or Over 100 3000 - - - :  
: : Less Less 3000 or Over :  
.....  
:  
: Use credit rating of : F G E S - - :  
: Use existing credit rating : - - - - X X :  
.....
```

- Elaboration of A2. The tabular list provided is used to build the table.

```
.....  
: Look up credit limit value :  
.....  
:  
: Credit rating = : Z Y X P F G E :  
.....  
:  
: Use limit of : $100 $200 $900 $100 $500 $900 $2000 :  
.....
```

- Elaboration of A3. This is the numerical calculations required to answer the condition. All conditions affecting this are used to build the table.

```
.....  
: Test the balance :  
.....  
: Balance + order < Limit : Y N N N N N :  
: Credit level = P : - Y N N N N :  
: Account over 1 year : - - Y Y Y N :  
: Bills paid in 20 days : - - Y Y N - :  
: Balance + order : - - Y N - - :  
: Greater 130% limit : :  
.....  
:  
: Limit is exceeded : - X X - X X :  
: Limit is not exceeded : X - - X - - :  
.....
```

This completes the elaboration of conditions from the overall table.

- Elaboration of action A1 in "overall table". Actual processing of the order depends on where the customer lives and what is ordered.

```
.....  
: Process the order :  
.....  
: Order for appliance: Y : N : N : N : N : N :  
:  
: Customer resides : - : ALONG : ELSEWHERE : NEW YORK : ELSEWHERE :  
: : PACIFIC : WEST OF : OR : EAST OF :  
: : : MISS. : NEW ENG. : MISS. :  
.....  
: : LOS : : : :  
: Ship copy to : TEXAS : ANGELES : OMAHA : BOSTON : CLEVELAND :  
:  
: Copy to billing : X : X : X : X : X : X :  
:  
: Acknowledgment : : : : : : :  
: to customer : X : X : X : X : X : X :  
.....
```

- Elaboration of action A2 in "overall table". The customer record is updated to show the affect of the order and possibly a revised rating.

```
.....  
: Update a customer record :  
.....  
: Account over 6 months : Y Y N :  
: Credit level = (X or Y or Z) : Y N - :  
.....  
: Assign new credit level : X :  
: Add amount of order to balance : X X X :  
.....  
.....  
: Assign new credit level :  
.....  
: Paid all bills within 30 days : Y : Y : Y : N :  
: Customer monthly income is : 3000 : : :  
: : >3000 : >1000:<1000: - :  
.....  
: Assign credit rating : E : G : F : P :  
.....
```

As previously identified actions A3, A4, A5, A6 in the "overall table " will not be further elaborated. This then completes the specification assuming a review with the user uncovers no changes.

5.3.1.3.4.3.3. The final solution

The producer has partitioned the problem sufficiently to present it to the user. If the user agrees with the problem statement, the producer can complete the system specifications by adding the details necessary to finalize the system specifications. This would include such things as where data is stored, how the forms are rejected, etc.. When all this information is compiled, it forms the complete system specifications. After the user has reviewed and approved it, the system specifications are passed to the System Design Phase.

The following figures represent the final solution for the problem. They contain the hierarchical structure (Figure 5-9) and component decision tables (Figures 5-10 through 5-18) contained in the system specifications.

```
.....  
: OVERALL TABLE :  
.....  
:  
.....  
:  
:  
:  
.....  
.....  
:  
:  
:  
:  
.....  
.....  
:  
:  
:  
:  
.....  
.....  
:  
:  
:  
:  
.....  
.....  
:  
:  
:  
:  
.....  
.....  
:  
:  
:  
:  
.....
```

Figure 5-9
Final Problem Hierarchy

```
.....  

:          OVERALL TABLE :  

:.....  

: C1 Does order match customer record on file : Y Y Y Y N N :  

: C2 Does credit rating = B                  : Y Y Y N - - :  

: C3 Is order in acceptable format          : Y Y N - Y N :  

: C4 Is order within credit limit           : Y N - - - :  

:.....  

: A1 Process the order                      : X - - - - :  

: A2 Update customer record                 : X - - - - :  

: A3 Reject on form BA                     : - X - - - :  

: A4 Reject on form BB                     : - - X - - X :  

: A5 Reject on form BC                     : - - - - X X :  

: A6 Reject on form BD                     : - - - X - - :  

:.....
```

Figure 5-10

```
.....  

: Is order in acceptable format :  

:.....  

: C1 The order is for      : Furniture : Other :  

:.....  

: C2 the order form is    : Blue : Other : Blue : Other :  

:.....  

: A1 The format is acceptable : X : : X :  

: A2 The format is not acceptable : : X : X : :  

:.....
```

Figure 5-11

```
.....  

: Is order within credit limit :  

:.....  

: A1 Establish credit rating   : X :  

: A2 Look up credit limit value : X :  

: A3 Test the balance        : X :  

:.....
```

Figure 5-12

```
.....  

: Establish credit rating :  

:.....  

: C1 Is account over 6 months   : Y     Y     Y     Y Y N :  

: C2 Is credit level = (X, Y, OR Z) : Y     Y     Y     Y N - :  

: C3 All bills paid in 30 days   : Y     Y     Y     N - - :  

: C4 Customer earns             : 1000 or Over 1000 Over - - - :  

:                               : Less    Less 3000 3000 :  

:.....  

: A1 Use credit rating of       : F     G     E     P - - :  

: A2 Use current credit rating  : -     -     -     - X X :  

:.....
```

Figure 5-13

```
.....  

: Look up credit limit :  

:.....  

: C1 Credit rating : N T K P F G E :  

:.....  

: A1 Use limit of : $100 $200 $900 $100 $500 $900 $2000 :  

:.....
```

Figure 5-14

```
.....  

: Test account balance :  

:.....  

: C1 Balance + order <= limit : Y N N N N N :  

: C2 Credit rating = P : - Y N N N N :  

: C3 Account over 1 year old : - - Y Y Y N :  

: C4 Bills paid in 20 days : - - Y Y N - :  

: C5 Balance + order >  
130% of limit : - - Y N - - :  

:.....  

: A1 Limit is exceeded : - X X - X X :  

: A2 Limit is not exceeded : X - - X - - :  

:.....
```

Figure 5-15

```
.....  

: Process the order :  

:.....  

: C1 Order in for furniture : Y N N N N N :  

: :  

: C2 Customer resides : - Along Other New York Other :  

: : Pacific West or East :  

: : Miss. New Eng. Miss. :  

: :  

:.....  

: A1 All send ship copy to : Texas Los Ang Omaha Boston Cleveland :  

: :  

: A2 Add copy of billing : X X X X X :  

: acknowledgment to :  

: customer :  

:.....
```

Figure 5-16

```
.....  

: Update Customer Record :  

.....  

: C1 Account over 6 months old : Y Y Y :  

: C2 Credit level = (X, Y, or Z) : Y N - :  

.....  

: A1 Assign new credit rating : X - - :  

: A2 Add amount of order to balance : X X X :  

.....
```

Figure 5-17

```
.....  

: Assign new credit rating :  

.....  

: C1 Paid all bills withing 30 days : Y Y Y N :  

: C2 Customer income/month :>3000 3000 <1000 - :  

.....  

: A1 Assign credit rating : E G F P :  

.....
```

Figure 5-18

5.3.1.3.4.3.4 Conclusions

The hierarchy and associated decision tables are the final system specifications. They describe a complete and unambiguous statement of the problem and represent it in a structure that can easily be understood and verified.

The process demonstrated in the example indicates that the correct and complete interpretation of a problem statement is not a simple task. It is easy to make assumptions and accept incorrect or partial statements if a rigorous form like decision tables is not used to express the problem.

5.3.2 System design phase

The system design is an architectural blueprint for a system solution that meets the user's needs. In this phase the producer uses the requirements documented in the system specifications to define what the system product must do. He then defines a solution for the operational environment that will perform the required activities.

The producer applies knowledge of the operational environment to develop the system design. The operational environment can be completely manual, in which people perform all the tasks, or fully automated. For manual systems, the system design is an architecture concerned with human engineering. The producer develops a system solution which people can use effectively. For such a system the logical relationships between conditions and the resulting actions must be written so they are easy for people to read and process. The producer can accomplish this with decision tables by incorporating the principles described in section 5.2.1. This will result in an effective system design for a manual system.

When the system design is for an automated operational environment, the producer is concerned with fulfilling the system specifications with computer software and hardware. The operational environment is a specific computer, and the system design is for computer programmers to develop the software. In this case the producer is concerned with developing a system design that is easy to translate into computer programs, which will use the computer resources efficiently, and which can be maintained and enhanced at a reasonable cost.

The more complex of the two types of operational environments is the automated one. Many of the concepts and approaches from it are also useable for the manual environment. Thus the principal emphasis in this section will be on developing a system design for an automated environment.

5.3.2.1 Product of this phase

The system design is a solution to the requirements in the system specifications. The solution is in the form of a blueprint which will be constructed by the Implementation Phase, the next phase, into an operational product.

The level and depth of the system design depends on the operational environment for which it will be developed. For a totally manual operational environment, the system design might define every step in the process to be performed by people. This would eliminate the need for an Implementation Phase since the system design would be directly useable.

Computer applications require the producer to develop a system design for a computer environment. The system design specifies the architecture for the computer software that will be constructed by the Implementation Phase. The system design will specify the computer programs in terms of the functions they will perform and relationships between them.

The architecture of a system design for a computer application recommended here is a hierarchical structure of decision tables. This structure is used because it has been shown to be an effective way of representing software in several disciplines of structured design methodologies. Decision tables are naturally compatible in the hierarchical structure, because they lead to a natural tree structure through elaboration of conditions and actions into subtables.

5.3.2.2 Method used in this phase

The producer builds the system design as an extension of the system specifications. The basic difference between the two documents is that the system design adds procedural software details to the system specifications. This is done by translating the wording of the condition and action stubs into software terms that will be better understood in the Implementation Phase, and changing or adding decision tables that will control the processing of the system.

The producer translates the system specifications into the system design in a six step process:

1. Analyze the system specifications for completeness.
2. Add the movement of data to the system specifications structure.
3. Add iterations to the system specifications structure.
4. Define the system architecture for more effective computer processing.
5. Add data access and process control.
6. Partition into program modules.

The objective of this process is to design a system of functions that perform the desired operations. The result may be the design for a single or many computer programs. It may run as a continuous process or be broken up into separately executed steps. Only the last step in the process defines the physical partitioning of the system. The preceding steps are the same regardless of the size or modularity of the end result.

5.3.2.2.1 Analyze the system specifications for completeness

The producer can only develop an effective system design if the system's requirements are complete and well documented. Section 5.3.1 described how to develop a system specifications document that meets these objectives. It is assumed that the producer has such a document to work from. If he does not, then the steps defined in Section 5.3.1 should be performed before the formal system design can begin.

With the system specifications in hand, the producer must review it in detail to insure it contains all criteria relevant to the end computer product. Such items as expected terminal response time, desired execution time, predicted frequency of changes to the system, etc., must be included in the system specifications if they do not already appear in it.

The producer must also be concerned with the physical environment in which the product will run, and the tools that are available to build it. The physical environment includes the computer hardware, and its operating system and user support software (e.g. data base, utility software, etc.). The tools for building the software include a decision table processor (if available), the programming language and its compiler, and the skill level of the programmers.

In developing the system design, the producer must know the physical environment to understand the limitations of the environment in which the system will run. The producer uses knowledge of the tools available to build the system, and to provide a vocabulary and grammar in which to express the system design. This is common to any software development whether decision tables are or are not used, and consequently will not be considered in any depth here.

5.3.2.2.2 Add the movement of data

The next step in developing the system design is for the producer to understand the flow of data through the system. It is necessary to know how and where data must be provided. This information will be used by the producer to define the computer data interfaces needed by the system.

The producer begins by depicting the data flowing through the various functions defined in the system specifications. Representing the system as a black box, each source of data or generated output can be depicted graphically; diagram 5-19 does this for the system specifications example of Section 5.3.1.3.4.3.3, Figure 5-9. This is followed by depicting the data processed by each functional node of the system, Figure 5-21. The result is the hierarchical structure of the system specifications with external and internal data represented, as in Figure 5-22.

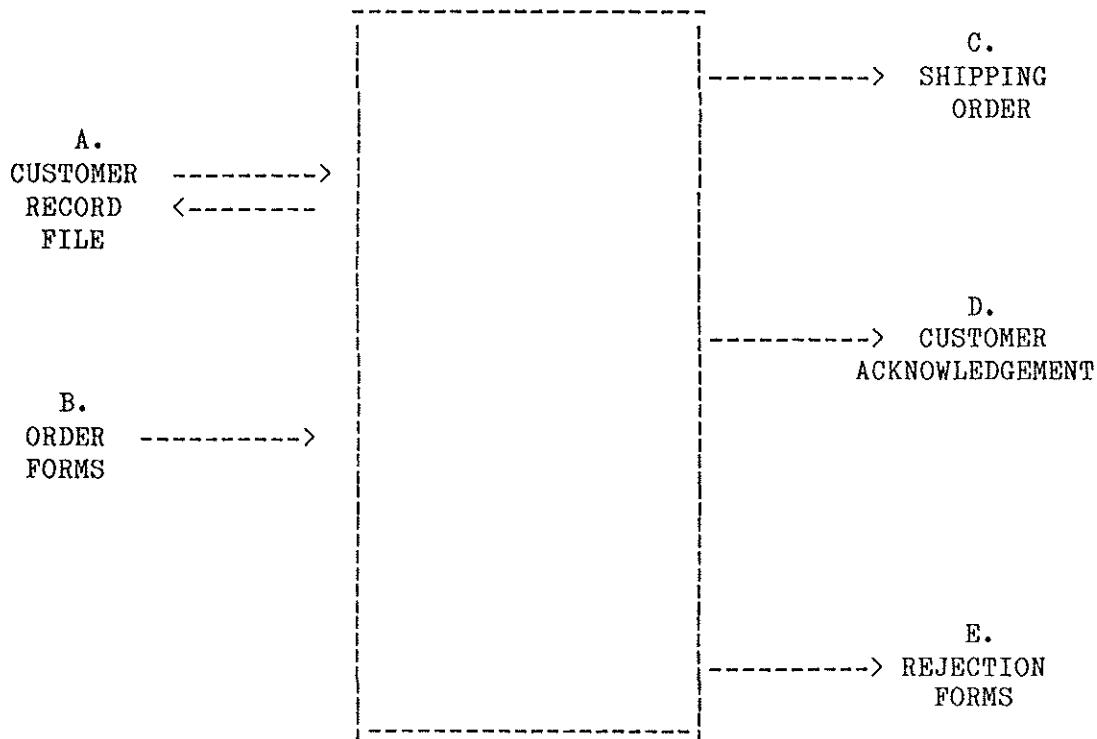


Figure 5-19
Data Sources for Figure 5-9 System Specifications

```
.....  
A -----> : OVERALL TABLE : -----> E  
B -----> .....  
.....  
B -----> : Is order :  
          : in      :  
          : acceptable :  
          : format   :  
.....  
.....  
          : Is order :  
          : within   :  
          : credit    :  
          : limit     :  
.....  
.....  
A -----> : Process  : -----> C  
          : the      :  
B -----> : order    : -----> D  
          : format   :  
.....  
.....  
A -----> : Update   :  
          : customer : -----> A  
B -----> : record   :  
.....  
.....  
A-----> : Establish :  
          : credit    :  
          : limit     :  
.....  
.....  
A-----> : Look up  :  
          : credit    :  
          : limit     :  
.....  
.....  
A-----> : Test     :  
          : account  :  
          : balance  :  
.....  
.....  
A-----> : Assign   :  
          : new      : -----> A  
          : credit   :  
          : rating   :  
.....
```

Figure 5-20
External Interfaces to Nodes (From Figure 5-19)

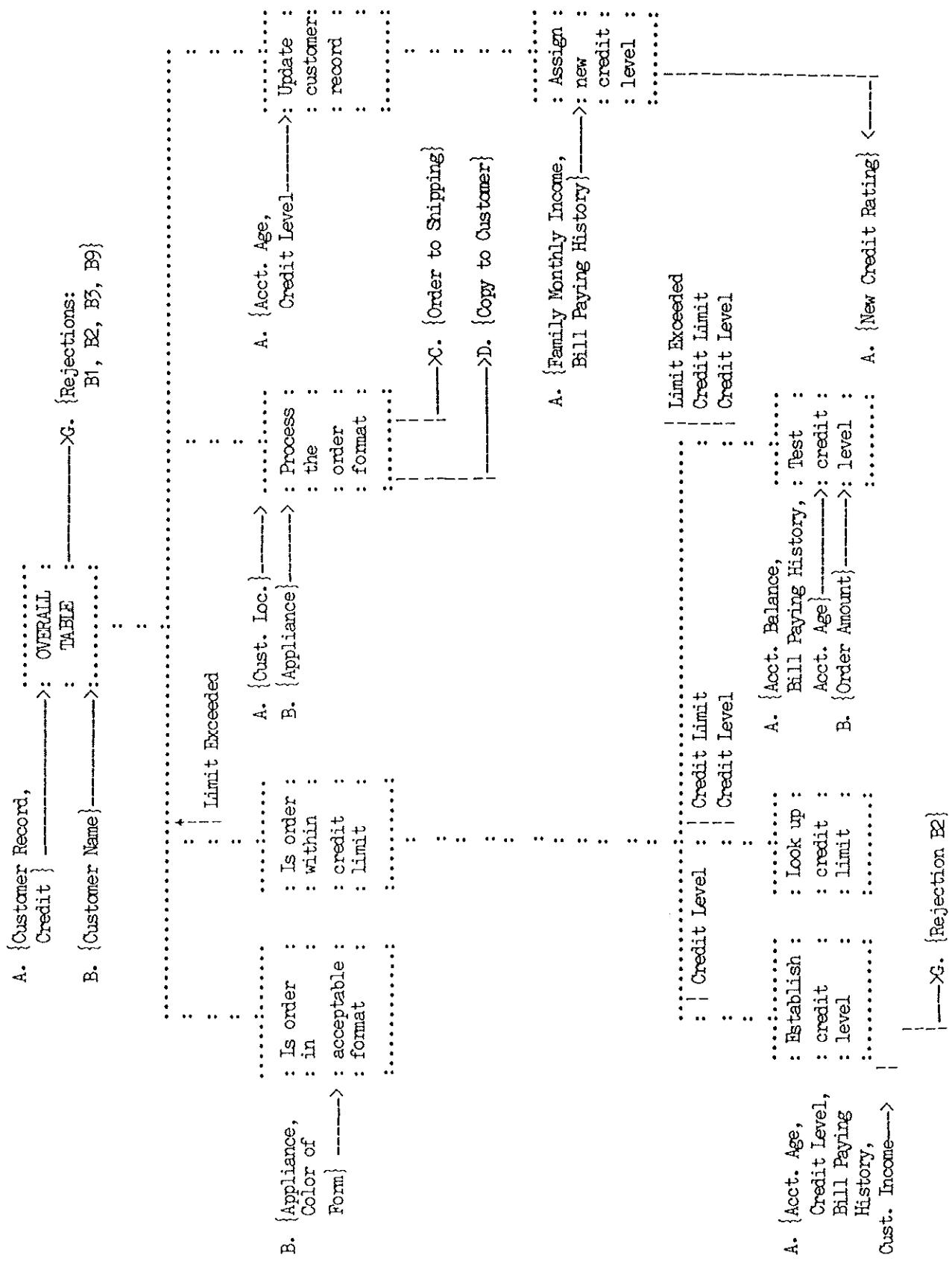


Figure 5-21
External and Internal Data Added to System Specifications Structure

5.3.2.2.3 Add iterations

The producer must understand the iterative portions of the system and include this in the system's design. The objective of a system's design that seeks to minimize computer execution time is to minimize both the number of instructions that need to be executed and the handling of data. In order to do this, the producer must understand the nature of the iterations specified by the system specifications. This can be accomplished by using arrows around the branches to nodes in the hierarchical structure diagram to graphically represent iterations in the system. Figure 5-22 depicts that nodes B and C are executed iteratively by A, and that nodes D and E are iterated by B.

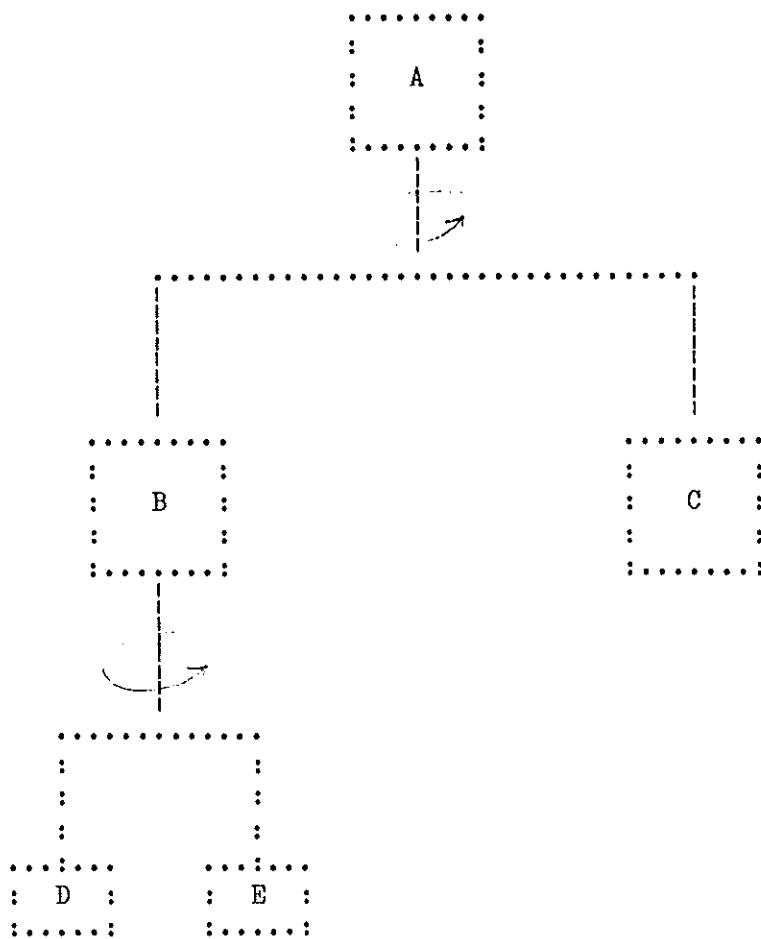


Figure 5-22
Example of Iteration Depiction

The hierarchical diagram of the system specifications is used to depict the iterative processes in the system. The diagram in Figure 5-23 illustrates this for the system of Figure 5-9.

```
.....  
: OVERALL TABLE :  
.....  
:  
:  
:  
: ^ one iteration  
: for each order  
.....  
:  
:  
:  
.....  
: Is order : Is order : Process : Update :  
: in       : within   : the    : customer :  
: acceptable : credit  : order  : record  :  
: format    : limit   : format : :  
.....  
:  
:  
:  
.....  
:  
:  
:  
.....  
: Establish : Look up : Test  : Assign :  
: credit    : credit   : account: new   :  
: rating    : limit   : balance: credit :  
.....
```

Figure 5-23
Iterations of System Specifications

5.3.2.2.4 Define the system architecture

The producer, at this point, has sufficient information on the functions and interfaces of the system to define its architectural structure. In general it is advisable to use the basic hierarchical structure of the system specifications as the system design's structure. This practice is beneficial because it maintains continuity between the specifications and design, and because it may very well be a cost effective approach for the design.

It is now almost universally acknowledged that the major life-cycle cost of software is in its maintenance. The costs of correcting and enhancing software systems significantly exceeds the cost of initially developing them.

Several of the popular structured methodologies have identified the principal culprit in high maintenance costs to be poor system designs. The objectives for a cost effective system design are that functions performed within the system can be located quickly; that they must be correctable with a minimum of disruption to the rest of the system; and that enhancements to a system can be made simply and directly.

Using the architecture of the system specifications and decision tables for the system design meets the objectives for cost effective software. Decision tables make it easy to locate and correct problems without having a major disruptive effect on the rest of system. If the architecture of the analysis and design are closely related, enhancements to the system can be directly transferred from the updated system specifications to the system design and then easily implemented (Section 5.3.6, Maintenance and Extensions, addresses this issue in greater depth). It is therefore advisable to use the basic architectural structure of the system specifications for that of the system design.

There are cases in which cost effective operation is a secondary consideration, and the system design must be developed more around efficiency. Real-time systems are typical examples where rapid operation is the highest design criteria. In such cases the system specifications may not be structured to provide sufficient efficiency for the system design. In this situation it is advisable to modify the system specifications only in the areas where high efficiency is necessary. By retaining as much of the system specifications architecture as possible in the system design, the system can achieve an acceptable balance between efficiency and cost effectiveness.

The producer defines the architecture for the system design most suitable for the objectives of the system. In the example of Figure 5-9 the objective of cost effectiveness is more important than efficiency. Therefore, the producer will use the system specifications architecture of Figure 5-9 as the basic structure for the system design. This architecture is acceptable for the system design because it provides a decomposition of the specification into functions that can easily be developed into programs when the proper data access and control functions are added.

5.3.2.2.5 Add data access and control functions

To complete the system design requires the addition of functions necessary for it to work in an operational computer environment, in particular data access and control functions.

The data access functions are those that either bring data to the system to be processed, or transfer processed data to its external destination. These functions are unique for each system being developed. They depend on the computer hardware and software available for data access.

The control functions are needed for coordinating the activities of the working functions of the system design. The control functions are defined as needed to meet the practical requirements of implementing the system in a computer environment.

The producer adds data access functions where data handling is required and control functions for coordination of either data handling or the processing functions to the system design as shown in Figure 5-24.

5.3.2.2.6 Partition into programmable modules

The system design is now ready to be packaged into computer programs. This step is necessary to fit the system design structure into the physical environment of the computer. The limitations of the computer's memory and capability to access data determine the extent of this packaging. In pageable computer operating systems where memory is virtually limitless, packaging can be simply to make each function a separate computer program and plan on having all programs in memory at the same time. In a more limited environment, with definite limits to the memory capacity of computer, the system design structure must be partitioned into related functions that can be written as a single computer program. Since the actual packaging is unique to a specific operational environment, no attempt will be made here to partition the system design of Figure 5-24.

5.3.3 Implementation phase

The Implementation Phase is the process of constructing an operational product from the system design. In this context, the Implementation Phase will only apply to computer applications. Thus it becomes the coding and testing stages of a typical computer application life cycle.

The first step involved in this phase is to translate the decision tables of the system design into program code. This can either be done by translations, if a decision table translator is used, or by interpretation without a translator.

The second step, although it can be done concurrently, is to test the translated modules using top-down integrated testing techniques.

5.3.3.1 Implementing using a decision table translator

The system design decision tables can be directly converted to program code when a decision table translator is used. The method employed is to translate the decision table condition and action stubs to their program code equivalents. This is accomplished while maintaining the framework of the design decision table, therefore the integrity of the system design is carried directly through to the programs without requiring extensive interpretation.

The actual syntax and semantics for translating the decision tables to programs depends upon the translator and the program language being used. Ideally the translator should be part of the language compiler. This allows compilation errors to be related directly back to the inputted decision table rather than its expanded program code representation. An additional feature of the translator should be the capability to process both limited entry and extended entry decision tables for greater flexibility.

As the translation is being performed, the translated decision tables can be integrated and tested into the final structure of the system. This process should proceed using top-down integration. This is the process of testing and verifying the program modules from the top of the hierarchy first. Once they are verified, the process proceeds to the next lower level. This process is well defined in structured programming literature.

The distinction decision tables have when testing is that through the use of a diagonal matrix the number of times each rule is executed can be signalled. Consider the following decision table:

	1	2	3	4					
: C1	:	Y	:	Y	:	N	:	N	:
:	:	:	:	:	:	:	:	:	:
: C2	:	Y	:	N	:	Y	:	N	:
:	:	:	:	:	:	:	:	:	:
: A1	:	X	:	:	:	:	X	:	:
:	:	:	:	:	:	:	:	:	:
: A2	:	X	:	X	:	X	:	X	:
:	:	:	:	:	:	:	:	:	:
: A3	:	X	:	:	X	:	:	:	:
:	:	:	:	:	:	:	:	:	:
: A4	:	:	X	:	:	:	X	:	:
:	:	:	:	:	:	:	:	:	:
: Signal 'RULE 1':	X	:	:	:	:	:	:	:	:
:	:	:	:	:	:	:	:	:	:
: Signal 'RULE 2':	:	X	:	:	:	:	:	:	:
:	:	:	:	:	:	:	:	:	:
: Signal 'RULE 3':	:	:	:	X	:	:	:	:	:
:	:	:	:	:	:	:	:	:	:
: Signal 'RULE 4':	:	:	:	:	:	X	:	:	:
:	:	:	:	:	:	:	:	:	:

Diagonal Testing Matrix

The actions following action 4 are add to the original decision table to signal each time a particular rule is executed. This diagonal matrix is only added during testing to insure that each rule is tested at least once. When testing is completed, the diagonal matrix should be removed.

5.3.3.2 Manual translation

Manually mapping decision tables into program syntax can be done quite efficiently if the decision table is factored into procedures to avoid duplicating statements. However, this may mask the logic of the original decision table.

It is not unusual for identical series of actions to appear in two or more rules within a decision table when manually translated. Often when the table's logic is represented in structured programming style, the statements forming these action series must be duplicated for each rule, a circumstance fostering the factoring (i.e., extraction) of statements into a separate procedure. Once factored the replication of the several statements can be replaced by a single statement invoking the procedure.

Figure 5-26 illustrates this point. Part (A.) displays a decision table in which the series of three actions, "Increment the number of rules by 1", "Set current rule-index to restart-rule-index", and "Zero H-count" are to be executed for rules R1, R2 and R3. These statements must be written three times if the table is expressed using structured programming constructions as in part B. of Figure 5-26. As shown in part c. of the figure, one can reduce the amount of writing required by extracting the three statements and synthesizing them into a procedure NEWTABLE and replacing each of their occurrences by 'CALL NEWTABLE'.

Such factoring has two disadvantages: First, the statements have been isolated from the context in which they are executed. To see what actions CALL NEWTABLE implies, one must divert attention from the program's flow and examine a separate selection of statement. Second, there is the chance that the pattern may change, necessitating a change in the procedure. Suppose for the previous example, it is discovered that 'zero H-count' should not be executed for rule R2. To make the change in the decision table, one need only change an 'X' into a '-'. In the case of the factored procedure, one must remove 'zero H-count' from NEWTABLE and insert it at two points in the calling BACK-UP procedure.

A. Procedure BACK-UP written as a single decision table.

		R1	R2	R3	R4
: C1	Is S a potential table entrance	:	Y	N	N
: C2	Is S the first stack entry	:	-	Y	N
: C3	Do size considerations justify a new table	:	-	-	Y
:		:			:
: A1	Designate S as a table entrance	:	-	X	X
: A2	Increment the number of tables by 1	:	X	X	X
: A3	Set current rule-index to restart-rule-index	:	X	X	X
: A4	Increment the rule count by 1	:	-	-	-
: A5	Zero H-count	:	X	X	X
: A6	Zero clause count	:	-	X	-
: A7	Pop-up the stack	:	X	-	X
:		:			:

B. Procedure BACK-UP written as a structured program

Procedure BACK-UP

BEGIN IF S is a potential table entrance

 THEN BEGIN Increment the number of tables by 1;
 Set current rule-index to restart-rule-index;
 Zero H-count;
 Pop-up the stack; END;
 ELSE IF S is the first stack entry
 THEN BEGIN Designate S as a table entrance;
 Increment the number of tables by 1;
 Set current rule-index to restart-rule-index;
 Zero H-count;
 Zero clause count; END;
 ELSE IF SIZE considerations justify creation of a new table
 THEN BEGIN Designate S as a table entrance;
 Increment the number of tables by 1;
 Set current rule-index to restart-rule-index;
 Zero H-count;
 Pop-up the stack; END;
 ELSE BEGIN Increment the rule count by 1;
 pop-up the stack; end;

END BACK-UP;

C. Avoid duplication by factoring

Procedure BACK-UP

```

BEGIN IF S is a potential table entrance

    THEN BEGIN CALL NEWTABLE:
            Pop-up stack; END;
    ELSE IF S is the first stack entry
        THEN BEGIN Designate S as a table entrance;
                  CALL NEWTABLE;
                  Zero clause count; END;
    ELSE IF SIZE considering justify creation of a new table
        THEN BEGIN Designate S as a table entrance;
                  CALL NEWTABLE;
                  Pop-up the stack; END;
    ELSE BEGIN Increment rule count by 1;
              Pop-up the stack; END;

END BACK-UP;
```

Procedure NEWTABLE

```

BEGIN Increment the number of tables by 1;
      Set current rule-index to restart-rule-index;
      Zero H-count; END;

END NEWTABLE;
```

Testing and integration of the program code from manual conversion should be performed top-down as with non-decision table structured programs.

5.3.4 OPERATION

5.3.4.1 The problem

When an information processing system is in operation, it has to interact with human beings. Three categories of human interfaces can be distinguished:

- people who provide input for the system;
- people who operate the system;
- people who use the output of the system.

The effectiveness of any information processing system is heavily influenced by correct or incorrect actions taken by its human partners. It is important therefore, that they are properly instructed.

5.3.4.2 The role of decision tables

For procedural decision situations, the use of decision tables, in many ways, improve the correctness and speed of decision making. Therefore, it is extremely useful to provide instructions in the form of decision tables to the people in these situations.

In practical cases, decision tables have been found to be an excellent tool for this purpose; evidence exists that, when the normal input preparers or operators are absent, their replacements can be instructed more quickly and make correct decisions faster when decision tables are used.

5.3.4.3 The format for decision tables

The decision tables used in this phase are exclusively for human use. Therefore in constructing them, the human engineering considerations discussed in section 5.2 should be applied.

5.3.4.4 An example

The following is an example of a decision table which was actually used in a computer center for instructing the operators of a small computer system.

```
.....: Console message = : 0 : 1 : 2 : 3 : 4 : 5 : OTHER :  

.....: System in WAIT mode : - : - : - : - : - : Y : N : Y :  

.....: Console message = "Disc X Unit Y" : - : - : - : - : - : Y : Y : N :  

.....: Console = "Unit X not ready" : - : - : - : - : - : - : N :  

.....: Paper properly aligned in printer : Y : N : - : - : - : - : - : Y :  

.....: Align paper properly : - : X : - : - : - : - : - : - : - :  

.....: Supply new paper : - : - : X : - : - : - : - : - : - :  

.....: Type "J" on console : X : X : X : - : - : - : - : - : - :  

.....: Stop processing : - : - : - : X : X : - : - : - : - : - :  

.....: Call responsible maintenance-programmer : - : - : - : X : - : - : - : - : - :  

.....: Start recovery procedure : - : - : - : X : - : - : - : - : - :  

.....: Type variable currency on console for : - : - : - : - : X : - : - : - : - :  

.....: Pounds Sterling : : : : : : : : : : : : :  

.....: Mount subroutine disc : - : - : - : - : - : X : - : - : - :  

.....: Mount disc X on Unit Y : - : - : - : - : - : - : X : X : - :  

.....: Press START button after READY signal : - : - : - : - : - : - : X : - : X :  

.....:
```

5-27

Decision table for instructing computer operation

5.3.5 User acceptance

5.3.5.1 Objective

The user must be sure that the product contracted for in the System Specifications is the one delivered. The User Acceptance Phase is the user's first opportunity to see the final product and determine its acceptability.

5.3.5.2 Procedure

The user should test the product to determine that it performs what the system specifications have defined. With decision tables used to describe the system specifications, the user can now easily identify the key functions for testing the product.

The user begins constructing tests by formulating specific situations from the system specifications. First, the most important general cases should be chosen, and then variations of those cases. These tests are submitted to the product for processing, applying procedures for manual systems or executing the computer system. The results are then checked against expectations.

This phase is complete when the user is satisfied that the operational product fulfills the requirements of the system specifications. At this point the producer's work is done.

5.3.6 Maintenance and extensions

5.3.6.1 Introduction

This phase of redesign and development is the continuing process that takes place after system implementation has been completed. When decision tables have been used in the initial development, problem identification and solution during this period is similar to the initial development process. Because most of the software being maintained today was not developed using decision tables, this discussion will include traditionally and structured coded systems. The examples used here are of necessity not complex, but will serve as illustrations.

Section 5.3.6.2 discussed the maintenance of programs using decision tables, traditional coding, and structured coding. Section 5.3.6.3. discusses extensions and improvements to programs done in each of the three techniques. Section 5.3.6.4 discusses how decision tables can aid in analyzing and documenting existing programs that were not originally designed using decision tables.

5.3.6.2 Maintenance

5.3.6.2.1 Need for Maintenance

The need for changing software arises from two types of symptoms:

- . The expected output is not produced.
- . The system has abnormally terminated.

The first results from either beginning with an inaccurate statement of what is wanted or incorrectly translating the requirements into executable instructions. The second should not occur in a properly designed system. Finding out whatever caused the error and resolving it is a two-step process:

- . Find the cause of the error.
- . Correct it.

To describe this process of finding and fixing errors, a computer program is represented here in three forms for reference in following discussions:

- . decision table - Figure 5-28
- . traditional coding - Figure 5-29
- . structured coding - Figure 5-30

	R1	R2	R3	R4	R5	R6	R7	R8	R9	R10	R11	R12	R13	R14	R15	R16
:C1 :	Y	Y	Y	Y	Y	Y	Y	Y	N	N	N	N	N	N	N	N
:C2 :	-	-	-	-	-	-	-	-	Y	Y	Y	N	N	N	N	N
:C3 :	Y	Y	Y	Y	Y	N	N	N	-	-	-	-	-	-	-	-
:C4 :	Y	Y	Y	N	N	-	-	-	-	-	-	-	-	-	-	-
:C5 :	Y	Y	N	-	-	Y	Y	N	-	-	-	Y	Y	Y	N	N
:C6 :	Y	N	-	Y	N	Y	N	-	Y	N	-	-	Y	N	Y	N
:C7 :	-	-	-	-	-	-	-	-	-	-	-	Y	N	N	-	-
:C8 :	-	-	-	-	-	-	-	-	Y	Y	N	-	-	-	-	-
:A1 :	-	-	X	-	-	-	-	X	-	-	X	-	-	-	-	-
:A2 :	-	-	-	X	X	-	-	-	X	X	-	-	-	-	-	-
:A3 :	-	-	-	-	-	-	-	-	X	X	-	-	-	-	-	-
:A4 :	-	-	-	X	X	-	-	-	-	-	-	-	-	-	-	-
:A5 :	-	-	-	-	-	-	-	-	-	-	X	-	-	-	-	-
:A6 :	-	X	-	X	-	-	X	-	-	-	-	-	-	-	-	-
:A7 :	X	-	-	X	X	-	-	X	-	-	X	X	-	X	-	-
:A8 :	-	-	-	-	-	-	-	-	-	-	-	-	-	X	-	X
:A9 :	X	X	-	X	X	X	X	-	X	X	-	X	X	X	X	X

Figure 5-28
Decision Table

<u>STMT</u>	<u>CODE</u>	<u>STMT</u>	<u>CODE</u>
1 P0.		22 P7.	
2 IF C1 GO TO P8.		23 DC ACTION A2.	
3 IF C2 GO TO P5.		24 DO ACTION A3.	
4 IF C5 GO TO P4.		25 IF C6 GO TO P3.	
5 P1.		26 DO ACTION A5.	
6 IF C6 GO TO P3.		27 GO TO P2.	
7 DO ACTION A8		28 P8.	
8 P2.		29 IF C3 GO TO P12.	
9 DO ACTION A9		30 P9.	
10 EXIT.		31 IF C5 GO TO P10.	
11 P3.		32 GO TO P6.	
12 DO ACTION A7		33 P10.	
13 GO TO P2.		34 IF C6 GO TO P3.	
14 P4.		35 P11.	
15 IF C7 GO TO P3		36 DO ACTION A6.	
16 GO TO P1.		37 GO TO P2.	
17 P5.		38 P12.	
18 IF C8 GO TO P7.		39 IF C4 GO TO P9.	
19 P6.		40 DO ACTION A2.	
20 DO ACTION A1.		41 DO ACTION A4.	
21 EXIT.		42 IF C6 GO TO P11.	
		43 GO TO P3.	

Figure 5-29
TRADITIONAL (UNSTRUCTURED) CODING

<u>STMT</u>	<u>CODE</u>	<u>STMT</u>	<u>CODE</u>
1	MAINLINE	33	ELSE
2	IF C1	34	IF C5
3	IF C3	35	IF C7
4	IF C4	36	DO ACTION A7
5	DO SUBROUT A	37	ELSE
6	ELSE	38	END IF
7	DO ACTION 2	39	ELSE
8	DO ACTION 4	40	IF C6
9	IF C6	41	DO ACTION A7
10	DO ACTION A6	42	ELSE
11	ELSE	43	DO ACTION A8
12	DO ACTION A7	44	END IF
13	END IF	45	END IF
14	DO ACTION A9	46	DO ACTION A9
15	END IF	47	END IF
16	ELSE	48	END IF
17	DO SUBROUT A	49	EXIT
18	END IF	50	SUBROUT A
19	ELSE	51	IF C5
20	IF C2	52	IF C6
21	IF C8	53	DO ACTION A7
22	DO ACTION A2	54	ELSE
23	DO ACTION A3	55	DO ACTION A6
24	IF C6	56	END IF
25	DO ACTION A7	57	DO ACTION A9
26	ELSE	58	ELSE
27	DO ACTION A5	59	DO ACTION A1
28	END IF	60	END IF
29	DO ACTION A9	61	EXIT.
30	ELSE		
31	DO ACTION A1		
32	END IF		

Figure 5-30
STRUCTURED CODING

5.3.6.2.2 Finding the cause

As an example, assume that action A9 is being performed and should not be, when C1 = YES, C3 = YES, C4 = NO, and C6 = NO.

- Using the decision table logic in Figure 5-28 it is easy to see that rule 5 represents the processing situation in question and that action A9 should be removed from that rule.
- In the traditional coding (Figure 5-29) action A9 only appears in statement 9, but it is not easy to see the flow of control that leads to statement 9. It becomes necessary to trace the program's logic through statements 1, 2, 28, 29, 38, 39, 40, 41, 42, 35, 36, 37, and 8 to finally arrive at statement 9.
- In the structured coding of Figure 5-30, it is also necessary to trace the processing situation through statements 1, 2, 3, 4, 6, 7, 8, 9, 11, 12, 13, and 14. One other alternative reaches action A9 in statement 14 -- the alternative is like the one under study except C6 = YES. With this structured code this trace was simpler than with the traditional code.

When a program terminates abnormally, the location is known but not necessarily the logic path used to get there. It becomes necessary to follow the program in reverse, i.e., read it from the bottom up. For purposes of demonstration, assume the program terminated at action A7.

- In the decision table (Figure 5-28) it is quickly and easily discernable that action A7 is executed in seven decision rules: 1, 5, 6, 9, 12, 13, and 15. Isolating the exact cause is then a relatively straightforward process of comparing the data being processed (input transaction) against the various rules.
- Using the traditional coding in Figure 5-29, the termination occurs in statement 12. Following the control flow backwards is not possible, so the approach must be to start at the top of the code and proceed down, using the input transaction as a guide. Even then it is difficult to determine that there are 7 paths leading to statement 12 and it may be more difficult to identify which one was followed.
- In the structured coding in Figure 5-30, the termination could have occurred in statement 12, 25, 36, 41, or 53. Whichever one it is, it is possible to read from there up.

5.3.6.2.3 Fixing the cause

Once an error is isolated, attention can be directed toward correcting it. It is of paramount importance that the correction of the error not create additional problems. This means that not only the transaction being processed be considered, but all others.

- Decision Table

The unacceptable result is fixed by removing action A9 from rule 5. No other effect will be introduced as no other occurrences of action A9 have been changed.

The abnormal termination requires further diagnosis. Assume the cause is found to be the absence of A2 prior to action A7. Assuming the input transaction to be for rule 15, action A2 is needed there. In addition, rules 1, 6, 12, and 13 have like action sets and must be considered for the possible addition of action A2.

- Traditional Coding

There is no explicit separation of the 13 and 7 paths as in the decision table. Determination of where and how the changes are to be made is difficult. When the changes have been made, there is usually some uncertainty as to whether all necessary (or possibly some unnecessary) changes have been made.

- Structured Coding

Provides explicit capability for identifying both the paths to be changed to correct this error and those affected by such changes. What is not directly identifiable are those other paths which should be considered for change.

Structured coding is far superior to traditional coding, but it can be more complex than decision tables. Tables provide a far superior representation of the relations between paths.

Changes, for example, the addition, deletion, or rearranging of conditions, would increase the difficulty of error correction but the degree of difficulty would remain relatively the same between the three forms.

5.3.6.3 Extensions and Improvements

Enhancing or making major modifications to a system after it has been implemented, requires the ability to research and learn the system functions based upon what can be learned from the system itself. This requires interpreting the system's logic. A task that is often tedious

and error prone if done with traditional software. Decision tables, however, provide a very readable tool for showing the relationship between conditions and actions associated with a system's logic, and they can help insure that all possible logic paths have been considered for the decision situation. Therefore, using decision tables provides a far easier method to:

- Determine the full impact of adding/deleting conditions,
- Determine the full impact of adding/deleting actions,
- Minimize the condition that changes implemented to resolve one problem unintentionally create others.

Traditionally designed systems are normally difficult and treacherous to modify. This is due in part to the inability to prove their correctness. The typical procedure in such systems is to make a series of fixes and corrections to the existing code and then to test it hoping that it will work. If documentation was lacking before the process, it can be assured no upgrade of it will result.

Hierarchically structured designs employing the concept of "Levels of Abstraction" are easier to handle because they provide the modularity necessary to insure ease of extension. In such structure decision tables can be used effectively, as has been shown.

5.3.6.4 Analysis and documentation

The previous sections of Design Methodology have attempted to show the value of decision tables in system design and development. The question arises, can decision tables improve analysis and documentation processes in systems originally constructed without the use of decision tables?"

Figure 5-31 is a representation of traditional coding from an actual program which will be used in the discussion of analysis that follows.

STMT

1	P1.		27	P4.
2	. IF A = "P" MOVE QQ TO YY.		28	MOVE RR TO WW
3	IF B = SPACE GO TO P10.		29	P5.
4	IF (C = "P" and A = "M")		30	MOVE "P" TO CC
5	GO TO P9.		31	IF (J = "C" or J = "N")
6	IF C = "P" DO ERR - C		32	NEXT SENTENCE
7	GO TO P9.		33	ELSE GO TO P7.
8	IF C = "N" DO ERR - L		34	IF K = SPACE MOVE 1 TO RR
9	GO TO P9.		35	IF L = SPACE or L = "R"
10	IF A = "M" MOVE MM TO YY		36	GO TO P6.
11	GO TO P9.		37	MOVE 1 TO SS
12	MOVE A TO ZZ.		38	GO TO P7
13	IF A NOT = "P" GO TO P9.		39	P6.
14	IF C NOT = "A" GO TO P9.		40	IF M = SPACE or M = "N"
15	MOVE SPACE TO YY		41	GO TO P7.
16	MOVE PP TO QQ		42	MOVE 1 TO S
17	IF D = "E" or "N" GO TO P2.		43	P7.
18	IF F NOT = SPACE GO TO P2.		44	MOVE AA TO BB
19	MOVE PP TO ZZ		45	P8.
20	P2.		46	IF N NOT = SPACE GO TO P9.
21	DO LI-ROUTINE.		47	DO PRE-PRINT
22	IF G = SPACE GO TO P4.		48	P9.
23	P3.		49	EXIT.
24	IF H = SPACE GO TO P4.		50	P10.
25	MOVE QQ TO WW		51	MOVE CC TO DD
26	GO TO P5		52	GO TO P7

Figure 5-31

5.3.6.4.1 Record all possible paths

Following one path, list the first condition encountered and the first alternative for it.

```
.....  
: A = : "P" :  
.....:
```

Follow that path and add the next stub encountered (condition or action). If it is a condition, list it and its first alternative.

```
.....  
: A = : "P" :  
.....:  
: MOVE QQ TO YY : X :  
.....:
```

Repeat this process for all other stubs encountered in the selected path.

```
.....  
: A = : "P" :  
: B = : SPACE :  
: N = : SPACE :  
:.....:  
: MOVE QQ TO YY : X :  
: MOVE CC TO DD : X :  
: MOVE AA TO BB : X :  
: DO PRE-PRINT : X :  
: EXIT : X :  
:.....:
```

Follow a second path by listing the condition entries again (ADDITIONAL RULE) with a different alternative for the last condition listed.

```
.....  
: A = : "P" : "P;  
: E = : SPACE : SPACE :  
: N = : SPACE : NOT SPACE :  
:.....:  
: MOVE QQ TO YY : X :  
: MOVE CC TO DD : X :  
: MOVE AA TO BB : X :  
: DO PRE-PRINT : X :  
: EXIT : X :  
:.....:
```

Follow the path through the coding, listing the previous actions encountered and adding any additional actions and/or conditions encountered.

```
.....  
: A = : "P" : "P"  
: B = : SPACE : SPACE :  
: N = : SPACE : NOT SPACE :  
:.....:  
: MOVE QQ TO YY : X : X  
: MOVE CC TO DD : X : X  
: MOVE AA TO BB : X : X  
: DO PRE-PRINT : X : -  
: EXIT : X : X  
:.....:
```

Repeat this for all alternatives of the last condition. Then follow another path by listing the condition entries with no entry for the last one and a different alternative for the next to last condition.

```
.....:  

: A = : "P" : "P" : "P" :  

: B = : SPACE : SPACE : NOT SPACE :  

: N = : SPACE : NOT SPACE :  

:.....:  

: MOVE QQ TO YY : X : X :  

: MOVE CC TO DD : X : X :  

: MOVE AA TO BB : X : X :  

: DO PRE-PRINT : X : - :  

: EXIT : X : X :  

:.....:
```

Again follow this path listing action and conditions encountered. For a listed condition not encountered show a dash (-) for the entry and also in the new conditions in the previous rules.

```
.....:  

: A = : "P" : "P" : "P" :  

: B = : SPACE : SPACE : NOT SPACE :  

: N = : SPACE : NOT SPACE : - :  

: C = : - : - : "P" :  

:.....:  

: MOVE QQ TO YY : X : X : - :  

: MOVE CC TO DD : X : X : - :  

: MOVE AA TO BB : X : X : - :  

: DO PRE-PRINT : X : - : - :  

: DO ERR-C : - : - : X :  

: EXIT : X : X : X :  

:.....:
```

(Note that in following this path the test (IF A = "M") was encountered. Since we were following the path when A = "P", then A is NOT = "M" and we fall through and no extra entry is made. This begins to highlight the real value of this method.)

```

: A = : "P" :
:      :     :     : NOT : NOT : NOT : NOT : NOT :
: B = : SPACE :
:      :     : NOT :     :     :     :     :     :
: N = : SPACE : SPACE : - : - : SPACE : SPACE : SPACE :
: C = : - : - : "P" : "N" : "A" : "A" : "A" :
: D = : - : - : - : - : "E" : "E" : "E" :
: G = : - : - : - : - : SPACE : SPACE : SPACE :
: J = : - : - : - : - : "C" : "C" : "C" :
: K = : - : - : - : - : SPACE : SPACE : SPACE :
: L = : - : - : - : - : SPACE : SPACE : SPACE :
:      :     :     :     :     : NOT : NOT : NOT :
: M = : - : - : - : - : SPACE : SPACE : SPACE :
: M = : - : - : - : - : - : - : "N" : NOT "N" :
:
: MOVE QQ TO YY : X : X : X : X : X : X : X : X :
: MOVE CC TO DD : X : X : - : - : - : - : - : - :
: MOVE AA TO BB : X : X : - : - : - : X : X : X :
: DO PRE-PRINT : X : - : - : - : - : X : X : X :
: DO ERR-C : - : - : - : X : - : - : - : - :
: DO ERR L : - : - : - : - : X : - : - : - :
: MOVE A TO ZZ : - : - : - : - : - : X : X : X :
: MOVE SPACE TO : - : - : - : - : - : X : X : X :
: YY : - : - : - : - : - : - : - : - :
: MOVE PP TO QQ : - : - : - : - : - : X : X : X :
: DO L1-ROUTINE : - : - : - : - : - : X : X : X :
: MOVE RR TO WW : - : - : - : - : - : X : X : X :
: MOVE P TO CC : - : - : - : - : - : X : X : X :
: MOVE I TO RR : - : - : - : - : - : X : X : X :
: MOVE I TO S : - : - : - : - : - : - : - : X :
: EXIT : X : X : X : X : X : X : X : X :

```

Figure 5-32

This process can show patterns leading to a consideration of partitioning. One has developed here in the last three rules listed. A study will show that starting with P2 at statement 20 all paths eventually end up at P7 at statement 43. Identify this as a lower level action table and restart the rules involved (fifth and up). (This table is shown in limited entry form because of page size.)

Continue this process of adding paths (rules), condition stubs and action stubs until the table is complete (i.e., all possible paths have been followed).

```
.....:  

: A = 'P' : Y Y Y Y Y Y Y Y Y Y N N N N N N N N N N :  

: A = 'M' : - - - - - - - - Y Y Y Y N N N N N N :  

: B = SPACE : Y Y N N N N N N N N N N N Y Y N N N N N N Y Y :  

: N = SPACE : Y N - - Y N Y N Y N Y N - Y N - - - - Y N :  

: C = 'P' : - - Y N N N N N N N N N N - - Y N N Y N N - - :  

: C = 'N' : - - - Y N N N N N N N N N N - - Y N - Y N - - :  

: C = 'A' : - - - - Y Y Y Y Y Y Y Y N - - - - - - - - - - :  

: D = 'E' : - - - - Y Y N N N N N N - - - - - - - - - - - - :  

: D = 'N' : - - - - - Y Y N N N N N - - - - - - - - - - - - :  

: F = SPACE : - - - - - Y Y N N - - - - - - - - - - - - - - :  

.....:  

: MOVE QQ TO YY : X X X X X X X X X X X X X - - - - - - - - :  

: MOVE CC TO DD : X X - - - - - - - - X X - - - - - - X X :  

: MOVE AA TO BB : X X - - X X X X X X X X - X X - - - - - X X :  

: DO ERR-C : - - X - - - - - - - - X - - - - - - X - - - - :  

: DO ERR L : - - - X - - - - - - - - X - - X - - - - :  

: MOVE MM TO YY : - - - - - - - - - - - - - - X - - - - - - :  

: MOVE A TO ZZ : - - - - X X X X X X X X X - - - - - - X - - :  

: MOVE SPACE TO : - - - - X X X X X X X X X - - - - - - - - :  

: YY :  

: MOVE PP TO QQ : - - - - X X X X X X X X - - - - - - - - :  

: MOVE PP TO ZZ : - - - - - - X X - - - - - - - - - - - - :  

: DO P2-TO-P6 : - - - - X X X X X X X X - - - - - - - - - :  

: DO PRE-PRINT : X - - - X - X - X - X - - X - - - - - X - :  

: EXIT : X X X X X X X X X X X X X X X X X X X X X X X X X X :  

.....:
```

Figure 5-33

The lower level table P2-to-P7 can be done in a similar manner. This gives a table representing exactly what the program does.

5.3.6.4.2 Study for correctness

With the logic now in the form of decision tables, the structure can be checked for completeness, contradiction/redundancy and unnecessary condition/action sets. These attributes, together with the ability to visually and logically relate sets of conditions and their actions, provide an excellent aid to the analysis function. Figure 5-34 shows a rearrangement of the table taking advantage of these possibilities.

:	B	=	SPACE	:	Y	Y	Y	Y	N	N	N	N	N	N	N	N	N	N	N	N	:		
:	A	=	'P'	:	Y	Y	N	N	Y	Y	Y	Y	Y	Y	Y	Y	Y	N	N	N	N	:	
:	A	=	'M'	:	-	-	-	-	-	-	-	-	-	-	-	-	-	Y	Y	N	N	:	
:	C	=	'P'	:	-	-	-	-	Y	N	N	N	N	N	N	N	N	Y	N	N	Y	:	
:	C	=	'N'	:	-	-	-	-	Y	N	N	N	N	N	N	N	N	-	Y	N	-	Y	:
:	C	=	'A'	:	-	-	-	-	-	Y	Y	Y	Y	Y	Y	Y	Y	N	-	-	-	-	:
:	D	=	'E'	:	-	-	-	-	-	Y	Y	N	N	N	N	N	N	-	-	-	-	-	:
:	D	=	'N'	:	-	-	-	-	-	Y	Y	N	N	N	N	N	N	-	-	-	-	-	:
:	F	=	SPACE	:	-	-	-	-	-	-	Y	Y	N	N	N	N	N	-	-	-	-	-	:
:	N	=	SPACE	:	Y	N	Y	N	-	-	Y	N	Y	Y	N	Y	N	-	-	-	-	-	:
																							
:	MOVE CC TO DD	:	X	X	X	X	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	:	
:	MOVE AA TO BB	:	X	X	X	X	-	-	X	X	X	X	X	X	X	-	-	-	-	-	-	-	:
:	MOVE PP TO ZZ	:	-	-	-	-	-	-	X	X	-	-	-	-	-	-	-	-	-	-	-	-	:
:	MOVE QQ TO YY	:	X	X	-	X	X	X	X	X	X	X	X	X	X	X	-	-	-	-	-	-	:
:	MOVE A TO ZZ	:	-	-	-	-	-	-	X	X	X	X	X	X	X	X	-	-	-	-	-	X	:
:	MOVE SPACE TO	:	-	-	-	-	-	-	X	X	X	X	X	X	X	X	-	-	-	-	-	-	:
:	YY	:																					:
:	MOVE PP-TO-QQ	:	-	-	-	-	-	-	X	X	X	X	X	X	X	X	-	-	-	-	-	-	:
:	DO P2-TO-P6	:	-	-	-	-	-	-	X	X	X	X	X	X	X	X	-	-	-	-	-	-	:
:	MOVE MM TO YY	:	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	X	-	-	-	-	:
:	DO ERR-C	:	-	-	-	-	-	-	X	-	-	-	-	-	-	-	-	-	X	-	-	-	:
:	DO ERR-L	:	-	-	-	-	-	-	X	-	-	-	-	-	-	-	-	X	-	X	-	-	:
:	DO PRE-PRINT	:	X	-	X	-	-	-	X	-	X	-	X	-	X	-	-	-	-	-	-	-	:
:	EXIT	:	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	:
																							

Figure 5-34

5.3.6.4.3 Rewrite

The table(s) constructed in the analysis and study process graphically show the structure of the program or system. It becomes relatively easy to develop computer instructions that are well organized, simple to follow, accurate, covering all alternatives, and therefore, very maintainable.

To take full advantage of this, consideration should be given to conversion to decision table documentation giving the continuing benefits as discussed earlier in this chapter.

6. EDUCATION

6.1	INTRODUCTION	6- 1
6.2	THE SUBJECT MATTER MODULES	6- 1
6.3	COURSE OUTLINE	6- 3
6.4	MODULE DESCRIPTION	6- 4
6.4.1	Introduction	6- 4
6.4.2	Motivation	6- 6
6.4.3	Interpretation	6- 6
6.4.4	Structure	6- 7
6.4.5	Analysis	6- 8
6.4.6	Construction	6- 9
6.4.7	Manipulation	6-10
6.4.8	Multi-Table Systems	6-11
6.4.9	Human Factors	6-12
6.4.10	Translation	6-13
6.4.11	Decision Table Processors	6-14
6.4.12	Testing	6-15
6.4.13	Software Engineering	6-16
6.5	SAMPLE PROBLEMS	6-18
6.5.1	Credit Approval	6-18
6.5.2	Airline Reservation System	6-18
6.5.3	Filling Orders	6-18
6.5.4	Obtaining a Secret Clearance	6-18
6.5.5	Current or Expired Year Appropriations	6-19
6.5.6	To Quit or not to Quit	6-20
6.5.7	Stock Purchase	6-20
6.5.8	Craps	6-20
6.5.9	Inventory File Update	6-20

6. EDUCATION

6.1 INTRODUCTION

This chapter is designed to provide educators of decision tables with a flexible plan for presenting the subject. The intention is to provide a modular repertoire of decision table material which can easily be adapted for the interests and objectives of those being taught.

To determine the standing of decision tables in the academic world, a member of the DTG addressed a questionnaire to the "Chairman of the Computer Science Department" of one hundred and four (104) selected colleges and universities. Of the thirty-eight (38) schools that responded, only fifteen (15) showed any interest in decision tables. Six schools cover decision tables in formal classes; four of these give no more than passing reference in the context of another course while two schools treat decision tables in more depth. Obviously there is much work to be done in the area of education.

The history of decision table usage indicates it has had difficulty dislodging the more familiar forms of communication such as prose and flowcharting. However the authors have experienced that when properly educated, users of decision tables find it to be a far superior tool for dealing with complex problems.

6.2 THE SUBJECT MATTER MODULES

The course material here has been defined as independent subject matter modules, Figure 6-1. By including different sets of these modules in a decision table course, the course can be individually tailored to meet its audience's needs.

No.	Module Id.	Objective
1	Introduction/ Background	Discuss the evolution, purpose and general applications of decision tables at varying levels of detail.
2	Motivation	Identify and discuss the advantages and tradeoffs of using decision table techniques.
3	Interpretation	Teach how to read and understand decision tables.
4	Structure	Identify and discuss the different types of decision tables and their areas of use.
5	Analysis	Identify the syntactical and semantic checking necessary to establish the correctness of decision table construction.
6	Construction	Teach the construction methods necessary to develop decision tables from decision-oriented problems.
7	Manipulation	Teach methods to refine and improve decision table structure.
8	Multi-Table Systems	Teach the organization of systems of tables.
9	Human Factor	Identify the influence of human limitations on the development process of decision tables.
10	Translation	Teach the translation processes necessary to transform decision tables to programs.
11	Decision Table Processors	Teach the use of translators and interpreters as software tools.
12	Testing	Relate decision tables to the testing of programs and systems.
13	Software Engineering	Relate the use of decision tables to other software engineering techniques, e.g., top down design/development and structured programming.

Figure 6-1
Training Modules

People for whom these modules are intended have been categorized into functional classes based upon what they do. Figure 6-2 identifies the five functional classes and their attributes.

Functional Class	Attributes
1. Review	People who manage and direct activities.
2. Analysis	People who analyze the problem... answer the "What".
3. Design	People who resolve the problem... answer the "How".
4. Implement	People who perform the translation from the design to the implementation.
5. Execute	People who are directed by decision tables.

Figure 6-2
Functional Classes

The construction industry can be used as an example of these functional classes and their relationship to job titles: the building client reviews; the architect analyzes; the engineer designs; the contractor implements; and the laborer/craftsman executes.

6.3 COURSE OUTLINE

A course of instruction has been developed by selecting and sequencing the subject matter modules by functional class. Each module contains an overview section which describes the material to be covered, followed by a detail section that further describes the module's content with references to pertinent source materials and problems to illustrate the concepts. Figure 6-3 shows a suggested selection and order of instruction arranged by the functional class of the student.

<u>Key:</u>							
: 1)	Entries = X,Y where:	X = sequence number					
		Y = level of detail (O = Overview, D = Detail)					
: 2)	The function "Implement" is divided into two instructional						
	sequences -- the mechanized and manual translation processes.						
: 3)	Hyphen (-) indicates not required.						
: Mod.	: Review	: Analysis	: Design	: Impl.-Mech.	: Impl.-Man.	: Execute	
: Intro.	: 1 D	: 1 D	: 1 D	: 1 D	: 1 D	: 1 D	:
: Motiv.	: 2 D	: 2 D	: 2 D	: 2 D	: 2 D	: 2 O	:
: Interp.	: 3 O	: 3 D	: 3 D	: 3 D	: 3 D	: 3 D	:
: Struct.	: 4 O	: 4 D	: 4 D	: 4 D	: 4 D	: 4 O	:
: Analysis	: 5 O	: 5 D	: 5 D	: 5 D	: 5 D	: -	:
: Const.	: 6 O	: 6 D	: 6 D	: 6 D	: 6 D	: -	:
: Manip.	: -	: 7 D	: 7 D	: 7 D	: 7 D	: -	:
: M-Table	: 7 O	: 8 D	: 8 D	: 11 O	: 8 O	: 5 O	:
: H. Fact.	: -	: 9 O	: 9 D	: 12 O	: 9 D	: -	:
: Transl.	: 8 O	: 10 O	: 10 O	: 8 D	: -	: -	:
: DT Proc.	: 9 O	: 11 O	: 11 O	: 9 D	: -	: -	:
: Testing	: 10 O	: 12 D	: 12 D	: 10 D	: -	: -	:
: S. Eng.	: 11 O	: -	: 13 D	: 13 D	: -	: -	:

Figure 6-3
Module Selection and Order

6.4 MODULE DESCRIPTION

Each instruction module contains an overview section and a detail section. The overview section summarizes the content of the detail section in relation to the objective of the module. The detail section contains further explanation of the concepts covered.

References specified as: (ref. para. XX) are made to discussion of the subject matter in previous chapters of the report.

References specified as: [ref. bib. XX] are made to the Bibliography in Chapter 8.

6.4.1 Introduction

[Ref. bib. LON72, NCC70, POL71B, MCD68,
MON74, DEN00, KIN67B]

I. Overview Section.

Discusses the derivation and history of the decision table form, and the computer processing of decision tables into high level computer languages.

II. Detail Section.

A. Background of Decision Table Development.

Discusses the mathematical foundation associated with decision tables. Such things as: Boolean algebra, set theory and combinatorial mathematics (ref. para. 3.2).

B. Decision Tables in Use.

1. Manual.

Discuss the use of decision tables in the area of communication, documentation, and decision making. The work of the Sutherland Co. and Hunt Foods in the mid-1950s illustrates this early use.

2. Mechanized.

Discusses the early efforts of CODASYL, General Electric, IBM, Insurance Co. of North America and selected groups of interested individuals to provide computer translation techniques for decision table input.

C. Advantages and Disadvantages.

1. Manual.

Emphasize that few, if any, disadvantages can be found when decision tables are used to clarify and verify procedures. Only through lack of training or misuse can they lose their advantages. Three major problem areas are:

- Designer and interpreter must be familiar with the decision table concept.
- Procedures not suitable for the decision table format should not be forced to conform to them.
- Large and complex decision situations should not be designed as one large, difficult to follow decision table.
(ref. para. 5.2)

2. Mechanical.

Note that mechanized uses of decision tables were not well accepted. Early processors developed voluminous and inefficient code; programmers disliked the concept of automatic code generation which they considered their

private domain; and the time consumed in restructuring the tabular forms during system modification/maintenance were considered major disadvantages. In retrospect, these disadvantages were more imaginary than real.

6.4.2 Motivation

[ref. bib. DIX64, MCD70A]

I. Overview Section.

Discusses the advantages of using decision tables as a communication medium and a documentation device. Examples are provided which emphasize the completeness and accuracy of decision table logic.

II. Detail Section.

Attack the two big problems in communication and documentation-- incompleteness and misinterpretation. These problems which can be illustrated using well chosen examples, e.g., the Quit or not to Quit problem, section 6.5.6 and the Mail-Order Process problem, section 5.3.1.3.4 are representative of the misinterpretation and incompleteness that can exist in traditional methods of problem definition.

The following benefits of decision tables should be illustrated and discussed:

- (1) Clear problem definition in a systematic and precise way.
- (2) Elimination of unnecessary verbal description.
- (3) Forced factoring of a problem statement into well-defined units.
- (4) Indication of alternatives and exceptions.
- (5) Demonstration of meaningful relationships among variables.
- (6) Provision for a common structure/standard format.
- (7) Utility as a specification and documentation device.

6.4.3 Interpretation

[ref. bib. LON72, NCC70, POL71B, MCD68]

I. Overview Section.

Presents the terminology of the decision table format and form. By example, show the simple narrative to decision table and decision table to narrative transformation.

II. Detail Section.

A. Definition of Terms.

Basic elements relative to decision table format and form are defined here. The definitions should be general enough to be consistent with past and present instructional and published subject matter. These terms are: decision table, condition portion, condition statement, condition stub, condition entry, action portion, action statement, action stub, action entry and decision rule (ref. para. 2.3).

B. Decision Table Form.

Express the "If --- and --- and --- then" format of condition and action combinations of a decision statement relative to the condition and action portions of a decision table.

C. Decision Table Format.

The parts of a decision table are discussed with emphasis placed on the header, quadrant and table body as they relate to the tabular or graphic representation of a decision table form (ref. para. 3.4, 3.5).

D. The Transformation Process.

Express the transformation of decision logic from narrative to decision table and decision table to narrative. Specific examples emphasize the developmental and interpretive processes necessary for unambiguous communication; e.g., the Credit Approval problem, section 6.5.1 and the Stock Purchase problem, section 6.5.7.

6.4.4 Structure

[ref. bib. KIN68, KIN73]

I. Overview Section.

Discuss the types and classes of decision tables and the attributes of each. Some non-traditional subjects are discussed -- multiple hit tables, impossible combinations, and unconditional tables.

II. Detail Section.

A. Decision Table Types

The two formats of a decision table are presented in detail. Decision rules as columns and decision rules as rows are discussed using a common example (ref. para. 2.3).

B. Decision Table Classification.

Limited-entry (LEDT), extended entry (EEDT) and mixed entry (MEDT) are discussed. Examples should be used to represent these table classifications. The building of limited, extended, and mixed entry tables using the same problem parameters is illustrated (ref. para. 2.3).

C. Decision Table Rules.

Rule consolidation and expansion principles are applied in order to: equate simple rules to a complex rule; make a complex rule into two or more simple rules; and identify the rule(s) covered in an "else" rule.

D. Single and Multiple Hit Tables (ref. para. 3.7.9).**E. Unconditional Tables.****6.4.5 Analysis**

[ref. bib. POL71B, LON72]

I. Overview Section.

Discuss decision table primitive construction and the analysis necessary to refine the primitive into a well defined, consistent and complete document. Contradictory rules, redundant rules, impossible rules and exhaustive states of conditions are illustrated by inspecting a number of tables (ref. para. 5.2.1).

II. Detail Section.**A. Consistency Checking.**

Examples should be used to illustrate the analysis processes listed below (ref. para. 3.8.1).

1. Consistency of Conditions.**a. Order of Conditions.****b. Contradictions Between Conditions.****c. Redundancy of Conditions.****2. Consistency of Actions.****a. Contradictions Between Actions.****b. Redundancy of Actions.**

3. Consistency of Rules.

- a. Contradictory Rules
- b. Redundant Rules.
- c. Impossible Rules.

B. Completeness Checking (ref. para. 3.8.7).

Emphasize that once decision logic has been represented in decision table form, that form constitutes a powerful tool for analyzing whether the logic is complete.

1. Completeness of Conditions.

- a. Omitted Conditions.
- b. Exhaustive States of Conditions.
- c. Redundant Conditions.

2. Completeness of Actions.

3. Completeness of Rules.

- a. Checking whether all possible combinations of condition states are included in the decision table, and for constructing missing rules.
- b. Constructing Missing Rules.

6.4.6 Construction

[ref. bib. VER69A, LON72, STR73B]

I. Overview Section.

Discuss the two methods associated with the development and preparation of decision tables from the analysis of another media.

II. Detail Section.

A. Construction Methods for Decision Tables.

- 1. Direct Mode development technique -- exhaustive enumeration (ref. para. 5.2.2.2).
- 2. Search mode development technique -- progressive rule development (ref. para. 5.2.2.1).

B. Examples.

To illustrate the direct and search methods of decision table construction, the following problems are presented:

- (1) Airline Reservation System, section 6.5.2.
- (2) Current or Expired Year Appropriation, section 6.5.5.
- (3) Craps, sections 6.5.8.

6.4.7 Manipulation

[ref. bib. SHW75, CHE76]

I. Overview Section.

Illustrate the transformation methods used to change the decision table structure from one format to another.

II. Detail Section.

A. Objectives.

Emphasize that the manipulation of decision tables during the development process is done primarily to improve readability by the human translator or processability by a decision table processor. However, it should be made clear that manipulation will not affect the logical content of the tables but only the presentation of the logic.

B. Manipulation Procedures (ref. para. 3.8).

The following manipulation procedures should be explained by demonstration.

1. Transformation of decision table formats.

- . Horizontally ruled tables into vertically ruled tables and vice versa.
- . Limited-entry tables into extended-entry or mixed-entry tables and vice versa (ref. para. 3.8.6).

2. Transformation of the rule portion of a decision table.

- . Expansion of a decision table into its canonical form.
- . Consolidation of simple into complex rules (ref. para. 3.8.5).

3. Factoring of decision tables into subtables (ref. para. 3.6)
4. Combining of decision tables (ref. para. 3.8.10).

6.4.8 Multi-Table Systems

[ref. bib. TAY68, STR73B]

I. Overview Section.

Illustrate that single tables only cover a limited part of the logic of complex decision situations. The only way to describe complex problems in the decision table format is to link single tables to multi-table systems (decision table networks) or - in accordance with top-down design - to divide a complex system into subsystems of single decision tables.

II. Detail Section.

A. Parsing of Complex Decision Situations.

A complex decision situation, represented by many conditions and actions, can be parsed into single decision tables by grouping those conditions and actions together that have a strong influence on each other.

B. Table Linkage Mechanisms (ref. para. 3.6).

Tables within a multi-table system can be related through procedures or data.

1. In accordance with the control structures of structured programming, decision tables can be linked together by their actions to form sequences, selections and iterations which may be nested in each other.
2. The actions of one decision table may operate on data which are tested by the conditions of a decision table following afterwards and by this influence that decision table.
3. Various consequences of these relationships should be discussed.

C. Morphology of Decision Table Networks (ref. para. 5.3.1.3.3).

When decision tables are nested in each other, they form a hierarchical structure. The advantages of hierarchical constructed multi-table systems should be explained.

D. Example (ref. Mail Order Problem, para. 5.3.1.3.4).

6.4.9 Human Factors.

[ref. bib. CHE76, LEW70]

I. Overview Section.

Using different decision table formats for the same decision situation, it can be shown how various formats affect the ease and efficiency with which human beings understand and use decision tables.

II. Detail Section.

A. Human Limitations in Communications.

Describe the difficulty of precisely stating and interpreting information.

B. Advantages of Decision Tables for Human Use.

By their general and fundamental characteristics, decision tables are helpful for alleviating the human limitations in the designing and executing of decision logic. Some characteristics to be explained and exemplified are: conciseness, exhaustiveness, unambiguousness, and the possibility of consistency checks.

C. Influence of Human Characteristics upon Decision Table Development.

Various characteristics of decision tables crucial for their influence upon meeting the human limitations are reviewed. Some indication should be given about the range of each characteristic so as to alleviate any negative effect upon the human factors involved in making and using decision tables (ref. para. 5.2.1).

1. The size of the table -- number of conditions, actions and rules.
2. The structure of the table -- limited, extended or mixed entry; simple vs. complex rules; use of the ELSE rule; the method used for constructing rules; bound action; bound conditions and single-hit or multiple-hit tables.
3. The wording of conditions, actions and rules. For example, the condition stub entry "A>B" in connection with the condition entry "Y" is more readable than the condition stub entry "A not > B" in connection with the condition entry "N".

4. The different ways of connecting between tables (e.g., GO TO, PERFORM) and within the same table (e.g., REPEAT).

D. Examples.

The examples provide the vehicle for exercising the human factor concepts as stated above.

1. Filling Orders, section 6.5.3.
2. Obtaining a Secret Clearance, section 6.5.4.
3. Inventory File Update, section 6.5.9.

6.4.10 Translation

([ref. bib. P0074])

I. Overview Section.

Discuss the manual and automated approach to translating decision tables into computer program code. Within the automated translation, the conversion to program trees and rule masking techniques are presented.

II. Detail Section.

- A. There are two methods of converting a decision table to a computer program.

1. Manual -- Have a human being do it. With this translation considerable discipline is required to ensure that the resulting program correctly represents the original decision table and that verification of its correctness is relatively easy.
2. Automated -- Have a computer do it. Automated translation usually requires that the language used within a table is the same as the language into which the table is to be translated, e.g., COBOL conditions and actions are written in tables to be translated into COBOL.

B. Translation Techniques (ref. Chapter 4).

1. There are two translation techniques -- program trees and rule masking. Explain how they work and why the former yields more efficient programs.

2. Parsing Techniques -- Tree Generation
 - . Mechanics of tree generation
 - . Incorporated completeness and consistency checks
 - . Optimization -- Goals (minimize average processing time, minimize storage requirements and establish conventions)
 - . Optimization -- Methods (branch and bound, dynamic progress and single alternative using various selection criteria)
3. Parsing Techniques -- masking

6.4.11 Decision Table Processors

[ref. bib. MCD70B, HOF73]

I. Overview Section.

Describe how various decision table processors function and discuss their different kinds of input and output formats.

II. Detail Section (ref. Chapter 4).

A. Input Languages for Decision Table Processors.

Most decision table processors require a formal input language such as COBOL, FORTRAN, PL/I, etc. Describe typical input formats.

B. Decision Table Input Formats.

Explain why various limitations must be considered for decision table inputs, e.g., number and size of tables, rules, conditions, and action entries.

C. Translation Function of Decision Tables.

1. Discussion of the decision table as a formal language element.
2. Differences between decision table compilers, translators and interpreters.
3. Discuss basic translation and interpretation methods as far as these methods have been implemented in existing processors.
4. Explain processing steps, relevance of and criteria for optimization of existing processors.

D. Diagnostic Function of Decision Table Processors.

Besides syntactical checks of the decision table input format, processors provide for consistency checks (redundancy and contradiction) and completeness checks with hints toward the consolidation of rules.

E. Update and Editing Function of Decision Table Processors.

1. To achieve a vertical and horizontal update function, the deletion, insertion and replacement of rules, conditions, actions and entries should be supported either by the processor or with an external text editor.
2. To achieve readability processors should provide for a neat layout of the edited table.

F. Testing Support Function of Decision Table Processors.

The execution of test cases can be traced with satisfied condition entries, selected rules and performed actions listed. Frequency counts give the cumulative number of tested conditions, selected rules and performed actions to enable a complete decision table test and provide optimization hints.

G. Procurement of Decision Table Processors.

The advantages and disadvantages of manual coding vs. automatic translation of decision tables must be established. If a processor is required, the "make or buy" decision must be made. What kind of information sources on existing processors are available? The selection process is characterized by a comparison of the defined requirements and the measured capabilities.

6.4.12 Testing

[ref. bib. GO075, CAV74]

I. Overview Section.

Discuss the use of decision tables in the testing of computer programs and systems. Describe how the use of decision tables can result in complete and non-redundant testing.

II. Detail Section.

A. Survey.

1. The producer can use the design tables to prepare test data. However, the user may prepare an independent set from which to prepare tests.

2. Testing is greatly enhanced when decision tables are used to carefully identify all combinations of the condition set.
3. The monitoring of testing progress can be easily followed by benchmarking on a rule by rule basis.

B. Approach.

1. Explain why good design that is combinatorially exhaustive gives added assurance of initial success.
2. It has been recognized that module by module testing is an excellent approach. The individual rules of decision tables provide easily identified logical elements that are excellent for this purpose.
3. Describe why decision table specifications allow the user to more easily verify the system's correctness.

6.4.13 Software Engineering

[ref. bib. LEW75]

I. Overview Section.

The concepts and origins of software engineering are described. Within the framework of software engineering techniques, the role of decision tables as supplement or alternative to other techniques is highlighted.

II. Detail Section.

A. Background of Software Engineering.

1. Describe the economic pressures and technical needs for more reliable, readable, flexible, maintainable, portable, and efficient software. Discusses the improved methods for managing software projects that gave rise to the development of a discipline of software rather than an art form.
2. Explain the historical development of software engineering techniques. This should include the work of Dijkstra, Wirth, Mills and others -- brings the student up to the present.

B. Decision Tables in the Framework of Software Engineering.

Using the life cycle of a software system, describe the use of decision tables as supplement or alternative to other software engineering techniques.

1. Recording Specifications.

- a. Explain the use of decision tables to supplement documentation techniques such as HIPO. The benefits of decision tables in ensuring an unambiguous statement of the process while other techniques may be used to describe data flow and structure.
- b. Describe the benefits of using decision tables for analyzing problem statements (ref. para. 5.3.1).

2. Design the System Solution.

- a. Use of software engineering techniques such as the Constantine-Yourdon Structured Design approach to analyze the system design.
- b. Decision tables simplify partitioning the problem statement into the solution due to their readability and facility to define completeness.
- c. Decision Tables provide a higher level abstraction than coding-like formats for specifying the details of the solution process.

3. Developing the System.

- a. Describe the way decision tables, when used with a translator, are compatible with the concepts of structured programming.
- b. Decision tables provide the readability and verification of correctness that distinguishes structured programming. Therefore, the decision table should be considered an additional basic control structure of structured programming.
- c. Decision Tables provide a simple method for testing program paths.

4. Maintenance and Extensions.

Explains the ease with which decision tables can be modified and extended. Supported by a translator, decision tables can be more easily changed than structured coding (ref. para. 5.3.6).

6.5 SAMPLE PROBLEMS

6.5.1 Credit approval

If a customer's credit limit is okay, then the order may be approved. If the credit limit is questionable but the customer's pay experience is favorable, then the order may be approved. If, on the other hand, the credit limit is not favorable and the pay experience is not favorable, the order may be approved only if a special clearance has been obtained. Lastly, if the credit limit is not favorable and the pay experience is not favorable, and if no special clearance has been obtained, then the order should be returned to the sales department.

6.5.2 Airline Reservation System

Only two (2) classes of travel are offered by this airline -- First and Coach. When a customer requests a First Class reservation and a seat is available for that flight, a ticket can be issued. On transmission of the transaction the First Class reservation balance for that flight is updated. However, if a First Class seat is not available, then, at the wish of the customer, his name can either be placed on the First Class waiting list or if a seat is available for that flight in Coach Class, a ticket can be issued for that class. Again, on transmission of the transaction, the Coach Class reservation balance for that flight is updated.

6.5.3 Filling Orders

When the quantity ordered for a particular item equals or exceeds the minimum discount quantity and the order is from a wholesaler, give the customer a discount and make the shipment. This presumes that there is sufficient quantity on hand to fill the order.

If the quantity ordered is less than the discount quantity, bill at regular rates and make the shipments even if the customer is a wholesaler. Do the same if the sale is retail.

If there is not a sufficient quantity on hand, bill as above, ship what can be shipped and backorder the remainder of the order. It must be emphasized that, in this situation also, even if the discount quantity is ordered, if the customer is a retailer, the discount is not given.

6.5.4 Obtaining a Secret Clearance

A. Interim Clearance:

1. Civilian Personnel-US Citizens. Request a National Agency Check (NAC).
2. Military Personnel-US Citizens.

- a. For individuals who have served for 2 consecutive years or more on active duty immediately preceding the date of the request for current investigation, a check of the unit personnel records will be made to include available medical records and any Special File.
- b. For individuals who have not served for 2 consecutive years on active duty immediately preceding the date of the request for a current investigation, no interim clearance is authorized. Exceptions to this provision will not be authorized.

3. Immigrant Aliens. No interim clearance is authorized.

B. Final Clearance:

1. Civilian Personnel-US Citizens. Request an NAC and send written inquiries to appropriate local law enforcement agencies, former employers and supervisors, references, and schools attended.
2. Military Personnel-US Citizen. Request a limited NAC (LNAC) or an NAC.
3. Immigrant Aliens. Request a background investigation.

6.5.5 Current or Expired Year Appropriations

For the RDT&E appropriation, Basic Symbol 3600, there are presently two appropriations, Program Year 76 and 75, which are considered current and two appropriations, Program Year 74 and 73, considered expired. During Fiscal Year 7T and through Fiscal Year 77, the 3600 appropriation will have three current years and two expired years. During Fiscal Year 7T, 7T, 76 and 75 will be current. The necessary programming steps will have to be taken to properly reflect three current year RDT&E appropriations during Fiscal Year 7T and 77.

Like the RDT&E appropriation above, there will be four current year procurement appropriations (Basic Symbols 3010, 3020 and 3080) instead of the traditional three current year procurement appropriations. This condition will exist through Fiscal Year 78. The necessary programming steps should be taken to also accommodate this situation.

During Fiscal Year 7T, there will be two current year annual appropriations (7T and 76) instead of the normal one current appropriation. In Fiscal Year 77 only annual year 77 data will be current. However, three years (7T, 76 and 75) will be expired. Normally, there are only two expired years in annual appropriations. The necessary actions should be accomplished to reflect this data. The following Basic Symbols are annual appropriations: 3400, 3500, 3700, 3740, 3840, 3850, 0700, 0102, and 0030.

6.5.6 To Quit or Not to Quit

If I don't get a raise of at least 10 percent, I will find a job somewhere else. But if I get promoted, then I will expect my own office or I'll quit, unless the work is going to be more interesting; in which case, I'll stay with just a 10 percent raise.

6.5.7 Stock Purchase

There are two situations in which you might buy a particular stock: when the market as a whole is appreciating and when you have knowledge of a special situation. When the market in general is going up, you should buy the stock if it is appreciating faster than the market as a whole. Otherwise, you should look for another stock. If you have knowledge of a special situation, check the stock's recent price and volume action. If price has been increasing on expanding volume and decreasing on declining volume, buy the stock. Otherwise, your information is probably faulty, and you should hold back. In general, you should not borrow money to buy stock. However, if the market is going up as a whole, you have knowledge of a special situation, and the stock is behaving in such a fashion as to indicate that it should be bought, buy the stock on margin. Otherwise, when the market is going up, don't worry about special situations.

6.5.8 Craps

The game of craps involves a player rolling a pair of dice repeatedly. On an initial or "coming out" roll, the player wins with a roll of 7 or 11 and loses on 2, 3 or 12. If he rolls any other number (4, 5, 6, 8, 9 or 10), that number becomes the "point". When the player has a point he makes repeated rolls of the dice until he gets either the "point" or 7. If he gets his point, the player wins; if he gets a 7, he loses. After either winning or losing, the next roll is considered an initial or coming out roll.

6.5.9 Inventory File Update

An inventory master file is to be updated from a validated transaction file. Each transaction and master record on the file contains only three fields: stock number, unit price and quantity.

If a transaction stock number has no matching master record stock number, the transaction is to be printed on a control report.

If a transaction and master record match on stock number, but not on unit price, and if the transaction quantity is negative, print the transaction on the control report. If the stock numbers match, the unit prices do not match and the transaction quantity is positive, then add the transaction quantity to the master quantity.

If the stock numbers and the unit prices match, and the transaction quantity is positive, add the transaction quantity to the master quantity. However, if the transaction quantity is negative, add the transaction quantity to the master quantity only if the master quantity will not go negative; otherwise, print the transaction on the control report.

7. DECISION TABLE GLOSSARY

action

The statement performed or not performed, depending on the state of the conditions.

action entry

The part of the decision table that indicates whether an action must be performed or not for the combination of condition entries appearing in the same decision rule.

action set

A set of actions, each consisting of an action subject and a set of alternatives.

action stub

That quadrant of the decision table which holds the action subjects.

action subject

The action that is to be performed; that part of the action located in the action stub.

alternatives

The different states a condition or action can take.

'a priori' problem statement

A problem definition that already exists in the form of written laws, rules, regulations, procedures, or a narrative description. This 'a priori' statement can be transformed into a problem specification in decision table form using the direct mode.

branch and bound

An algorithm used to parse a decision table into its equivalent program tree, consisting of successively testing different "branches", possibly finding a lower "bound" that can be used to avoid testing other branches of the tree.

canonical form

Fully expanded set of alternatives for the condition entries in which all condition entries are combinatorically expressed.

cell

The intersection of a decision table row and column.

completeness

Completeness of a decision table can be defined on two levels. On a structural (syntactic) level completeness refers to the fact that all combinations of condition entries are represented by the decision rules. On the semantic level, completeness of a decision table means that all transactions produced by a real decision situation are satisfied by at least one rule of the decision table. This requires the complete description of condition, condition alternatives, actions, action alternatives in the decision rules. Completeness checks based on structure (syntax) are of value but are not sufficient to insure semantic completeness.

complete table

See completeness.

condition

The statement of a variable (or a function of variables) together with a specification of its alternatives.

condition entry

The upper right quadrant of a decision table which contains the set of alternatives for the conditions in the condition stub.

condition set

A set of conditions, each consisting of a condition subject and a set of alternatives.

condition stub

The upper left quadrant of a decision table, containing the condition subjects.

condition subject

The condition that is to be checked; that part of the condition located in the condition stub.

consistent decision table

A decision table is said to be consistent if its rules are pairwise mutually exclusive so that there is only one rule for each combination of conditions.

consolidation

Consolidation of a decision table implies the combining of all groups of rules with identical action sets into one rule. This can be done if the condition entries of these rules are irrelevant for the execution of the action.

decision rule

The set of condition alternatives and series of actions to be performed.

decision table

Tabular expression of procedural decision situations which is characterized by one or more conditions and a set of actions that are executed based upon the state of the conditions.

decision table horizontal format

A transposed image of a vertical decision table where the decision rules are listed vertically and extend horizontally.

decision table vertical format

The more commonly used decision table format where decision rules are listed horizontally and extend vertically.

direct mode

A method for constructing decision tables from a written, a priori, problem statement in which the condition alternatives are exhaustively enumerated.

elaboration

The process of refining conditions/actions to more specific levels of detail by expanding them into more detailed decision tables.

else rule

A rule that is executed when none of the other rules in a decision table are satisfied by a given transaction.

entry

A condition or action alternative; the contents of a cell in the rule portion of the table.

equivalence class

See rule classes.

expansion of condition entries

The process of replacing a rule containing an irrelevant "--" entry with rules containing each of the possible entry values for the condition that had the irrelevant entry. The opposite of consolidation.

expansion of decision table

The process of eliminating linkages between a table and its invoked tables by replacing each invoking reference by the invoked table.

explicit subtables

A subtable defined by a declarative action "subtable" and which is a separate structure from the original table.

extended entry decision table

Refers to the content of the condition/action entry portion of a decision table. The condition/action stub is a condition statement which assumes the values supplied in the entry portion of the table.

factoring

Extracting a subset of conditions, condition alternatives and actions from a decision table to form a subtable which is to be invoked by the original table.

ignore

See irrelevant entry.

implied limited entries

In case of logical relations among a table's conditions, one entry in a rule may be derivable from other entries in the rule. Three limited entries cover such situations:

- Y! meaning true by implication,
- N! meaning false by implication,
- # meaning undefined and cannot be evaluated.

impossible rules

If conditions in a decision table are logically dependent, one condition's entry may imply that another condition's entry cannot be satisfied. Rules that contain these entries are logically impossible. These rules can be removed but in their syntactic dimension must be considered when testing completeness.

inconsistency

On the structure (syntactic) level single hit decision tables are inconsistent if they provide more than one rule for a condition configuration. On the semantic level single hit decision tables are inconsistent if they provide more than one rule for a transaction processed against the table. Semantic analysis of inconsistency requires knowledge of the environment in which a decision table is used.

indifference entry

See irrelevant entry.

inputs for decision tables

When viewing a decision table as a mechanism which transforms a set of inputs into a set of outputs, the inputs to a decision table comprise variables directly tested by the table's conditions or utilized by its actions, and indirectly, those as input to procedures invoked by the table.

intermixing conditions and actions

An alternative to the linkage of decision tables via invocation is to replace each reference to the invoked table by the table invoked. This process, called "expansion", may result in the interleaving of actions and conditions.

interpretation of decision tables

The purpose of interpreting a decision table is to find the rule(s) that satisfy an actual transaction. The interpretation process consists of testing the condition entries with a specific transaction state to identify the rule (within single hit table) or the rules (within multiple hit table) that satisfy it.

interpretive masking

Interpretive masking is a method to convert decision tables into a computer program. The resulting program processes each transaction, mapped into a bit mask of one's and zero's, against a set of table masks to find the rule(s) satisfied by the transactions.

intersection of rules

Within consistent single hit decision tables the intersection of any two of the decision rules is empty. This means that two rules do not describe the same part of the decision situation, which is exactly what is intended by single hit decision tables. If the intersection of two rules of a single hit decision table is not empty, the table is inconsistent. Both rules describe one part of the decision situation which violates the logic behind single hit decision tables. If both rules include the same action set, the table is said to contain redundancy, if the action sets differ, the table is contradictory. Within multiple hit tables, non-empty intersections of rule-pairs are not only allowed but the normal case.

invocation of decision tables

One decision table invokes another table by DOing it, CALLing it, PERFORMing it, using it as a FUNCTION reference, etc. The invocation of the second table is done in the stub portion of the invoking table.

irrelevant entry

A "-" (dash) within a condition row of the decision rule represents all semantically possible alternatives of the condition. Hence within the dash-containing rule, this condition cannot affect the action set selection, therefore it is irrelevant. A "-" or a blank within an action row of a decision table shows that the action must not be executed, it is to be ignored within the dash-containing rule.

LEDT

See limited entry decision table.

limited entry decision table

a decision table which limits the condition and action alternatives in the entry portion of the table to Y (yes), N (no), - (irrelevant or ignore), Y! (true by implication), N! (false by implication), and # (do not test) in the conditon entry area, and X (perform) or - or blank (do not perform) in the action entry area.

linkage of decision tables

By linkage of decision tables, a system of decision tables is formed. Table linkage is performed by invocation.

mergeable entries

The elimination of a duplicate condition check in two or more branches of the decision tree based upon like action sets.

mixed entry decision table

A decision table that has limited and extened entry forms intermixed for the conditions/actions in the table.

multiple hit tables

Presents an alternative to the standard exclusive "or", single hit table. The table uses an inclusive "or" relationship to relate the rules of the table, thus allowing a transaction to satisfy more than one decision rule.

occurrence

A state of a condition.

order (rule, condition)

Relative to testing sequence; condition order is testing the conditions in the sequence listed. Rule order is testing in such a way as to determine the satisfaction of rules in the sequence listed.

outpath....

A term used to describe a rule path from and including the entry at condition c of rule r and those entries which follow it in rule r -- OP(r,c).

output

Semantically, the results of a transformation.

output generating clause

A clause (action or condition) which sets or alters the value of a variable.

outspan

Output generating clause.

parsing

Mapping a decision table into a control structure compatible with the test-and-branch logic of a computer.

producer

The system's developer. The person or people responsible for analyzing the user's requirements and then developing a system (either manual or for a computer) to perform the desired task.

quadrant

A quadrant is used to refer to one of the four sections which normally divides a decision table, viz condition subjects, condition alternatives, action subjects, and action alternatives.

redundancy

Unnecessary (overlapping, repeated) specification (coding). A condition is redundant in a table when none of its entries indicate it is to be tested (evaluated). Rule redundancy is the special case of rule overlap where two rules are also in the same rule class (have the same actions).

relevancy

a condition is relevant to a rule when it must be tested to determine satisfaction of the rule. An action is relevant when it is to be executed upon satisfaction of the rule.

rule class (equivalence class)

All simple rules within a table that have the same (identical or equivalent) actions.

rule masking

At execution (processing) time all conditions are evaluated and compared rule by rule to select the rule or rules satisfied by the transaction.

rule overlap

Two rules overlap when one or more transactions satisfy both rules. The extent of the overlap is the complete set of transactions that satisfy both rules.

rule satisfaction

At execution (processing) time a rule is satisfied when a transaction meets the alternatives specified in that rule.

search cost

The product of condition testing cost and the summation of rule probabilities for which that condition is irrelevant (i.e. where the condition entry is "-").

search free conditions

Conditions which do not contain irrelevant entries for any given rule thereby yielding a search cost of zero.

search free tables

Those tables which can be mapped into program trees that test only search free conditions and thereby yielding a search cost of zero for the table.

search mode

A technique for defining a problem structure through systematic inquiries and then expressing it in a decision table.

selector input

Any input to a decision table which is tested in one of the table's conditions.

semantic completeness

A decision table is semantically complete if it provides a rule for all possible transactions.

semantic consistency

A decision table is semantically consistent if no transaction satisfies more than one rule.

semantic function

A decision table is a semantic function if it is semantically complete and consistent.

single hit decision tables

Those tables that allow not more than one rule to support a given transaction.

space, action

The cartesian product formed by selecting one alternative each of a table's actions.

space, condition

The cartesian product formed by selecting one alternative from each of a table's conditions.

subject

See condition subject or action subject.

subspace

A collection of points within a space.

subtables

A selection of rows and columns that partition the decision table.

supported transaction

See transaction, supported.

transaction

Input to a decision table, contain a value for each variable tested by the table's conditions or used by its actions.

transaction-set

A rule transaction set comprises all transactions which satisfy the rule. A T-set.

transaction, supported

A transaction is supported by a table if it satisfies at least one of the tables rules.

T-set

See transaction-set.

verification cost

The product condition test cost and the summation of rule probabilities with relevant condition entries.

8. ANNOTATED BIBLIOGRAPHY (*)

Directions For Use

This bibliography, containing 581 entries, presents a fairly complete survey of the literature as it has been published up to December, 1981. Not covered are the publications treating the use of decision tables in the field of the syntactical analysis of programming languages. Parts of publications on systems analysis or computer programming, dealing with the decision table technique, have only been inserted as far as they offer an original treatment or a thought-provoking survey of the subject.

The publications are listed as follows:

ENTRY	AUTHOR(S)
	TITLE
	PERIODICAL, VOLUME (NUMBER)
	DATE, PAGES
	ANNOTATION

If the treated publication is a book or a separate issue, the name of the periodical is replaced by the publisher's name. Titles in a foreign language are followed by their English equivalent. The language code is as follows:

C : CZECK	J : HEBREW
D : DUTCH	N : NORWEGIAN
E : SPANISH	P : POLISH
F : FRENCH	R : RUSSIAN
G : GERMAN	S : SWEDISH
H : HUNGARIAN	

(*) The compilation of this bibliography was supported by the Onderzoeksfonds of the Catholic University of Leuven (Belgium) under Project No. OT/II/8. An earlier version, including an author's list and a KWIC-reference list appeared as a separate publication.

As far as periodicals are concerned, the abbreviations given below are freely used throughout the text:

BURO	: BUEROTECHNIK UND DATENVERARBEITUNG
CACM	: COMMUNICATIONS OF THE ASSOCIATION FOR COMPUTING MACHINERY
IBM JRD	: IBM JOURNAL OF RESEARCH AND DEVELOPMENT
IBM TDB	: IBM TECHNICAL DISCLOSURE BULLETIN
IBM TIE	: IBM TECHNICAL INFORMATION EXCHANGE
JACM	: JOURNAL OF THE ASSOCIATION FOR COMPUTING MACHINERY
PROC. IEEE	: PROCEEDINGS OF THE INSTITUTE OF ELECTRICAL AND ELECTRONICS ENGINEERS
RIDV	: RECHENTECHNIK UND DATENVERARBEITUNG

The indication "ALSO IN : ADT" on the other hand refers to the book edited by McDaniel: "Applications of Decision Tables" (Ref. MCD70A).

As many annotations as possible are drafted based upon the original head-notes. In a few cases, we departed from this general rule because of the discrepancy between the figuring abstract and its relevance with respect to this Report. Some of the entries are incomplete, due to a lack of adequate information.

- ACM66 ASSOCIATION FOR COMPUTING MACHINERY
 DETAB/65 PREPROCESSOR
 SHARE PROGRAM PACKAGE REFERENCE, SDA 3396
 JANUARY 1966.
 THIS PACKAGE DESCRIBES THE ORIGINAL (EXPERIMENTAL)
 DETAB/65 PREPROCESSOR AS COMPILED AND EXECUTED UPON IBM
 7040, 7044 AND 7090/94 COMPUTERS. PARTS OF IT HAVE BEEN
 PUBLISHED SEPARATELY (SEE : POL66, SHA65 AND CHA66B).
- ALB71 ALBRECHT, K. BEUKENBUSCH, H.
 KEMNITZ, J. SCHOETTLER, J.
 SYSTEM ARBEITSPAPIERE BEI VARIANTEN.
 PROBLEMLOESUNG MIT ENTSCHEIDUNGSTABELLEN
 (G)
 PROBLEM SOLVING BY MEANS OF DECISION TABLES.
 AEG-TELEFUNKEN AND HARTMANN UND BRAUN AG
 HEILIGENHAUS, SEPTEMBER 1971.
- ALL71 ALLEN, R.L.
 IMPROVING LEGISLATIVE AND ADMINISTRATIVE PROCESSES IN THE
 PUBLIC SECTOR.
 NAME OF THE PERIODICAL : UNKNOWN
 JANUARY 1971.
 THIS ARTICLE SHOWS IN A SIMPLE WAY HOW LEGISLATIVE AND
 ADMINISTRATIVE PROCEDURES CAN BE IMPROVED BY USING
 THE DECISION TABLE TECHNIQUE. THE EXPLANATION, HOWEVER,
 IS TOO SIMPLE TO BE REALLY CONVINCING.
- ARE67 ARENTZEN, M. NIELSEN, K.A.
 DECISIONTABELLER
 (S) DECISION TABLES
 AUTOMATISK DATABEHANDLING, (7)
 1967, 31.
- ARE68 ARENTZEN, M.
 HVORDAN MAN BENYTTER BESLUTNINGSTABELLerna SOM ARBEYDS- OG
 BESKRIVELSEMETODIK
 (N) DECISION TABLES AS A WORK AND DOCUMENTATION METHODOLOGY
 NORD DATA 68
 HELSINGFORS, 6-8 JUNE 1968, III/1081.
- ARM62 ARMERDING, G.W.
 FORTAB : A DECISION TABLE LANGUAGE FOR SCIENTIFIC COMPUTING
 APPLICATIONS
 PROC. DECISION TABLES SYMPOSIUM
 20-21 SEPTEMBER 1962, 81-87.
 ALSO QUOTED AS : RAND CORPORATION, RM-3306-PR
 SEPTEMBER 1962, 37 PP.
 SCIENTIFIC COMPUTER PROGRAMS, LIKE BUSINESS PROGRAMS,
 OFTEN INVOLVE PROGRAMMED DECISION LOGIC. ALTHOUGH THE
 DECISION TABLE TECHNIQUE IS MAINLY USED IN BUSINESS AND
 COMMERCIAL COMPUTER APPLICATIONS, IT CAN ALSO BE APPLIED
 TO SCIENTIFIC AND ENGINEERING PROBLEMS. FORTAB IS A
 DECISION TABLE LANGUAGE BASED ON THE FORTRAN LANGUAGE.
 PROGRAMS WRITTEN IN THE COMBINED FORTAB AND FORTRAN
 LANGUAGES CAN BE COMPILED BY A FORTAB PREPROCESSOR PRO-
 GRAM WHICH HAS BEEN CONSTRUCTED FOR THE IBM 7090
 COMPUTER.
- ARN71 ARNOLD, H.D.
 UTILISATION OF A DECISION TABLE TRANSLATOR FOR BASIC PROGRAM
 CREATION
 SIGPLAN NOTICES, 6(8)
 SEPTEMBER 1971, 12-19.
 THE DECISION TABLE TRANSLATOR DESCRIBED IS PART OF A
 PACKAGE OF PROGRAMS WRITTEN IN BASIC TO GENERATE AND
 EXECUTE BASIC PROGRAMS. THE DECISION TABLE IS GENERATED
 IN-LINE WITH THE REMAINDER OF THE PROGRAM. ONLY ONE TABLE
 HOWEVER CAN BE GENERATED WITHIN A PROGRAM. MOREOVER,
 THE PROGRAM GENERATED IS PROBABLY LARGER THAN THAT
 CREATED BY OTHER TRANSLATORS.
- ARN75 ARNOLD, G. HALBACH, H.D.
 (G) NEW ASPECTS ON DECISION TABLE TECHNIQUES
 ELEKTRON. RECHENANLAGEN, 17(6)
 DECEMBER 1975, 283-286.
 THIS PAPER TRIES TO PROVIDE A THEORETICAL FUNDATION OF
 THE DECISION TABLE TECHNIQUE BASED ON MATHEMATICAL

CONCEPTS.

- AUE68 AUERBACH CORPORATION
DECISION TABLES - THEIR GENERAL CONSTRUCTION AND ACCEPTANCE
IN PROGRAMMING
AUERBACH STANDARD EDP REPORTS, (12)
1968, 23:030:100-103.
- AVE69 AVEN, I.O. DUSINSKIJ, V.A.
(R) OPTIMIZATION OF MACHINE PROGRAMS BY MEANS OF DECISION
TABLES
EKONOMIKA I MATEMATICHESKIE METODY, 5(6)
1969, 902-908.
DECISION TABLES ARE PRESENTED AS VECTOR FUNCTIONS IN AN
N-DIMENSIONAL UNITY-SPACE. THE ALGORITHM OF REINWALD
AND SOLAND (REIGGB) IS EXTENDED IN ORDER TO INCLUDE THE
TREATMENT OF EXTENDED ENTRY TABLES. A NEW METHOD IS
PROPOSED FOR THE CONVERSION OF SUCH TABLES INTO LIMITED
ENTRY TABLES.
- BAG70 BAGLIN, G. KLEE, J.
LES TABLES DE DECISION : INITIATION PRATIQUE
(F) DECISION TABLES : PRACTICAL INITIATION
ENTREPRISE MODERNE D'EDITION
PARIS, 1970, 111 PP.
THIS MONOGRAPH PURSUES A DOUBLE OBJECT. FOR ONE THING
IT PRESENTS DECISION TABLES AS A VERY USABLE ANALYSIS
TECHNIQUE. ON THE OTHER HAND, IT GIVES A CONCISE
INTRODUCTION TO THE USE OF THE IBM DECISION LOGIC
TRANSLATOR.
- BAR67 BARNARD, T.J.
NITA USER'S GUIDE
NATIONAL COMPUTING CENTRE
1967.
THIS USER'S GUIDE INCLUDES A SECTION ON THE USE OF
DECISION TABLES FOR EXPRESSING SELECTION AND EDITING
PARAMETERS IN NITA, LATER RENAMED AS FILETAB. THE
SPECIFICATION AND THE REPORTS GENERATED.
ILLUSTRATIONS INCLUDE EXAMPLES OF DECISION TABLE
- BAR69A BARALT-TORRIJOS, J.B. PASS, E.M.
DECISION TABLE MINIMIZATION
GEORGIA INSTITUTE OF TECHNOLOGY, GITIS-69-23
ATLANTA, 1969, 21 PP.
THE PAPER PRESENTS A PRELIMINARY REVIEW OF THE WORK WHICH
HAS BEEN DONE ON DECISION TABLE OPTIMIZATION. A BASIC
ALGEBRAIC FORMULATION OF THE PROBLEM IS INTRODUCED,
AND A SOLUTION BASED ON THE GRAPHIC REPRESENTATION OF
A LATTICE IS DESCRIBED. A COMPARISON OF THE ADVANTAGES
AND DISADVANTAGES OF THE USE OF FLOWCHARTS AND DECISION
TABLES IS PRESENTED. THE PRESENTATION IS ILLUSTRATED
WITH A SIMPLE EXAMPLE.
- BAR69B BARNARD, T.J.
A NEW RULE MASK TECHNIQUE FOR INTERPRETING DECISION TABLES
THE COMPUTER BULLETIN, 19(5)
MAY 1969, 153-154.
THE AUTHOR DESCRIBES A TECHNIQUE OF RULE IDENTIFICATION
(BASED ON THE RULE MASK METHOD ORIGINALLY PROPOSED BY
KIRK) USED FOR PARAMETER SETTING IN FILETAB. IT UTILISES
A METHOD OF MATCHING A DATA SITUATION AGAINST ALL RULES
SIMULTANEOUSLY, CONDITION BY CONDITION, AND REJECTING
PROGRESSIVELY THOSE RULES THAT DO NOT MATCH. THE METHOD
DOES NOT SUPPRESS THE TESTING ON NONRELEVANT CONDITIONS
AS CAN BE DONE WITH THE INTERRUPTED RULE MASK TECHNIQUE.
- BAU69 BAUN, W.
ENTScheidungstabellen - Eine Methode zur Analyse und
Programmierung von systematischen Abläufen
(G) DECISION TABLES - A METHOD FOR ANALYSIS AND PROGRAMMING
IBM DEUTSCHLAND, IBM-FORM 81570
SINDELFINGEN, 1969, 18 PP.
THIS ARTICLE GIVES AN INTRODUCTION TO THE DECISION TABLE
TECHNIQUE. THE CONVERSION OF DECISION TABLES INTO
PROGRAMS, ESPECIALLY THE DECISION LOGIC TRANSLATOR, IS
THOROUGHLY TREATED. SOME PRACTICAL EXAMPLES ARE PROVIDED.
- BAU70A BAUN, W.

ARBEIT MIT ENTSCHEIDUNGSTABELLEN

(G) WORKING WITH DECISION TABLES

VDI-NACHRICHTEN, 24(8)

1970, 4.

THE AUTHOR DISCUSSES DECISION TABLES AND THEIR ADVANTAGES. THE CONSTRUCTION AND THE APPLICATION OF EXTENDED ENTRY DECISION TABLES ARE ILLUSTRATED.

- BAU70B BAUN, W.
PRAKTIISCHE ARBEIT MIT ENTSCHEIDUNGSTABELLEN
(G) PRACTICAL EXPERIENCES WITH DECISION TABLES
KEM-ZEITSCHRIFT FÜR KONSTRUKTION UND PRAXIS,
PART I : 7(4), 25-29
PART II : 7(5), 36-38 AND 43-44
1970.
THE BASIS OF THE DECISION TABLE TECHNIQUE IS DESCRIBED AND ILLUSTRATED BY MEANS OF DIFFERENT EXAMPLES. THE AUTHOR GOES ON TO DISCUSS THE USE OF PREPROCESSORS (PARTICULARLY THE DECISION LOGIC TRANSLATOR OF IBM). DECISION TABLES ARE COMPARED WITH FLOWCHARTS AND DIAGRAMS. FINALLY, THE PSYCHOLOGICAL ASPECTS OF THE INTRODUCTION OF THE DECISION TABLE TECHNIQUE IN AN ORGANIZATION ARE MENTIONED.
- BAY73 BAYES, A.J.
A DYNAMIC PROGRAMMING ALGORITHM TO OPTIMISE DECISION TABLE CODE
AUSTRALIAN COMPUTER JOURNAL, 5(2)
MAY 1973, 77-79.
A DYNAMIC PROGRAMMING ALGORITHM IS PRESENTED WHICH, GIVEN A DECISION TABLE, PRODUCES CODE WHOSE AVERAGE EXECUTION TIME IS A MINIMUM. THE ALGORITHM HAS BEEN CODED IN FORTRAN. IT SOLVES AN EIGHT QUESTION DECISION TABLE IN ABOUT 18 SECONDS COMPUTE TIME ON A /360 MODEL 67.
- BEI70 BEISTER, H.
ABLAUFTABELLEN ALS MITTEL DER DOKUMENTATION ZWISCHEN ABLAUFDIAGRAMM UND ENTSCHEIDUNGSTABELLE
(G) CONTROL TABLES, A COMBINATION OF FLOWCHARTS AND DECISION TABLES, AS TOOLS FOR DOCUMENTATION
ELEKTRONISCHE DATENVERARBEITUNG, 11(6)
1970, 283-286.
THE AUTHOR POINTS OUT THAT DECISION TABLES, ON ACCOUNT OF THEIR INHERENT AMOUNT OF CONDITIONS AND ACTIONS, AREN'T VERY WELL-ADAPTED TOOLS FOR DOCUMENTATION. THEREFORE, HE PROPOSES CONTROL TABLES, A COMBINATION OF FLOWCHARTS AND DECISION TABLES. BOTH METHODS ARE COMPARED BY MEANS OF A SIMPLE EXAMPLE.
- BEI73 BEIGANG, O.
ENTSCHEIDUNGSTABELLEN - VORZUEGE UND ANWENDUNGSGRENZEN BEI DER PROGRAMMGESTALTUNG
(G) DECISION TABLES - ADVANTAGES AND LIMITATIONS IN PROGRAM DEVELOPMENT
BURO, 21(1)
JANUARY 1973, 37-41.
THE QUESTION, WHETHER IT MAKES SENSE TO COMPILE A COMMERCIAL PROGRAM ENTIRELY ON THE BASIS OF DECISION TABLES IS EXAMINED WITH THE AID OF PRACTICAL EXAMPLES.
- BELOO BELL CANADA
P.E.T. (PREPROCESSOR FOR ENCODED TABLES) PROCESSOR
USERS MANUAL
PLACE OF ISSUE AND DATE UNKNOWN.
- BER73 BERKEY, M.A.
DECISION TABLES FOR CLERICAL PROCEDURES
MANAGEMENT ACCOUNTING, 55(4)
OCTOBER 1973, 21-26.
DECISION TABLES WERE DEVELOPED AS AN ADDITION TO THE CLASSICAL COMMUNICATION TECHNIQUES. THEY DEAL WITH THE LOGIC OF ANALYSIS OF COMPLEX SITUATIONS. ORIGINALLY USED IN THE SYSTEMS ANALYSIS ENVIRONMENT, DECISION TABLES CAN BE USEFUL IN OTHER AREAS OF COMMUNICATION AND DOCUMENTATION OR IN DESCRIBING PROCEDURES. IN THIS ARTICLE, ONLY LIMITED ENTRY DECISION TABLES ARE CONSIDERED.
- BJO68 BJORK, H.
DECISION TABLES IN ALGOL 60

BIT, 8
1968, 147-153.

THIS PAPER DEFINES AN EXTENSION TO ALGOL 60, WHICH ALLOWS THE PROGRAMMER TO WRITE DECISION TABLES IN HIS ALGOL PROGRAM. A PREPROCESSOR CONVERTS THE DECISION TABLES TO ALGOL, AND ITS OUTPUT IS USED AS INPUT TO THE ORDINARY ALGOL COMPILER. THE GENERATED ALGOL PROGRAM USES A STRAIGHTFORWARD AND EFFICIENT ALGORITHM FOR CHOOSING THE APPROPRIATE DECISION RULE.

- BJ069 BJORK, H.
BESLUTSTABELLER I ALGOL 60
(S) DECISION TABLES IN ALGOL 60
DATASAAB 90006
JANUARY 1969.
- BJ076 BJORK, H.
TEKNIK FOR OVERSATTNING AV BESLUTSTABELLER TILL DATORPROGRAM
(S) TECHNIQUES FOR TRANSLATING DECISION TABLES IN COMPUTER
PROGRAMS
UMEA UNIVERSITET, INST. FOR MATEMATIK OCH STATISTIK,
RAPPORT UMINF-55.76
1976, 31 PP9
DIFFERENT TECHNIQUES OF IMPLEMENTING DECISION TABLES IN
A COMPUTER PROGRAM ARE DISCUSSED. SOME PREPROCESSORS
ARE DESCRIBED, AND DECISION TABLES ARE PUT INTO CONTEXT
WITH THE TECHNIQUE OF STRUCTURED PROGRAMMING.
- BLA73 BLAU, H.
DIE DELEGATION VON VERANTWORTUNG IST NICHT NUR EINE FRAGE
DES FUEHRUNGSSTILS
(G) DELEGATION OF RESPONSABILITY IS NOT ONLY A QUESTION
OF MANAGEMENT STYLE
BURO, 21(1)
JANUARY 1973, 17-21.
THE AUTHOR INTRODUCES THE USE OF DECISION TABLES BY
MEANS OF AN ILLUSTRATIVE EXAMPLE. HE GOES ON TO PROCLAIM
DECISION TABLES THE BASIC OF A COMMUNICATION TOOL THAT
IS USABLE THROUGHOUT ALL HIERARCHICAL LEVELS OF A
COMPANY.
- BOD72 BODE, H.
ENTSCHEIDUNGSTABELLENTECHNIK - ANWENDUNG IN DER ARBEITS -
PLANUNG
(G) THE DECISION TABLE TECHNIQUE - ITS APPLICATION IN WORK
PLANNING
MANUAL
WUERTT. INGENIEURSVEREIN
STUTTGART, 1972.
- BOE00 BOEHM, H.H.
METHODEN DER UMSETZUNG VON ENTSCHEIDUNGSTABELLEN IN
PROGRAMME
(G) METHODS FOR THE CONVERSION OF DECISION TABLES INTO
PROGRAMS
SYSTEMBERATUNG, KARLSRUHE, DATE UNKNOWN.
- BOE62A BOEING
DECISION TABLES TRAINING MANUAL
THE BOEING COMPANY, AEROSPACE DIVISION,
TRAINING SECTION, INDUSTRIAL RELATIONS
1962.
- BOE62B BOERDAM, W.
DECISION TABLES IN SYSTEM DESIGN
UNPUBLISHED PAPER
ATLANTIC RICHFIELD CO.
LOS ANGELES, NO DATE, 9 PP.
- BRA71 BRAUCHLE, W. MELEKIAN, N.
ENTSCHEIDUNGSTABELLENTECHNIK
(G) THE DECISION TABLE TECHNIQUE
MANUAL
WUERTT. INGENIEURSVEREIN
STUTTGART, 1971.
- BRA73 BRAUCHLE, H.
PRAXIS DER ENTSCHEIDUNGSTABELLENTECHNIK
(G) PRACTICE OF THE DECISION TABLE TECHNIQUE

GEHLEN
BADEN-BADEN, 1973, 178 PP.

- BRO00 BROWN, L.M.
DECISION TABLES - AN ELECTRONIC SYSTEMS TOOL
PLACE OF ISSUE AND DATE UNKNOWN.
- BRO62 BROWN, L.M.
DECISION TABLE EXPERIENCE ON A FILE MAINTENANCE SYSTEM
PROC. DECISION TABLES SYMPOSIUM
SEPTEMBER 1962, 75-80.
A DECISION TABLE LANGUAGE AND A COMPUTER PROGRAM PRECOM-PILER WERE DEVELOPED AT THE INSURANCE COMPANY OF NORTH AMERICA TO FACILITATE DESIGN, IMPLEMENTATION AND MAINTENANCE OF A LARGE FILE MAINTENANCE PROGRAM. THE RESULTS OF THIS EFFORT INDICATE THAT DECISION TABLES CAN HAVE APPLICATION OVER THE ENTIRE SYSTEMS DESIGN-PROGRAMMING AREA. DECISION TABLES ALSO FORCE A DISCIPLINED MODULARITY IN THE DESIGN OF A PROGRAM WHICH CAN ENABLE A COMPILER TO ACCOMPLISH SOME OF THE PROGRAM ORGANIZATION FUNCTIONS.
- BUC67 BUCKERFIELD, P.S.T.
A TECHNIQUE FOR THE CONSTRUCTION AND USE OF A GENERALISED INFORMATION TABLE
COMPUTING & DATA PROCESSING DEPARTMENT, ROLLS-ROYCE LIMITED
REPORTED TO HAVE BEEN DELIVERED AT DATAFAIR 67
ALSO IN : INFORMATION PROCESSING 68, 395-402.
THIS PAPER IS CONCERNED WITH THE ESTABLISHMENT AND MAINTENANCE OF A DATA RETRIEVAL SYSTEM FOR DESIGN INFORMATION. IT CONSIDERS TABULAR FORMAT AS A BASIS FOR RETRIEVAL PARAMETER SPECIFICATION. SETS OF DECISION TABLES, CONSTRUCTED BY DIFFERENT DESIGNERS ACCORDING TO THEIR OWN CRITERIA, NEED TO BE COMBINED INTO A GENERALIZED TABLE SPECIFYING ONLY THE ESSENTIAL RELATIONSHIPS.
- BUE70 BUECHNER, O.
ENTSCHEIDUNGSTABELLEN UND IHRE ANWENDUNG
(G) DECISION TABLES AND THEIR APPLICATION
SIEMENS, ERLANGEN, 12.5.1970.
- BUL66 BULL - GENERAL ELECTRIC
LES TABLES DE DECISION
(F) DECISION TABLES
BULLETIN TECHNIQUE, (79)
MARCH 1966, 12 PP.
THE BASIC PROGRAMMING TECHNIQUE OF THE TABSOL LANGUAGE IS EXPLAINED.
- BUR70 BURROUGHS
NAVY DECISION TABLE TRANSLATOR (CODEC) : USER GUIDE
RELEASED UNDER B2508/B3500
TECHNICAL NEWSLETTER NO. 67
26 JUNE 1970.
THE FIRST PART OF THIS MANUAL INTRODUCES AND ILLUSTRATES LIMITED, EXTENDED AND MIXED ENTRY TABLES. THIS IS FOLLOWED BY A SHORT DESCRIPTION OF CODEC, TABLE LINKAGE AND INPUT FORMATS. THE FINAL TWO SECTIONS DESCRIBE CONVENTIONS, RESTRICTIONS, ERRORS AND WARNINGS.
- BUS00 BUSCH, H.J.
BEISPIEL ZUR ANWENDUNG DER ET ZUR BESCHREIBUNG DER STEUERUNG EINES CHARGENPROZESSES DER CHEMISCHEN INDUSTRIE
(G) APPLICATION OF DECISION TABLES IN STEERING CHEMICAL PROCESSES
FORSCHUNGSBERICHT DER SEKTION INFORMATIONSVERARBEITUNG DRESDEN, DATE UNKNOWN.
IN THIS RESEARCH PAPER, DECISION TABLES ARE SUCCESSFULLY USED TO DESCRIBE THE STEERING OF THE REACTOR OF A PVC-PLANT.
- BUS76 BUSCH, H.J. ENGELIEN, M. STAHN, H.
(G) DECISION TABLE TECHNIQUE - AN EXTENDED CONCEPT APPLICABLE TO PROCESSES OF DISCONTINUOUS DISCRETE TYPE
WISS. Z. TECH. UNIV. DRESDEN, 25(1-2), PP 179-187, 1976.
ON THE BASIS OF A FORMAL DESCRIPTION OF COMMONLY USED DECISION TABLE PROCEDURES, A GENERAL, SYSTEMATIC THEORY IS ESTABLISHED. EXAMPLES ILLUSTRATING THE SUITABILITY OF

THE THEORY OF THE DESCRIPTION OF OPERATIONAL TYPE AUTOMATA ARE GIVEN.

- BYR00 BYRNE, P.B.
APPLICATION OF DECISION TABLES TO PROGRAM DESIGN.
IBM TIE 277-4108.
DATE UNKNOWN.
- CAL62 CALKINS, L.W.
PLACE OF DECISION TABLES AND DETAB-X
PROC. DECISION TABLES SYMPOSIUM
SEPTEMBER 1962, 9-12.
THIS ARTICLE BRIEFLY STATES THE HISTORY OF DETAB-X AND ITS RELATIONSHIP TO THE GENERAL DEVELOPMENT OF COBOL.
FOUR MAIN OBJECTIVES OF DETAB-X ARE DEFINED.
- CAM70 CAMPBELL, J.R.
LIFE WITH DECISION TABLES AT MANUFACTURERS LIFE.
NAME OF THE PERIODICAL : UNKNOWN
APRIL 1970.
THE ADOPTION OF THE DECISION TABLE TECHNIQUE AT MANUFACTURERS LIFE INC. IS REVIEWED.
THE ARTICLE DISCUSSES THE ROLE OF DECISION TABLES IN THE SYSTEMS DEVELOPMENT PROCESS, TWO CONTRASTING TYPES OF DECISION TABLES, HOW THEY USE DECISION TABLES AND THE BENIFITS TO BE EXPECTED FROM USING THEM.
- CAN61 CANTRELL, H.N. KING, J. KING, F.E.H.
LOGIC-STRUCTURE TABLES
CACM, 4(6)
JUNE 1961, 272-275.
LOGIC TABLES ARE AN EXCELLENT WAY OF DEVELOPING AND EXPRESSING THE LOGIC REQUIRED IN PROCEDURES, OPERATIONS, SYSTEMS AND CIRCUITS. A SET OF RULES FOR WRITING AND USING LOGIC TABLES IS EXPLAINED BY MEANS OF SOME SIMPLE EXAMPLES. THEN THE LOGIC STRUCTURE OF A VENDING MACHINE IS GIVEN IN WHICH TWO LOGIC TABLES ARE USED.
- CAN62 CANTRELL, H.N.
COMMERCIAL AND ENGINEERING APPLICATIONS OF DECISION TABLES
PROC. DECISION TABLES SYMPOSIUM
SEPTEMBER 1962, 55-61.
SOME OF THE DIFFICULTIES THE AUTHOR HAS HAD IN USING DECISION TABLES ARE MATCHED AGAINST THE ADVANTAGES HE HAS GAINED FROM THEM.
- CAN63 CANNING, R.G. (ED.)
TIME TO CONSIDER DECISION STRUCTURE TABLES AND EDP DESIGN SESSIONS
EDP ANALYZER, 1(4)
MAY 1963, 1-6.
THIS ARTICLE BRIEFLY DESCRIBES THE INITIAL USE OF DECISION TABLES AT GENERAL ELECTRIC AND GOES ON TO DESCRIBE BASIC FORMAT AND CONSTRUCTION OF A TABLE, IN TERMS OF EXTENDED ENTRY TABLES.
- CAN66 CANNING, R.G. (ED.)
HOW TO USE DECISION TABLES
EDP ANALYZER, 4(5)
MAY 1966, 1-14.
THE ARTICLE DESCRIBES EXPERIENCES IN USING DECISION TABLES AT THE RICHFIELD OIL CORPORATION AND LISTS THE WAYS IN WHICH THEY ARE BEING USED. GOALS IN THE USE OF TABLES AND REASONS FOR RESISTANCE ARE DISCUSSED ; THIS SECTION INCLUDES A USEFUL 8-POINT TABLE SHOWING SOME APPROACHES FOR REDUCING PROGRAMMING TIME, AND A 10-POINT LIST OF DECISION TABLES. THE USE OF DECISION TABLES IN SYSTEMS WORK AND AS A PROGRAMMING TOOL ARE CONSIDERED AT SOME LENGTH IN TWO SEPARATE SECTIONS. AN APPENDIX SUMMARISES TYPES OF DECISION TABLES, GENERAL CHARACTERISTICS, RULES, RECURSION, USE OF ELSE, INITIALIZATION, LINKAGE AND THE USE OF THE OR CONNECTOR.
- CAR70 CARRICK, D.A.
INTERPRETATION OF LIMITED ENTRY DECISION TABLE FORMAT
THE COMPUTER JOURNAL, 13
MAY 1970, 220-221.
LETTER REFERRING TO KING69, AND ARGUING, IN OPPOSITION TO THE VIEW EXPRESSED IN THAT PAPER, THAT IT IS NECESSARY TO

ADHERE TO STANDARD CONVENTIONS IN USING DECISION TABLES.
SEE ALSO : KIN71.

- CAV74 CAVOURAS, J.C.
ON THE CONVERSION OF PROGRAMS TO DECISION TABLES : METHOD
AND OBJECTIVES
CACM, 17(8)
AUGUST 1974, 456-462.
THE PROBLEMS OF CONVERTING PROGRAMS TO DECISION TABLES
ARE INVESTIGATED. OBJECTIVES OF THESE CONVERSATIONS ARE
MAINLY PROGRAM DEBUGGING AND OPTIMIZATION IN PRACTICE.
EXTENSIONS TO THE THEORY OF COMPUTATION AND COMPUTABILITY
ARE SUGGESTED.
- CEG66 CEGOS-INFORMATIQUE
LES TABLES DE DECISION : BULLETIN NO. 1 ET 2
(F) DECISION TABLES : BULLETIN 1 AND 2
NEUILLY-SUR-SEINE, 21 JANUARY 1966.
THE FIRST BULLETIN DISCUSSES THE PRINCIPLES OF THE
DECISION TABLE TECHNIQUE. THE SECOND BULLETIN GOES ON
TO DEAL WITH THE APPLICATION AREAS.
- CHA66A CHAPIN, N.
A GUIDE TO DECISION TABLE UTILIZATION
PROC. 1966 DPMA CONFERENCE
LOS ANGELES, OCTOBER 1966.
ALSO IN : DATA PROCESSING, 9 DPMA
PARK RIDGE ILLINOIS, 1967, 327-329.
THE USE OF DECISION TABLES IN PROGRAMMING IS
INVESTIGATED. SOME ADVANTAGES ARE OUTLINED AND A
PRELIMINARY "GUIDE" TO CHOICE BETWEEN MATHEMATICAL
FORMULAS, FLOW-DIAGRAMS AND DECISION TABLES IS PROPOSED.
- CHA66B CHAPMAN, A.E.
DETAB/65 PREPROCESSOR
SHARE PROGRAM LIBRARY PACKAGE, SDA3396
1966.
ALSO IN : ACM DETAB/65 PREPROCESSOR (ACM66).
- CHA67A CHAPIN, N.
AN INTRODUCTION TO DECISION TABLES
DPMA QUARTERLY, 3(3)
APRIL 1967, 3-23.
THE ARTICLE STARTS WITH A GUIDE TO CHOICE BETWEEN
MATHEMATICAL FORMULAS, FLOW-DIAGRAMS/CHARTS AND DECISION
TABLES. THIS IS FOLLOWED BY A BRIEF HISTORY OF THE
DEVELOPMENT OF THE LATTER. GENERAL FORMAT, CONTENTS AND
INTERPRETATION ARE THEN DISCUSSED ; EXAMPLES OF LIMITED,
EXTENDED AND MIXED ENTRY TABLES ARE GIVEN.
THE MAIN PART OF THE PAPER DEALS WITH THE PARSING OF
DECISION TABLES AND COVERS IN ALMOST IDENTICAL TERMS THE
MAIN MATERIAL IN THE AUTHOR'S "PARSING OF DECISION
TABLES" (CHA67B).
- CHA67B CHAPIN, N.
PARSING OF DECISION TABLES
CACM, 10(8)
AUGUST 1967, 507-512.
REDUCTION IN THE SIZE OF DECISION TABLES CAN BE
ACCOMPLISHED BY SEVERAL TECHNIQUES. THE TECHNIQUES
CONSIDERED IN THIS PAPER ARE ON THE PARSING OF DECISION
TABLES WITH REGARD TO HORIZONTAL AND VERTICAL DATA
STRUCTURES, JOB IDENTITY, HARDWARE AND JOB PRIORITIES,
AND CONTEXT RELATIONSHIPS. SUCH PARSING RESTS UPON SOME
CONVENTIONS FOR THE LINKAGE OF DECISION TABLES.
- CHA67C CHAPMAN, A.E. CALLAHAN, M.
A DESCRIPTION OF THE BASIC ALGORITHM USED IN THE DETAB/65
PREPROCESSOR
SYSTEMS DEVELOPMENT CORPORATION, SP-2534/000/00
7 JULY 1966, 18 PP.
ALSO IN : CACM, 10(7)
JULY 1967, 441-446.
THE BASIC ALGORITHM FOR THE CONVERSION OF DECISION TABLES
INTO COBOL CODE CONTAINED IN THE GENERATOR PORTION OF
THE DETAB/65 PREPROCESSOR IS DESCRIBED. THE GENERATOR
ANALYZES A DECISION TABLE AND PRODUCES SIMPLE COBOL
CONDITIONAL STATEMENTS. CORE STORAGE IS SAVED BY USING
QUEUEING TECHNIQUES AND EXTENSIVE INDEXING AND ALSO BY

OUTPUTTING THE CODE AS IT IS GENERATED, A LINE AT A TIME. THE ONLY OPTIMIZATION ATTEMPTED IS THE ELIMINATION OF OBVIOUSLY UNNECESSARY TESTS ON CERTAIN CONDITIONS IN THE DECISION TABLES. SINCE THE PREPROCESSOR AND THE LANGUAGE ASSOCIATED WITH IT WERE DEVELOPED FOR COBOL USERS, THE PREPROCESSOR WAS WRITTEN IN A MODULAR FORM IN REQUIRED COBOL-61.

- CHE70 CHESEBROUGH, W.C.
DECISION TABLES AS A SYSTEMS TECHNIQUE
COMPUTERS AND AUTOMATION, 19(4)
APRIL 1970, 30-33.
THIS ARTICLE DISCUSSES DECISION TABLES AS A METHOD OF BUSINESS PROBLEM ANALYSIS.
THE MAIN TOPICS OF THE ARTICLE ARE DECISION TABLE SYMBOLS AND CONSTRUCTION, SITUATION DEFINITION, EXPANDING THE CONDITION ENTRY QUADRANT, COMPLETING THE ACTION ENTRY QUADRANT, COMPRESSING THE DECISION TABLE, INTUITIVE CONSTRUCTION AND SOLVING LARGE PROBLEMS WITH DECISION TABLES.
- CHE76 CHENG, C.W. RABIN, J.
SYNTHESIS OF DECISION RULES
CACM, 19(7)
JULY 1976, 404-406
DECISION TABLES CAN BE USED AS AN EFFECTIVE TOOL DURING AN INTERVIEW TO RECORD THE LOGIC OF PROCESSES TO BE AUTOMATED. THE RESULT OF SUCH AN INTERVIEW IS NOT A STRUCTURE OF COMPLETE DECISION TABLES BUT RATHER SETS OF DECISION RULES. THE PURPOSE OF THIS PAPER IS TO PROVIDE A PROCEDURE FOR SYNTHESIZING THE DECISION RULES AND THUS PROVIDE AN AID IN DEVELOPING A STRUCTURE OF COMPLETE DECISION TABLES.
- CHI74 CHISWELL, P.R.G.
THE PRACTICAL USE OF DECISION TABLES AS A PROGRAMMING AID
SYSTEMS (S. AFRICA), 4(9)
SEPTEMBER 1974, 10-14.
THIS ARTICLE DEMONSTRATES HOW THE DT-TECHNIQUE CAN BE USED SUCCESSFULLY AND WHAT BENEFITS RESULT FROM ITS IMPLEMENTATION. IT IS INTENDED TO CORRECT ANY MIS-CONCEPTIONS THAT EXIST ABOUT THE TECHNIQUE.
- CHI79 CHINNASWAMY, V.
TRANSLATION OF DECISION TABLES INTO COMPUTER PROGRAMS WITH THE USE OF CARNAUGH MAPS
SIGPLAN NOTICES, 14(5)
MAY 1979, 21-23.
A METHOD FOR CONVERSION LIMITED-ENTRY DECISION TABLES INTO MINIMAL PROGRAM CODE, WHICH IS BASED ON THE WELL-KNOWN USE OF CARNAUGH MAPS, IS PRESENTED.
- CHV71 CHVALOVSKY, V.
POUZHITI ROZHODVACICH TABULEK PRZI VYVOJI SOFTWARE (C) USE OF DECISION TABLES IN SOFTWARE DEVELOPMENT
INORGA PRAG, (1)
1971, 9-15.
- CHV76 CHVALOVSKY, V.
PROBLEMS WITH DECISION TABLES
CACM, 19(12)
DECEMBER 1976, 705-707.
SOME COMMENTS ON THE IMPACT OF CONVERSION ALGORITHMS ON THE USE OF DECISION TABLES ARE GIVEN; THEY WERE STIMULATED BY THE ARTICLE SCH76. THE AUTHOR'S REJOINDER IS ADDED.
- CLA62 CLARKE, D.S.
STRUCTURE TABLE LOGIC
SHARE SECRETARY DISTRIBUTION, (96)
OCTOBER 1962.
ALSO PRESENTED AT THE SHARE MEETING IN TORONTO 11-13 SEPTEMBER, 1962.
A PAPER PRESENTED AT THE SHARE MEETING IN TORONTO, SEPTEMBER 11-13, 1962 DESCRIBING A LANGUAGE BASED ON DECISION TABLES FOR USE IN SCIENTIFIC APPLICATIONS.
- COD62A CODASYL
PROCEEDINGS OF THE DECISION TABLES SYMPOSIUM

SPONSORED BY THE CODASYL SYSTEMS GROUP AND THE JOINT USERS
GROUP OF ACM
NEW YORK, 20-21 SEPTEMBER 1962, 116 PP.

THIS DOCUMENT COUNTAINS THE PROCEEDINGS OF A SYMPOSIUM
ON DECISION TABLES. IN THIS BIBLIOGRAPHY, THE INSERTED
PAPERS ARE ANNOTATED SEPARATELY.

- COD62B CODASYL
DECISION TABLE TUTORIAL USING DETAB-X
PREPARED BY INSTRUCTION TASK FORCE - CODASYL SYSTEMS
DEVELOPMENT GROUP
NEW YORK, 1962, 99 PP.
THIS EARLY TEXT WAS DEVELOPED :
1. TO GIVE AN INTRODUCTORY LEARNING EXPERIENCE IN THE
USE OF DECISION TABLES THEMSELVES.
2. TO GIVE SPECIFIC INSTRUCTIONS IN THE USE OF DETAB-X,
AN EXPERIMENTAL LANGUAGE COMBINING DECISION TABLE
STRUCTURE AND COBOL-61 LANGUAGE WITH MINOR MODIFICATION.
- COD62C CODASYL
DETAB-X, PRELIMINARY SPECIFICATIONS FOR A DECISION TABLE
STRUCTURED LANGUAGE
PREPARED BY CODASYL SYSTEMS GROUP
NEW YORK, 1962, 119 PP.
THE SYSTEMS GROUP OF CODASYL, AS A FIRST STEP IN CREATING
A DATA-PROCESSING LANGUAGE BASED ON DECISION TABLES,
HAS DEVELOPED DETAB-X, A DECISION TABLE LANGUAGE BASED
ON COBOL-61. BECAUSE DECISION TABLES ARE STRUCTURED
DIFFERENTLY FROM THE FREE-FORM PROCEDURE STATEMENTS OF
COBOL-61, SOME MODIFICATIONS TO COBOL-61 ARE REQUIRED ;
HOWEVER, THESE ARE HELD TO A MINIMUM AND ARE OF SUCH A
NATURE AS TO ENABLE A RELATIVELY SIMPLE PREPROCESSOR TO
CONVERT THE DECISION TABLES STATEMENTS TO COBOL-61
STATEMENTS WHICH CAN THEN, IN TURN, BE PROCESSED BY A
COBOL-61 COMPILER (OR PROCESSOR). THIS DETAB-X MANUAL
HAS BEEN PREPARED AS A LANGUAGE SPECIFICATION REFERENCE
PUBLICATION, SUPPLEMENTARY TO THE OFFICIAL COBOL-61
MANUAL PUBLISHED BY THE UNITED STATES GOVERNMENT PRINTING
OFFICE. IT PROVIDES SUFFICIENT INFORMATION TO PERMIT
EXPERIMENTATION BY MANY COBOL USERS.
- COD66 CODASYL
DRAFT OF DECISION TABLE STANDARDS
DECISION TABLE TASK FORCE OF SYSTEMS COMMITTEE
NEW YORK, MARCH 1966.
- CON63 CONNOR, A.C.
DETAB-X - FROM A CRITICAL POINT OF VIEW
GUIDE
OCTOBER 1963.
- CON71 CONNORS, M.M. CORAY, C. CUCCARO, C.J.
GREEN, W.K. LOW, D.W.
PRODUCTION OF CUSTOMIZED PROGRAMS BY QUESTIONNAIRE AND
DECISION TABLES
IBM TDB, 13(11)
APRIL 1971, 3384.
DECISION TABLES MAY PRODUCE CUSTOMIZED PROGRAMS IN ACCORDANCE WITH ANSWERS TO A QUESTIONNAIRE. DECISION TABLES ARE PROCESSED SEQUENTIALLY, EACH ONE CORRESPONDING TO A SUBSET OF POTENTIAL ENTRIES IN THE QUESTIONNAIRE. THE SUBSET MAY COMprise A GROUPING OF ALTERNATIVES, ONE OF WHICH IS TO BE SELECTED. THE DECISION TABLE COMPRISSES INSTRUCTIONS FOR EXAMINING "YES" OR "NO" ANSWERS TO THE CORRESPONDING QUESTIONS OF THE QUESTIONNAIRE TO SELECT PROGRAM SOURCE STATEMENTS. THE INSTRUCTIONS ARE IN THE FORM OF A MATRIX ARRAY.
- CON72 CONNORS, M.M. CORAY, C. CUCCARO, C.J.
GREEN, W.K. LOW, D.W. MARKOWITZ, H.M.
THE DISTRIBUTION SYSTEM SIMULATOR
MANAGEMENT SCIENCE, 18(8)
APRIL 1972, B/425-B/453.
THE BASIC CONCEPTS OF PROGRAMMING BY QUESTIONNAIRE, AS PROPOSED IN LOW73, ARE ILLUSTRATED BY MEANS OF AN ELABORATED EXAMPLE. THE USER OF THE DSS-SYSTEM SPECIFIES THE CHARACTERISTICS OF HIS OWN DISTRIBUTION SYSTEM IN ANSWER TO A QUESTIONNAIRE. THE DSS-SYSTEM GENERATES THE RESULTING SIMULATION MODEL MATCHING THE ANSWERS AGAINST

PREDEFINED TEMPLATES, WHICH ARE STORED IN DECISION TABLE FORM.

- COU67 COULTER, K.J.
PREPROCESSOR FOR ENCODED TABLES PET
IBM REF. 03.2.004
APRIL 1967.
PET IS A DECISION TABLE TRANSLATOR. THE SOURCE LANGUAGE IS A COMBINATION OF PL/1 AND PET VERBS. THROUGH RULE SORTING AND THE CROSS-LINKING OF DUPLICATE ACTION STRINGS, EFFICIENT STRUCTURING OF THE INPUT TABLE'S LOGIC PROCEDURES OPTIMIZED OUTPUT PROGRAM CODE. A COMPLETE ERROR CHECK PLUS A TABLE UPDATE FACILITY PERMITS DEBUGGING AT THE LOGIC LEVEL.
- COU69 COULTER, K.J.
A DECISION TABLE TRANSLATOR FOR MODULAR COBOL PROGRAMMING
UNIVAC SCIENTIFIC EXCHANGE
MARCH 1969.
THIS IS BASICALLY A DESCRIPTION OF MOST (MODULAR SYSTEM TRANSLATOR) DEVELOPED FOR USE ON THE 1108 BY THE BELL TELEPHONE CO. OF CANADA ; THIS INCORPORATES A DECISION TABLE TRANSLATOR. THE PAPER CONCERNS ITSELF TO A GREAT EXTENT WITH PROBLEMS OF CORE STORAGE ORGANIZATION WHEN USING MODULAR PROGRAMS ; IN ADDITION, IT DESCRIBES THE OPERATION OF THE DECISION TABLE TRANSLATOR WHICH READS RATHER LIKE PET (SEE : COU67).
- CSA68 CANADIAN STANDARDS ASSOCIATION
DECISION TABLES
A DRAFT CSA STANDARD
SEPTEMBER 1968, 24 PP.
THIS STANDARD APPLIES TO THE FORMAT AND USE OF LIMITED ENTRY DECISION TABLES ON 80 COLUMN PUNCHED CARDS A LANGUAGE IS NOT DEFINED ; IT IS STATED THAT EFFORTS HAVE BEEN MADE TO ENSURE COBOL COMPATABILITY AND A COBOL CODING SHEET IS RECOMMENDED FOR USE WHEN PRODUCING A TABLE. A SHORT LIST OF DEFINITIONS IS FOLLOWED BY LAYOUTS AND FULL DESCRIPTIONS OF NINE CARDS ; THESE INCLUDE INITIALIZATION, STUB CONTINUATION AND COMMENT FACILITIES. AN APPENDIX DEALS WITH DECISION TABLE CONCEPTS IN DEPTH.
- CUN62 CUNNINGHAM, J.
DECISION TABLES SYMPOSIUM
PROC. DECISION TABLES SYMPOSIUM
SEPTEMBER 1962, 7-8.
- CUV60 CUVELETTE, J.N.
DECISION TABLES
IBM TIE 277-7051
DATE UNKNOWN.
- CUV66 CUVELETTE, J.N.
APPLICATION DES TABLES DE DECISION A L'AUTOMATISATION DES ETUDES DE PRODUCTION ET DES PLANNINGS DE FABRICATION
(F) APPLICATION OF DECISION TABLES FOR AUTOMATED DESIGN ENGINEERING AND AUTOMATED MANUFACTURING PLANNING
AUTOMATISME, 11(10)
OCTOBER 1966, 552-559.
DECISION TABLES ARE OF GREAT VALUE IN MANUFACTURING PROBLEMS (AUTOMATED DESIGN ENGINEERING - ADE). BECAUSE OF THE AUTOMATIC CREATION OF MANUFACTURING FILES, MANUFACTURING ENGINEERS ARE NOW ABLE TO SPEND MORE VALUABLE TIME ON MORE INTERESTING AND MORE PAYING PROBLEMS. THIS TECHNIQUE PERMITS TO DETECT SPECIFICATION ERRORS OR TECHNICAL INCOMPATIBILITIES BEFORE COMPUTING AND NOT AT THE MOMENT OF ASSEMBLING. THIS CAN AVOID WASTAGE OF MATERIAL, LOSS OF TIME AND HENCE ALSO PRODUCTION DELAYS.
- CUV70 CUVELETTE, J.N.
DECISION TABLE TRANSLATOR
IBM TDB, 13(6)
NOVEMBER 1970, 1710-1713.
THE DECISION TABLE TRANSLATOR PROGRAM PROPOSED ACCEPTS DECISION TABLES WRITTEN WITH A FORTRAN, COBOL, OR PL/1 ORIENTED LANGUAGE AND TRANSLATES THEM INTO A FORTRAN, COBOL OR PL/1 PROGRAM WHICH IS STORED ON DISK OR TAPE READY FOR COMPILE WITHOUT MANIPULATION. TO GENERATE

THE MINIMUM NUMBER OF STATEMENTS, EACH PROGRAM MUST PERFORM SOME OPERATIONS. THESE OPERATIONS AND ACTIONS OF THE DECISION TABLE ARE DESCRIBED.

- CUV71 CUVLETTE, J.N.
DECTAT : DECISION TABLE TRANSLATOR FOR COBOL AND PL/1
DECISION TABLES CONFERENCE
AMSTERDAM, DECEMBER 1971.
- CUV73 CUVLETTE, J.N.
LA TRADUCTION DES TABLES DE DECISION
(F) THE TRANSLATION OF DECISION TABLES
INFORMATIQUE ET GESTION, (45)
1973, 101-106.
AFTER INTRODUCING DECISION TABLES AND THEIR TRANSLATORS,
THE AUTHOR GOES ON TO PRESENT THE CONVERSION METHODS
USED IN THE IBM DECTAT-PREPROCESSOR.
- DAI71 DAILEY, W.H.
SOME NOTES ON PROCESSING LIMITED ENTRY DECISION TABLES
SIGPLAN NOTICES, 6(8)
SEPTEMBER 1971, 81-89.
THIS PAPER DEALS WITH THE AUTOMATIC GENERATION OF CODE
STATEMENTS FOR INPUT TO A COMPUTER. THE TREATMENT IS
RATHER SUPERFICIAL.
- DAI75A DAILEY, W.H.
A TABULAR APPROACH TO PROGRAM OPTIMIZATION
SIGPLAN NOTICES, 10(11)
NOVEMBER 1975, 17-22.
A PROCEDURE FOR OPTIMIZING PROGRAMS IS PRESENTED WHICH
IS NOT DEPENDENT UPON EXISTING CONTROL STRUCTURE. IT
BEGINS BY SELECTING OUT LISTS OF ESSENTIAL CONDITION AND
ACTION STATEMENTS FROM WHICH A DECISION GRID CHART IS
BUILT. FROM THE DECISION GRID CHART AND THE STATEMENT
LISTS A SET OF DECISION TABLES IS DRAWN. FINALLY THE
DECISION TABLES ARE PROCESSED TO GENERATE AN OPTIMAL
PROGRAM.
- DAI75B DAILEY, W.H.
ON GENERATING BINARY DECISION TREES WITH MINIMUM NODES
SIGPLAN NOTICES, 10(12)
DECEMBER 1975, 14-21.
THIS PAPER DESCRIBES A TECHNIQUE FOR GENERATING BINARY
DECISION TREES FROM DECISION TABLES. IT STARTS WITH A
FULLY EXPANDED TABLE AND DOES A PARTIAL REDUCTION WITH
RESPECT TO THE ACTIONS. THE REMAINDER OF THE REDUCTION IS
DONE WITHOUT REFERENCE TO ACTIONS. POLLACK'S ALGORITHM IS
FINALLY APPLIED TO DETERMINE THE NEXT DECISION TO
CONSIDER.
- DAP70 DAPRON, F.E.
OPTIMIZING DECISION TABLES
IBM TDB, 13(7)
DECEMBER 1970, 2033-2036.
THE DECISION TABLE CAN BE CONSIDERED AS A NONAMBIGUOUS
STATEMENT OF THE LOGICAL RELATIONSHIPS BETWEEN THE CONDI-
TIONS AFFECTING A PROBLEM. IN REDUCING THIS STATEMENT TO
A SEQUENTIAL TESTING TREE IT IS DESIRABLE TO DEVISE A
TREE WHICH WILL GIVE MINIMUM AVERAGE TESTING PATH LENGTH
(MINIMUM AVERAGE NUMBER OF TESTS). THE PROPOSED ALGO-
RITHM DERIVES QUANTITATIVE MEASURES OF THE PERTINENT
FACTORS FROM THE CONDITION RULES OF THE DECISION TABLE
DESCRIPTION OF THE PROBLEM STRUCTURE.
- DAP74A DAPRON, F.E.
DETERMINING THE LOGICAL SEQUENCE IN DECISION TABLES
IBM TDB, 16(12)
MAY 1974, 4051-4054.
WHEN THE LOGIC OF AN ALGORITHM IS TO BE IMPLEMENTED BY
SEQUENTIAL TESTING OF THE LOGICAL VARIABLES AND THE LOGIC
IS REPRESENTED BY A DECISION TABLE, THE TEST FOR WHETHER
A DISCONTINUITY IS DETERMINATE IS THAT SEQUENTIALLY
PRECEDENT TO IT, THERE MUST HAVE BEEN SOME DIFFERENCE
BETWEEN THE RULE CONTAINING THE DISCONTINUITY AND ALL
RULES NOT CONTAINING IT, AND THAT THIS DIFFERENCE BE
SOME EXPLICIT VALUE.
AN EXTENSION OF A PREVIOUSLY PUBLISHED TECHNIQUE IS
PRESENTED THAT GREATLY SIMPLIFIES DETERMINATION OF

LOGICAL SEQUENCE CONSTRAINTS.

- DAP74B DAPRON, F.E.
DETERMINING INDEPENDENCE/DEPENDENCE OF STATE FUNCTIONS IN A
MULTIPATH ALGORITHM
IBM TDB, 16(12)
MAY 1974, 4055-4057.
- IN PARTITIONING AN ALGORITHM, THE BEST CANDIDATES FOR EACH PARTITION ARE THOSE STATE FUNCTIONS WHICH ARE DEPENDENT ON EACH OTHER AND INDEPENDENT OF ALL STATE FUNCTIONS IN OTHER PARTITIONS. THE METHOD PRESENTED WAS DEVELOPED PRIMARILY FOR ANALYZING THE SEQUENCE RELATIONSHIPS IN A SET OF DECISION TABLE ACTION RULES AND IS PRESENTED IN THAT NOTATION, WITH THE RESOLVING MATRIX BEING A PRECEDENCE MATRIX.
- DAR71 DARWOOD, N.
THE SIMPLIFICATION OF DECISION TABLES
COMPUTER WEEKLY, (249)
JULY 1971, 6.
- TABLES ARE USED BY SYSTEMS ANALYSTS AND PROGRAMMERS AS AN AID TO DOCUMENTATION AND WILL IN THE FUTURE BE PART OF DATA PROCESSING'S STOCK-IN-TRADE. THIS ARTICLE DESCRIBES DECISION TABLES UP TO FOUR CONDITION VARIABLES.
- DAT72A DATHE, G.
ENTScheidungstabellen : WIRTSCHAFTLICHKEIT
(G) DECISION TABLES : EFFICIENCY
INFORMATION, 26(4)
1972, 19-21.
- IT IS SAID IN DAT72C THAT THIS ARTICLE DESCRIBES THE AGENTA-PREPROCESSOR.
- DAT72B DATHE, G.
CONVERSION OF DECISION TABLES BY RULE MASK METHOD WITHOUT
RULE MASK
CACM, 15(10)
OCTOBER 1972, 906-909.
- TWO ALGORITHMS FOR GENERATING COMPUTER PROGRAMS FROM DECISION TABLES ARE DESCRIBED. THE ALGORITHMS ALLOW HANDLING LIMITED ENTRY, EXTENDED ENTRY AND MIXED ENTRY TABLES. THE ALGORITHMS ARE BASED ON THE RULE MASK METHOD BUT NEED NOT HAVE THE MASKS AT EXECUTION TIME. THEY PERFORM THE LOGICAL OPERATIONS IMMEDIATELY RATHER THAN AT THE END OF THE INTERPRETING PROCESS. EXECUTION TIME CAN BE CONSIDERABLY REDUCED BY INSTANTLY MARKING RULES WHICH ARE NOT APPLICABLE (ALGORITHMS 1 AND 2) OR CONDITIONS WHICH ARE ALREADY TESTED (ALGORITHM 2). THE NEW ALGORITHMS COMBINE TO A CERTAIN DEGREE THE ADVANTAGES OF MASK METHODS WITH THOSE OF TREE METHODS.
- DAT72C DATHE, G.
MEHRDEUTIGE ENTScheidungstabellen
(G) MULTIPLE HIT DECISION TABLES
IN : GI - 2. JAHRSTAGUNG
KARLSRUHE, 2. - 4.10.1972.
- THE MULTIPLE HIT TABLE, I.E. A DECISION TABLE WITH MUTUALLY NON-EXCLUSIVE RULES, IS PRESENTED AND ITS ADVANTAGES ARE INDICATED.
- DAV70 DAVIS, R.A.
TABLE DRIVEN PROGRAM
IBM PATENT USA 3702007
7 DECEMBER 1970 (PUBLISHED 31 OCTOBER 1972).
- A NEW METHOD OF USING DECISION TABLES WITH COMPUTERS APPARATUS IS DISCLOSED. A GENERAL PURPOSE DRIVER IS PROVIDED FOR EXECUTING VARIOUS DECISION TABLES. WHEN A PROBLEM PROGRAM REACHES THE POINT WHERE A DECISION TABLE IS TO BE EXECUTED, THE PROBLEM PROGRAM CALLS THE DRIVER AND IDENTIFIES THE SELECTED DECISION TABLE. THE CONDITION STUB OF THE DECISION TABLE INCLUDES A SERIES OF INSTRUCTIONS FROM WHICH THE DRIVER FORMS A CONDITION MASK. THE DRIVER SELECTS THE APPROPRIATE ACTION ACCORDING TO THE CONDITION MASK AND A SET OF RULES AND AN ACTION STUB IN THE DECISION TABLE.
- DAV73 DAVIES, G. WELLAND, R.
A PREPROCESSOR USING RULE MASK TECHNIQUES FOR EXTENDED ENTRY DECISION TABLES

SOFTWARE PRACT. & EXPER., 3(3)
JULY-SEPTEMBER 1973, 227-234.

THE AUTHOR DESCRIBES THE IMPLEMENTATION OF A PREPROCESSOR FOR EXTENDED ENTRY DECISION TABLES. THE ACTIONS TO BE TAKEN IN CASE OF AMBIGUITY ARE DISCUSSED.

- DAV74 DAVIES, N.R.
DECISION TABLES IN DISCRETE-SYSTEM SIMULATION
SIMULATION, 22(2)
FEBRUARY 1974, 39-44.
DECISION TABLES PROVIDE A CLEAR AND CONCISE FORMAT FOR SPECIFYING A COMPLEX SET OF CONDITIONS AND THE VARIOUS CONSEQUENT COURSES OF ACTION. THEY ARE THEREFORE IDEAL FOR DESCRIBING THE CONDITIONS FOR INTERACTION BETWEEN COMPONENT PARTS OF A SIMULATION MODEL AS SPECIFIED IN THE USER-DEFINED EVENT ROUTINES. A PREPROCESSOR TO MAKE DECISION TABLES COMPUTER-READABLE WOULD GREATLY ENHANCE THE PROCESS OF MODELLING AND WOULD ALLOW A CONSIDERABLE EXTENSION OF THE RANGE OF CONDITIONED STATEMENTS TO BE USED IN THE CONDITION STUBS OF THE TABLE. A DECISION-TABLE FACILITY WOULD FORM A USEFUL EXTENSION TO THE MANY EVENT-ORIENTED SIMULATION PROGRAMMING LANGUAGES.
- DEA71 DEAN, R.N.
A COMPARISON OF DECISION TABLES AGAINST CONVENTIONAL COBOL AS A PROGRAMMING TOOL FOR COMMERCIAL APPLICATIONS
SOFTWARE WORLD, 2(3)
SPRING 1971, 26-30.
THE AUTHOR PROVIDES A COMPARISON OF THE USE OF COBOL AND DECISION TABLES AS A PROGRAMMING TOOL BY MEANS OF FIVE DETAILED CASE STUDIES. SOME INDIVIDUAL COMMENTS ARE ADDED.
- DEC73 DE COCQUEAU DES MOTTES, E.
ANALYSE AUTOMATIQUE D'UN PROBLEME PAR TABLES DE DECISION ET EQUATIONS LOGIQUES
(F) AUTOMATIC PROBLEM ANALYSIS BY MEANS OF DECISION TABLES AND LOGIC EQUATIONS
FAC. UNIV. NOTRE DAME DE LA PAIX, INSTITUT D'INFORMATIQUE NAMUR, 1973.
AFTER A GENERAL PROBLEM DEFINITION AND AN INTRODUCTION TO THE USE OF DECISION TABLES, THE AUTHOR CALLS ATTENTION TO THE STUDY OF LOGICAL BOOLEAN EQUATIONS. IN A FOURTH CHAPTER, HE DESCRIBES A PROGRAM GENERATOR ACCEPTING SIMPLE BOOLEAN EQUATIONS AND/OR DECISION TABLES AS INPUT. AN ASSEMBLER PROGRAM IS GENERATED MAKING USE OF THE DECISION TABLE TECHNIQUE. SOME SIMPLE EXAMPLES ARE GIVEN.
- DEF67 DEFORT, W.F.
BESLISSINGSTABELLEN
(D) DECISION TABLES
INFORMATIE, 9(10)
OCTOBER 1967, 206-213.
- DEL70 DELDYCKE, J.
HET OMZETTEN VAN BESLISSINGSTABELLEN TOT COMPUTERPROGRAMMA'S VIA NETWERKEN
(D) THE CONVERSION OF DECISION TABLES INTO COMPUTER PROGRAMS BY NETWORK METHODS
KUL, INSTITUUT VOOR TOEGEPASTE ECONOMISCHE WETENSCHAPPEN SEMINARIE ELEKTRONISCHE INFORMATIEVERWERKING LEUVEN, 1970-1971, 54 PP.
THIS INTERNAL SEMINAR PAPER COMPARES THREE CONVERSION ALGORITHMS WITH RELATION TO MINIMUM EXECUTION TIME : THE OPTIMUM-APPROACHING ALGORITHM PROPOSED BY POLLACK (POL64A) AND THE OPTIMUM-FINDING ALGORITHMS OF REINWALD & SOLAND (REI66B) AND OF VERHELST.
- DEM68 DEMOTES-MAINARD, G.
UNE METHODE D'ANALYSE : CORIG
(F) A METHOD OF ANALYSIS : CORIG
INFORMATIQUE ET GESTION, 10(1)
DECEMBER 1968, 84-89.
THE AUTHOR PRESENTS CORIG, A SYSTEMS ANALYSIS METHOD AIMING AT A FULLY LINEAR CONSTRUCTION OF PROGRAMS. EACH PROBLEM THEREFORE MUST BE SPLIT UP IN TWO CONCEPTUALLY DISTINGUISHED PARTS. FIRST OF ALL, THE ANALYST EXAMINES THE PROBLEM STATEMENT TAKING THE VIEW OF THE ORGANISATION ; AFTERWARDS, HE LOOKS AT IT FROM THE POINT

OF VIEW OF THE COMPUTER IMPLEMENTATION.
IN THE RESULTING FLOW DIAGRAM, EACH TREATMENT HAS TO BE
EXECUTED DEPENDING UPON THE RESULT OF A CERTAIN NUMBER OF
BOOLEAN CONDITION TESTS. ESSENTIALLY, SUCH A TEST CAN BE
COMPARED WITH A DECISION TABLE CALL.

- DEM77 . DE MEERSMAN, F.
DE OMZETTING VAN LIMITED-ENTRY BESLISSINGSTABELLEN TOT
COMPUTERPROGRAMMA'S
(D) THE CONVERSION OF LIMITED-ENTRY DECISION TABLES INTO
COMPUTER PROGRAMS
INFORMATIE, 19(11)
NOVEMBER 1977, 653-662.
THE AUTHOR PRESENTS, WITHOUT EXPLICITLY MAKING REFERENCE
TO THE ORIGINAL PAPER, A VEINOTT-LIKE ALGORITHM FOR THE
CONVERSION OF LIMITED-ENTRY DECISION TABLES INTO COMPUTER
PROGRAMS (SEE VEIGGA AND VEI66B) ; THIS ALGORITHM
EXPLICITLY DEALS WITH CONTRADICTIONS AND IMPLIED
CONDITION STATES.
- DENO0 DENT, J.L.
DECISION TABLES. A STATE-OF-THE-ART REPORT
MONOGRAPH 2, WORKING PAPER SERIES
SCHOOL OF BUSINESS ADMINISTRATION, UNIV. OF MINNESOTA
DATE UNKNOWN, PROBABLY 1969, 60 PP.
THE TOPICS THAT ARE COVERED IN THIS STATE-OF-THE-ART
REPORT ARE : THE HISTORY OF DECISION TABLE DEVELOPMENT,
THE BASIC FORMS, HOW THEY ARE WRITTEN, THEIR CONVERSION
TO COMPUTER PROGRAMS, CURRENT APPLICATIONS AND AN
EVALUATION OF THE TECHNIQUE.
- DEN68A DENOLF, H.
DECISION TABLES - AN ANNOTATED BIBLIOGRAPHY
IAG QUARTERLY JOURNAL, 1(1)
1968, 67-82.
THIS BIBLIOGRAPHY CONTAINS 76 REFERENCES, MOSTLY ANNOTA-
TED. MANY ANNOTATIONS APPEAR TO BE THE ORIGINAL ABSTRACTS
OR HEAD-NOTES ; SOME OTHERS ARE SIMILAR TO SHAW'S EARLIER
NOTES.
- DEN68B DENOLF, H.
DECISION TABLES - A STATUS QUAESTIONIS
KUL, INSTITUUT VOOR TOEGEP. EKON. WETENSCHAPPEN
LEUVEN, 1968.
THE INTENTION OF THIS THESIS IS TO PRESENT A STATUS
QUAESTIONIS CONCERNING THE DEVELOPMENT AND APPLICATION
AREAS OF DECISION TABLES. IT IS THE AIM OF PART I TO
SHOW THAT DECISION TABLES ARE HIGHLY VISUAL, EASY TO
GRASP, FORMS OF EXPRESSING COMPLEX LOGIC. PART II
DEMONSTRATES SCHEMATICALLY THE SPECTACULAR RESULTS OF THE
USE OF DECISION TABLES. PART III INDICATES CLEARLY THAT
THE USE OF PROCESSORS OR COMPILERS CAN INCREASE THE
POWER OF THIS PROGRAMMING TOOL. IN PART IV, TWO "SIMILAR"
METHODS ARE DESCRIBED BRIEFLY.
- DEPO0 DEPARTMENT OF THE AIR FORCE
DECISION LOGIC TABLES
BOLLING AIR FORCE BASE
WASHINGTON, DATE UNKNOWN.
- DEP65 DEPARTMENT OF THE AIR FORCE
DECISION LOGIC TABLE TECHNIQUE
USAF MILITARY PERSONNEL CENTRE, RANDOLPH AIR FORCE BASE,
PAMPHLET AFP5-1-1
TEXAS, 1965, 42 PP.
- DER74 DERAS, R.
OMZETTING VAN BESLISSINGSTABELLEN TOT COMPUTERPROGRAMMA'S -
BESPREKING EN EVALUATIE VAN ALGORITHMEN
(D) CONVERSION OF DECISION TABLES INTO COMPUTER PROGRAMS -
DISCUSSION AND EVALUATION OF SOME ALGORITHMS
KUL, INSTITUUT VOOR TOEGEPASTE ECONOMISCHE WETENSCHAPPEN
LEUVEN, 1974, 121 PP.
AFTER A SURVEY OF THE EXISTING CONVERSION ALGORITHMS, THE
AUTHOR OF THIS THESIS GOES ON TO COMPARE THOROUGHLY THE
OPTIMUM APPROACHING ALGORITHMS OF POLLACK (POL64A),
SHWAYDER (SHW71) AND VERHELST (VER72A). FOR THIS PURPOSE,
HE USES A SAMPLE OF 84 DECISION TABLES. IN A THIRD PART,
THE USE OF AN ALGORITHM THAT SEARCHES MORE THAN ONE STEP

AT A TIME IS INVESTIGATED.

- DEV62 DEVINE, D.J.
LOBOC, LOGICAL BUSINESS-ORIENTED CODING
INSURANCE COMPANY OF NORTH AMERICA
DISTRIBUTED TO ATTENDEES OF THE GUIDE
INTERNATIONAL MEETING
PHILADELPHIA, OCTOBER 1962.
- DEV65 DEVINE, D.J.
DECISION TABLES AS THE BASIS OF A PROGRAMMING LANGUAGE
PROC. 1965 INTERNATIONAL DATA PROCESSING CONFERENCE
1965, 461-466.
ALSO IN : ADT, 155-161.
AFTER A BRIEF HISTORY OF THE DEVELOPMENT OF LOBOC IN THE
INSURANCE COMPANY OF NORTH AMERICA, THIS PAPER GOES ON
TO DEFINE THE NECESSARY SUB-DIVISIONS OF A DECISION
TABLE PROGRAMMING LANGUAGE AS DATA DIVISION, ACTION
DIVISION, DECISION DIVISION AND DIRECTOR. THESE ARE
FURTHER EXPANDED BUT NOT ILLUSTRATED.
- DEV66 DEVINE, D.J.
DECISION TABLE SEMINAR
STUDENT TEXT, TRILOG ASSOCIATES
1966.
- DEV67 DEVINE, D.J.
DECISION TABLES IN SYSTEMS DESIGN AND MAINTENANCE
A PAPER PRESENTED TO THE SPRING JOINT CONFERENCE
ATLANTIC CITY, APRIL 1967.
THIS PAPER INCLUDES SOME PERFORMANCE STATISTICS ON THE
USE OF DECISION TABLES.
- DEV71 DE VRIES, P. FILIPPO, G.
CASE STUDY REGARDING THE USE OF DECISION TABLES IN
COMPUTER APPLICATIONS
DECISION TABLES CONFERENCE
AMSTERDAM, DECEMBER 1971.
A CASE STUDY, TREATING THE USE OF DECISION TABLES IN ALL
STAGES OF SYSTEMS DEVELOPMENT, IS PRESENTED.
- DIA70 DIAL, R.B.
ALGORITHM 394 - DECISION TABLE TRANSLATION
CACM, 13(9)
SEPTEMBER 1970, 571-572.
AN ALGOL 60-ALGORITHM CONVERTING A LIMITED-ENTRY DECISION
TABLE INTO A MACHINE PROCESSABLE CODE MATRIX IS
PRESENTED. THE ALGORITHM'S TECHNIQUE IS DUE TO POLLACK
(POL64A).
FOR A REMARK : SEE MAR728.
- DIX62 DIXON, P.
SPECIAL REPORT, DECISION TABLES SYMPOSIUM
STANDARD EDP REPORTS, 1
DECEMBER 1962, 23:030.100-23:030.601.
- DIX64 DIXON, P.
DECISION TABLES AND THEIR APPLICATION
COMPUTERS AND AUTOMATION, 13(4)
APRIL 1964, 14-19.
THIS ARTICLE DESCRIBES THE FUNDAMENTAL PRINCIPLES OF
DECISION TABLE DESIGN, WITH EXAMPLES. IT INDICATES THE
POWER AND APPLICABILITY OF THE TECHNIQUE TO INCREASE THE
EFFICIENCY OF SYSTEMS ANALYSIS AND PROGRAMMING. IT
INCLUDES DIRECTIONS FOR FURTHER DEVELOPMENT AND AN
ELEVEN-POINT SUMMARY OF THE ADVANTAGES.
- DOB73A DOBOZY, S.
PRAXIS DER ENTSCHEIDUNGSTABELLENTECHNIK (I) :
ZUSAMMENFUEHREN VON N DATEIEN
(G) PRACTICE IN THE DECISION TABLES TECHNIQUE (I) : MIXING
OF N FILES
ONLINE, 11(5)
MAY 1973, 367-370.
BY MEANS OF A PRACTICAL PROBLEM (THE MIXING OF SEVERAL
FILES), THE AUTHOR SHOWS THAT THE FIELD OF APPLICATION
OF THE TRADITIONAL DT-TECHNIQUE CAN BE CONSIDERABLY
ENLARGED.

- DOB73B DOBOZY, S.
 PRAXIS DER ENTSCHEIDUNGSTABELLENTECHNIK (II) :
 AENDERUNGSDIENST
 (G) PRACTICE IN THE DECISION TABLES TECHNIQUE (II) :
 ALTERATION SERVICE
 ONLINE, 11(9)
 SEPTEMBER 1973, 620-622.
- THIS SECOND ARTICLE DEALS WITH THE UPDATING OF FILES
 USING A DECISION TABLE APPROACH. A SITUATION ANALYSIS
 IS PERFORMED BY ATTACHING FIGURES 1,2,3, ETC. TO THE
 CLASSIFICATIONS TO ENABLE COMBINATION TO BE EXPLORED
 AND THE APPROPRIATE DECISION TABLE TO BE DEVISED.
- DOE73C DOBOZY, S.
 PRAXIS DER ENTSCHEIDUNGSTABELLENTECHNIK (III) :
 HIERARCHISCHE ORDNUNG DER BEDINGUNGEN IN EINER BINÄREN
 ENTSCHEIDUNGSTABELLE
 (G) PRACTICE IN THE DECISION TABLES TECHNIQUE (III) :
 HIERARCHICAL ARRANGEMENT OF CONDITIONS IN A BINARY DECISION
 TABLE
 ONLINE, 11(11)
 NOVEMBER 1973, 797-804.
- THIS THIRD ARTICLE DEALS WITH THE BASIC THEORY, I.E.,
 THE DISCOVERY OF THE HIERARCHICAL ORDER OF THE CON-
 DITIONS. IT IS POSSIBLE THEN TO REALISE A MINIMAL
 COMPUTER FLOW CHART WITH THE COMBINATORIAL COMPLETE
 BINARY DECISION TABLE. THE BASIC PRINCIPLE OF THE THEORY
 IS THE RECOGNITION OF THE CONNECTION BETWEEN A BINARY
 DECISION TABLE WITH N-CONDITIONS AND THE N-DIMENSIONAL
 UNIT STRUCTURE.
- DRE71 DRESSLER, H. HAMMELMANN, J. WILHELM, H.A.
 ENTSCHEIDUNGSTABELLEN - EIN LEHRPROGRAMM
 (G) DECISION TABLES - AN INSTRUCTIONAL COURSE
 AIV DIDAKTEAM, AIV DRUCK UND VERLAG DARMSTADT
 DARMSTADT, 1971.
- THIS PROGRAMMED COURSE ENABLES THE READER TO ACQUIRE AN
 INSIGHT INTO THE FUNDAMENTALS OF THE DECISION TABLE
 TECHNIQUE. ALL THE USUAL INTRODUCTORY TOPICS ARE
 TREATED BY MEANS OF EXAMPLES.
- DRE75 DRESSLER, H.
 PROBLEMLOESSEN MIT ENTSCHEIDUNGSTABELLEN
 (G) PROBLEM SOLVING WITH DECISION TABLES
 OLDENBURG, MUENCHEN 1975, 292 PP.
- THIS PROGRAMMED INSTRUCTION TEXT CHIEFLY DEALS WITH
 DECISION TABLE FUNDAMENTALS, MULTIPLE HIT TABLES, THE
 CONSTRUCTION OF DECISION TABLES AND PROGRAMMING BY MEANS
 OF DECISION TABLE PREPROCESSORS. NUMEROUS EXAMPLES ARE
 PROVIDED
- DWY77A DWYER, B.
 HOW TO WRITE DECISION TABLES FOR USE WITH COPE
 SOUTH AUSTRALIAN INSTITUTE OF TECHNOLOGY
 NORTH TERRACE, JANUARY 1977, 68 PP.
- THIS "PROGRAMMED INSTRUCTION" - LIKE TEXT DESCRIBES THE
 USE OF THE COPE PREPROCESSOR ; THE CONSTRUCTION OF
 APPROPRIATE DECISION TABLES IS THOROUGHLY DEALT WITH.
- DWY77B DWYER, B. HUTCHINGS, K.
 FLOWCHAT OPTIMISATION IN COPE, A MULTI-CHOICE DECISION TABLE
 THE AUSTRALIAN COMPUTER JOURNAL, 9(3)
 SEPTEMBER 1977, 92-100.
- DWY78 DWYER, B.
 EXPERIENCE WITH COPE, A MULTI-CHOICE DECISION TABLE PROCESSOR
 PROC. 8TH AUSTRALIAN COMP. SOC. CONFERENCE
 1978, P. 302.
- DWY79 DWYER, B.
 DECISION TABLES
 THE COMPUTER JOURNAL, 22(4)
 NOVEMBER 1979, 381-382.
- THIS LETTER PRESENTS SOME ALTERNATIVE SOLUTIONS TO PROBLEMS
 POSED IN MAE78B. THE USE OF COPE PREPROCESSOR IS.
 ADVOCATED.
- EGL63 EGLER, J.F.
 A PROCEDURE FOR CONVERTING LOGIC TABLE CONDITIONS INTO AN

EFFICIENT SEQUENCE OF TEST INSTRUCTIONS

CACM, 6(8)

SEPTEMBER 1963, 510-514.

THIS ARTICLE DESCRIBES A PROCEDURE FOR CONVERTING DECISION TABLES INTO A COMPUTER PROGRAM, INTENDED TO MINIMIZE PROGRAMMING TIME, STORAGE REQUIREMENTS AND PROCESSING TIME; IT IS ALSO SAID TO DETECT LOGICAL INCONSISTENCIES. RELATIVE RULE FREQUENCIES ARE SHOWN BUT NOT USED. SEE MONTALBANO'S LETTER "EGLER'S PROCEDURE REFUTED" FOR COMMENTS (MONG64).

- ELB71 ELBEN, W. SCHOERNIG, G.
ENTSCHEIDUNSTABELLEN, EINE WIRKSAME UND LEICHT ERLERNBARE PROGRAMMIERSPRACHE
(G) DECISION TABLES AS AN EFFICIENT AND EASY TO LEARN PROGRAMMING LANGUAGE
DATENVERARBEITUNG (AEG-TELEFUNKEN), 4(3)
1971, 96-103.
THE DECISION TABLE TECHNIQUE IS BRIEFLY DESCRIBED AND AN ENGINEERING APPLICATION IS OUTLINED.
- ELB73 ELBEN, W.
ENTSCHEIDUNGSTABELLENTECHNIK
(G) DECISION TABLES TECHNIQUE
DE GRUYTER
BERLIN, 1973, 140 PP.
IN A FIRST PART, THE AUTHOR DESCRIBES THE DECISION TABLE TECHNIQUE AS A COMPUTER-INDEPENDENT TOOL FOR SYSTEMS ANALYSIS AND DOCUMENTATION. THE SECOND PART DEALS WITH THE CONVERSION OF DECISION TABLES TO COMPUTER PROGRAMS.
- ELL67 ELLIS, J.
DECISION TABLES, A USER'S GUIDE
WESTERN ELECTRIC COMPANY
JUNE 1967.
- ELL74 ELLENRIEDER, B.
SYSTEMTECHNIK ZU EINER NORMIERTEN SYSTEMANALYSE,
-ENTWICKLUNG UND -DOKUMENTATION SOWIE MODELLGESTALTUNG
(G) DEVELOPING ABSTRACT MODELS AND CONCRETE SYSTEMS BY THE "SYSTEMS TECHNIQUE"
ANGEWANDTE INFORMATIK, 16(8)
1974, 353-366.
THIS ARTICLE DESCRIBES METHODS OF LOGICAL PROCEEDING AND TECHNIQUES OF GRAPHIC REPRESENTATION FOR DEVELOPING ABSTRACT BUSINESS ORGANIZATION MODELS AND CONCRETE BUSINESS ORGANIZATION SYSTEMS WHICH ARE DESIGNATED AS "SYSTEMS TECHNIQUE". THE METHODS ARE BROUGHT INTO A LOGICAL CONTEXT USING THE PRINCIPLE OF MODULARITY AND ARE MEANT TO SERVE AS AN INSTRUMENT FOR THE SYSTEMS AND MODEL PLANNER. ON THE LOWEST LEVEL OF ABSTRACTION, THE DECISION TABLE TECHNIQUE IS USED.
- ELS73 ELSHOLZ, G.
DATEISTEUERUNG, ENTSCHEIDUNGSTABELLEN UND DOCUMENTATION IM VERBUND
(G) DATA CONTROL, DECISION TABLES AND DOCUMENTATION UNIFIED BURO, 21(1)
JANUARY 1973, 42-46.
A DESCRIPTION IS GIVEN OF THE ARIANE SYSTEM FOR INTEGRATED TREATMENT OF SYSTEM ANALYSIS, PROGRAMMING, CODING AND DOCUMENTATION.
- ENG00 ENGELIEN, M.
BEISPIEL DER ANWENDUNG VON ET ZUR UEBERWACHUNG UND STEUERUNG EINER BRIKETTFABRIK
(G) APPLICATION OF DECISION TABLES IN MANAGING A BRIQUETTE FACTORY
AUSSCHNITT AUS EINEM FORSCHUNGSBERICHT DER SEKTION INFORMATIONSVERARBEITUNG DRESDEN, DATE UNKNOWN.
IN THIS RESEARCH PAPER, THE APPLICATION OF THE DECISION TABLE TECHNIQUE IN MANAGING A BRIQUETTE FACTORY IS EXPLAINED. SOME SERIOUS DIFFICULTIES, HOWEVER, APPEAR WHEN REALIZING THE CONCRETE IMPLEMENTATION.
- ERB72 ERBACH, K.F.
ENTSCHEIDUNGSTABELLEN - THEORIE UND PRAXIS
(G) DECISION TABLES - THEORY AND PRACTICE

COMPUTER PRAXIS, 5(12)
1972, 353-361.

THIS ARTICLE PRESENTS A GENERAL INTRODUCTION TO DECISION TABLES. A METHOD FOR CONSTRUCTING DECISION TABLES IS EXPLAINED AND THE AGENTA-PREPROCESSOR IS INTRODUCED BY MEANS OF A SIMPLE EXAMPLE.

- ERB76 ERBESDOBLER, R. HEINEMANN, J. MEY, P.
ENTSCHEIDUNGSTABELLEN-TECHNIK : GRUNDLAGEN UND ANWENDUNG
VON ENTSCHEIDUNGSTABELLEN
(G) THE DECISION TABLE TECHNIQUE : FUNDAMENTALS AND
APPLICATION OF DECISION TABLES
SPRINGER-VERLAG
BERLIN-HEIDELBERG-NEW YORK, 1976.
THIS BOOK COVERS DECISION TABLE FUNDAMENTALS, ANALYSIS
AND CONVERSION OF DECISION TABLES, DECISION TABLE
PREPROCESSORS (CORTEL, FORTET, ASSMET), CONSTRUCTION AND
APPLICATION FIELDS OF DECISION TABLES. AN ILLUSTRATING
EXAMPLE IS EXTENSIVELY DEALT WITH.
- ESP71 ESPRESTER, A.
GUT GEPLANT MIT ENTSCHEIDUNGSTABELLEN
(G) GOOD PLANNING BY MEANS OF DECISION TABLES
DATA REPORT (SIEMENS), 6(6)
1971, 5-9.
THE SPECIFIC SIEMENS DECISION TABLE FORMAT AND THE
PREPROCESSOR FORTET ARE INTRODUCED IN THIS REPORT.
- EVA60 EVANS, O.Y.
AN ADVANCED ANALYSIS METHOD FOR INTEGRATED ELECTRONIC DATA
PROCESSING
NATIONAL MACHINE ACCOUNTANTS
LONG BEACH, MARCH 1960.
CONDENSED VERSION : IBM GENERAL INFORMATION MANUAL, REF.
F20-8047
1960, 21 PP.
THE ANALYSIS METHOD PRESENTED HERE CAN BEST BE DESCRIBED
AS A SYSTEMATIC METHOD FOR COLLECTING, RECORDING AND
MAINTAINING ALL PERTINENT INFORMATION REGARDING A
COMPLETE DATA PROCESSING SYSTEM. THE METHOD IS
USEFUL IN THAT IT REQUIRES A COMPLETE EXPLANATION OF
THE CHARACTERISTICS AND UTILIZATION OF EACH PIECE OF
DATA INVOLVED IN THE SYSTEM. IN ADDITION, IT INTRODUCES
A TABULAR FORMAT FOR THE DEFINITION OF PROCEDURES.
- EVA61 EVANS, O.Y.
REFERENCE MANUAL FOR DECISION TABLES
IBM CORPORATION
SEPTEMBER 1961.
THIS MANUAL PROVIDES A BASE LANGUAGE THAT INTERESTED
PERSONS CAN USE TO EXPERIMENT WITH DECISION TABLES IN
DOCUMENTING PROBLEM DEFINITIONS. THIS LANGUAGE IS VERY
RIGOROUS IN ORDER THAT IT MAY BE USED AT THE DETAIL
LEVEL OF DOCUMENTATION. PEOPLE EXPERIMENTING AT HIGHER
LEVELS OF MAN TO MAN COMMUNICATION CAN ADJUST THIS
RIGOR TO THEIR NEEDS.
- EVA62 EVANS, O.Y.
A METHOD FOR SYSTEMATIC DOCUMENTATION - KEY TO IMPROVED DATA
PROCESSING ANALYSIS
IN : COMPUTER APPLICATIONS - 1961
MACMILLAN,
NEW YORK, 1962, 14-34.
A PAPER DELIVERED AT THE ARMOUR RESEARCH FOUNDATION
SYMPOSIUM IN 1961.
- EVA67 EVANS, O.Y.
(R) DESCRIPTION OF THE SOLUTION STRUCTURE OF A PROBLEM BY
MEANS OF LOGICAL TABLES
CONTEMPORARY PROGRAMMING. LANGUAGES FOR ECONOMICAL
COMPUTATIONS
SOVET-SKOE RADIO, 1967, 40-62.
- FAI81 FAICT, N.
TRANSFER ARRANGEMENTS OF THE K.B.V.B. : ANALYSIS BY MEANS
OF DECISION TABLES
K. U. LEUVEN, DEPT. OF APPLIED ECONOMICS, THESIS
1981, 154 PP.
THE CONSISTENCY AND COMPLETENESS OF THE INTERNAL REGULA-

TIONS OF THE BELGIAN FOOTBALL ASSOCIATION ARE SYSTEMATICALLY LOOKED FOR BY MEANS OF THE DECISION TABLE TECHNIQUE. EXTENSIVE USE IS MADE OF THE PRODEMO SYSTEM (SEE MAE81A).

- FEI70A FEITSCHER, W.
KOMBINATION DER SYSTEMATISCHEN HEURISTIK MIT DER TECHNIK DER
ENTSCHEIDUNGSTABELLEN
(G) COMBINATION OF SYSTEMATIC HEURISTICS AND THE DECISION
TABLE TECHNIQUE
DIE TECHNIK, 25(10)
1970, 625-627.
THE AUTHOR APPLIES THE DECISION TABLE TECHNIQUE TO
SPECIAL HEURISTIC PROBLEMS ; IN THIS WAY, HE
INVESTIGATES THE USE OF DECISION TABLES IN SOCIOLOGY.
- FEI70B FEITSCHER, W.
EIN BEITRAG DER TECHNIK DER ENTSCHEIDUNGSTABELLEN ZUR
THEORIE UND PRAXIS DER SYSTEMATISCHEN HEURISTIK
(G) A CONTRIBUTION OF THE DECISION TABLES TECHNIQUE TO
THE THEORY AND PRACTICE OF SYSTEMATIC HEURISTICS
WISS. ZEITSCHRIFT DER TECH. HOCHSCHULE ILMENAU, 16(4)
1970, 71-78.
THE AUTHOR MAKES A DISTINCTION BETWEEN PROBLEM-ORIENTED
DECISION TABLES AND ANALYTICAL DECISION TABLES. THE
TECHNIQUE IS DEMONSTRATED WITH THE HELP OF EXAMPLES TAKEN
FROM THE AUTHOR'S EXPERIENCE OF ITS APPLICATION IN PRO-
BLEMS OF DESIGN, LINGUISTICS AND INFORMATION.
- FEI71 FEITSCHER, W.
TECHNIK DER ENTSCHEIDUNGSTABELLEN
(G) DECISION TABLE TECHNIQUE
FORSCHUNGSBERICHT DER TH ILMENAU, SEKTION INER
1971.
- FEI72 FEITSCHER, W. HOFFMANN, P. REICHLING, C.
(G) ANALYSIS, DESIGN AND CONSTRUCTION OF ALGORITHMS FOR
INFORMATION PROCESSES
7TH COLLOQUIUM ON INFORMATION AND DOCUMENTATION
ILMENAU, 17-19 NOVEMBER 1971.
ALSO IN : DOR./INF., (19)
1972, 27-38.
DECISION TABLES ARE USED FOR ANALYSING AND MODELLING
THE INFORMATION PROCESS. ADVANTAGES ARE THAT COMPLEX
PROCESSES (WITH SIMULTANEOUS AND/OR SEQUENTIAL PARTS)
CAN BE DESCRIBED IN A CLEAR AND UNDERSTANDABLE MANNER.
THE ALGORITHMS PROPOSED MUST BE CONSIDERED AS A FIRST
APPROXIMATION, WHICH CAN AND MUST BE IMPROVED DURING
RUNNING.
- FEN69 FENVES, S.J. GAYLORD, E.H. GOEL, S.K.
DECISION TABLE FORMULATION OF THE 1969 AISC SPECIFICATION
CIVIL ENGINEERING STUDIES
STRUCTURAL RESEARCH SERIES NO.347
UNIVERSITY OF ILLINOIS
AUGUST 1969.
- FEN71 FENVES, S.J.
PROCESSING OF DESIGN SPECIFICATIONS USING A RECURSIVE
DECISION-TABLE PROCESSOR
PAPER PREPARED FOR PRESENTATION AT IFIP CONGRESS 71
LJUBLJANE, AUGUST 23-28, 1971, 21 PP.
AN ALGORITHM IS PRESENTED FOR THE INTERPRETIVE EXECUTION
OF HIERARCHIES OF DECISION TABLES. THE CRUX OF THE
ALGORITHM IS THAT A CONDITION IS EVALUATED ONLY WHEN
NEEDED TO PERFORM A TEST. IF THE CONDITION DEPENDS ON
ACTIONS OF LOWER-LEVEL TABLES, THESE ARE RECURSIVELY
EXECUTED UNTIL THE NECESSARY OPERATIONS ARE ALL
PERFORMED. AN EFFICIENT PROCEDURE FOR HANDLING MULTIPLE
PASSES THROUGH THE TABLES IS ALSO PRESENTED.
THE CONCEPT AND ALGORITHM ARE ILLUSTRATED WITH AN
EXAMPLE BASED ON THE 1969 AISC SPECIFICATION FOR DESIGN
OF STRUCTURAL STEEL FOR BUILDINGS.
- FEN72 FENVES, S.J.
REPRESENTATION OF THE COMPUTER-AIDED DESIGN PROCESS BY A
NETWORK OF DECISION TABLES
INT. SYMP. ON COMPUTER-AIDED STRUCTURAL DESIGN
COVENTRY (U.K.), 10-19 JULY 1972, E1031-45.
ALSO IN :

COMPUT. & STRUC., 3(5)
SEPTEMBER 1973, 1099-1107.

IT IS SHOWN THAT DECISION TABLES ARE AN IDEAL TOOL FOR REPRESENTING THE DETAILED DECISIONS AND PROCESSING STEPS OF LARGE-SCALE IMPLEMENTATION OF COMPUTER-AIDED DESIGN OF STRUCTURAL SYSTEMS. THE INTERACTION BETWEEN THE DATA ITEMS AND THE TRANSFORMATIONS INCLUDING DECISION TABLES IS CONCISELY REPRESENTED BY NETWORKS OF INGREDIENCE AND DEPENDENCE. FINALLY, THE INTERATIVE, OPEN-ENDED NATURE OF THE DESIGN PROCESS IS ACCOMMODATED BY ASSIGNING TO EACH DATA ITEM A STATUS INDICATOR, WHICH SPECIFIES WHETHER THE ITEM IS VALID OR VOID. IT IS SHOWN THAT ONLY TWO BASIC RECURSIVE ROUTINES ARE NEEDED TO LINK THE DATA TOGETHER, EITHER TO EVALUATE A GIVEN ITEM BY EVALUATING ALL OF ITS INGREDIENTS OR TO MAKE ALL DEPENDENTS OF A MODIFIED DATA ITEM VOID.

- FEN73 FENVES, S.J.
DECOMPOSITION OF PROVISIONS OF DESIGN SPECIFICATIONS.
IN : DECOMPOSITION OF LARGE-SCALE PROBLEMS (ED. D.M. HIMMELBLAU)
NORTH-HOLLAND PUBLISHING CO.
AMSTERDAM, 1973, 43-56.
BUILDING CODES AND DESIGN SPECIFICATIONS ARE WRITTEN SO AS TO BE INTERPRETED BY EXPERIENCED DESIGN ENGINEERS. IT IS CONVENIENT TO DECOMPOSE THEIR TEXTUAL ORGANIZATION AS WELL AS THEIR COMPUTER IMPLEMENTATION INTO THREE HIERARCHIAL LEVELS. IN THIS PAPER, IT IS ARGUED THAT DECISION TABLES ARE GOOD REPRESENTATIONS OF THE PROVISIONS AT THE DETAILED (THIRD) LEVEL.
- FER67 FERGUS, R.M.
DECISION TABLES - AN APPLICATION ANALYST/PROGRAMMER VIEW
PROC. 1967 INTERNATIONAL DATA PROCESSING CONFERENCE AND BUSINESS EXPOSITION
JUNE 1967, 85-110.
THIS PAPER DESCRIBES WHAT DECISION TABLES ARE AND HOW THEY CAN BE USED IN AN ORGANISATION.
- FER68A FERGUS, R.M.
AN INTRODUCTION TO DECISION TABLES
SYSTEMS AND PROCEDURES JOURNAL, 19(4)
JULY 1968, 24-27.
A DESCRIPTION OF THE CHARACTERISTICS OF DECISION TABLES, AND GUIDELINES AND EXAMPLES OF TABLE CONSTRUCTION ARE GIVEN.
- FER68B FERGUS, R.M.
GOOD DECISION TABLES AND THEIR USES
JOURNAL OF SYSTEMS MANAGEMENT, 9
SEPTEMBER-OCTOBER 1968, 18-21.
- FER69 FERGUS, R.M.
DECISION TABLES : WHAT, WHY AND HOW
PROC. COLLEGE AND UNIVERSITY MACHINE RECORDS CONFERENCE,
UNIVERSITY OF MICHIGAN
1969, 1-20.
ALSO IN : SYSTEM ANALYSIS TECHNIQUES (ED. COUGER & KNAPP)
J. WILEY & SONS
NEW YORK, 1974, 162-179.
THIS ARTICLE PRESENTS A CONCISE INTRODUCTION TO DECISION TABLES AND THEIR APPLICATIONS.
- FIF64 FIFE, R.C.
DECISION TABLES : A TOOL FOR ANALYST AND PROGRAMMER
UNPUBLISHED PAPER IN COOPERATION WITH GEORGE FRY & ASSOCIATES
LOS ANGELES, 1964.
- FIF65 FIFE, R.C.
DECISION TABLES
UNPUBLISHED PAPER PRESENTED TO THE JOINT SPRING CONFERENCE OF THE SYSTEMS AND PROCEDURES ASSOCIATION DETROIT, 1965.
THIS EARLY PAPER WAS PUBLISHED AS A UNIVAC APPLICATION REPORT. IT IS UNDERSTOOD THAT FIF66 IS AN UPDATED PRESENTATION OF THE SAME MATERIAL.
- FIF66 FIFE, R.C.

DECISION TABLES
PROC. UNIVAC USERS ASSOCIATION
PHILADELPHIA, 1966, 15-27.
ALSO IN : ADT, 126-133.

A GENERAL INTRODUCTION ON DECISION TABLES IS CONCLUDED BY A SHORT LIST OF THEIR ADVANTAGES. ONE EXAMPLE SHOWS SOLUTIONS BOTH IN DECISION TABLE AND FLOW CHART FORMAT. ANOTHER ONE SHOWS BOTH LIMITED AND EXTENDED ENTRY VERSIONS OF THE SAME PROCEDURE.

- FIS66 FISHER, D.L.
DATA, DOCUMENTATION AND DECISION TABLES
CACM, 9(1)
JANUARY 1966, 26-31.
IN BUSINESS DATA PROCESSING SYSTEMS, IT IS NECESSARY TO BE ABLE TO DEFINE AND DOCUMENT DATA, FILES, PROGRAMS, AND DECISION RULES IN A WAY THAT ADEQUATELY REPRESENTS BOTH 1) THEIR CHANGING INFORMATION CONTENT, AND 2) THEIR CONTINUOUS INTERACTION. TABULAR DESCRIPTION MAKES THIS POSSIBLE, BEING NOTABLY OBJECTIVE, THOROUGH AND ECONOMIC IN COST AND TIME WHEN SYSTEMS MUST BE ANALYZED AND PROGRAMS PREPARED OR MODIFIED. TO SHOW HOW QUICKLY TABULAR TECHNIQUES MAKE AN UNFAMILIAR SYSTEM MANAGABLE, A DETAILED EXAMPLE AND A SELF-TEST ARE PROVIDED.
- FIS67 FISKE, F.H.
A CASE HISTORY : ACCOUNTING USE OF DECISION TABLES FOR DEFINITION OF COMPUTER PROGRAM LOGIC
ONE-DAY TECHNICAL SYMPOSIUM OF THE LOS ANGELES CHAPTER OF THE ACM (UNPUBLISHED PAPER)
MARCH 1967.
THIS PAPER DESCRIBES NORTH AMERICAN AVIATION'S (NAA) SPACE DIVISION'S USE OF DECISION TABLES AS ADAPTED AND PRECISELY DEVELOPED FOR COMMUNICATING ITS DETAILED REQUIREMENTS TO THE DATA PROCESSING DEPARTMENT WHICH, THROUGH THE USE OF THE DETAB-65 PRE-PROCESSOR AND STANDARD COMPILERS, IS ABLE TO CONVERT THE TABLES INTO PROGRAMS FOR COMPUTERS. CASE HISTORY BACKGROUND IS LAID DOWN; THE PROBLEM SOLVED BY THE PROCEDURE IS DEFINED; VARIOUS ADVANTAGES ARE BRIEFLY OUTLINED; AND A MODEL OF CERTAIN OF THE DECISION TABLES IS PRESENTED.
- FIS75A FISCHBACH, F. GROSS, J. OTT, W.
ENTSCHEIDUNGSTABELLEN ; HILFSMITTEL ZUR ENTSCHEIDUNGSFINDUNG, DOKUMENTATION UND PROGRAMMIERUNG
(G) DECISION TABLES; A TOOL FOR DECISION FINDING, DOCUMENTATION AND PROGRAMMING.
MUELLER, KOELN - BRAUNSFEILD 1975, 139 PP.
THIS BOOK DEALS WITH THE BASIC CONCEPTS OF THE DECISION TABLE TECHNIQUE. APPLICATION AREAS AND THE USE OF PREPROCESSORS ARE OUTLINED. SINGLE HIT AND MULTIPLE HIT TABLES ARE TREATED ON THE SAME LEVEL.
- FIS75B FISCHBACH, F. GROSS, J.
(G) USE OF DECISION TABLES
ONLINE, 13(9)
SEPTEMBER 1975, 576-582.
THE USE OF DECISION TABLES FOR A RATIONAL INFORMATION EXCHANGE BETWEEN MAN AND COMPUTER IS EXPLAINED. THE STEPS LEADING TO THE FORMATION OF THE TABLES I.E. DOCUMENTATION, ACTION DEFINITION ETC. ARE DESCRIBED. FINALLY, THE DISTINCTION BETWEEN |SPECIFIC| AND AND |AMBIGUOUS| DECISION TABLES AND ALSO THE RELATIONSHIP BETWEEN THE LATTER AND THE COMPUTER PROGRAM ARE DISCUSSED.
- FLE64 FLETCHER, H.R.
DECISION TABLE PREPROCESSOR
SEMINAR ON DECISION TABLES
SEPTEMBER 1964.
- FLE67 FLETCHER, H.R.
DATA VALIDATION
SYMPOSIUM ON DECISION TABLES
USASI AD HOC X 3.6.7/X3.4.2D
ATLANTIC CITY, APRIL 1967.
- GAB71A GABRIEL, H.
DIE ANWENDUNG VON ENTSCHEIDUNGSTABELLEN ZUR DARSTELLUNG VON

ARBEITSABLAUFEN IN DER VERWALTUNG
(G) THE APPLICATION OF DECISION TABLES IN MAN-POWER PLANNING
ORGANISATION, 5(2)

1971, 9-16.

THE AUTHOR OF THIS ARTICLE PROPOSES THE APPLICATION
OF DECISION TABLES IN MAN-POWER PLANNING. THE CONSTRUC-
TION, THE ADVANTAGES AND THE DISADVANTAGES CONNECTED
WITH THE USE OF DECISION TABLES ARE ILLUSTRATED BY
MEANS OF SEVERAL EXAMPLES.

- GAB71B GABRIEL, H.
ENTSCHEIDUNGSTABELLEN - EIN NEUES HILFSMITTEL ZUR
ERARBEITUNG VON EDVA-PROJEKTEN
(G) DECISION TABLES - A NEW TOOL FOR EDP-PROJECTS
RTDV, 8(5)
1971, 30-33.
THE AUTHOR PRESENTS A GENERAL INTRODUCTION TO DECISION
TABLES BY MEANS OF SIMPLE EXAMPLES. DECISION TABLES ARE
COMPARED WITH FLOWCHARTS AND COMBINATIONS OF DECISION
TABLES AND PAP-FLOWCHARTING ARE DISCUSSED.
- GAL73 GALAN, I.
LES TABLES DE DECISION ET LEURS APPLICATIONS
(F) DECISION TABLES AND THEIR APPLICATIONS
DUNOD
PARIS, 1973, 121 PP.
A SHORT INTRODUCTION TO THE USE OF DECISION TABLES IN
SYSTEMS ANALYSIS AND PROGRAMMING. MANY PRACTICAL EXAMPLES
ILLUSTRATE THE TRAIN OF THOUGHT.
- GAN69 GANAPATHY, S.
INFORMATION THEORY APPLIED TO DECISION TABLES
M. TECH TH., INDIAN INSTITUTE OF TECHNOLOGY
KAMPUR, 1969.
- GAN73 GANAPATHY, S. RAJARAMAN, V.
INFORMATION THEORY APPLIED TO THE CONVERSION OF DECISION
TABLES TO COMPUTER PROGRAMS
CACM, 16(9)
SEPTEMBER 1973, 532-539.
USING IDEAS FROM INFORMATION THEORY, THIS PAPER DEVELOPS
A HEURISTIC ALGORITHM THAT CONVERTS A LIMITED ENTRY
DECISION TABLE TO A TREE STRUCTURED COMPUTER PROGRAM
WITH NEAR MINIMUM AVERAGE PROCESSING TIME. THE METHOD IS
APPLICABLE TO ANY LIMITED ENTRY DECISION TABLE AND DOES
NOT REQUIRE THAT ACTIONS HAVE SINGLE RULES OR THAT THE
COST OF TESTING CONDITIONS BE EQUAL. IT IS THUS MORE
GENERAL THAN THE PREVIOUSLY PUBLISHED HEURISTIC
ALGORITHMS. COMPARED TO THE OPTIMAL ALGORITHM OF REINWALD
AND SOLAND, THIS ALGORITHM IS EASY TO CODE AND TAKES A
MUCH SMALLER TRANSLATION TIME; IT IS THUS FELT THAT IT
IS MORE USEFUL IN PRACTICE. THE ALGORITHM IS WELL SUITED
FOR MANUAL CONVERSION OF DECISION TABLES TO FLOWCHARTS.
- GEI00 GEIGER, D. MOCKLER, A.
ENTSCHEIDUNGSTABELLENTECHNIK FUER ORGANISATION UND
PROGRAMMIERUNG
(G) THE DECISION TABLE TECHNIQUE IN THE ORGANIZATION AND IN
PROGRAMMING
SIEMENS-DATA-PRAXIS, D 14/438.
DATE UNKNOWN.
- GEN61A GENERAL ELECTRIC
GE-225 TABSOL
REFERENCE MANUAL
APPLICATION MANUAL CPB-147B
PHOENIX, 1961.
ILLUSTRATIONS OF POTENTIAL APPLICATIONS OF THE TABSOL
LANGUAGE IN THE AREAS OF MANUFACTURING, ENGINEERING,
AND FINANCE ARE DESCRIBED IN DETAIL. TABSOL, WHICH
STANDS FOR TABULAR SYSTEMS ORIENTED LANGUAGE, IS
BASICALLY A STRUCTURING TECHNIQUE USED TO SYSTEMATICALLY
DESCRIBE THE STEP BY STEP DECISION LOGIC IN THE PROCESS
OF SOLVING A PROBLEM. THE BASIC ADVANTAGE OF TABSOL IS
PROBABLY ONE OF THE MOST EASILY LEARNED AND UNDERSTOOD
AND CAN BE APPLIED TO MANY ANALYTICAL SITUATIONS.
- GEN61B GENERAL ELECTRIC
THE DECISION LANGUAGE OF THE GE-225

GENERAL ELECTRIC COMPUTER DEPARTMENT
PHOENIX, 1961.

- GIL70 GILDERSLEEVE, T.R.
DECISION TABLES AND THEIR PRACTICAL APPLICATION IN DATA
PROCESSING
PRENTICE-HALL
ENGLEWOOD-CLIFFS, 1970, 206 PP.
THIS IS A SELF-TEACHING BOOK TREATING THE CONSTRUCTION
AND THE USE OF DECISION TABLES IN ANALYSIS AND
PROGRAMMING. AS A RESULT, IT HAS TO BE READ ENTIRELY
FROM THE BEGINNING TO THE END. ALTHOUGH A CONCISE INDEX
IS ADDED, ITS USE AS A REFERENCE-BOOK IS VERY LIMITED.
- GIL75 GILDERSLEEVE, T.R.
THE DARK SIDE OF STRUCTURED PROGRAMMING
DATAMATION, 21(11)
NOVEMBER 1975, 178
DECISION TABLES ARE SAID TO BE SUPERIOR TO MORE CLASSICAL
STRUCTURED PROGRAMMING TECHNIQUES.
- GIN65 GINBERG, A.S. MARKOWITZ, H.M. OLDFATHER, P.M.
PROGRAMMING BY QUESTIONNAIRE
RAND CORPORATION, RM-4460-PR
SANTA MONICA, 1965.
PROGRAMMING BY QUESTIONNAIRE, A COMBINATION OF DECISION
TABLE PROGRAMMING AND GENERAL PURPOSE PROGRAMMING, IS
PROPOSED. A THOROUGH DISCUSSION OF THE DECISION TABLE
TECHNOLOGY AS WELL AS AN ILLUSTRATIVE EXAMPLE (JOB SHOP
SIMULATOR) ARE PROVIDED.
- GLA62 GLANS, T.B. GRAD, B.
7080 DECISION TABLE SYSTEM PRELIMINARY MANUAL
IBM TECHNICAL REPORT 2D1
APRIL 1962.
- GLA63 GLANS, T.B.
PROGRESS IN DECISION TABLE APPLICATION
IDEAS FOR MANAGEMENT (PAPERS PRESENTED AT THE 1963 INTER-
NATIONAL SYSTEMS MEETING)
1963, 183-190.
ALSO IN : ADT, 139-144.
THIS PAPER CITES SOME SUCCESSFULL EXPERIMENTS AND
SUGGESTS A COURSE OF ACTION FOR FURTHER EXPLORATION
OF THE ADVANTAGES OFFERED BY DECISION TABLES IN SYSTEMS
ANALYSIS, DOCUMENTATION AND PROGRAMMING.
- GOE71 GOEL, S.K. FENVES, S.J.
COMPUTER PROCESSING OF DESIGN SPECIFICATIONS
JOURNAL STRUCT. DIV., 97(ST1)
JANUARY 1971, 463-479
GENERALIZATION OF DESIGN PROCESS HAS BEEN HAMPERED BY
THE FACT THAT DESIGN CONSTRAINTS, AS PRESENTED IN THE
VARIOUS CODES AND SPECIFICATIONS ARE WRITTEN WITH HAND
COMPUTATION IN MIND AND ARE NOT READILY ADAPTABLE TO
COMPUTER PROCESSING. DECISION TABLE FORMAT FOR
SPECIFICATIONS HAS BEEN SUGGESTED TO OVERCOME THIS
PROBLEM. A GENERAL ALGORITHM FOR CONSTRAINT CHECKING,
USING THE DESIGN CONSTRAINTS IN THE FORM OF DECISION
TABLES, IS PRESENTED. THE ALGORITHM MINIMIZES THE VOLUME
OF DATA REQUIRED AND OPTIMIZES ON EXECUTION TIME BY
(1) CONDITIONALLY EXECUTING SUBSEQUENT TABLES IN THE
HIERARCHY ONLY WHEN THE DATA GENERATED BY SUCH TABLES IS
ACTUALLY REQUIRED ; AND (2) GROUPING DATA IN THE FORM OF
INGREDIENCE AND DEPENDENCE ARRAYS. CONDITIONAL EXECUTION
IS IMPLEMENTED BY RECURSIVE USE OF THE SAME PROGRAM
SEGMENT. PROVISIONS ARE INCLUDED FOR RECYCLING OF THE
CONSTRAINT CHECKING FOR TESTING ALTERNATE DESIGNS OR
ALTERNATE DESIGN APPROACHES BY CHANGING ONLY A FEW
ELEMENTS OF DATA, OR BOTH. EXAMPLES HAVE BEEN SOLVED
USING THE 1969 AISC SPECIFICATION.
- GOL66 GOLD, V. BISMUTH, C.
LES TABLES DE DECISION APPLIQUEES A LA PAYE
(F) DECISION TABLES APPLIED TO PAYROLL
AUTOMATISME, 11(10)
OCTOBER 1966, 560-565.
AFTER INTRODUCING DECISION TABLES, THE AUTHORS GO ON TO
DISCUSS THE ADVANTAGES GAINED BY PROGRAMMING USING THIS

TECHNIQUE. THEY BRIEFLY DESCRIBE THE SNETAB DECISION TABLE FORM. IN A SECOND PART, THEY DEMONSTRATE THE EASE OF UPDATING A RAPIDLY CHANGING PROGRAM BY MEANS OF AN EXAMPLE TAKEN FROM A PAYROLL SYSTEM.

G0075 GOODENOUGH, J.B. GERHART, S.L.
TOWARD A THEORY OF TEST DATA SELECTION
PROC. INT. CONFERENCE ON RELIABLE SOFTWARE
LOS ANGELES, 21-23 APRIL 1975
ALSO IN : SIGPLAN NOTICES, 10(6)
493-510

THIS PAPER EXAMINES THE THEORETICAL AND PRACTICAL ROLE OF TESTING IN SOFTWARE DEVELOPMENT. THE AUTHORS PROVE A FUNDAMENTAL THEOREM SHOWING THAT PROPERLY STRUCTURED TESTS ARE CAPABLE OF DEMONSTRATING THE ABSENCE OF ERRORS IN A PROGRAM. THE THEOREM'S PROOF HINGES ON THE DEFINITION OF TEST RELIABILITY AND VALIDITY, BUT ITS PRACTICAL UTILITY HINGES ON BEING ABLE TO SHOW WHEN A TEST IS ACTUALLY RELIABLE. THEY EXPLAIN WHAT MAKES TESTS UNRELIABLE (FOR EXAMPLE, THEY SHOW BY EXAMPLE WHY TESTING ALL PROGRAM STATEMENTS, PREDICATES, OR PATHS IS NOT USUALLY SUFFICIENT TO INSURE TEST RELIABILITY). AND THEY OUTLINE A POSSIBLE APPROACH TO DEVELOPING RELIABLE TESTS. THEY ALSO SHOW HOW THE ANALYSIS REQUIRED TO DEFINE RELIABLE TESTS CAN HELP IN CHECKING A PROGRAM'S DESIGN AND SPECIFICATION AS WELL AS IN PREVENTING AND DETECTING IMPLEMENTATION ERRORS.

THE BASIC DECISION TABLE SCHEME IS THOROUGHLY USED.

GOU64 GOULD, K.H.
AN INTRODUCTION TO DECISION TABLES
UNPUBLISHED PAPER
1964.

GRA61A GRAD, B.
TABULAR FORM IN DECISION LOGIC
DATAMATION, 7(7)
JULY 1961, 22-26.

THIS REPORT HAS THE DUAL PURPOSE OF SKETCHING THE HISTORICAL BACKGROUND ON THE DEVELOPMENT OF TABULAR FORMS AND INDICATING THEIR POSSIBLE ADVANTAGES.

GRA61B GRAD, B.
TABLES SIGNAL BETTER COMMUNICATIONS
PAPER PRESENTED AT IBM GUIDE XII INTERNATIONAL MEETING
JUNE 1961.

GRA62A GRAD, B. GLANS, T.B.
TABULAR DESCRIPTIVE LANGUAGE
IBM TECHNICAL REPORT 245
JANUARY 1962.

GRA62B GRAD, B.
DECISION TABLES IN SYSTEMS DESIGN
DIGEST OF TECHNICAL PAPERS, ACM NATIONAL CONFERENCE
SYRACUSE (N.Y.), 4-7 SEPTEMBER 1962, 76-77.
THE ADVANTAGES OF USING DECISION TABLES IN ANALYSIS,
DOCUMENTATION AND PROGRAMMING ARE BRIEFLY EXPLAINED.

GRA62C GRAD, B.
STRUCTURE AND CONCEPT OF DECISION TABLES
PROC. DECISION TABLES SYMPOSIUM
20-21 SEPTEMBER 1962, 19-28.
IN THIS PAPER, IT IS ARGUED THAT DECISION TABLES PROVIDE A MEANS OF PRESENTING COMPLEX DECISION LOGIC IN A WAY THAT IS RELATIVELY EASY TO PREPARE AND UNDERSTAND. A DECISION TABLE SHOWS THE SPECIFIC ALTERNATIVE COURSES OF ACTION TO BE TAKEN UNDER VARIOUS COMBINATIONS OF CONDITIONS. THIS PERMITS AN ANALYST OR PROGRAMMER TO CONCISELY AND COMPLETELY RECORD LOGICAL DECISION RULES FOR ANALYSIS, DOCUMENTATION AND PROGRAMMING.

GRA62D GRAD, B.
USING DECISION TABLES FOR PRODUCT DESIGN ENGINEERING
AIEE WINTER GENERAL MEETING
NEW YORK, FEBRUARY 1962.

GRA65 GRAD, B.
ENGINEERING DATA PROCESSING USING DECISION TABLES

PROC. 1965 INTERNATIONAL DATA PROCESSING CONFERENCE
1965, 467-475.

ALSO IN : ADT, 32-38.

AFTER A GENERAL INTRODUCTION TO THE CONCEPT AND FORM OF DECISION TABLES, THIS ARTICLE GOES ON TO CONSIDER THEIR USE IN ENGINEERING DESIGN AND MANUFACTURING. ILLUSTRATIONS ARE TAKEN FROM ARMATURE WINDING AND SHEET METAL ROUTING. CONVERSION OF DECISION TABLES TO PROGRAMS IS DISCUSSED AND FOUR APPROACHES ARE BRIEFLY DESCRIBED : MANUAL, SEMI-AUTOMATIC, INTERPRETATION AND AUTOMATIC CONVERSION.

- GRE73 GREVE, R.H.
ENTSCHEIDUNGSTABELLEN ALS SYSTEMATISCHES HILFSMITTEL IN DER ADV
(G) DECISION TABLES AS SYSTEMATIC AIDS IN AUTOMATIC DATA PROCESSING
BURO, 21(1)
JANUARY 1973, 22-26.
THE ROLE OF THE DECISION TABLE TECHNIQUE AS AN EFFICIENT METHOD FOR THE DESCRIPTION OF A SYSTEM, APART FROM THE COMPUTER, IS DISCUSSED AND THE ADVANTAGES ARE EXPLAINED WHICH THE TECHNIQUE OFFERS IN SYSTEMS ANALYSIS AS COMPARED WITH COLLOQUIAL EXCHANGE AS A COMMUNICATION MEDIUM.
- GRI66 GRINDLEY, C.B.B.
SYSTEMATICS - A NON-PROGRAMMING LANGUAGE FOR DESIGNING AND SPECIFYING COMMERCIAL SYSTEMS FOR COMPUTERS
THE COMPUTER JOURNAL, 9(2)
AUGUST 1966, 124-128.
THE PAPER DESCRIBES A NEW LANGUAGE WHICH PROVIDES TOOLS AND TECHNIQUES FOR THE SYSTEMS ANALYST. THE MAIN FEATURE OF THE LANGUAGE IS THAT, LIKE MATHEMATICS, IT IS AN AID TO PROBLEM SOLVING, BUT, AGAIN LIKE MATHEMATICS, IT IS NOT PROVIDED WITH A COMPILER. A SHORT EXAMPLE IS GIVEN SHOWING HOW THE ANALYST IS ENABLED TO CONSTRUCT A MODEL OF THE INFORMATION SYSTEM HE IS DESIGNING INDEPENDENTLY OF ANY COMPUTER CONSIDERATIONS.
- GRI68A GRINDLEY, C.B.B.
THE USE OF DECISION TABLES WITHIN SYSTEMATICS
THE COMPUTER JOURNAL, 11(2)
AUGUST 1968, 128-133.
SYSTEMATICS EMBRACES A RANGE OF TECHNIQUES FOR DESIGNING AND DESCRIBING INFORMATION SYSTEMS. ONE OF THESE TECHNIQUES, USED TO CONSTRUCT A MODEL OF AN INFORMATION SYSTEM IS SHOWN TO USE A SPECIAL CASE OF DECISION TABLES. THE FORM OF THIS TABLE AND CERTAIN ADVANTAGES GAINED FROM USING IT ARE DESCRIBED. THE PAPER ENDS WITH A SYNTACTIC DESCRIPTION OF THE SYSTEMATICS LANGUAGE, TO A LARGE EXTENT IN BACKUS-NAUR FORM.
- GRI68B GRINDLEY, C.B.B. STEVENS, W.G.R.
PRINCIPLES OF THE IDENTIFICATION OF INFORMATION
REPRINTED FROM VOL. I OF WORKING PAPERS FOR FILE 68
NOVEMBER 1968, 243-260.
THIS PAPER PRESENTS SOME OF THE BASIC PRINCIPLES OF SYSTEMATICS AS A THEORY OF INFORMATION STRUCTURE (PART I) AND OF INFORMATION SELECTION (PART II), TO AN IFIP ADMINISTRATIVE DATA PROCESSING GROUP CONFERENCE ON INFORMATION FILE STRUCTURE (SEE ALSO : GRI66 AND GRI68A).
- HAB71 HABERMEHL, J.H.
TABELLARISCHE ORGANISATION VEREINFACHT PROGRAMMIERUNG
(G) TABLE METHOD AND PROGRAMMING
BURO, 19(7)
JULY 1971, 639-640.
TABLE METHODS SIMPLIFY PROGRAMMING. TWO PARTICULARLY USEFUL METHODS, AMONG WHICH THE DECISION TABLE TECHNIQUE, ARE DESCRIBED.
- HAC70 HACCOUR, R.
EEN METHODE VOOR HET CONSTRUEREN VAN BESLISSINGSTABELLEN
(D) A METHOD FOR CONSTRUCTING DECISION TABLES
KUL, INSTITUUT VOOR TOEGEPASTE ECONOMISCHE WETENSCHAPPEN
LEUVEN, 1970, 96 PP.
TWO DIFFERENT METHODS FOR CONSTRUCTING DECISION TABLES, THE ANALYTICAL APPROACH OF POLLACK (POL63C) AND THE SYNTHETIC ONE OF VERHELST (VER69), ARE COMPARED. A

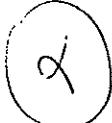
FAIRLY RUDIMENTARY COMPUTER PROGRAM, BASED ON THE
METHOD OF VERHELST, IS GIVEN.

- HAI66 HAIGHT, R.
DECISION TABLES
GUIDE
NOVEMBER 1966.
- HAK65 HAKANSSON, G.
BESLUTSTABELLER
(S) DECISION TABLES
AUTOMATISK DATABEHANDLING, (1)
NOVEMBER 1965.
- HAR67 HARRIS, E.R.
ALP DOS DECISION TABLE MACROS
IBM CORPORATION, CONTRIBUTED PROGRAM LIBRARY,
NO. 360D-03.7.017
1967.
THE USE OF LIMITED-ENTRY DECISION TABLES IN ASSEMBLER
PROGRAMS IS ENABLED BY THE IBM MACROS SETIT AND TABLE.
BOTH FACILITIES ARE DESCRIBED.
- HAR71 HARRISON, W.J.
PRACTICALLY COMPLETE DECISION TABLES : A RANGE APPROACH
SIGPLAN NOTICES, 6(8)
SEPTEMBER 1971, 89-93.
IN THIS PAPER, A NEW APPROACH TO COMPLETENESS OF
DECISION TABLES IS TAKEN. THE MAJOR DEPARTURE FROM THE
TRADITIONAL VIEWPOINT IS THE PREMISE THAT A CONDITIONAL
VARIABLE CAN BE DEFINED AS A ONE-DIMENSIONAL SCALAR
VARYING FROM ONE EXTREME TO THE OPPOSITE. THIS SINGLE
EXTENT CAN BE SEGMENTED INTO A SERIES OF SMALLER RANGES.
THE VERY IMPORTANT CASE OF THE EQUALITY CAN THEN BE
DEFINED AS A SUBRANGE OF LENGTH ZERO NESTED BETWEEN OTHER
LONGER SUBRANGES. THIS VIEWPOINT ALLOWS MULTIPLE
EQUALITIES, RANGES OF VALUES, AND EVEN OVERLAPPING RANGES
OF VALUES ON A SINGLE CONDITIONAL VARIABLE TO BE FORMALLY
HANDLED FOR PRACTICAL COMPLETENESS CHECKING.
- HAS71 HASLETT, J.W.
DECISION TABLE FOR ENGAGING A CONSULTANT
JOURNAL OF SYSTEMS MANAGEMENT, 22(7)
1971, 12-19.
- HAU68 HAUCK, R.F.
THE ORTHOGONALITY PRINCIPLE OF DECISION TABLES AND ITS AP-
PLICATION TO COMPUTER IMPLEMENTATION
UNITED STATES STEEL REPORT OF STEEL TAB COMPILER METHODS,
UNITED STATES STEEL CORPORATION,
OCTOBER 1968.
THE AUTHOR OF THIS REPORT PRESENTS A CONVERSION ALGORITHM
BASED ON THE ORTHOGONAL LOGICAL RELATIONSHIP WHICH EXISTS
BETWEEN STATEMENTS AND RULES IN A DECISION TABLE. THE
DEVELOPMENT OF THIS ALGORITHM IS BASED ON THE CONCEPT OF
RULE INDEPENDENCE.
- HAU72 HAUG, R.
(G) STANDARDISED PROGRAMMING, OR THE BUILDING BLOCK
PRINCIPLE FOR SOFTWARE
ZEITSCHRIFT FUER DATENVERARBEITUNG, 10(2)
MARCH 1972, 97-98
THE PRESENT POSITION OF THE SOFTWARE FIELD REVEALS A
CLEAR POLARIZATION BETWEEN CONVENTIONAL SOFTWARE
SOLUTIONS, REFERRING TO DEFINITE APPLICATIONS AND
PROCESSES, AND FLEXIBLE PROGRAM SYSTEMS WHICH WORK WITH
PROGRAM GENERATORS. IN THE LATTER CONNECTION WITH
ADOPTION OF MODULAR CONSTRUCTION FOR PROGRAM DEVELOPMENT,
WITH AUTOMATIC PROGRAM GENERATION IN ASSOCIATION WITH
THE USE OF DECISION TABLES IS SUGGESTED.
- HAW62 HAWES, M.K.
THE NEED FOR PRECISE PROBLEM DEFINITION
PROC. DECISION TABLES SYMPOSIUM
20-21 SEPTEMBER 1962, 13-18.
THE NEED FOR PRECISE PROBLEM DEFINITION IS DISCUSSED. IT
IS INDICATED THAT OVER 65 % OF THE COST ASSOCIATED WITH
PROGRAMMING DATA PROCESSING PROBLEMS CAN BE ATTRIBUTED
TO THIS NEED. THE USE OF DECISION TABLES IS PROPAGATED.

- HAW65 HAWES, M.K.
THE USE OF DECISION TABLES FOR PROBLEM SPECIFICATION
PROC. UNIVAC USERS ASSOCIATION
APRIL 1965, 55-61.
THIS REPRINT GIVES A GENERAL INTRODUCTION TO DECISION TABLES, ILLUSTRATING IT WITH A MIXED ENTRY TABLE USED IN THE ANALYSIS OF A RETRIEVAL REQUEST. THE AUTHOR GOES ON TO MENTION AND COMMENT ON EARLY WORK AT GENERAL ELECTRIC AND SUTHERLAND AND THE DEVELOPMENT OF DETAB PREPROCESSORS AND ENDS WITH A 14-POINT LIST OF ADVANTAGES.
- HAW67 HAWES, M.K.
DATA REFERENCING IN A MAN-TO-MAN COMMUNICATION ENVIRONMENT
SYMPORIUM ON DECISION TABLES
USASI AD HOC X3.6.7/X3.4.2D
ATLANTIC CITY, APRIL 1967.
- HER75 HERNANDEZ RUIZ, J.J.
(E) DECISION TABLES AND THEIR IMPLEMENTATION IN A CID-201A COMPUTER
CONTROL CIBERNETICA Y AUTOMATIZATION (CUBA9, 9(3)
JULY-SEPTEMBER 1975, 32-34.
THE USE OF DECISION TABLES IN CONJUNCTION WITH A CID-201A COMPUTER IS DEALT WITH. THE GENERAL CONCEPT OF A DECISION TABLE IS FIRST GIVEN BRIEFLY. ONE POSSIBLE APPLICATION IS FOR PROCESS CONTROL IN THE SUGAR INDUSTRY. A TYPICAL TABLE IS GIVEN WHICH COVERS A FILTER OPERATION. A FURTHER SECTION COVERS A DECISON TABLE PROGRAM AND THE DATA WHICH CAN BE FED INTO IT.
- HEI68 HEINASMAKI, P.
BESLUTSTABELLerna SOM HJALPMEDEL I SYSTEMARBETET
(S) DECISION TABLES AS TOOLS IN SYSTEMS ANALYSIS AND DESIGN
NORD-DATA-68
HELSINGFORS, OTNAS, 6.-8.6.1968, PART II, 544.
- HEN72 HENTSCHLE, B.
ENTSCHEIDUNGSTABELLEN - HILFSMITTEL FUR DEN TECHNOLOGISCHEN WERKSTAETTENENTWURF
(G) DECISION TABLES - A TOOL FOR THE TECHNOLOGICAL DESIGN OF WORKSHOPS
FERTIGUNGSTECHNIK UND BETRIEB, 22(3)
1972, 135-140.
- HIC68 HICKS, H.T. POLLACK, S.L.
DECISION TABLES FOR COMPUTER SYSTEM DESIGN AND PROGRAMMING
INFORMATION MANAGEMENT INC.
SAN FRANCISCO, MARCH 1968.
- HOE73 HOELZ, L.
DETAB/GT - EIN ENTSCHEIDUNGSTABELLENPROZESSOR
(G) DETAB/GT - A DECISION TABLE PREPROCESSOR
COMPUTER-PRAXIS, 6(2)
1973, 51-55.
THE AUTHOR SUCCESSIVELY TREATS EXPERIMENTAL PREPROCESSORS (DETAB-X, DETAB/65, DETAB/66), THE DETAB/GT PREPROCESSOR, DETAB/GT AS A SYSTEM, CONVERTING DECISION TABLES BY MEANS OF DETAB/GT AND NORMALIZED PROGRAM DEVELOPMENT. AN EXAMPLE IS ADDED.
- HOF73 HOFFMANN, K.
AUTOMATISCHE UMWANDLER FUR ENTSCHEIDUNGSTABELLEN
(G) AUTOMATIC TRANSLATORS FOR DECISION TABLES
BURO, 21(1)
JANUARY 1973, 28-35.
THE AUTOMATIC CONVERSION OF DECISION TABLES IS EXPLAINED. TYPICAL DIFFERENCES BETWEEN VARIOUS PROCESSORS COMMERCIALLY AVAILABLE ARE DESCRIBED.
- HOG75 HOGGER, E.I.
THE USE OF DECISION TABLES IN CONTROL SOFTWARE
TREND IN ON-LINE COMPUTER CONTROL SYSTEMS
SHEFFIELD, 21-24 APRIL 1975, 238-245.
RELIABILITY IS AN ESSENTIAL PROPERTY OF SOFTWARE FOR THE ON-LINE COMPUTER CONTROL OF POWER STATIONS. HOWEVER, TRADITIONAL PROGRAMMING METHODS AREN'T WELL ADAPTED TO CHECK THE CORRECTNESS OF THE PROGRAM LOGIC. IN THIS PAPER

AN ALTERNATIVE APPROACH USING DECISION TABLES IS PROPOSED. IT ENSURES AUTOMATIC AND COMPREHENSIVE CHECKING OF THE BUILT-IN PROGRAM LOGIC.

- HOL00A HOLSTEIN, D. SMITH, C.F.
AUTOMATING DESIGN PROCEDURES WITH DECISION-MAKING SYSTEMS
DATA SYSTEMS DESIGN
DATE UNKNOWN.
- HOL00B HOLSTEIN, D.
THE CONCEPT AND USE OF DECISION TABLES IN ENGINEERING
APPLICATIONS
IBM TIE, 6 309-0708
DATE UNKNOWN.
- HOL62A HOLSTEIN, D.
DECISION TABLES, A TECHNIQUE FOR MINIMIZING ROUTINE
REPETITIVE DESIGN
MACHINE DESIGN, 34(18)
AUGUST 1962, 76-79.
THIS IS A BRIEF INTRODUCTION TO THE USE OF DECISION TABLES FOR ENGINEERING DESIGN. THE MAIN ILLUSTRATIONS DEAL WITH THE WINDING OF ARMATURE COILS.
- HOL62B HOLSTEIN, D.
USE OF DECISION TABLES IN SYSTEM DESIGN
AMERICAN MANAGEMENT ASSOCIATION, 8TH ANNUAL EDP CONFERENCE
FEBRUARY 1962.
- HOL71 HOLLEY, J.K.
UNTERSUCHUNGSERGEBNISSE UBER DEN EINSATZ DER
ENTSCHEIDUNGSTABELLENTECHNIK ALS ORGANISATIONSMITTEL
(G) RESULTS OF RESEARCH ON THE USE OF THE DECISION TABLE
TECHNIQUE AS A TOOL FOR ORGANIZATION
UNPUBLISHED PAPER (C-E-I-R)
1971.
- HON69 HONEYWELL
INTRODUCTION TO DECISION TABLES - A PROGRAMMED TEXT
HONEYWELL MANUAL D42
OCTOBER 1969.
THIS IS A PROGRAMMED TEXT MANUAL COVERING BASICS,
BUILDING AND CONDENSING A LIMITED ENTRY TABLE AND
LINKING TABLES.
- HON70 HONEYWELL
DECISION TABLE PREPROCESSOR (DETAB I)
HONEYWELL EDP UK SOFTWARE BULLETIN B 70013/EXT
3RD APRIL 1970.
THIS BULLETIN DESCRIBES HARDWARE REQUIREMENTS, DETAILS OF INPUT AND THE STRUCTURE OF THE GENERATED SOURCE PROGRAM ; IT DISCUSSES PROGRAMMING CONSIDERATIONS AND SETS OUT THE SEQUENCE OF INPUT CARDS AND THE SET-UP FOR OPERATION OF THE PREPROCESSOR. AN APPENDIX DESCRIBES SEGMENT 99, IN EASYCODER, WHICH PROVIDES THE INTERFACE BETWEEN DETAB I AND COBOL I.
- HON71 HONEYWELL
DECISION TABLES
STUDY GUIDE
1971.
THIS STUDENT GUIDE IS INTENDED TO COMPLEMENT THE DECISION TABLES VIDEOTAPES DISTRIBUTED BY HONEYWELL.
- HSR66 HOSKYNS SYSTEMS RESEARCH
TAP DECISION TABLE PROCESSOR
UNDATED BUT APPARENTLY NOVEMBER 1966.
THIS SHORT REPORT OUTLINES THE GENESIS AND DEVELOPMENT OF TAP AND GOES ON TO DESCRIBE THE FORMAT USED AND THE LOGIC FEATURES OF THE PROCESSOR. IT INCLUDES A COPY OF JACKSON'S TWO ARTICLES JAC68A AND JAC68B.
- HSR70A HOSKYNS SYSTEMS RESEARCH
TABLEMASTER SPECIFICATION MANUAL
1970.
- HSR70B HOSKYNS SYSTEMS RESEARCH
TABLEMASTER COBOL/360 SPECIFICATION
1970.

- HUE75 HUET, A.
(F) AN ALGORITHM FOR CONVERTING LIMITED-ENTRY DECISION TABLES
REVUE HF (BELGIUM). 9(12)
1975, 356-360.
- THE ALGORITHM PRESENTED IN THIS PAPER HAS BEEN REALIZED IN A SIMPLIFIED VERSION IN THE FORM OF MACRO-INSTRUCTIONS FOR THE IBM 360/370 ASSEMBLER LANGUAGE.
- HUG68 HUGHES, M.L. SHANK, R.M. STEIN, E.S.
DECISION TABLES
MANAGEMENT DEVELOPMENT INSTITUTE
WAYNE, 1968, 176 PP.

THIS BOOK COVERS TABULAR TECHNIQUES, LIMITED-ENTRY AND EXTENDED-ENTRY DECISION TABLES, TABLE LINKAGE, "MECHANICS" (CHECKING, TABLE SIZE, COMPLETENESS, REDUNDANCY, ETC.) AND SOME APPLICATIONS. IT ALSO INCLUDES A CHAPTER ON DECISION TABLE SOFTWARE, WITH SHORT WRITE-UPS ON DETAB/65, PET, TABSOL AND DETRAN. THE TOPICS ARE MAINLY TREATED BY MEANS OF THE DEVELOPMENT OF EXAMPLES, STEP BY STEP.
- HUG72 HUGHES, P.
AN APPLICATION OF DECISION LOGIC TABLES IN AN AIR FORCE COMBAT ENGAGEMENT SIMULATION MODEL
BULL. OPER. RES. SOC. AM., ORSA 41ST ANNUAL MEETING,
VOLUME 20, SUPPL. 1.

NEW ORLEANS, 26-28 APRIL 1972, B/159
THIS PAPER DESCRIBES A TECHNIQUE FOR SIMULATING THE TACTICAL DECISION MAKING PROCESS OF AIRCRAFT PILOTS IN A COMBAT ENGAGEMENT MODEL. THIS TECHNIQUE UTILIZES THE THEORY OF DECISION TABLES TO AFFORD USERS OF THE MODEL A GREAT DEAL OF FLEXIBILITY IN DEFINING THE TACTICS TO BE SIMULATED.
- HUM73 HUMBY, E.
PROGRAMS FROM DECISION TABLES
MAC DONALD
LONDON, 1973, 91 PP.
THIS BOOK EXPLAINS GRAPHICALLY MOST OF THE POPULAR TRANSLATION TACTICS, IDENTIFIES THEIR RELATION TO A THEORETICALLY BEST APPROACH AND POINTS THE WAY TO SOME PRACTICAL COMPROMISES.
- HUROO HURLBERT, H.R.
THE USE OF DECISION TABLES AS A SYSTEMS DESIGN TECHNIQUE
IBM TIE Z 77-4376.
DATE UNKNOWN.
- IBMOOA IBM
AUTOMATED DESIGN ENGINEERING IN THE TRANSFORMER & RECTIFIER DIVISION OF I-T-E CIRCUIT BREAKER COMPANY
IBM APPLICATION BRIEF, FORM NO. K20-0016-0
DATE UNKNOWN.
THIS DESCRIPTION OF AN APPLICATION OF AUTOMATED DESIGN ENGINEERING CONTAINS THREE OF "HUNDREDS OF TABLES" CREATED AT I-T-E TO DOCUMENT THE EXISTING LOGIC FOR VARIOUS PRODUCT LINES. FOR ADE : SEE IBM63A.
- IBMOOB IBM
THE SURVEY AND IMPLEMENTATION OF AN AUTOMATED DESIGN ENGINEERING SYSTEM
IBM FORM NO. C20-8153
DATE UNKNOWN.
- IBMOOC IBM
SYSTEM/360 AND SYSTEM/370 DECISION TABLE TRANSLATOR (DECTAT)
FOR COBOL AND PL/I
APPLICATION DESCRIPTION MANUAL, FORM NO 5734-XR2
PROGRAM DESCRIPTION MANUAL, FORM NO 5734-XR2
OPERATIONS MANUAL, FORM NO 5734-XR2
DATE UNKNOWN.
THESE THREE PUBLICATIONS CONSTITUTE THE COMPLETE MANUAL FOR THE DECTAT-TRANSLATOR. THIS PREPROCESSOR TRANSLATES DECISION TABLES INTO A COBOL OR PL/I PROGRAM. THE SOURCE PROGRAM PRODUCED IS READY FOR COMPILEATION.
- IBMOOD IBM

IBM 1401 DECISION LOGIC TRANSLATOR (1401-SE-05X)
APPLICATION DESCRIPTION MANUAL, H20-0063-0, 2 PP.
PROGRAM REFERENCE MANUAL, H20-0068-0, 54 PP.
UNDATED.

THIS PROGRAM ACCEPTS DECISION TABLES WRITTEN IN A
FORTRAN-ORIENTED DECISION TABLE LANGUAGE AND
AUTOMATICALLY TRANSLATES THEM INTO A FORTRAN SOURCE
PROGRAM, GIVING PERTINENT DIAGNOSTICS IN THE PROCESS.

- IBM62 IBM
DECISION TABLES - A SYSTEMS ANALYSIS AND DOCUMENTATION
TECHNIQUE
GENERAL INFORMATION MANUAL, F20-8102
1962, 21 PP.
DESCRIBES THE BASIC CONCEPTS OF DECISION TABLES AND A
MINIMUM OF CONVENTIONS FOR THEIR USE IN SYSTEMS ANALYSIS,
PROCEDURE DESIGN, AND DOCUMENTATION. WHILE THE CONCEPTS
IN THE TEXT ARE PRESENTED ON A LEVEL FOR COMPREHENSION
BY STUDENTS IN BASIC COMPUTER COURSES, THE PROPOSED
TECHNIQUES ARE APPLICABLE AT ALL LEVELS OF SOPHISTICATION
BY EVERYONE IN A DATA PROCESSING ENVIRONMENT.
- IBM63A IBM
GENERAL INFORMATION MANUAL - AUTOMATED DESIGN ENGINEERING
IBM FORM NO. E20-8151
1963.
THIS MANUAL DESCRIBES A GENERALISED SYSTEM FOR AUTOMATED
DESIGN ENGINEERING (ADE), BASED ON THE USE OF A COMPUTER.
STARTING FROM CUSTOMER REQUIREMENTS, IT PRODUCES
COMPLETED DESIGN INFORMATION FOR THE MANUFACTURING
PROCESS. A SECTION IS DEVOTED TO DECISION TABLES,
DESCRIBED AS A PRIMARY TOOL IN THE SYSTEM. THEY ARE USED
MAINLY FOR RECORDING (IN EXTENDED ENTRY FORMAT) PRODUCT
CHARACTERISTICS OVER A RANGE OF PRODUCTS ; THE TABLES
CAN THEN BE USED TO FACILITATE MATCHING OF CUSTOMER
SPECIFICATIONS AGAINST EXISTING PRODUCTS. THIS USE OF
TABLES IS ILLUSTRATED BY AN EXAMPLE DEALING WITH AN
ARMATURE COIL, OVER THREE TABLES. THE SECTION CLOSES
BY SUMMARISING SOME ADVANTAGES OF USING DECISION TABLES
IN ADE.
- IBM63B IBM
DECISION TABLES, EDUCATION GUIDE
IBM FORM NO. R25-1684-0
1963.
THIS IS A LOOSE-LEAF TRAINING MANUAL, DIVIDED INTO THREE
MAIN SECTIONS : INSTRUCTOR'S TEXT, PRACTICE PROBLEMS
AND COPIES OF OVERHEAD PROJECTOR FOILS PREPARED FOR USE
WITH THE MANUAL.
- IBM63C IBM
DECISION TABLES, PRACTICE PROBLEMS AND SOLUTIONS
IBM FORM NO. R25-1685-1
1963, 11 PP.
THE FOUR PRACTICE PROBLEMS, WITH SOLUTIONS, ARE DESIGNED
TO AID THE STUDENT IN LEARNING HOW TO USE AND PREPARE
LIMITED ENTRY DECISION TABLES.
- IBM64 IBM
BASIC COMPUTER SYSTEMS PRINCIPLES. PROGRAMMED INSTRUCTION
COURSE
IBM FORM NO. R29-0071-4, 1
1964.
THIS PROGRAMMED INSTRUCTION TEXT INTRODUCES DECISION
TABLES AS A "MAJOR TOOL IN THE ANALYSIS AND SOLUTION OF
DATA PROCESSING PROBLEMS" IN SOME 70 FRAMES.
- IBM67 IBM
CONTRIBUTED PROGRAM NO. 0360D/03.2.004 : PREPROCESSOR FOR
ENCODED TABLES (PET)
S/360 GENERAL PROGRAM LIBRARY, FORMS NO. J40-6339 AND
J43-6340
APRIL 1967.
- IBM68A IBM
SYSTEM/360 DECISION LOGIC TRANSLATOR
OPERATIONS MANUAL, FORM NO. H20-0573
APPLICATION DESCRIPTION MANUAL, FORM NO. H20-0492-1
PROGRAM DESCRIPTION MANUAL, FORM. NO. H20-0572

1968.

THESE THREE PUBLICATIONS CONSTITUTE THE COMPLETE MANUAL FOR THE SYSTEM/360 DECISION LOGIC TRANSLATOR. THIS PREPROCESSOR ACCEPTS DECISION TABLES PUNCHED IN CARDS AND TRANSLATES THE TABLES INTO A FORTRAN PROGRAM. THE FORTRAN SOURCE PROGRAM IS PUNCHED IN CARDS OR STORED ON DISK OR TAPE READY FOR COMPILEATION.

IBM68B IBM
CONTRIBUTED PROGRAM NO. 0360-D.03.7.017 : SYSTEM/360 MACROS
(ALP DOS)
IBM FORM NOS. J40-6417 AND J40-6418
1968.

IBM68C IBM NEDERLAND
ALGEMENE HANDLEIDING BESLISSINGSTABELLEN
(D) GENERAL MANUAL OF DECISION TABLES
1968, 40 PP.

THE AIM OF THIS PAPER IS TO PROVIDE THE READER WITH THE FUNDAMENTALS OF DECISION TABLES. MOREOVER, IT GIVES ADDITIONAL INFORMATION FOR THE USE OF THESE TABLES BY STATING EIGHT SITUATIONS THAT MAY POSSIBLY APPEAR IN A COMPANY.

IBR71 IBRAMSHAH, M. RAJARAMAN, V.
A FAST RULE MASK ALGORITHM FOR THE CONVERSION OF DECISION TABLES TO COMPUTER PROGRAMS
TECHNICAL REPORT COMPUTER CENTRE, INDIAN INSTITUTE OF TECHNOLOGY
KANPUR, 1971.

IBR78 IBRAMSHAH, M. RAJARAMAN, V.
DETECTION OF LOGICAL ERRORS IN DECISION TABLE PROGRAMS
CACM, 21(12)
DECEMBER 1978, 1016-1025
IN THIS PAPER AN ALGORITHM TO DETECT LOGICAL ERRORS IN A LIMITED-ENTRY DECISION TABLE AND IN LOOP-FREE PROGRAMS WITH EMBEDDED DECISION TABLES IS DEVELOPED. ALL THE CONDITIONS IN THE DECISION TABLES ARE ASSUMED TO BE INEQUALITIES OR EQUALITIES RELATING LINEAR EXPRESSIONS. IT IS ALSO ASSUMED THAT ACTIONS ARE LINEAR IN VARIABLES WHICH OCCUR IN THE CONDITION STUB OF THE DECISION TABLE(S) TO WHICH CONTROL IS TRANSFERRED FROM THE TABLE. THE ALGORITHM IS BASED ON DETERMINING WHETHER A SET OF LINEAR INEQUALITIES HAS OR DOES NOT HAVE A SOLUTION. THE ALGORITHM DESCRIBED IN THE PAPER IS IMPLEMENTED IN FORTRAN IV.

ICL00 ICL
DECISION TABLE PREPROCESSORS XEH2, XEH3 AND XEH4
GENERAL MANUAL
DATE UNKNOWN.
THREE COBOL-PREPROCESSORS ARE DESCRIBED. BY SELECTING THE APPROPRIATE PREPROCESSOR, THE USER CAN OPT FOR MINIMUM COMPUTER STORAGE OR MINIMUM EXECUTION TIME.

ICL69A ICL
INTRODUCTION TO DECISION TABLES
ICL TECHNICAL PUBLICATIONS DEPARTMENT, REFERENCE NO. 4139
FEBRUARY 1969.
THIS TRAINING MANUAL CONTAINS FOUR PARTS : 1. BASIC STRUCTURE, PRINCIPLES AND CONVENTIONS, 2. A SERIES OF PROBLEMS WITH QUESTIONS AND ANSWERS, 3. DRAWING UP AND CHECKING LIMITED ENTRY TABLES, USING THE "FULL SPECIFICATION" METHOD AND 4. A SUMMARY OF THE FIRST THREE PARTS.

ICL69B ICL
DECISION TABLES WITH COBOL
ICL 1900 SERIES PROVISIONAL MANUAL
LONDON, JANUARY 1969.
THIS IS A MANUAL DESCRIBING A DECISION TABLE PREPROCESSOR WHICH GENERATES COBOL. THE MANUAL IS IN TWO PARTS, THE FIRST DEALING WITH THE USE OF DECISION TABLES IN COMMERCIAL PROGRAMMING. THE SECOND PART DISCUSSES THE USE OF THE LANGUAGE CALLED DETAB/65 AND THE PREPROCESSOR UTILISED. THE MANUAL IS COMPLETE IN THE USUAL MANNER, WITH HARDWARE REQUIREMENTS AND OPERATING INSTRUCTIONS. APPENDICES COVER THE PROGRAMMING TECHNIQUES USED IN THE PREPROCESSOR.

- INC80 INCE, D.C.
AN INTERPRETATIVE IMPLEMENTATION OF LIMITED-ENTRY DECISION TABLES IN ALGOL 68
SIGPLAN NOTICES, 15(2)
FEBRUARY 1989, 43-45.
THE INTERPRETATIVE CONVERSION OF DECISION TABLES INTO LINEAR PROGRAM CODE IS ACCOMPLISHED USING PARTICULAR LANGUAGE FEATURES OF ALGOL 68.
- INFOO INFORMATION MANAGEMENT
DETAP MANUAL
DATE UNKNOWN.
THIS MANUAL IS REFERRED TO IN THE CANADIAN STANDARDS ASSOCIATION DRAFT STANDARD ON DECISION TABLES (CSA68).
- ING68 INGLIS, J. KING, P.J.H.
FLOWCHARTS AND DECISION TABLES
THE COMPUTER JOURNAL, 11(1)
MAY 1968, 117-118.
LETTER FROM KING PRESENTING SOME CORRESPONDENCE BETWEEN HIMSELF AND INGLIS REGARDING AN OMISSION IN THE FORMER'S PAPER KING67B.
- IRM73 IRMSCHER, M.
ZUR DARSTELLUNG VON ABLAUFPLAENEN MIT ESER-STANDARDS
(G) PRODUCTION OF FLOWCHARTS BY MEANS OF ESER-STANDARDS
RTDV, 2(10)
1973, 16-20.
- JAC64 JACKSON, L.
EXPERIENCE OF THE FOREIGN TRADE DIVISION SEMINAR ON DECISION TABLES
SEPTEMBER 1964.
- JAC68A JACKSON, M.
THE NEED FOR IMPRECISION
DATAMATION, 14(2)
FEBRUARY 1968, 143-144.
THE AUTHOR'S THEME IS THE NEED TO RECOGNISE THOSE SITUATIONS "WHERE CONFUSION AND INCONSISTENCY ARE INHERENT ELEMENTS OF THE PROBLEM". HE ARGUES IN FAVOUR OF DEVELOPING PROGRAMMING METHODS THAT DEAL WITH THESE BY ALLOWING "INCONSISTENCY, REDUNDANCY, AMBIGUITY AND INCOMPLETENESS". SOME TYPICAL SITUATIONS ARE OUTLINED AND REASONS ANALYSED FOR THE USER'S FREQUENT INABILITY TO DEFINE NEEDS PRECISELY.
SEE POLLACK'S "DECISION, NOT IMPRECISION" (POL68A) FOR COMMENTS.
- JAC68B JACKSON, M.
THE MEANING OF IMPRECISION
DATAMATION, 14(7)
JULY 1968, 170.
FOLLOWING POLLACK'S COMMENTS (POL68A) ON THE AUTHOR'S FIRST REMARKS, THIS ARTICLE ARGUES HIS CASE AFRESH AND SUGGESTS THAT DECISION TABLES "PROPERLY USED" OFFER "ONE OF THE FEW AVENUES OF ESCAPE FROM THE RIGOURS OF PROCEDURAL PROGRAMMING" FOR THOSE SITUATIONS THAT CANNOT BE SUCCESSFULLY HANDLED BY THE LATTER. THE APPROACH ADOPTED BY TAP IS BRIEFLY DESCRIBED (THOUGH THE PACKAGE IS NOT SPECIFICALLY IDENTIFIED) AS AN ILLUSTRATION OF A "RELATIVELY CRUDE" FACILITY THAT ALLOWS A "PROGRAM TO BE DEVELOPED BY A SERIES OF SUCCESSIVE APPROXIMATIONS".
- JAN71 JANISCH, R.
CODIEREN VON ENTSCHEIDUNGSTABELLEN
(G) CODING DECISION TABLES
MANUAL
WUETT. INGENIEURSVEREIN, STUTTGART, 1971.
- JAR68 JARVIS, J.M.
THE APPLICATION OF DECISION TABLES TO COMPUTER PROGRAMMING
THE COMPUTER BULLETIN (SOUTH AFRICA),
PART I : 9(6), JULY-AUGUST 1968, 5-13
PART II : 9(7), SEPTEMBER-OCTOBER 1968, 12-24.
THESE TWO ARTICLES ARE A PRACTICAL STUDY IN THE APPLICATION OF THE CDC DETAB/65 PREPROCESSOR TO A PROGRAMMING PROJECT AT THE SOUTH AFRICAN IRON & STEEL INDUSTRIAL

CORPORATION. THE AUTHOR IDENTIFIES, DESCRIBES IN DETAIL AND ILLUSTRATES WITH ACTUAL MATERIAL, EACH SEPARATE STAGE IN THE PROJECT FROM SYSTEM DESIGN THROUGH PROGRAMMING AND KEYPUNCHING TO DEBUGGING. THE FINAL SUMMARY CONSIDERS THE ADVANTAGES ACCRUING AT VARIOUS STAGES FOR SYSTEMS DESIGN AND PROGRAM MAINTENANCE.

- JAR71 JARVIS, J.M.
AN ANALYSIS OF PROGRAMMING VIA DECISION TABLE COMPILERS
SIGPLAN NOTICES, 6(8)
SEPTEMBER 1971, 20-32.
THIS PAPER COVERS WORK DONE DURING THE PERIOD 1966-1971 USING DECISION TABLES AND DECISION TABLE COMPILERS IN THE BUSINESS APPLICATIONS AREA. MOST OF THE PAPER IS DEVOTED TO SETTING OUT THE CHARACTERISTICS OF DECISION TABLES THAT HAVE EMERGED FROM AN ANALYSIS OF SYSTEMS AND PROGRAMS THAT HAVE ACTUALLY BEEN INSTALLED USING THE TECHNIQUE. THE STATISTICS GIVEN IN THE TEXT ARE DRAWN FROM A COST AND FINANCIAL SYSTEM INSTALLED AT A STEELWORKS AND SEVERAL SYSTEMS INSTALLED AT A LIFE INSURANCE COMPANY.
- JOH68 JOHNSTON, F.J.J. DAVIS, J.C.
SURVEY REPORT ON THE USE OF DECISION TABLES IN DATA PROCESSING
NATIONAL COMPUTING CENTRE
MANCHESTER, AUGUST 1968.
THIS REPORT IS THE RESULT OF A SURVEY UNDERTAKEN TO ASSESS THE DEGREE OF INTEREST AMONG COMPUTER USERS IN THE USE OF DECISION TABLES AND THE EXTENT TO WHICH THE TECHNIQUE IS APPLIED IN PRACTICE. THE SURVEY IS DIVIDED INTO THREE SECTIONS : METHOD AND SCOPE OF THE SURVEY ; ANALYSIS OF SURVEY FINDINGS ; THE NEXT STAGE IN THE DEVELOPMENT OF THE TECHNIQUE. APPENDICES CONTAIN : THE QUESTIONNAIRE IN THE USE OF DECISION TABLES ; TYPES OF BUSINESS AND MACHINES REPRESENTED ; AND AN ANALYSIS OF RESPONSE TO THE SURVEY.
- JOH69 JOHNSTON, F.J.J.
DECISION TABLES
BUSINESS SYSTEMS AND EQUIPMENT.
SEPTEMBER 1969, 41, 43-44, 46 AND 48.
THIS IS A STRAIGHTFORWARD ARTICLE ON DECISION TABLES. IT DESCRIBES AND ILLUSTRATES LIMITED AND EXTENDED ENTRY FORMATS AND CONCENTRATES ON THE FORMER. COMPARISON IS MADE WITH FLOWCHARTING AND ILLUSTRATED THROUGH A PROBLEM IN CAPITAL GAINS TAX. AREAS OF USE ARE BRIEFLY OUTLINED.
- JON74 JOHNSON, R.G.
LOGICAL RELATIONS BETWEEN CONDITIONS IN DECISION TABLES
PH. D. THESIS, UNIVERSITY OF LONDON
1974.
THE MAIN RESULTS OF THIS THESIS ARE SUMMARIZED IN KIN75.
- JO081 JOOKEN, F.
VERGELIJKENDE STUDIE VAN OPTIMALE ALGORITMEN VOOR HET OMZETTEN VAN BELISSINGSTABLLEN IN COMPUTERPROGRAMMA'S
(D) COMPARATIVE STUDY OF OPTIMAL ALGORITHMS FOR THE CONVERSION OF DECISION TABLES INTO COMPUTER PROGRAMS
K.U.LEUVEN, DEPT. OF APPLIED ECONOMICS, THESIS 1981,
110 PP.
THIS THESIS INVESTIGATES THE CONVERSION ALGORITHMS OF REINWALD & SOLAND, BAYES, LEW AND MARTELLI & MONTANARI IN A COMPARATIVE WAY. IT IS CONCLUDED THAT THE ALGORITHM OF LEW IS THE OPERATIVELY MOST EFFICIENT AND FLEXIBLE ALGORITHM.
- JUN71 JUNGE, O.
ENTScheidungstabellen-technik bei der Erarbeitung wissenschaftlich-technischer Leitungsinformationen
(G) DECISION TABLE TECHNIQUE IN SCIENTIFIC AND TECHNICAL INFORMATION PROCESSING
ORGANISATION, 5(6)
1971, 29-31.
THE AUTHOR EMPHASIZES THE NEED OF A THOROUGH DOCUMENTATION IN RESEARCH AND DEVELOPMENT. AN EXAMPLE TAKEN FROM THE INFORMATION PROCESSING DEPARTMENT AND TACKLED BY MEANS OF DECISION TABLES DEMONSTRATES THE USEFULNESS

OF THIS TECHNIQUE.

- KAD72 KADAR, I. KOVACS, P.
A DONTESI TABLAKROL
(H) ABOUT DECISION TABLES
INFORMACIO ELEKTRONIKA, 7(3)
1972, 216-221.
DECISION TABLES MAY BE MADE BY APPLYING THE DIRECT OR
INDIRECT METHOD. THIS ARTICLE TRIES TO ILLUSTRATE BOTH
METHODS WITH THE HELP OF ILLUSTRATIONS. IT SHOWS HOW
DECISION TABLES MAY BE SPLIT AND COUPLED, HOW ERRORS
MAY BE DETECTED AND REMOVED AND ALSO HOW THESE TABLES
MAY BE CHECKED.
- KAH64 KAHN, J.
REFLECTIONS ON DECISION TABLES AND FOREIGN TRADE EXPERTS
SEMINAR ON DECISION TABLES
SEPTEMBER 1964.
- KAV60 KAVANAGH, T.F.
TABSOL, A FUNDAMENTAL CONCEPT FOR SYSTEMS-ORIENTED
LANGUAGES
PROC. EASTERN JOINT COMPUTER CONFERENCE, 18
13-15 DECEMBER 1960, 117-136.
THIS PAPER INTRODUCES AND DESCRIBES DECISION STRUCTURE
TABLES, THE ESSENTIAL ELEMENT IN TABSOL, A TABULAR
SYSTEMS-ORIENTED LANGUAGE DEVELOPED BY THE GENERAL
ELECTRIC COMPANY. DECISION STRUCTURE TABLES CAN BE USED
TO DESCRIBE COMPLICATED, MULTI-VARIABLE, MULTI-RESULT
DECISION SYSTEMS. VARIOUS APPROACHES TO THE AUTOMATIC
COMPUTER SOLUTION OF DECISION STRUCTURE TABLES ARE
PRESENTED. SOME BENEFITS WHICH HAVE BEEN OBSERVED IN
APPLYING THIS LANGUAGE CONCEPT ARE ALSO DISCUSSED.
- KAV61 KAVANAGH, T.F.
TABSOL - THE LANGUAGE OF DECISION MAKING
COMPUTERS AND AUTOMATION, 10(9)
SEPTEMBER 1961, 15 AND 18-22.
THIS IS A SLIGHTLY CONDENSED VERSION OF THE AUTHOR'S
PAPER KAV60.
- KAV62 KAVANAGH, T.F.
MANUFACTURING APPLICATIONS OF DECISION STRUCTURE TABLES
PROC. DECISION TABLES SYMPOSIUM
SEPTEMBER 1962, 89-97.
A DESCRIPTION IS GIVEN OF THREE AREAS OF APPLICATION
WITHIN GENERAL ELECTRIC IN WHICH DECISION TABLES HAVE
BEEN USED. THE APPLICATIONS INCLUDE PLANNING THE
MANUFACTURE OF CAST ROTORS FOR ELECTRIC MOTORS, THE
MANUFACTURING INSTRUCTIONS FOR PRODUCING GEAR WHEELS, AND
THE SPECIFICATION OF STOCK CONTROL RULES IN TRIM, A
COMPUTER SIMULATION MODEL USED BY GE TO CONSTRUCT AND
TEST ALTERNATIVE TYPES OF INVENTORY SYSTEMS.
- KAV63A KAVANAGH, T.F.
DECISION STRUCTURE TABLES - A TECHNIQUE FOR BUSINESS
DECISION MAKING
JOURNAL OF INDUSTRIAL ENGINEERING
SEPTEMBER-OCTOBER 1963, 249-258.
- KAV63B KAVANAGH, T.F. ALLEN, M.
THE USE OF DECISION TABLES
PROC. 1963 INTERNATIONAL DATA PROCESSING CONFERENCE
1963, 318.
ALSO IN : ADT, 39-61.
DECISION TABLES ALLOW MORE DIRECT COMMUNICATION BETWEEN
THE ULTIMATE USER AND THE COMPUTER. HOWEVER, THE ADDED
POWER OF AN ALGORITHMIC LANGUAGE SEEMS VERY APPEALING.
AS A RESULT, GENERAL ELECTRIC HAS INCORPORATED DECISION
STRUCTURE TABLES UNDER THE TRADE NAME TABSOL AS AN
INTEGRAL PART OF GECOM, A COBOL-BASED COMPILER. THIS
PAPER GIVES AN INTRODUCTION TO DECISION TABLES IN TERMS
OF TABSOL.
- KEP77 KEPNER, H.M.
TRANSACTION PROCESSING PROGRAM GENERATION
PROC. USAFA COMPUTER RELATED INFORMATION
SYSTEMS SYMPOSIUM
JANUARY 1977, 17.1-17/22.

THIS PAPER DISCUSSED THE USE OF DECISION TABLES IN THE ADVANCED PERSONNEL DATA SYSTEM OF THE US DEPARTMENT OF THE AIR FORCE. DECISION TABLES ARE USED AS A COMMUNICATION TOOL. THE DETAP PROCESSOR IS USED TO TRANSLATE THE TABLES INTO COBOL STATEMENTS.

- KES71 KESHNER, J.
BLOKOVA SCHEMATA NEBO ROZHODOVACI TABULKY
(C) BLOCK DIAGRAMS OR DECISION TABLES
INORGA PRAG, 5(1)
1971, 4-8.
- KES75 KESSEL, R.
A NEW APPROACH TO SYSTEM AND PROGRAM DEVELOPMENT SYSTEMS
(S. AFRICA), 5(10) AND 5(11)
PART I: OCTOBER 1975, 15-18.
PART II: NOVEMBER 1975, 24-26.
THE STEPS TAKEN IN THE DEVELOPMENT OF A COMPUTER PROGRAM USING A DECISION TABLE PROCESSOR ARE SHOWN WITH REFERENCE TO AN ERROR CHECKING EXAMPLE.
- KIN59 KING, J.E.
LOGTAB - A LOGIC TABLE TECHNIQUE
GENERAL ELECTRIC CO.
SCHEECTADY, 1959, 11 PP.
LOGTAB IS A METHOD FOR EXPRESSING THE LOGIC OF A COMPUTER PROGRAM IN TABLE FORM AND THE SUBSEQUENT INTERPRETATION OF THE TABLE BY THE COMPUTER. THE EXTERNAL AND INTERNAL LANGUAGE AND THE METHOD OF INTERPRETATION ON THE IBM 704 ARE DESCRIBED. LOGTAB IS AN EXTENSION OF TABSOL FOR THE IBM 702. THE TABLE REPLACES THE FLOW-CHART AND HAND CODING OF A PROGRAM AND REMAINS AS A FINAL AND UNDERSTANDABLE RECORD OF THE PROGRAM.
- KIN66 KING, P.J.H.
CONVERSION OF DECISION TABLES TO COMPUTER PROGRAMS BY RULE MASK TECHNIQUES
CACM, 9(11)
NOVEMBER 1966, 796-801.
IN THIS PAPER A MODIFICATION OF THE RULE MASK TECHNIQUE ("INTERRUPTED RULE MASK") FOR CONVERTING LIMITED ENTRY DECISION TABLES TO COMPUTER PROGRAMS IS DISCUSSED. IT TAKES INTO ACCOUNT BOTH RULE FREQUENCIES AND THE RELATIVE TIMES FOR EVALUATING CONDITIONS. FOUR OF THE POSSIBLE STRATEGIES FOR THE DESIGN OF INTERRUPTED RULE MASK PROCEDURES ARE SET OUT. COMPARISON OF THESE WITH KIRK'S METHOD (KIRG5) SHOWS A SUBSTANTIAL IMPROVEMENT IN RUN TIME, IF THE BEST OF THE FOUR STRATEGIES IS SELECTED.
- KIN67A KING, P.J.H.
SOME COMMENTS ON SYSTEMATICS
THE COMPUTER JOURNAL, 10(1)
MAY 1967, 116-118.
IN THIS NOTE, THE AUTHOR POINTS OUT SOME INCONVENIENCES IN SYSTEMATICS AS FIRST PROPOSED BY GRINDLEY (GRIG6). THE TWO MAJOR FEATURES HE FINDS UNSATISFACTORY ARE ITS USE OF A DATA DICTIONARY AND A BOOLEAN AND/OR MATRIX, NEITHER OF WHICH HAS TAKEN ACCOUNT OF DEVELOPMENTS IN BOTH FIELDS, ESPECIALLY IN DECISION TABLES.
- KIN67B KING, P.J.H.
DECISION TABLES
THE COMPUTER JOURNAL, 10(2)
AUGUST 1967, 135-142.
THE PURPOSE OF THIS PAPER IS TO EXPLAIN THE BASIC IDEAS OF THE DECISION TABLE APPROACH, TO ILLUSTRATE ITS WIDE APPLICABILITY BY MEANS OF EXAMPLES, AND TO PRESENT A BIBLIOGRAPHY. THE USE OF DECISION TABLES IN THE CONTEXT OF MORE COMPREHENSIVE SYSTEMS IS ALSO MENTIONED.
- KIN68 KING, P.J.H.
AMBIGUITY IN LIMITED ENTRY DECISION TABLES
CACM, 11(10)
OCTOBER 1968, 680-684.
THE AUTHOR SUGGESTS THAT THE RULES LAID DOWN BY POLLACK FOR LIMITED ENTRY TABLES IN HIS ANALYSIS OF DECISION RULES IN DECISION TABLES ARE UNSATISFACTORY. THIS IS BECAUSE THEY MAY LEAD TO LONGER RUN TIMES THAN STRICTLY NECESSARY AND ALSO IMPOSE UNNECESSARY, BURDEN-

SOME AND SEEMINGLY OBSCURE PROCEDURES ;
THEY ARE IN ANY CASE FREQUENTLY APPLIED
WITHOUT REFERENCE TO THE INDEPENDENCE OF CONDITIONS
WHICH THEY ASSUMED. THE IMPORTANT ASPECT OF CHECKING IS
TO ELIMINATE AMBIGUITY FROM THE TABLES.
HE GOES ON TO DEFINE AMBIGUITY AND SUGGESTS THAT DIAG-
NOSTIC TESTS IN A DECISION TABLE PROCESSOR SHOULD INCLUDE
ONE FOR AMBIGUITY, AND DESCRIBES A PROCEDURE FOR
THIS. THIS IS ILLUSTRATED BY TWO EXAMPLES TAKEN RESPEC-
TIVELY FROM A (SIMPLIFIED) HIRE PURCHASE ARREARS PROCE-
DURE AND FROM AN INTEGRATION SUB-ROUTINE IN NUMERICAL
ANALYSIS.

- KIN69 KING, P.J.H.
THE INTERPRETATION OF LIMITED ENTRY DECISION TABLE FORMAT
AND RELATIONSHIPS AMONG CONDITIONS
THE COMPUTER JOURNAL, 12(4)
NOVEMBER 1969, 320-326.
BASIC DECISION TABLE FORMAT AND THE DEPENDENCE/INDEPEN-
DENCE OF CONDITIONS IN A TABLE ARE SEPARATE MATTERS BUT
TEND TO BE CONFUSED. THIS PAPER EMPHASISES THE INTERPRE-
TATION OF BASIC FORMAT. RELATIONSHIPS AMONG CONDITIONS
ARE DISCUSSED AND IT IS SHOWN THAT THIS IS A SEPARATE
MATTER FROM BASIC FORMAT. SOME FORMAL DEFINITIONS ARE
PROPOSED.
FOR A CRITICAL REMARK : SEE CAR70.
- KIN71 KING, P.J.H.
INTERPRETATION OF LIMITED ENTRY DECISION TABLE FORMAT
THE COMPUTER JOURNAL, 14(1)
FEBRUARY 1971, 54.
LETTER DISCUSSING THE POINTS RAISED IN CAR70, WHICH
REFERRED CRITICALLY TO THE AUTHOR'S PAPER KIN69.
- KIN73 KING, P.J.H. JOHNSON, R.G.
SOME COMMENTS ON THE USE OF AMBIGUOUS DECISION TABLES AND
THEIR CONVERSION TO COMPUTER PROGRAMS
CACM, 16(5)
MAY 1973, 287-290.
COMMENTS UPON PUBLISHED WORK ON DECISION TABLE TRANSLA-
TION USING METHODS SIMILAR TO THE RULE-MASK TECHNIQUE.
THE APPLICABILITY OF THESE METHODS UNDER VARIOUS POSSI-
BLE CONVENTIONS ON OVERALL TABLE MEANING IS DISCUSSED,
AND IT IS ARGUED THAT THERE IS A PLACE FOR THE MULTI-
RULE AND THE SINGLE-RULE CONVENTION IN DECISION TABLE
USAGE.
- KIN74 KING, P.J.H. JOHNSON, R.G.
COMMENTS ON THE ALGORITHMS OF VERHELST FOR THE CONVERSION
OF LIMITED ENTRY DECISION TABLES TO FLOWCHARTS
CACM, 17(1)
JANUARY 1974, 43-45.
THE AUTHORS COMMENT THE ALGORITHMS PROPOSED BY VERHELST
(VER72A). THEY ARGUE THAT THE KNOWLEDGE OF ANY LOGICAL
RELATIONSHIPS AMONG CONDITIONS, THE FREQUENCIES OF THE
RULES AND THE RELATIVE TIMES FOR TESTING THE CONDITIONS
DOESN'T SUFFICE TO CONSTRUCT AN OPTIMAL ALGORITHM.
MOREOVER, THEY DEMONSTRATE THAT THE LOWER BOUND FOR THE
TEST TIME OF A DECISION TABLE GIVEN IN VER72A IS INVALID.
IN A REPLY, VERHELST DISCUSSES THE CONCEPT OF OPTIMI-
ZATION USED BY HIM AND SUGGESTS HIS "OPTIMUM-FINDING"
ALGORITHM SHOULD BE RENAMED AS AN "OPTIMUM-APPROACHING"
ONE.
- KIN75 KING, P.J.H. JOHNSON, R.G.
THE CONVERSION OF DECISION TABLES TO SEQUENTIAL TESTING
PROCEDURES
THE COMPUTER JOURNAL, 18(4)
NOVEMBER 1975, 298-306.
THIS PAPER PRESENTS A GENERAL METHOD FOR THE CONVERSION
OF A DECISION TABLE TO A SEQUENTIAL TESTING PROCEDURE
REPRESENTED AS A TREE. THE METHOD RESOLVES DIFFICULTIES
WHICH HAVE BEEN DISCUSSED PREVIOUSLY IN THE LITERATURE
BUT LEAVES OPEN THE QUESTION OF THE PRODUCTION OF
OPTIMUM OR NEAR OPTIMUM TREES RELATED TO PARTICULAR
CRITERIA. IT PROVIDES, HOWEVER, A FRAMEWORK WITHIN WHICH
SUCH OPTIMISING MAY BE DEVELOPED. SUGGESTIONS ARE ALSO
MADE FOR IMPROVING THE CONCEPTS AND NOTATIONS USED IN
EXTENDED ENTRY TABLES AND FOR THEIR USE IN TRANSLATION.

- KIR65 KIRK, H.W.
USE OF DECISION TABLES IN COMPUTER PROGRAMMING
CACM, 8(1)
JANUARY 1965, 41-43.
THIS PAPER INTRODUCES THE RULE MASK TECHNIQUE FOR THE CONVERSION OF (LIMITED ENTRY) DECISION TABLES INTO COMPUTER PROGRAMS. THE TECHNIQUE IS BASED ON THE CREATION OF A BINARY IMAGE OF THE DECISION TABLE IN COMPUTER MEMORY. A BINARY IMAGE OF A GIVEN SET OF INPUT CONDITIONS CAN ALSO BE CREATED. THIS DATA IMAGE IS USED TO SCAN THE DECISION TABLE IMAGE TO ARRIVE AT THE PROPER COURSE OF ACTION.
- KLI61 KLICK, D.C.
TABSOL : A DECISION TABLE LANGUAGE FOR THE GE 225
PREPRINTS OF SUMMARIES OF PAPERS, ACM NATIONAL MEETING,
PAPER 10 B-2
LOS ANGELES, 5-8 SEPTEMBER 1961.
- KNI70 KNIGHT, J.
THE DESIGN OF THE CHECKING PHASE OF A DECISION TABLE
PREPROCESSOR
M. SC. THESIS, UNIVERSITY COLLEGE OF WALES
ABERYSTWYTH, 1970.
- KN073 KNOCH, W.
ENTSCHEIDUNGSTABELLEN UND IHRE PRAKTISCHE ANWENDUNG
(G) DECISION TABLES AND THEIR PRACTICAL APPLICATION
RTDV, 10(12)
DECEMBER 1973, 33-37.
THE CONSTRUCTION OF SIMPLE DECISION TABLES IS DESCRIBED IN TERMS OF COMPARISON MATRICES, A CONDITION TABLE AND AN ACTION TABLE, WHICH INTER-RELATE FOR VARIOUS OPERATIONS. ELIMINATION OF REDUNDANCIES AND THE RESULTING SAVING IN RUNNING TIME ARE DISCUSSED.
- KN074 KNOCH, W. DOEMELAND, S.
AUTOMATISCHE HERSTELLUNG UND REPRODUKTION VON ABLAUFPLAENEN
IN FORM VON ENTSCHEIDUNGSTABELLEN
(G) AUTOMATIC PRODUCTION AND REPRODUCTION OF FLOWCHARTS IN THE FORM OF DECISION TABLES
RTDV, 11(4)
APRIL 1974, 39-40.
THE DECISION TABLE TECHNIQUE IS PRESENTED AS A RATIONALIZED DOCUMENTATION TECHNIQUE. A COMPARISON WITH EXISTING TECHNIQUES IS GIVEN.
- KOC70 KOCH, R.F. KROHN, M.J.
MC GREW, P.W. SIBLEY, E.H.
PSL VERSION 2 RELEASE 1 :
A PSL LANGUAGE PRIMER
ISDOS WORKING PAPER NO. 33
DEPARTMENT OF INDUSTRIAL ENGINEERING
UNIVERSITY OF MICHIGAN
ANN ARBOR, AUGUST 1970, 10/1-17.
IN THIS SECTION OF AN INTERNAL WORKING PAPER, DECISION TABLES ARE PRESENTED IN TERMS OF THE PSL LANGUAGE AS DEVELOPED IN THE ISDOS RESEARCH PROJECT.
- KOL64 KOLENCE, K.W.
DECISION TABLE - DATA STRUCTURE RELATIONSHIPS
PRESENTED TO THE SAN FRANCISCO CHAPTER OF ACM
20 FEBRUARY 1964.
- KOL67 KOLENCE, K.W.
DECISION TABLES AND DATA STRUCTURES -
A DESIGN METHODOLOGY
SYMPOSIUM ON DECISION TABLES,
USASI AD HOC X3.6.7/3.4.20
ATLANTIC CITY, APRIL 1967.
- KOL71 KOLTAI, T.
BEVEZETES A DONTESI TABLAZATOKBA
(H) INTRODUCTION TO DECISION TABLES
STATISZTIKAI KIADO VALLALAT
1971.
- KOS74 KOSYACHENKO, S.A. KUL'BA, V.V. TSVIRKUN, A.D.

OPTIMIZATION OF A SYSTEM DEBUGGING PROCESS BY THE USE OF
DECISION TABLES
AUTOMATION & REMOTE CONTROL, 35(8)
AUGUST 1974, 1359-1364
TRANSLATION OF AN ARTICLE PUBLISHED IN : AVTOMATIKA I
TELEMEKHANIKA, 35(8)
AUGUST 1974, 171-177 (IN RUSSIAN).

THE AUTHORS GIVE A METHOD OF OPTIMIZING THE SYSTEM
DEBUGGING PROCESS BY THE USE OF DECISION TABLES. THEY
GIVE FORMULATIONS OF SYSTEM DEBUGGING OPTIMIZATION
PROBLEMS FOR VARIOUS ACTUAL SITUATIONS ARISING IN
AUTOMATED CONTROL SYSTEMS.

- KOL75 KOLTAI, T.
(H) REAL SIGNIFICANCE OF DECISION TABLES
INFORMACIO ELEKTRONIKA, 10(2)
1975, 143-148.
THE ADVANTAGES OF USING DECISION TABLES ARE DISCUSSED
WITHIN A GENERALIZED PROGRAMMING CONTEXT.
- KRA66 KRAMER, F.A. KIRK, G.J.
DECISION TABLE TECHNIQUES IN COMPUTER CONTROL
IEEE TRANSACTIONS ON POWER APPARATUS AND SYSTEMS, 85(5)
MAY 1966, 495-498.
DECISION TABLES FOR REAL TIME PROCESS CONTROL LOGICALLY
PRESENT THE SOLUTION TO A PROBLEM AND DIRECTLY REPLACE
FLOWCHARTS. THEIR POWER IS THAT FORMAT AND PROGRAMMING
ARE INDEPENDENT OF THE PROBLEM, AND IT IS THEREBY
POSSIBLE TO AUTOMATE MUCH OF THE EFFORT REQUIRED TO
DESIGN, PROGRAM, TEST, AND DOCUMENT SOLUTIONS TO COMPLEX
CONTROL PROBLEMS.
- KRA74 KRATOCHVIL, E.
(C) AUTOMATIC CONVERSION OF DECISION TABLES
MECHANIZACE AUTOMATIZACE ADMINISTRATIVY, 14(11)
1974, 430-433.
THIS ARTICLE OUTLINES THREE METHODS OF REALIZATION OF
THE DECISION TABLE CONVERSION : THE USE OF COMPILERS
TRANSLATING DECISION TABLES DIRECTLY INTO MACHINE CODE,
USING INTERPRETIVE PROGRAMS AND USING PREPROCESSORS.
IT DISCUSSES OPTIMIZATION WITH RESPECT TO CONVERSION
TIME, PROGRAM PROCESSING TIME AND REQUIRED INTERNAL
STORAGE. THE PRINCIPLE OF THE TWO CONVERSION APPROACHES -
THE FLOWCHART ALGORITHMS AND THE MATRIX ALGORITHMS - IS
SURVEYED.
- KRA75 KRATOCHVIL, E.
(C) PROTAB - THE TRANSLATOR OF DECISION TABLES FOR THE
EC 1021 COMPUTER
MECHANIZACE AUTOMATIZACE ADMINISTRATIVY, 15(10)
1974, 403-404.
THE ADVANTAGES OF USING DECISION TABLES IN EFFECTIVE
EVALUATION OF DECISION PROCESSES ARE OUTLINED, THE COBOL
PROTAB-PREPROCESSOR IS DESCRIBED.
- KRO73 KROEMER, H.
RATIONALISIEREN ENTSCHEIDUNGSTABELLEN DEN
PROGRAMM-AENDERUNGSDIENST ? - EIN BEISPIEL AUS DER PRAXIS
(G) CAN PROGRAM MODIFICATION SERVICE BE RATIONALIZED BY
MEANS OF DECISION TABLES ? - A PRACTICAL EXAMPLE
ONLINE, 11(11)
NOVEMBER 1973, 790-796.
THIS ARTICLE DESCRIBES AND EXPLAINS, WITH THE USE OF
EXAMPLES, THE PRACTICAL PROCEDURES FOR RATIONALIZING
MODIFICATIONS TO PROGRAMS, BASED ON THE USE OF THE
VORELLE ADVOR-641 PRETRANSLATOR AND DECISION TABLES. THE
PRETRANSLATOR CONVERTS THE PROGRAM, SET UP ACCORDING TO
THE DECISION TABLES, INTO A SYMBOLIC PROGRAM WHILE
TESTING FOR COMPLETENESS, REDUNDANCY AND CONTRADICTIONS
INCLUDING LOGIC ERRORS.
- KUN73 KUNETZ, L.
ALGORITMUSOK A DONTESI TABLAK PROGRAMOZASARA ES SZAMITOGEPI
FORDITASARA
(H) ALGORITHMS FOR THE PROGRAMMING OF DECISION TABLES AND
FOR TRANSLATION BY COMPUTER
INFORMACIO ELEKTRONIKA, 8(2)
1973, 92-98.
THE AUTHOR OF THE ARTICLE PRESENTS THE TYPES OF DECISION

TABLES AND SOME ALGORITHMS FOR PREPARING THE PROGRAMS FROM THE FORMER. FINALLY, HE COMPARES VARIOUS ALGORITHMS FROM THE POINT OF VIEW OF THEIR MAIN FUNCTIONS.

- LAF71 LA FLEUR, T.G.
DECISION TABLES ~ A TOOL FOR DOCUMENTING LOGICAL CONDITION RELATIONSHIPS
SIGPLAN NOTICES, 6(8)
SEPTEMBER 1971, 9-12.
THIS PAPER SUPPLIES AN UNINTERESTING INTRODUCTION TO THE DECISION TABLE TECHNIQUE.
- LANG66 LANGENWALTER, D.F.
DECISION TABLES AS A TOOL
IDEAS FOR MANAGEMENT (PAPERS PRESENTED AT THE 1966 INTERNATIONAL SYSTEMS MEETING)
1966, 146-157.
THIS PAPER IS MAINLY DEVOTED TO DEMONSTRATING THE ADVANTAGES OF DECISION TABLES. EXAMPLES ARE TAKEN FROM THE US TAX DEPARTMENT AND THE US AIR FORCE PERFORMANCE REVIEW PROCEDURES. THE SELECTION OF AUDIO STAGE MODULES FOR A PORTABLE RADIO IS SHOWN IN DECISION TABLE FORM, IN FLOWCHART FORM AND AS A SERIES OF FORTRAN STATEMENTS FOR PURPOSES OF COMPARISON. OTHER APPLICATIONS ARE BRIEFLY MENTIONED.
- LANG68 LANGENWALTER, D.F. CLARK, R.W.
DECISION TABLES
IDEAS FOR MANAGEMENT (PAPERS PRESENTED AT THE 1968 INTERNATIONAL SYSTEMS MEETING)
1968, 87-102.
IN THIS ARTICLE, THE DECISION TABLE TECHNIQUE IS PRESENTED AS ANOTHER METHOD OF FORMATTING COMPLEX PROGRAM LOGIC. THE ADVANTAGES OF DECISION TABLE PROGRAMMING ARE DISCUSSED. IN AN APPENDIX, A DETAILED DESCRIPTION OF DECISION TABLE APPLICATIONS IS GIVEN.
- LANG69 LANGENWALTER, D.F.
THE USE OF DECISION TABLES TO SOLVE BUSINESS PROBLEMS
IDEAS FOR MANAGEMENT (PAPERS PRESENTED AT THE 1969 INTERNATIONAL SYSTEMS MEETING)
1969.
IN THIS ARTICLE, IT IS ARGUED THAT DECISION TABLES, COMBINED WITH THE ALGEBRAIC, FILE MANAGEMENT, REPORT WRITER, AND CONTROL STATEMENTS FOUND IN COMPUTER LANGUAGES, BECOME A POWERFUL TOOL FOR ANALYZING AND SOLVING BUSINESS PROBLEMS. SOME ELEMENTARY EXAMPLES ARE GIVEN.
- LAR66 LARSEN, R.P.
DATA FILTERING APPLIED TO INFORMATION STORAGE AND RETRIEVAL APPLICATIONS
CACM, 9(11)
NOVEMBER 1966, 785-789 AND 793.
THIS PAPER DISCUSSES IN GENERALIZED TERMS A CONCEPT OF DATA FILTERING UNDERLYING THE SYSTEM DESIGN OF AN INFORMATION STORAGE AND RETRIEVAL SYSTEM CALLED ABACUS (AIR BATTLE ANALYSIS CENTER UTILITY SYSTEM). THE INTEREST LIES IN THE NATURAL USE OF MIXED ENTRY DECISION TABLES TO PRESENT TO THE READER THE PROCESSING MECHANICS UNDERLYING BOTH THE DATUM RETRIEVAL FUNCTION AND DATA STRING CONVERSION.
- LAR77 LARSON, L.E.
HIGH LEVEL LANGUAGE DECISION TABLE DECOMPILER
IBM TDB, 20(6)
NOVEMBER 1977, 2194-2195.
DESCRIBES AN APL PROCESS FOR CONVERTING HIGH LEVEL LANGUAGE (ACTUALLY PL/I AND APL ITSELF) PROGRAMS TO DECISION TABLES.
- LAU74 LAUER, H.
FORMALISMEN ZUR AUFLÖSUNG VON ENTSCHEIDUNGSTABELLEN
(G) FORMAL SYSTEMS TO SPLIT DECISION TABLES
ANGEWANDTE INFORMATIK, 16(10)
OCTOBER 1974, 437-443.
DECISION TABLES OFTEN EXCEED THE GIVEN FORMATS. TABLES OF THIS KIND HAVE TO BE SPLIT INTO SUB-TABLES. THE PROBLEMS ARISING CAN BE TACKLED BY MEANS OF THE FORMAL

- SYSTEMS DESCRIBED.
- LEC66 LECOINT, R.
LES TABLES LOGIQUES OU TABLES DE DECISION
(F) LOGIC OR DECISION TABLES
REVUE DE LA MECANOGRAPHIE, (220)
MARCH 1966, 193-194.
DECISION TABLES ENABLE THE ANALYST TO PRESENT HIS RESULTS
IN A SIMPLE AND DEFINITIVE WAY. THIS ARTICLE DISCUSSES
THE BASIC NOTIONS, THE SCHEMATIC REPRESENTATION, THE
DIFFERENT TYPES AND COMBINATION OF TABLES AND THE
ADVANTAGES GAINED BY USING DECISION TABLES.
- LEC74 LECKEBUSCH, N.
ENTSCHEIDUNGSTABELLENTECHNIK BEI DER STADTSPARKASSE KOELN
(G) THE DECISION TABLE TECHNIQUE AT THE STADTSPARKASSE KOELN
DIE COMPUTER-ZEITUNG, (10)
MAY 1974.
- LEM71 LEMOINE, D.J.
AN AUTOMATIC DECISION-LOGIC-TABLE PROCESSOR
SIGPLAN NOTICES, 6(8)
SEPTEMBER 1971, 40-45.
THE DECISION LOGIC TABLE PROGRAMMING LANGUAGE, DELTA,
DESCRIBED HERE, IS AN AUTOMATIC DECISION TABLE PROCES-
SOR, ELIMINATING THE NEED TO TRANSLATE THE INPUT INTO
SOME INTERMEDIATE SOURCE LANGUAGE. THE CODE GENERATED
IS NOT MACHINE LANGUAGE, BUT IS AN INTERNAL REPRESEN-
TATION OF THE OBJECT LANGUAGE PROGRAM. BOTH A SYNTAX ANA-
LYSIS AND THE GENERATION OF CODE ARE PERFORMED IN ONE
PASS.
- LES66A LESKINEN, L.E.
GENERAL PURPOSE SYSTEMS DESIGN AND PROGRAMMING TECHNIQUES
PROC. UNIVAC USERS ASSOCIATION
PHILADELPHIA, FALL 1966.
ALSO IN : ADT, 134-138.
IN THIS ARTICLE, DECISION TABLES ARE DISCUSSED AS A
VERY USEFUL FORM OF DOCUMENTATION AND COMMUNICATION
DURING THE SUCCESSIVE DESIGN PHASES. A PRACTICAL EXAMPLE
IS PROVIDED.
- LES66B LESKINEN, L.E.
PROGRAMMING IN A MODULAR FORM WITH OR WITHOUT DECISION
TABLES
PROC. UNIVAC USERS ASSOCIATION
PHILADELPHIA, FALL 1966.
ALSO IN : ADT, 168-170.
PROGRAMMING IN A MODULAR FORM IS DISCUSSED. IT IS SHOWN
BY MEANS OF A SIMPLE EXAMPLE THAT DECISION TABLES LEND
THEIRSELF TO THIS METHOD OF PROGRAMMING.
- LEW70 LEWIS, B.N.
DECISION LOGIC TABLES FOR ALGORITHMS AND LOGICAL TREES
CAS OCCASIONAL PAPER NO. 12
CIVIL SERVICE DEPARTMENT
LONDON, 1970, 23 PP.
FLOWCHARTS AND DECISION TABLES ARE COMPARED FROM THE
POINT OF VIEW OF PSYCHOLOGY. IT IS ARGUED THAT
DECISION TABLES ARE MUCH EASIER TO MANIPULATE.
- LEW75 LEW, A.
OPTIMAL CONVERSION OF EXTENDED-ENTRY DECISON TABLES
TECHN. REPORT AEO-21R, DEPT. OF INF. AND COMP. SCI.,
UNIVERSITY OF HAWAII, JULY 1975.
- LEW76 LEW, A. TAMANAH, D.
DECISION TABLE PROGRAMMING AND RELIABILITY
PROC. 2ND INT. CONF. ON SOFTWARE ENGINEERING
SAN FRANCISCO, 13-15. OCTOBER 1976, 345-349.
THE USE OF DECISION TABLES AS PROGRAMMING AIDS AND RELA-
TED RELIABILITY QUESTIONS ARE DISCUSSED. IT IS SUGGESTED
THAT DECISION TABLE PROGRAMMING IS WORTH MORE SERIOUS
CONSIDERATION FOR GENERAL PURPOSE COMPUTING.
- LEW78 LEW, A.
OPTIMAL CONVERSION OF EXTENDED-ENTRY DECISION TABLES WITH
GENERAL COST CRITERIA
CACM, 21(4)

APRIL 1978, 269-279.

A GENERAL DYNAMIC PROGRAMMING ALGORITHM FOR CONVERTING LIMITED, EXTENDED OR MIXED ENTRY DECISION TABLES TO OPTIMAL DECISION TREES IS PRESENTED WHICH CAN TAKE INTO ACCOUNT RULE FREQUENCIES OR PROBABILITIES, MINIMUM TIME AND/OR SPACE COST CRITERIA, COMMON ACTION SETS, COMPRESSED RULES AND ELSE RULES, SEQUENCING CONSTRAINTS ON CONDITION TESTS, EXCLUDABLE COMBINATIONS OF CONDITIONS, CERTAIN AMBIGUITIES AND INTERRUPTED RULE MASTERS.

- LEW80 LEW, A.
ON THE EMULATION OF FLOWCHARTS BY DECISION TABLES
UNIV. OF HAWAII AT MANOA, TECHNICAL REPORT NO. ARO-41R
1980, 35 PP.
ANY FLOWCHART CAN BE EMULATED BY A DECISION TABLE, WHOSE 'COMPLEXITY' DEPENDS ON THAT OF THE FLOWCHART. HOWEVER, IT MAY BE NECESSARY TO INTRODUCE A NEW CONTROL VARIABLE WITH ASSOCIATED TESTS AND SETS, OR TO PERMIT CHANGES IN EXECUTION SEQUENCES PROVIDED ACTION-TEST INDEPENDENCE HOLDS. TWO MEASURES OF DECISION TABLE COMPLEXITY ARE DISCUSSED AND INTERRELATED. FINALLY, CONDITIONS AND PROCEDURES FOR REDUCING COMPLEXITY ARE PRESENTED.
- LEW81 LEW, A. TAMANAH, D.
A NOTE ON PROVING THE CORRECTNESS OF A DECISION TABLE PROGRAM
UNIV. OF HAWAII AT MANOA, RESEARCH REPORT.
1981, 11PP.
THE 'ASSERTION' METHOD IS USED TO PROVE THE CORRECTNESS OF A DECISION TABLE PROGRAM FOR BUBBLE SORTING AN ARRAY.
- LIE75 LIEBSCH, R.
ANWENDUNG SCHALTALGEBRAISCHER METHODEN IN DER
ENTSCHEIDUNGSTABELLENTECHNIK
(G) THE APPLICATION OF SWITCHING ALGEBRA METHODS IN THE
DECISION TABLE TECHNIQUE
RTDV, 12(3)
1975, 25-31.
IN THIS ARTICLE, A FORMAL ALGORITHM IS PROPOSED FOR THE CONVERSION OF DECISION TABLES INTO A SEQUENTIAL PROCESS. IT IS BASED ON SOME SWITCHING ALGEBRA METHODS.
- LOB65 LOBEL, J.
STRUCTURE TABLES - A NEW TOOL FOR DECISION MAKING
AUTOMATION, 12(11)
NOVEMBER 1965, 109-114.
DECISION STRUCTURE TABLES ARE PRESENTED. IT IS ARGUED THAT THE TABLES OFFER A WAY TO RELATE TYPICAL CONDITIONS TO APPROPRIATE ACTIONS. COMPARISON OF PROBLEM PARAMETERS TO THE CONDITIONS IN THE TABLES LEADS STEP BY STEP TO THE SUITABLE ACTION DECISION.
- LOB69 LOBEL, J.
DECISION TABLE LOGIC
AUTOMATION, 16(11)
NOVEMBER 1969, 78-82.
AFTER A CONCISE INTRODUCTION TO DECISION TABLE FORMAT AND LOGIC, THE AUTHOR LOOKS FOR INDUSTRIAL APPLICATIONS OF THE DECISION TABLE TECHNIQUE. HE SHOWS BY MEANS OF AN EXAMPLE HOW DECISION TABLES CAN BE USED TO EXPRESS THE LOGIC OF A DECISION PROBLEM.
- LOM61 LOMBARDI, L.A.
INEXPENSIVE PUNCHED CARD EQUIPMENT
JOURNAL OF MACHINE ACCOUNTING, 12(8)
AUGUST 1961, 11-18.
THE AUTHOR REPORTS ELSEWHERE THAT IN THIS ARTICLE HE "USED DECISION TABLES QUITE EFFECTIVELY TO LOGICALLY DESIGN A HARDWARE TWO-WAY-MERGING RECORD SORTER".
- LOM62 LOMBARDI, L.A.
ON TABLE OPERATING ALGORITHMS
2ND IFP CONGRESS
AUGUST 1962.
- LOM64 LOMBARDI, L.A.
A GENERAL BUSINESS-ORIENTED LANGUAGE BASED ON DECISION EXPRESSIONS
CACM, 7(2)

FEBRUARY 1964, 104-111.

THIS PAPER WAS PRESENTED AT A WORKING CONFERENCE ON MECHANICAL LANGUAGE STRUCTURES IN AUGUST 1963 AND PUTS FORWARD AN APPROACH WHICH STRESSES THE STRUCTURAL ANALYSIS OF THE CLASS OF PROCESSES TO BE REPRESENTED. IT USES A NON-PROCEDURAL REPRESENTATION OF PROCESSES AS SETS (TABLES) OF RELATIONS BETWEEN DATA AND RESULTS INSTEAD OF PROCEDURE DESCRIPTIONS. PROCESSING PROCEDURES WHICH LEAD TO VERY LARGE DECISION TABLES IN IMPORTANT APPLICATIONS SHOULD BE HANDLED BY STANDARD SERVICE ROUTINES AS THEY ARE NOT THE CONCERN OF THE BUSINESS PROGRAMMER. THE APPROACH IS SAID TO UNDERLY THE EXPERIMENTAL LANGUAGE N1 USED ON THE OLIVETTI ELEA 9003. THE APPROACH IS CLAIMED TO HAVE THE POTENTIAL TO OVERCOME THE DEFICIENCIES OF OTHER BUSINESS-ORIENTED LANGUAGES ANALYZED IN THE PAPER AND MEETS IN FULL THE CODASYL REQUIREMENTS FOR SUCH LANGUAGES.

- LON72 LONDON, K.R.
DECISION TABLES
AUERBACH PUBL.,
PRINCETON (PHILADELPHIA), 1972, 205 PP.
THIS BOOK LOOKS IN DETAIL AT DECISION TABLES AND THEIR CURRENT APPLICATIONS. SUBJECTS TREATED : CONSTRUCTION OF DECISION TABLES, INITIALIZATION, RECURSION, PARSING, USE OF DECISION TABLES IN SYSTEMS ANALYSIS AND PROGRAMMING. DETAILED EXAMPLES AND A THOROUGH BIBLIOGRAPHY COMPLETE THE EXTENSIVE TREATMENT.
- LOW69 LOW, D.W.
PROGRAM/TEXT GENERATION : A DECISION TABLE APPROACH
IBM LOS ANGELES SCIENTIFIC CENTER TECHNICAL REPORT 320-2633
LOS ANGELES, 1969.
THE FUNCTIONS AND ORGANIZATION OF THE EDITOR AS WELL AS THE DECISION TABLE TECHNOLOGY, AS THEY ARE USED IN PROGRAMMING BY QUESTIONNAIRE (SEE LOW73), ARE FULLY DESCRIBED.
- LOW73 LOW, D.W.
PROGRAMMING BY QUESTIONNAIRE : AN EFFECTIVE WAY TO USE DECISION TABLES
CACM, 16(5)
MAY 1973, 282-286.
PROGRAMMING BY QUESTIONNAIRE COMBINES ASPECTS OF DECISION TABLE PROGRAMMING AND GENERAL PURPOSE PROGRAMMING BY USING DECISION TABLES TO CONSTRUCT AN APPLICATION PROGRAM THROUGH THE SELECTION OF CERTAIN SOURCE STATEMENTS FROM A PREDEFINED FILE. IT IS PROPOSED THAT SUCH PROGRAMMING IS A USEFUL COMPROMISE BETWEEN GENERAL AND SPECIAL PURPOSE PROGRAMMING. THE ELEMENTS OF THE APPROACH ARE DISCUSSED AND AN EXISTING APPLICATION DESCRIBED.
- LUD68A LUDWIG, H.R.
DECISION TABLES AND SIMULATION
PH. D. DISSERTATION
SCHOOL OF ENGINEERING, NORTHWESTERN UNIVERSITY
1968.
- LUD68B LUDWIG, H.R.
SIMULATION WITH DECISION TABLES
JOURNAL OF DATA MANAGEMENT, 6(1)
6 JANUARY 1968, 20-27.
ALSO IN : ADT, 115-125.
THIS ARTICLE DISCUSSES THE USE OF DECISION TABLES FOR SIMULATING QUEUING SYSTEMS. IT HANDLES THE DEVELOPMENT OF HIGHER LEVEL LANGUAGES FOR SIMULATION. A KNOWLEDGE OF FORTRAN IS REQUIRED. THREE MODELS ARE REVIEWED. IT ALSO DISCUSSES HOW TO WORK WITH DECISION TABLES.
- MAE71 MAERZ, W.
DER EINSATZ VON ENTSCHEIDUNGSTABELLEN IN DER PROGRAMMIERUNG (G) USING DECISION TABLES IN PROGRAMMING
UNPUBLISHED PAPER
JANUARY 1971.
THIS PAPER PRESENTS A CONCISE INTRODUCTION TO THE DECISION TABLE TECHNIQUE. A DETAILED DETAB/65 EXAMPLE IS PROVIDED.

- MAE76A MAES, R.
ENKELE RECENTE ONTWIKKELINGEN IN DE STUDIE VAN DE BESLIS-
SINGSTABELLEN
(D) SOME RECENT DEVELOPMENTS OF THE DECISION TABLE
TECHNIQUE
BELEIDSDINFORMATICA-BERICHTEN, 2(3)
OCTOBER 1976, 4-19.
THE USE OF DECISION TABLES DURING SYSTEMS ANALYSIS AND
SYSTEMS DESIGN AND AS A MANAGEMENT TOOL ARE OUTLINED.
SOME IMPLICATIONS FOR THE BASIC DECISION TABLE FORMAT ARE
INDICATED.
- MAE76B MAES, R.
EEN WEGWIJZER DOORHEEN DE RECENTE BESLISSINGSTABELLENLITE-
RATUUR
(D) A SURVEY OF THE RECENT LITERATURE ON DECISION TABLES
BELEIDSDINFORMATICA-BERICHTEN, 2(3)
OCTOBER 1976, 20-29.
REPRINTED AS MAE77C.
- MAE76C MAES, R.
BESLISSINGSTABELLEN : EEN GREEP UIT DE PRAKTIJK
(D) DECISION TABLES : SOME PRACTICAL APPLICATIONS
BELEIDSDINFORMATICA-BERICHTEN, 2(3)
OCTOBER 1976, 30-36.
THIS PAPER SUCCESSIVELY DEALS WITH A CONCISE DESCRIPTION
OF THREE PREPROCESSORS (AGENTA, DETAB/GT AND VORELLE),
THE APPLICATION OF DECISION TABLES IN ANALYZING MANUAL
PROCEDURES AND AN INTERACTIVE SYSTEM UNDER DEVELOPMENT
FOR THE CONSTRUCTION AND MANIPULATION OF DECISION TABLES.
- MAE76D MAES, R.
DECISION TABLES : AN ANNOTATED BIBLIOGRAPHY
K.U.LEUVEN, DEPT. OF APPLIED ECONOMICS
LEUVEN, 1976.
THIS ANNOTATED BIBLIOGRAPHY COVERS 495 ENTRIES. AN
AUTHORS' LIST AND A KWIC-INDEX ARE ADDED.
- MAE77A MAES, R.
BESLISSINGSTABELLEN EN BESLUITVORMING : MOGELIJKHEDEN EN
PERSPEKIEVEN
(D) DECISON TABLES AND DECISION MAKING : POTENTIALITIES AND
PERSPECTIVES
BEDRIJFSECONOMISCHE VERHANDELING 7701, DEPT. OF APPLIED
ECONOMICS, K.U.LEUVEN
LEUVEN, 1977 24 PP.
THE DECISION TABLE IS INVESTIGATED AS A MANAGEMENT TOOL.
DECISION SITUATIONS ARE FITTED IN TWO DISTINCT CATE-
GORIES : A PRIORI STRUCTURED AND A PRIORI NON STRUCTURED
SITUATIONS. THE RELATIVE IMPORTANCE OF THE DECISION
TABLE TECHNIQUE FOR BOTH CATEGORIES IS OUTLINED.
- MAE77B MAES, R.
AN ALGORITHMIC APPROACH TO THE CONVERSION OF DECISION GRID
CHARTS INTO COMPRESSED DECISION TABLES
RESEARCH REPORT 7712, DEPT. OF APPLIED ECONOMICS,
K.U.LEUVEN, 1977, 26 PP.
THE DECISION GRID CHART IS INVESTIGATED AS AN INTERMEDI-
ATE FORM IN CONSTRUCTING DECISION TABLES. A VERY NATURAL
EXTENSION TO THE BASIC FORMAT IS PROPOSED. FIVE ALTERNA-
TIVE PROCEDURES FOR THE CONVERSION OF DECISION GRID
CHARTS INTO COMPRESSED DECISION TABLES ARE GIVEN.
- MAE77C MAES, R.
RECETE EVOLUTIES VAN DE BESLISSINGSTABELLENTECHNIEK : EEN
LITERATUROVERZICHT
(D) RECENT DEVELOPMENTS OF THE DECISION TABLE TECHNIQUE : A
SURVEY OF THE LITERATURE
INFORMATIE, 19(2)
FEBRUARY 1977, 74-78.
RECENT PUBLICATIONS ON THE DECISION TABLE TECHNIQUE ARE
SURVEYED. IT IS ARGUED THAT THE USE OF DECISION TABLES
IN SYSTEMS ANALYSIS AND DESIGN IS STRESSED AT THE EXPENSE
OF FURTHER RESEARCH ON CONVERSION ALGORITHMS.
- MAE78A MAES, R.
ON THE REPRESENTATION OF PROGRAM STURCTURES BY DECISION
TABLES : A CRITICAL ASSESSMENT
THE COMPUTER JOURNAL, 21(4)

NOVEMBER 1978, 290-295.
ALSO AS : RESEARCH REPORT 7707, DEPT. OF APPLIED ECONOMICS
K.U.LEUVEN, 1977, 20 PP.
RECENT PUBLICATIONS SUGGEST THE USE OF DECISION TABLES IN
ANALYZING CONVENTIONAL COMPUTER PROGRAMS. IN THIS PAPER,
IT IS ARGUED THAT THE CLASSICAL DECISION TABLE FORMAT IS
NOT WELL SUITED TO REPRESENT FLOWCHART-LIKE PROGRAMS.
SOME ALTERNATIVE DECISION TABLE CONVENTIONS ARE INVESTI-
GATED. FINALLY, A VERY BASIC DISTINCTION IS MADE BETWEEN
THE USE OF DECISION TABLES IN THE PROBLEM STATEMENT PHASE
AND IN PROGRAMMING.

- MAE78B MAES, R.
DE BESLISSINGSTABEL EN HET STRUKTUREREN VAN PROCEDURE-BE-
SLISSINGEN
(D) DECISION TABLES AND STRUCTURING PROCEDURAL DECISIONS
PAPER PRESENTED AT THE TWEEJAARLIJKSE STUDIEDAG VAN DE
VERENIGING VOOR ECONOMIE
BRUSSELS, 29TH SEPT. 1978, 26 PP.
RECENT DEVELOPMENTS IN THE STUDY OF DECISION TABLES ARE
COMPARED WITH THE THEORIES UNDERLYING GENERAL DECISION
SCIENCES. IT IS SHOWN HOW DECISION TABLES CAN CONTRIBUTE
TO THE STRUCTURING OF PROCEDURAL DECISIONS. FURTHER,
DECISION TABLES ARE DEMONSTRATED TO BE 'META-INDICATIONS'.
- MAE81A MAES, R. VANTHIENEN, J.
PRODEMO : PROCEDURAL DECISION MODELING THROUGH THE USE OF
DECISION TABLES
BEDRIJFSECONOMISCHE VERHANDELING 8101, DEPT. OF APPLIED
ECONOMICS, K.U.LEUVEN
LEUVEN, 1981, 36 PP.
IN THIS REPORT, THE PRODEMO SYSTEM FOR CONSTRUCTING,
MANIPULATING AND USING DECISION TABLES IS EXTENSIVELY
PRESENTED. FURTHER, IT IS DEMONSTRATED BY A SIMPLE
EXAMPLE HOW DECISION TABLES CAN CONTRIBUTE TO THE QUALI-
TY OF EVERYDAY DECISION MAKING.
- MAE81B MAES, R.
BIJDRAEGE TOT EEN KRITISCHE HERWAARDERING VAN DE BESLISSINGS-
TABELLENTECHNIK
(D) CONTRIBUTION TO A CRITICAL EVALUATION OF THE DECISION
TABLE TECHNIQUE
K.U.LEUVEN, FAC. OF APPLIED SCIENCES
DOCTORAL THESIS
LEUVEN, 1981, 397 PP.
THE SUCCESSIVE CHAPTERS OF THIS DOCTORAL THESIS DEAL WITH
THEORETICAL FOUNDATIONS, DECISION TABLES AS A PROGRAMMING
TECHNIQUE, ALGORITHMS FOR THE COMPUTER-ASSISTED MANIPULA-
TION OF DECISION TABLES AND THE USE OF DECISION TABLES
DURING THE WHOLE SYSTEMS LIFE CYCLE. IT IS SAID THAT
DECISION TABLES, WHEN USED AS A PROGRAMMING TECHNIQUE,
HAVE A DEFINITE YET LIMITED APPLICATION. MUCH MORE IMPOR-
TANT IS THEIR USE DURING THE PRELIMINARY, USER-ORIENTED
DEVELOPMENT PHASES. NUMEROUS ALGORITHMS, ENABLING THE
INTRODUCTION OF THE COMPUTER IN THE CONSTRUCTION OF DECI-
SION TABLES, ARE PRESENTED.
- MAE81C MAES, R. VANTHIENEN, J.
PRODEMO, COMPUTER-ONDERSTEUND ONTWERP EN GEBRUIK VAN BESLIS-
SINGSTABELLEN
(D) PRODEMO, COMPUTER-SUPPORTED DESIGN AND USE OF DECISION
TABLES
TIJDSCHRIFT VOOR ECONOMIE EN MANAGEMENT, 26(3)
1981, 301-336.
A LESS TECHNICAL VERSION OF MAE81A.
- MAE81D MAES, R. VANTHIENEN, J.
PRAKTIJKERVARINGEN MET BESLISSINGSTABELLEN EN PRODEMO
(D) PRACTICAL EXPERIENCES WITH DECISION TABLES AND PRODEMO
TIJDSCHRIFT VOOR ECONOMIE EN MANAGEMENT, 26(3)
1981, 337-348.
THREE REAL-LIFE CASE STUDIES AND SOME GENERAL CONCLUSIONS
ON THE USE OF DECISION TABLES FOR ANALYZING PRESCRIPTIONS,
LAWS, REGULATIONS ETC. ARE PRESENTED.
- MAE81E MAES, R. VANTHIENEN, J. VERHELST, M.
PROCEDURAL DECISION SUPPORT THROUGH THE USE OF PRODEMO
PROC. 2ND INT. CONF. ON INFORMATION SYSTEMS
CAMBRIDGE, MASS., DEC. 7-9, 1981, 135-152.

THE PRODEMO SYSTEM (FOR A DESCRIPTION, SEE MAE81A) IS PRESENTED AS THE CORE OF A DECISION SUPPORT SYSTEM. REAL-LIFE CASE STUDIES, CARRIED OUT THROUGH THE USE OF PRODEMO, ARE BRIEFLY MENTIONED.

- MAE81F MAES, R.
ON MINIMIZING DECISION GRID CHARTS
UNIV. OF AMSTERDAM, DEPT. OF ECONOMICS
RESEARCH MEMORANDUM NO. 8124
1981, 15 PP.
THE DECISON GRID CHART IS INVESTIGATED AS AN INTERMEDIATE FORM WHEN ANALYZING PROCEDURAL DECISIONS VIA DECISION TABLES. A NEW ALGORITHM IS PROPOSED WHICH ALLOWS MINIMIZING THE NUMBER OF RULES OF A DECISION GRID CHART. IT IS SHOWN IN THAT WAY IMPLICATIONS BETWEEN CONDITIONS CAN BE DEALT WITH.
- MAROO MARTIN, M.
ENTSCHEIDUNGSTABELLEN - EINE METHODE ZUR ANALYSE UND PROGRAMMIERUNG VON SYSTEMATISCHER ABLAUFEN
(G) DECISION TABLES - A METHOD FOR ANALYSIS AND PROGRAMMING
VDI - BILDUNGSWERK NO. 1311
PLACE OF ISSUE AND DATE UNKNOWN.
THIS PAPER GIVES A BRIEF PRESENTATION OF THE DECISION TABLE TECHNIQUE.
- MAR72A MARKLOF, E. STERZINGER, U.
DIE ENTSCHEIDUNGSTABELLE IN NEUER PERSPEKTIVE
(G) DECISION TABLES IN NEW PERSPECTIVE
ANGEWANDTE INFORMATIK, 14(10)
OCTOBER 1972, 480-482.
IN ADDITION TO THE USUAL CONDITION- AND ACTION-SECTION OF A DECISION TABLE, THE AUTHORS DISTINGUISH A SUPPLEMENTARY PART-SECTION. THIS ALLOWS AN EXTENSION OF THE APPLICATION FIELD. SOME EXPLANATORY EXAMPLES ARE PROVIDED.
- MAR72B MARSHALL, D.R.T.
REMARK ON ALGORITHM 394 : DECISION TABLE TRANSLATION
CACM, 15(12)
DECEMBER 1972, 1074.
REMARK REFERRING TO THE ALGORITHM PROPOSED IN DIA70.
- MAR72C MARTIN, P.
ENTSCHEIDUNGSTABELLEN ALS FUEHRUNGSHILFEN
(G) DECISION TABLES AS TOOLS FOR MANAGEMENT
TRUPPENPRAXIS, (1)
1972, 1-9.
THIS PAPER GIVES A BRIEF INTRODUCTION TO THE DECISION TABLE TECHNIQUE.
A DETAILED PRACTICAL EXAMPLE IS PRESENTED.
- MAR75 MARTZLOFF, C.
DE LA PROGRAMMATION STRUCTUREE AU LANGUAGE DE CONCEPTION
(F) STRUCTURED PROGRAMMING AND CONCEPTUAL LANGUAGES
INFORMATIQUE ET GESTION, (67)
MAY 1975, 45-48.
IN THIS ARTICLE, THE DECISION TABLE IS PRESENTED AS "THE ONLY TOOL FOR PROBLEM FORMULATION PRESERVING THE ORGANIZATIONAL HIERARCHY".
- MAR78 MARTELLI, A. MONTANARI, U.
OPTIMIZING DECISION TREES THROUGH HEURISTICALLY GUIDED SEARCH
CACM, 21(12)
DECEMBER 1978, 1025-1039.
THIS PAPER DEALS WITH THE CONVERSION OF DECISION TABLES INTO (SEMI-)OPTIMAL PROGRAMS. A HEURISTICALLY GUIDED TOP-DOWN SEARCH ALGORITHM IS PROPOSED, WHICH LIKE DYNAMIC PROGRAMMING, RECOGNIZES IDENTICAL SUBPROBLEMS BUT WHICH CAN ALSO BE USED TO FIND QUASIOPTIMAL SOLUTIONS. THE METHOD INTRODUCED IN THIS PAPER IS CLAIMED TO COMBINE THE POSITIVE ASPECTS OF BOTH BRANCH-AND-BOUND TECHNIQUES AND DYNAMIC PROGRAMMING. COMPRESSED TABLES WITH A LARGER NUMBER OF VARIABLES CAN BE HANDLED WITHOUT DERIVING EXPANDED TABLES FIRST.
- MAT71 MATHIA, P.
(J) TRUTH AND DECISION TABLES
THE ISRAEL INSTITUTE OF PRODUCTIVITY

TEL AVIV, 1971.

A TEST ALGORITHM ALG-TQ IS PROPOSED WITH THE INTENTION TO DEVELOP A STANDARD ALGORITHM THAT WILL AUTOMATICALLY MAP A GIVEN TREE STRUCTURE (BRANCH-OFF POINTS) INTO A COMPUTERIZED DECISION TABLE.

- MAU73 MAURER, G.
EDV-CASE-STUDY : BEWAG BERLIN, VERSUCH MIT
ENTSCHEIDUNGSTABELLEN
(G) EDP CASE STUDY BY MEANS OF DECISION TABLES
ONLINE, 11(3)
1973, 128-129.
THE AUTHOR DESCRIBES THE INTRODUCTION OF THE DETAB/GT
PREPROCESSOR IN AN EDP ENVIRONMENT.
- MAY73 MAYEDA, W.
DESIGN GRAPHS FOR INFORMATION PROCESS
16TH MIDWEST SYMP. ON CIRCUIT THEORY
VOL. 2, PAP. XII.3
APRIL 12-13, 1973, 24 PP.
A PROCEDURE IS DEVELOPED TO DESIGN A REASONABLE TESTING
PROCESS TO IDENTIFY INFORMATION FROM A DECISION TABLE.
THIS PROBLEM IS ATTACKED BY THE USE OF GENERAL GRAPH
THEORY.
- MAY75 MAYER, H.C.
DECISION TABLES IN EDUCATION
PROC. 13TH ANN. CONV. 26-29, ASSOC. EDUC. DATA SYSTEMS
WASHINGTON D.C., 1973.
- MAZ74 MAZEL, B.
(C) SYSTEM ASPECTS OF DECISION MAKING PROCESSES
INORGA, 8(5)
1974, 223-228.
THE ADVANTAGES OF USING DECISION TABLE PRE-PROCESSORS
ARE OUTLINED.
- MBP74 MATHEMATISCHER BERATUNGS- UND PROGRAMMIERUNGSDIENST
VORELLE - VORUEBERSETZER FUER ENTSCHEIDUNGSTABELLEN.
COBOL-VERSION
(G) VORELLE - A PREPROCESSOR FOR DECISION TABLES. COBOL
VERSION
DORTMUND, 1974.
- MCD68 MC DANIEL, H.
AN INTRODUCTION TO DECISION LOGIC TABLES
JOHN WILEY
NEW YORK, 1968, 96 PP.
THIS BOOK SETS OUT TO GIVE AN INTRODUCTION MAINLY THROUGH
EXAMPLES. IT INCLUDES AN INTRODUCTION TO PROGRAM
CONVERSION, SOME EXERCISES AND A BRIEF BIBLIOGRAPHY.
- MCD70A MC DANIEL, H.
APPLICATIONS OF DECISION TABLES
BRANDON/SYSTEMS PRESS
PRINCETON, 1970, 226 PP.
THIS BOOK IS A COLLECTION OF ARTICLES WHICH EXPLAINS THE
USE OF DECISION TABLES IN MANUFACTURING, FINANCE,
TRADING, GOVERNMENT, ENGINEERING, UTILITIES, AND
EDUCATION. THE DIFFERENT ARTICLES WERE PUBLISHED DURING
THE YEARS 1963-1968. IN THIS BIBLIOGRAPHY, THEY ARE
ANNOTATED SEPARATELY. A PARTIAL LIST OF SPECIFIC
APPLICATIONS OF DECISION TABLES IS GIVEN IN AN APPENDIX.
- MCD70B MC DANIEL, H.
DECISION TABLE SOFTWARE - A HANDBOOK
BRANDON/SYSTEMS PRESS
PRINCETON, 1970, 84 PP.
MORE THAN 30 PREPROCESSORS ARE DISCUSSED WITH RELATION TO
LANGUAGE AND HARDWARE FACILITIES, TABLE CONSTRAINTS, COST
PRICE, EXISTING MANUALS,...
- MCD70C MC DANIEL, H.
DECISION TABLES FOR EFFECTIVE PROCEDURES
IN : ADT, 145-149.
THREE EXAMPLES OF DECISION TABLES BEING USED FOR
PROCEDURES ARE GIVEN. THE FIRST TWO ARE FROM THE
ACCOUNTING DEPARTMENT OF A BUSINESS FIRM. THE THIRD IS
AN AIR FORCE PROCEDURE FOR SHIPPING CONTROLLED TEST

ITEMS.

- MEE74 MEERSMAN, E.
AUTOMATISERING VAN HET OPSTELLEN VAN BESLISSINGSTABELLEN
(D) THE AUTOMATION OF THE CONSTRUCTION OF DECISION TABLES
KUL, INSTITUUT VOOR TOEGEPASTE ECONOMISCHE WETENSCHAPPEN
LEUVEN, 1974, 2 PARTS.
THE AUTHOR TREATS THE CONSTRUCTION OF DECISION TABLES BY
MEANS OF THE INDIRECT METHOD AS PROPOSED BY VERHELST
(VER69A AND VER73). HE GOES ON TO DESCRIBE A FORTRAN-
PROGRAM THAT APPLIES THE METHOD TO AN INPUT OF SIMPLE
LOGICAL EQUATIONS. THE SECOND PART IS A REPRODUCTION OF
THE FORTRAN SOURCE PROGRAM.
- MEL74 MELEKIAN, N.
RATIONELLE PROGRAMMIERUNG MIT ENTSCHEIDUNGSTABELLENTECHNIK
UND DECTAT
(G) RATIONALIZED PROGRAMMING WITH THE DECISION TABLE
TECHNIQUE AND DECTAT
IBM-SEMINAR
SINDELFINGEN, JUNE 1972.
- MET72 METZNER, J.R.
DECISION TABLE PROGRAMMING LANGUAGES, GENERALIZATION AND
DESIGN
PH. D. THESIS, THE PENN. STATE UNIV.
PENNSYLVANIA, 1972.
- MET77 METZNER, J.R. BARNES, B.H.
DECISION TABLE LANGUAGES AND SYSTEMS
ACADEMIC PRESS
NEW YORK, 1977, 172 PP.
THIS MONOGRAPH CONSIDERS DECISION TABLES AT THREE
DISTINCT LEVELS : AT THE LOWEST LEVEL IS THE DECISION
TABLE ITSELF. AT THE SECOND LEVEL IS THE DECISION TABLE
LANGUAGE, OF WHICH DECISION TABLES ARE COMPONENTS. THE
THIRD LEVEL INCLUDES THE DECISION TABLE LANGUAGE, ITS
PROCESSOR OR TRANSLATOR, ITS USERS AND THE APPLICATION
AREA.
- MEYOO MEYER, H.I.
AUTOCODER DECISION TABLE ASSEMBLER
IBM, REF 7070-01.1.002.
DATE UNKNOWN.
- MEYG5 MEYER, H.I.
DECISION TABLES AS AN EXTENSION TO PROGRAMMING LANGUAGES
PROC. 1965 INTERNATIONAL DATA PROCESSING CONFERENCE
1965, 477-484.
ALSO IN : ADT, 162-167.
THIS PAPER SUGGESTS HOW LIMITED ENTRY DECISION TABLES
MAY BE INCORPORATED INTO HIGHER LEVEL LANGUAGE ; THE TWO
MAIN ILLUSTRATIONS RELATE TO THE USE OF AUTOCODER.
ASSEMBLY, TABLE LINKAGE AND DOCUMENTATION ARE BRIEFLY
DISCUSSED.
- MEY75 MEYHAK, H.
ENTSCHEIDUNGSTABELLENTECHNIK
(G) DECISION TABLE TECHNIQUE
SAUER-VERLAG
HEIDELBERG, 1975, 295 PP.
THIS INTRODUCTORY BOOK SUCCESSIVELY DEALS WITH DECISION
TABLE FUNDAMENTALS, FORMS, CONSTRUCTION, CONVERSION AND
APPLICATION FIELDS. A THOROUGH BIBLIOGRAPHY OF THE
(GERMAN) LITERATURE OF DECISION TABLES IS ADDED.
- MIL74 MILNER, D.A. TUGRUL, H.F.
SOME ASPECTS OF THE USE OF DECISION TABLES IN MANUFACTURING
SYSTEMS
PROC. INT. CONF. ON PRODUCTION TECHNOLOGY
MELBOURNE (AUSTR.), AUGUST 19-21, 1974, 56-64.
THE GENERAL CHARACTERISTICS, WRITING AND DEVELOPMENT OF
DECISION TABLES ARE DESCRIBED. DETAILS OF CASE STUDIES
FOR THE IMPLEMENTATION OF DECISION TABLE APPLICATIONS IN
MANUFACTURING SYSTEMS ARE INDICATED.
- MOE72 MOECKEL, M.
ERFAHRUNGEN BEIM EINSATZ VON ENTSCHEIDUNGSTABELLEN
(G) EXPERIENCES WITH DECISION TABLES

- RTDV, 3
1972, 16-19.
THE AUTHOR COMPARES DECISION TABLES WITH FLOWCHARTING
AND WITH THE NARRATIVE PROBLEM DESCRIPTION.
- MON62 MONTALBANO, M.
TABLES, FLOWCHARTS, AND PROGRAM LOGIC
IBM SYSTEMS JOURNAL, 1
SEPTEMBER 1962, 51-63.
DECISION TABLES ARE INTRODUCED WITH REFERENCE TO BUSINESS DATA PROCESSING. A METHOD OF VERIFYING BOTH THE COMPLETENESS AND CONSISTENCY OF A PROBLEM DESCRIPTION IS GIVEN. THE CONVERSION OF TABLES TO COMPUTER PROGRAMS IS CONSIDERED AND A TECHNIQUE OF OBTAINING A COMPUTER PROGRAM WHICH MINIMIZES THE BRANCHING REQUIREMENTS WITH RESPECT TO BOTH MEMORY AND EXECUTION TIME IS INCLUDED. PROGRAM DEBUGGING AND PROGRAM MODIFICATION ARE ALSO DISCUSSED.
- MON64 MONTALBANO, M.
EGLER'S PROCEDURE REFUTED
CACM, 7(1)
JANUARY 1964, 1.
IN THIS LETTER, MONTALBANO POINTS OUT THAT EGLER'S PROCEDURE (EGL63) DOES NOT ALWAYS SUCCEED IN DOING WHAT IS CLAIMED. HE GOES ON TO POINT OUT SOME OTHER DIFFICULTIES AND REFERS THE READER TO HIS OWN PAPER "TABLES, FLOWCHARTS AND PROGRAM LOGIC" (MON62).
- MON67 MONTALBANO, M.
DECISION TABLES AS A COMMUNICATION MEDIA
SYMPORIUM ON DECISION TABLES
USASI AD HOC X3.6.7/3.4.2D
ATLANTIC CITY, APRIL 1967.
- MON74 MONTALBANO, M.
DECISION TABLES
SCIENCE RESEARCH ASSOCIATES, INC.
CHICAGO, 1974, 186 PP.
THE OBJECTIVE OF THIS BOOK IS TO INTRODUCE THE READER TO THE DECISION TABLE AS A TOOL FOR PUTTING PROCEDURE DESCRIPTIONS IN COMPUTER READABLE FORM. SUBJECTS INCLUDE THE SETTING, STRUCTURE AND USE, NARRATIVE DESCRIPTION AND FLOWCHARTS, SIMPLE AND COMPOSITE RULES, COMPLETENESS AND CONSISTENCY, THE ALGEBRA AND GEOMETRIC OF LOGIC, DECISION TABLES AND FLOWCHARTS, DECISION TABLE TRANSLATORS, OPEN QUESTIONS AND OPINIONS. EXERCISES ARE PROVIDED.
- MOR65 MORGAN, J.J.
DECISION TABLES
MANAGEMENT SERVICES, 2(1)
JANUARY-FEBRUARY 1965, 13-18.
AN INTRODUCTION INTO THE BASICS OF DECISION TABLES. FIRST SOME RATHER SIMPLE SITUATIONS OF INVENTORY CONTROL ARE TREATED AS EXAMPLES, AND NEXT SOME ADDITIONAL CONDITIONS ARE INCLUDED IN THESE EXAMPLES. A FEW FURTHER APPLICATIONS ARE STATED, AS WELL AS FUNDAMENTAL REQUIREMENTS AND ADVANTAGES OF THE DECISION TABLES.
- MOR72 MORIYA, S. HIRAMATSU, K.
TABLE PROGRAM AND ITS CONVERSION TO COMPUTER PROGRAM - AN EXTENSION OF DECISION TABLE
INFORMATION PROCESSING IN JAPAN, (12)
1972, 51-57.
IN THIS ARTICLE, IT IS ARGUED THAT PROGRAM SEGMENTS WITH LOOPS OR WITH ACTIONS BETWEEN CONDITIONS CANNOT BE SATISFACTORILY DESCRIBED BY DECISION TABLES. THESE DEFECTS OF DECISION TABLES ARE OVERCOME BY ADDING TWO FUNCTIONS. ONE IS TO ALLOW PROGRAM LOOPS BY ADDING STATEMENT LABELS TO BOTH CONDITIONS AND ACTIONS OF DECISION TABLES. THE OTHER IS TO DEFINE TWO INTERPRETATIONS, "DON'T CARE" AND "NEGLECT", FOR BLANKS IN CONDITION ENTRY TO AVOID THE AMBIGUITY INTRODUCED BY ADDING THE STATEMENT LABELS. THE IMPROVED DECISION TABLE, CALLED "TABLE PROGRAM", BECOMES A PROGRAMMING LANGUAGE WHICH DESCRIBES AN ALGORITHM IN VERY SIMPLE FORM.
- MOR75 MORGAN, B.T. KAMAT, S.

DYNAMIC PROGRAMMING WITH DECISION TABLES
BULL. OPER. RES. SOC. AM., ORSA/TIMS NAT. MEETING,
VOL. 23, SUPPL. 1.
CHICAGO, 30 APRIL-2 MAY 1975, SPRING 1975, B/107.
THE RELATIVE MERITS OF USING DECISION TABLES FOR
DESCRIBING DYNAMIC PROGRAMMING ALGORITHMS AND GENERATING
THEIR CODE IS DISCUSSED. TWO ELABORATED EXAMPLES ARE
GIVEN. THE TAB40 PREPROCESSOR IS USED FOR THE GENERATING
PROCESS.

- MUL75 MULLER, G. VANDEWYHAEGHE, C. BEUZART, A.
CARTENTIER, A. DUBOST, C. JOLY, H.
(F) PATIENT MONITORING BY COMPUTER AFTER OPEN HEART SURGERY
USING SUGGESTION TABLES
PROC. INT. CONF. ON BIOMEDICAL TRANSDUCERS
PARIS, 3-7 NOVEMBER 1975, 383-391.
THE PRINCIPLE OF THE METHOD PRESENTED IN THIS PAPER IS TO
STORE THE HEMODYNAMIC PROFILE OF THE MOST COMMON SYN-
DROMES IN A DECISION TABLE AND COMPARE FOR A GIVEN
PATIENT HIS PROFILE WITH THE TABLE IN ORDER TO SUGGEST A
DIAGNOSIS.
- MUN73 MUNTZ, C.
MULTIPLE CHOICE DECISION TABLES
MASSACHUSETTS COMPUTER ASSOCIATES, RESEARCH PAPER
WAKEFIELD, MASS., DECEMBER 1973.
- MUT69 MUTHUKRISHNAN, C.R.
ANALYSIS AND CONVERSION OF DECISION TABLES TO COMPUTER
PROGRAMS
COMPUTER CENTRE, INDIAN INSTITUTE OF TECHNOLOGY
KANPUR, 1969.
- MUT70 MUTHUKRISHNAN, C.R. RAJARAMAN, V.
ON THE CONVERSION OF DECISION TABLES TO COMPUTER PROGRAMS
CACM, 13(6)
JUNE 1970, 347-351.
THE USE OF EXECUTION TIME DIAGNOSTICS IN PINPOINTING AM-
BIGUITIES IN DECISION TABLES IS DISCUSSED. IT IS POIN-
TED OUT THAT ANY ATTEMPT AT RESOLVING AMBIGUITIES AT
COMPILE TIME WILL, IN GENERAL, BE IMPOSSIBLE. IT IS
SHOWN THAT, AS A CONSEQUENCE, TREE METHODS OF CONVER-
TING DECISION TABLES TO PROGRAMS ARE INADEQUATE IN RE-
GARD TO AMBIGUITY DETECTION. TWO ALGORITHMS FOR PROGRAM-
MING DECISION TABLES WHOSE MERITS ARE SIMPLICITY OF IM-
PLEMENTATION AND DETECTION OF AMBIGUITIES AT EXECUTION
TIME ARE PRESENTED. THE FIRST ALGORITHM IS FOR LIMITED
ENTRY DECISION TABLES AND CLARIFIES THE IMPORTANCE OF
PROPER CODING OF THE INFORMATION OF THE DECISION TABLE.
THE SECOND ALGORITHM PROGRAMS A MIXED ENTRY DECISION
TABLE DIRECTLY WITHOUT GOING THROUGH THE INTERMEDIATE
STEP OF CONVERSION TO A LIMITED ENTRY FORM, THEREBY RE-
SULTING IN STORAGE ECONOMY. A COMPARISON OF THE ALGO-
RITHMS AND OTHERS PROPOSED IN THE LITERATURE IS MADE.
SOME FEATURES OF A DECISION TABLE TO FORTRAN IV TRANS-
LATOR FOR THE IBM 7044 DEVELOPED BY THE AUTHORS ARE
GIVEN.
- MUT71 MUTHUKRISHNAN, C.R.
REPLY TO THE COMMENT BY POLLACK
CACM, 14(1)
JANUARY 1971, 52.
THE AUTHOR REPLIES TO THE COMMENT MADE BY POLLACK IN
POL71A ON MUT70 BY EXPLAINING THE KIND OF AMBIGUITIES
HE MENTIONED IN HIS ORIGINAL ARTICLE.
- MYE72 MYERS, H.J.
COMPILING OPTIMIZED CODE FROM DECISION TABLES
IBM JRD, 16(5)
SEPTEMBER 1972, 489-503.
THIS PAPER REVIEWS THE STRUCTURE OF DECISION TABLES AND
METHODS FOR CONVERTING THEM INTO PROCEDURAL CODE. IT
DESCRIBES NEW OPTIMIZATION METHODS, WHICH ARE APPLIED
BEFORE, DURING, AND AFTER CODE GENERATION. SOME RESULTS
FROM AN EXPERIMENTAL DECISION TABLE PROCESSOR ARE
PROVIDED.
- NAR62 NARAMORE, F.
APPLICATION OF DECISION TABLES TO MANAGEMENT INFORMATION

SYSTEMS
PROC. DECISION TABLES SYMPOSIUM
SEPTEMBER 1962, 63-74.
THE PAPER OUTLINES THE USE OF DECISION TABLES AS PART
OF DATIS, A SYSTEM FOR DOCUMENTING MANAGEMENT INFORMATION
SYSTEMS DEVELOPED BY THE SUNDERLAND COMPANY. THE MAIN
EMPHASIS IS PLACED ON THE USE OF TABLES IN ACQUIRING
A PRECISE DEFINITION OF A SYSTEM'S REQUIREMENTS RATHER
THAN THEIR SUBSEQUENT USE FOR SPECIFYING HOW THE NEW
SYSTEM IS TO OPERATE. REFERENCE IS MADE TO USE IN
COMPLEX FILE MAINTENANCE APPLICATIONS AND THE BENEFITS
OBTAINED. IT IS CLAIMED THAT AN AVERAGE REDUCTION IN THE
WORK CONTENT OF ABOUT 30 % CAN BE GAINED BY USING TABLES
COMPARED WITH MORE CONVENTIONAL METHODS, WITH THE BULK OF
THIS SAVING BEING IN THE ANALYSIS PHASE.

- NAT76 NATTER, L.
ENTScheidungstabellen in Adv und Fachbereich
(G) DECISION TABLES IN DP AND SPECIALIST DEPARTMENTS
BURO, 24(3)
MARCH 1976, 42-47.
THE AUTHOR MAINLY DEALS WITH THE INTRODUCTION OF THIS
METHOD OF PROCEDURE FORMULATION INTO ORGANIZATIONS.
- NCC69 NATIONAL COMPUTING CENTRE
SIMPLIFYING REPORT PREPARATION
NCC NEWSLETTER, (15)
MAY 1969, 13.
A VERY BRIEF ACCOUNT OF THE USE OF NITA (THE FIRST
VERSION OF FILETAB) AT PLESSEY TELECOMMUNICATIONS AND
COLVILLES OF GLASGOW, FOLLOWED BY A GENERAL DESCRIPTION
OF WHAT THE PACKAGE CAN DO.
- NCC70 NATIONAL COMPUTING CENTRE
DECISION TABLES IN DATA PROCESSING
SCIENCE ASSOCIATES INTERNATIONAL
NEW YORK, 1970, 107 PP.
THIS REPORT CONSIDERS THE USE OF DECISION TABLES IN THE
U.K. WITHIN THE DATA PROCESSING FIELD AND IS MORE
CONCERNED WITH PRACTICAL EXPERIENCE THAN THEORETICAL
FOUNDATIONS. BEFORE EXAMINING IN DEPTH THE USE OF
DECISION TABLES, HOWEVER, A BRIEF DESCRIPTION IS GIVEN
OF THE BASIC CONCEPTS AND METHODS OF TABLE CONSTRUCTION.
AT THE END, CONVERTING TABULAR LOGIC TO PROGRAMS,
APPLICATIONS OF DECISION TABLES AND DEVELOPMENTS IN THE
USE OF DECISION TABLES ARE COMPREHENSIBLY TREATED. THE
APPENDICES DEAL WITH SOME DECISION TABLE FORMATS,
DECISION TABLE SOFTWARE, SOME NOTES ON RULE
IDENTIFICATION METHODS AND A THOROUGHLY ANNOTATED
BIBLIOGRAPHY.
- NCC73 NATIONAL COMPUTING CENTRE
FILETAB USER MANUAL
1973.
THIS IS A DETAILED DESCRIPTION OF A TYPICAL PIECE OF
DECISION TABLE SOFTWARE, WHICH IS BEING USED BY OVER
150 INSTALLATIONS FOR COMMERCIAL APPLICATIONS. THE
FACILITIES INCLUDE ALL THE USUAL DATA PROCESSING
ACTIVITIES OF DATA INPUT, VALIDATION, UPDATE, MERGE,
SORT, AND REPORTING. HENCE, FILETAB PROVIDES A METHOD
FOR PUTTING TOGETHER A CONVENTIONAL PROGRAM BUT DOES IT
BY COMPILING A DECISION TABLE LANGUAGE. ALL THE LOGIC
AND PROCESSING ARE EXPRESSED AS DECISION TABLES AND
THE COMPILER CONVERTS THEM BY A RULE MASK TECHNIQUE
INTO CODE FOR THE TARGET MACHINE.
- NIC61 NICKERSON, R.C.
AN ENGINEERING APPLICATION OF LOGIC-STRUCTURE TABLES
CACM, 4(11)
NOVEMBER 1961, 516-520.
THE AUTHOR DESCRIBES AND ILLUSTRATES IN SOME DETAIL AN
EARLY APPLICATION OF LOGIC-STRUCTURE TABLES TO
ENGINEERING CALCULATIONS, USING LOGTAB, AT THE GENERAL
ELECTRIC COMPANY. THESE WERE REQUIRED FOR THE
MANUFACTURING, ON MAGNETIC-TAPE CONTROLLED CONTOUR
MILLING MACHINES, OF BUCKETS FOR THE EXHAUST STAGES OF
LARGE STEAM TURBINES.
- NIE72 NIEDZIECKI, J.

TABLICE DECYZJI - STRUKTURA I ZASTOSOWANIA
(P) DECISION TABLES - STRUCTURE AND APPLICATIONS
INFORMATIKA, 8(1)
JANUARY 1972, 16-20.

- NIL75 NILL, H.
ENTSCHEIDUNGSMATRIZEN - EINE FORM DER ANWENDUNG VON
ENTSCHEIDUNGSTABELLEN ALS HILFSMITTEL FUR DIE PROGRAMMIERUNG
DER EDVA
(G) DECISION GRID CHARTS - AN APPLICATION FORM OF DECISION
TABLES IN PROGRAMMING
RTDV, 12(8)
SEPTEMBER 1975, 66-68.
THE DECISION GRID CHART, AS PROPOSED IN STR71B, IS PRE-
SENTED AS AN ALTERNATIVE LOGICAL DESCRIPTION FORM. SOME
ADVANTAGES ARE OUTLINED.
- NOL71 NOLAND, J.L. FENG, C.C.
FORMULATION OF DECISION LOGIC TABLES
JOURNAL STRUCT. DIV., 97(ST1)
JANUARY 1971, 463-479.
THIS PAPER PRESENTS DECISION TABLES AS TOOLS FOR DEFINING
AND PRESENTING ENGINEERING DECISION-MAKING PROBLEMS.
A TECHNIQUE FOR FORMULATION SUCH TABLES IS PROSENTED.
TWO EXAMPLE PROBLEMS ARE GIVEN.
- NOL75 NOLAND, J.L.
FORMULATION OF DECISION LOGIC TABLES AND THEIR APPLICATION
TO ENGINEERING SPECIFICATIONS
PH. D. THESIS
UNIV. OF COLORADO, DEPT. CIVIL AND ENVIRONMENTAL ENGINEERING
1975
IN THE FIRST PART OF THIS THESIS, A METHOD FOR CONSTRUC-
TING DECISION TABLES IS PRESENTED. IN THE SECOND PART,
THE CONVERSION OF THE BUILDING CODE REQUIREMENTS FOR
REINFORCED CONCRETE INTO DECISION TABLE FORMAT IS GIVEN.
- NOR65 NORDBOTTEN, S.
DESISJONSTABELLER OG GENERERING AV MASKIN - PROGRAMMER FOR
GRANSKNING AV STATISTISK PRIMAERMATERIALE
(N) DECISION TABLES AND THE GENERATION OF MACHINE PROGRAMS
FOR THE ANALYSIS OF STATISTICAL DATA
STATISTISK TIDSKRIFT, (44)
1975, 260.
- OEROO OERTER, G.W.
DECISION TABLES
UNPUBLISHED CHAPTER
LEEDS AND NORTHRUP
DATE UNKNOWN.
THE BASIS OF THE DECISION TABLE TECHNIQUE IS EXPLAINED
IN TERMS OF SPL IV, A FORTRAN-BASED PROGRAMMING
LANGUAGE.
- DER68 OERTER, G.W.
A NEW IMPLEMENTATION OF DECISION TABLES FOR A PROCESS
CONTROL LANGUAGE
IEEE TRANS. INDUSTRIAL ELECTRONICS AND CONTROL
INSTRUMENTATION, IECI-15 (2)
DECEMBER 1968, 57-61.
ALSO : NINTH JOINT AUTOMATIC CONTROL CONFERENCE
ANN ARBOR, 26-27 JUNE 1968, 972-973.
THE IMPLEMENTATION BY THE RULE MASK TECHNIQUE OF DECISION
TABLES IN A HIGH-LEVEL PROCESS CONTROL LANGUAGE, SPL, IS
DISCUSSED. THE TECHNIQUE IS ILLUSTRATED BY A SIMPLE
EXAMPLE ; NEXT, IT IS APPLIED TO CONVERT EXTENDED ENTRY
TABLES. THE ADVANTAGES GAINED BY THE AUTOMATIC TRANSLATI-
ON OF DECISION TABLES INTO CODED CONDITION MASKS ARE
STRESSED.
- OLI77 OLIVER, S.R.
NOTES ON THE USE OF DECISION TABLES
CACM, 20(11)
NOVEMBER 1977, 890 AND 896.
THE ADVANTAGES AND DISADVANTAGES OF DECISION TABLES ARE
DISCUSSED BASED ON THE AUTHOR'S EXPERIENCES IN USING
TRANSITION MATRICES, THAT ARE SAID TO BE A SPECIAL
VERSION OF THE DECISION TABLE CONCEPT.

- OPP72 OPPELLAND, H.J.
 ANSAETZE FUER EINE THEORIE DER ENTSCHEIDUNGSTABELLEN UNTER
 BERUECKSICHTIGUNG IHRE BEDEUTUNG FUER EIN MIS
 (G) CONTRIBUTION TO THE THEORY OF DECISION TABLES WITH A
 VIEW TO THIER MEANING ON DEVELOPING MIS
 TU BERLIN, THESIS
 1972.
- OPP77 OPPELLAND, H.J.
 DIE VERARBEITUNG VON ENTSCHEIDUNGSTABELLEN MIT HILFE EINES
 REGELZAHL-ALGORITHMUS
 (G) THE CONVERSION OF DECISION TABLES USING A RULE-NUMBER
 ALGORITHM
 ANGEWANDTE INFORMATIK, 19(5)
 MAY 1977, 202-209.
 A VEINOTT-LIKE ALGORITHM FOR THE CONVERSION OF DECISION
 TABLES IS GIVEN.
- PAG69 PAGER, D.
 ON THE PROBLEM OF FINDING MINIMAL PROGRAMS FOR TABLES
 INFORMATION AND CONTROL, 14
 1969, 550-554.
- PAG74 PAGER, D.
 FURTHER RESULTS ON THE PROBLEM OF FINDING MINIMAL LENGTH
 PROGRAMS FOR DECISION TABLES
 JACM, 21(2)
 APRIL 1974, 207-212.
 IN THIS PAPER IT IS SHOWN THAT WHATEVER THE LENGTH
 FUNCTION EMPLOYED, THE PROBLEM OF FINDING THE SHORTEST
 PROGRAM FOR A DECISION TABLE WITH TWO (OR MORE) ENTRIES
 IS NOT RECURSIVELY SOLVABLE (WHEREAS FOR DECISION TABLES
 WITH A SINGLE ENTRY THE PROBLEM IS SOLVABLE FOR SOME
 LENGTH FUNCTIONS AND UNSOLVABLE FOR OTHERS). MOREOVER, IT
 IS SHOWN THAT THERE IS A PAIR OF FINITE SETS OF PROGRAMS
 AND A SINGLE ENTRY E SUCH THAT THE SHORTEST PROGRAM FOR
 THE DECISION TABLE FORMED BY ADDING A SINGLE ADDITIONAL
 ENTRY TO E IS IN ALL CASES IN ONE OF THE TWO SETS, BUT IT
 IS UNDECIDABLE IN WHICH. SOME CONSEQUENCES OF THESE
 RESULTS ARE THEN PRESENTED, SUCH AS SHOWING THAT FOR A
 WIDE RANGE OF RESTRICTIONS THE RESULTS REMAIN TRUE, EVEN
 WHEN THE REPERTOIRE OF POSSIBLE PROGRAMS FOR A DECISION
 TABLE IS NARROWED BY ONLY CONSIDERING PROGRAMS WHICH
 MEET CERTAIN RESTRICTIONS.
- PAG75 PAGENKEMPER, K. HEITZ, B.
 ENTSCHEIDUNGSTABELLEN IN ORGANISATION UND DATENVERARBEITUNG
 (G) DECISION TABLES IN THE ORGANIZATION AND IN DATA
 PROCESSING
 LUCHTERHAND
 NEUWIED, 1975, 176 PP.
 THE USE OF THE DECISION TABLE AS A TOOL FOR COMMUNICATION
 IN THE ORGANIZATION IS EXPLAINED. MANY PRACTICAL EXAMPLES
 ARE PROVIDED.
- PAP80 PAPAKONSTANTINOU, G.
 A RECURSIVE ALGORITHM FOR THE OPTIMAL CONVERSION OF DECISION
 TABLES
 ANGEWANDTE INFORMATIK, 22(9)
 SEPTEMBER 1980, 350-354.
 THE CONVERSION ALGORITHM DEVELOPED IN THIS PAPER PRODUCES
 OPTIMAL PROGRAMS. IT IS BASED UPON A COMBINATION OF THE
 DYNAMIC PROGRAMMING APPROACH AND BRANCH-AND-BOUND TECH-
 NIQUES.
- PAROO PARSONS, A.C.
 AUTOCODER DECISION TABLE ASSEMBLY
 UNITED GAS CORPORATION, COMPUTER DEPARTMENT TECHNICAL
 REPORT 1-63
 SHREVEPORT, DATE UNKNOWN.
- PATOO PATTERSON, Z.R.
 A NEW TECHNIQUE FOR DECISION TABLES
 IBM TIE Z 77-3776
 DATE UNKNOWN.
- PEE69 PEEL, R.
 DECISION TABLE TRANSLATION
 THE COMPUTER BULLETIN, 19(12)

DECEMBER 1969, 419-421.

THE AUTHOR PROPOSES A METHOD, WHICH CONVERTS MIXED-ENTRY DECISION TABLES DIRECTLY, I.E. WITHOUT USE OF THE LIMITED ENTRY FORM, TO COMPUTER PROGRAMS. THE METHOD IS BASED ON THE RULE/MASK TECHNIQUE AS PROPOSED BY KIRK (KIRGG).

- PER67 PERSSON, P.O.
EN PRAKTISK TILLAMPNING AV BESLUTSTABELLER
(S) A PRACTICAL APPLICATION OF DECISION TABLES
AUTOMATISK DATABEHANDLING, (4)
1967.
- PER68 PERSSON, S.
KONSTRUKTION AV BESLUTSTABELLER
(S) THE CONSTRUCTION OF DECISION TABLES
DATABEHANDLING, (6-7)
1968, 4-14.
THE CONSTRUCTION OF DECISION TABLES STARTING FROM
A VERBAL DESCRIPTION OF THE DECISION SITUATION IS
ILLUSTRATED BY MEANS OF AN ELABORATED EXAMPLE
(REGULATIONS FOR OVERTIME).
- PER69 PERSSON, S.
BESLUTSTABELLER + FLODESPLANER = PROGRAM
(S) DECISION TABLE + FLOWCHART = PROGRAM
PART I : DATABEHANDLING, (3)
PART II : DATABEHANDLING, (4)
1969, 26-33 AND 24-28.
IN THESE ARTICLES, IT IS ARGUED THAT DECISION TABLES
AND FLOWCHARTS ARE TWO COMPLEMENTARY TECHNIQUES.
THE CONVERSION OF A USUAL DECISION TABLE INTO A DECISION
TABLE-FLOWCHART MIXED FORMAT IS EXPLAINED. THE
ADVANTAGES OF THE COMBINATION ARE GIVEN.
- PER71 PERSSON, S.
BESLUTSTABELLER, 1. BESKRIVING AV REGELSAMBAND
(S) DECISION TABLES : DESCRIPTION AND CONTEXT
STUDENTLITTERATUR
LUND, 1971, 307 PP.
THE AUTHOR SUCCESSIVELY DEALS WITH TABLE METHODS IN
GENERAL, DECISION TABLE FORMAT, PROPERTIES, CONSTRUCTION
AND CONTROL METHODS OF DECISION TABLES. THE TRANSFOR-
MATION AND THE MINIMIZATION OF A DECISION TABLE
ARE THOROUGHLY TREATED. AFTER DISCUSSING THE PARSING
METHODS, THE AUTHOR INDICATES THE APPLICATION AREA OF
THE DECISION TABLE TECHNIQUE.
- PET73 PETERS, H.
ENTSCHEIDUNGSTABELLEN ALS ORGANISATIONSMITTEL IN
SACHBEARBEITUNG UND EDV
(G) DECISION TABLES AS A TOOL FOR ORGANISATION IN COMMERCIAL
AND TECHNICAL PROCEDURES AND FOR ELECTRONIC DATA PROCESSING
BURO, 21(11)
NOVEMBER 1973, 1124-1129.
THIS ARTICLE DISCUSSES THE SHORTCOMINGS OF GENERALISED
UNIFYING STANDARDS AND METHODS USED BY ORGANISERS, SYSTEMS
ANALYSTS AND PROGRAMMERS IN THE SOLVING OF PROBLEM
COMPLEXES. THESE FAULTS ARE COMPOUNDED BY THE LACK OF A
UNIFORM AND GENERALLY COMPREHENSIBLE METHOD OF
PRESENTATION OF THE PROBLEMS TO BE HANDLED. THE BOOK
TRADE IS USED AS AN EXAMPLE TO ILLUSTRATE THE APPLICATION
OF DECISION TABLES TO A PROBLEM COMPLEX AND TO DEMON-
STRATE THE UNIFIED APPROACH TO ANY SET OF PROBLEMS WHICH
IS AVAILABLE THROUGH THE USE OF THESE TABLES.
- PIQ68 PIQUART, J.L.
LES TABLES DE DECISION
(F) DECISION TABLES
INFORMATIQUE ET GESTION,
PART I : (3), 103-113
PART II : (4), 89-98
PART III : (5), 97-105.
THESE SERIES OF ARTICLES PRESENT A RATHER UNUSUAL
INTRODUCTION TO THE USE OF DECISION TABLES. THE AUTHOR
SUCCESSIVELY DEALS WITH TABULAR FORMS, BASIC CONCEPTS
OF THE DECISION TABLE TECHNIQUE, PRACTICAL EXPERIENCES,
DIFFERENT TYPES OF DECISION TABLES, APPLICATIONS (ILLUS-
TRATED BY MEANS OF SIMPLE EXAMPLES) AND THE PROBLEM OF
PROGRAMMING AND COMPILEING DECISION TABLES.

- PLA75 PLATZ, G.
LOGIKBESCHREIBUNGSSPRACHEN UND IHRE ANWENDUNG, TEIL II
(G) LOGIC DESCRIPTION LANGUAGES AND THEIR APPLICATIONS,
PART II
ONLINE, 13(4)
APRIL 1975, 242-248.
IN THIS SECOND PART OF A TREATMENT OF LOGIC DESCRIPTION
LANGUAGES, THE AUTHOR PROPOSES DECISION TABLES AS A
UNIFIED TOOL FOR COMMUNICATION BETWEEN THE EDP-DEPARTMENT
AND THE REST OF THE ORGANIZATION. IN THIS MANNER,
DECISION TABLES CONSTITUTE THE BASIS OF A POSSIBLE
DESCRIPTION LANGUAGE.
- POL61 POLLACK, S.L. GRAD, B.
TABSOL APPLICATION MANUAL, INTRODUCTION TO TABSOL
GENERAL ELECTRIC COMPUTER DEPARTMENT, CPB-147A (5M-6-61)
PHOENIX, 1961.
- POL62A POLLACK, S.L. WRIGHT, K.R.
DATA DESCRIPTION FOR DETAB-X (DECISION TABLE, EXPERIMENTAL)
RAND CORPORATION, RM-3010-PR
MARCH 1962, 46 PP.
THIS MEMORANDUM IS THE FIRST OF A SERIES OF RAND PUBLICATIONS ON DECISION TABLES FOR THE UNITED STATES AIR FORCE; IT DESCRIBES PRELIMINARY RESULTS OF THE AUTHORS' WORK ON DATA DESCRIPTION. AFTER A BRIEF DESCRIPTION OF THE COBOL 61 DATA DIVISION, THE REST OF THE PAPER DESCRIBES A TABULAR STRUCTURE FOR DATA DESCRIPTION AND THE MODIFICATIONS REQUIRED TO COBOL 61 TO ENABLE DATA DESCRIPTION TO FIT THE TABULAR STRUCTURE PROPOSED.
- POL62B POLLACK, S.L.
DETAB-X : AN IMPROVED BUSINESS-ORIENTED COMPUTER LANGUAGE
RAND CORPORATION, RM-3273-PR
AUGUST 1962, 18 PP.
THIS MEMORANDUM DESCRIBES DETAB-X (DECISION TABLES, EXPERIMENTAL). IN AN EFFORT TO ILLUSTRATE SOME OF THE FEATURES OF DETAB-X, IT IS COMPARED WITH COBOL-61 (COMMON BUSINESS-ORIENTED LANGUAGE), USING EXAMPLES OF DATA AND PROCEDURES WRITTEN IN BOTH LANGUAGES.
- POL62C POLLACK, S.L.
WHAT IS DETAB-X ?
PROC. DECISION TABLES SYMPOSIUM
SEPTEMBER 1962, 29-39.
DETAB-X IS AN EXPERIMENTAL LANGUAGE THAT COMBINES COBOL-61 AND DECISION TABLES. IT IS A PROPOSED SUPPLEMENT TO, NOT A REPLACEMENT OF COBOL-61.
- POL62D POLLACK, S.L. GRAD, B.
DETAB-X. PRELIMINARY SPECIFICATIONS FOR A DECISION TABLE STRUCTURED LANGUAGE
DATA DESCRIPTION AND TRANSFORMATION LOGIC TASK FORCES OF THE CODASYL SYSTEMS GROUP
SEPTEMBER 1962.
- POL62E POLLACK, S.L. GRAD, B.
QUESTION AND ANSWER PERIOD . . . 9/20/62
PROC. DECISION TABLES SYMPOSIUM
SEPTEMBER 1962, 9-12.
- POL63A POLLACK, S.L.
CODASYL, COBOL, AND DETAB-X
DATAMATION, 9(2)
FEBRUARY 1963, 60-61.
THIS STATUS REPORT TREATS THE HISTORY OF CODASYL AND DETAB-X, A DECISION TABLE STRUCTURE USING MODIFIED COBOL-61 FOR BUSINESS-PROBLEM DESCRIPTION AND DEVELOPED BY THE SYSTEMS GROUP OF CODASYL.
- POL63B POLLACK, S.L.
ANALYSIS OF THE DECISION RULES IN DECISION TABLES
RAND CORPORATION, RM-3669-PR
SANTA MONICA, MAY 1963, 69 PP.
THIS MEMORANDUM DEVELOPS A THEORETICAL STRUCTURE FOR DECISION TABLES. THE THEOREMS DEVELOPED IN THIS PAPER PROVIDE A BASIS FOR SYSTEM ANALYSTS AND PROGRAMMERS TO VERIFY THE LOGIC OF THEIR ANALYSIS. RULES ARE ESTABLISHED

THAT ENABLE THEM TO INSURE THE FOLLOWING : (1) THAT ALL POSSIBLE COMBINATIONS OF CONDITIONS FOR THE PROBLEM HAVE BEEN CONSIDERED ; (2) THAT THE SYSTEM DOESN'T PRESCRIBE DIFFERENT ACTIONS FOR THE SAME SITUATION ; AND (3) THAT THE SYSTEM DESCRIBES EACH SITUATION AND ITS ACTION ONCE ONLY. THE IMMEDIATE EFFECT OF ACHIEVING THE ABOVE IS AN IMPROVEMENT IN COMPUTER PROGRAMMING BY REDUCING THE NUMBER OF COMPUTER INSTRUCTIONS, SHORTENING COMPUTER RUNNING TIME, AND DECREASING PROGRAMMING AND DEBUGGING TIME. THE TEXT ALSO PRESENTS AN EXTENSION OF DECISION TABLE THEORY : IT PROVIDES A BASIS FOR HAVING DECISION RULES IN WHICH A SERIES OF ACTIONS CAN BE TAKEN IF ANY ONE OF A SET OF SPECIFIED CONDITIONS IS SATISFIED. IT IS SAID THAT THIS TYPE OF DECISION RULE CAN BE EXTREMELY USEFUL IN EDITING AND INFORMATION RETRIEVAL.

- POL63C POLLACK, S.L.
HOW TO BUILD AND ANALYZE DECISION TABLES
RAND CORPORATION, P-2829
SANTA MONICA, NOVEMBER 1963, 17 PP.
THIS MEMORANDUM DESCRIBES THE CONVERSION OF SYSTEM APPLICATIONS TO DECISION TABLES, A PROCESS WHICH ENTAILS MAKING DECISIONS ON HOW LARGE THE INDIVIDUAL TABLES SHOULD BE AND WHAT SYSTEM PARAMETERS SHOULD BE INCLUDED. A TECHNIQUE FOR REDUCING THE NUMBER OF WRITTEN DECISION RULES IS ALSO DESCRIBED. ONCE THE DECISION TABLES ARE WRITTEN, THEY SHOULD BE CHECKED FOR COMPLETENESS AND CONSISTENCY. THIS PAPER DESCRIBES AND ILLUSTRATES THE RULES THAT ENABLE SYSTEMS ANALYSIS TO INSURE THE FEATURES MENTIONED IN POL63B.
- POL64A POLLACK, S.L.
CONVERSION OF LIMITED-ENTRY DECISION TABLES TO COMPUTER PROGRAMS
RAND CORPORATION, RM-4020-PR
SANTA MONICA, MAY 1964, 15 PP.
ALSO IN : CACM, 8(11)
NOVEMBER 1965, 677-682.
ALGORITHMS THAT CAN EFFICIENTLY CONVERT DECISION TABLES INTO COMPUTER PROGRAMS EXTEND THE USEFULNESS OF DECISION TABLES TO COMPUTER USERS. THIS MEMORANDUM DESCRIBES TWO SUCH ALGORITHMS, BASED ON WORK DONE BY M.S. MONTALBANO AND EXTENDED HERE TO HANDLE DASHES AND ELSE-DECISION RULES. THE FIRST ALGORITHM MINIMIZES THE COMPUTER STORAGE SPACE REQUIRED FOR THE RESULTING PROGRAM, THE SECOND MINIMIZES COMPUTER RUNNING TIME. DURING THE CONVERSION PROCESS, BOTH PINPOINT ANY CONTRADICTIONS OR REDUNDANCIES AMONG THE RULES IN A TABLE.
A NECESSARY ADJUNCT TO MINIMIZING COMPUTER STORAGE OR RUNNING TIME IS THE ALLOWABLE REDUCTION OF THE NUMBER OF RULES IN A DECISION TABLE. THIS MEMORANDUM DESCRIBES A TECHNIQUE TO EFFECT THIS REDUCTION FOR PAIRS, TRIPLES AND QUADRUPLES OF RULES. THE TECHNIQUE CAN BE APPLIED MANUALLY OR ACCOMPLISHED BY THE COMPUTER AS A PRELUDE TO EXECUTING ONE OF THE TWO ALGORITHMS.
- POL64B POLLACK, S.L.
THE DEVELOPMENT AND ANALYSIS OF DECISION TABLES
IDEAS FOR MANAGEMENT (PAPERS PRESENTED AT THE 1964 INTERNATIONAL SYSTEMS MEETING)
1964, 91-98.
THIS PAPER IS A SLIGHTLY EDITED VERSION OF THE SAME AUTHOR'S "HOW TO BUILD AND ANALYSE DECISION TABLES" (POL63C). ADDITIONAL MATERIAL SHOWS AN EXAMPLE IN BOTH FLOW CHART AND DECISION TABLE FORM AND BRIEFLY COMPARES BOTH TECHNIQUES.
- POL65 POLLACK, S.L.
DECISION TABLES FOR SYSTEM DESIGN
PROC. 19TH INTERNATIONAL DATA PROCESSING CONFERENCE
1965, 485-492.
AFTER A BRIEF STATEMENT OF THE TWO MAIN PROCEDURES FOR CONVERTING DECISION TABLES TO PROGRAMS, THE AUTHOR DESCRIBES AND ILLUSTRATES A PROCEDURE INTENDED TO ACHIEVE MINIMUM STORAGE. THIS IS THE FIRST OF THE TWO ALGORITHMS DESCRIBED IN HIS "CONVERSION OF LIMITED-ENTRY DECISION TABLES TO COMPUTER PROGRAMS" (POL64A).
- POL66 POLLACK, S.L.-

DECISION TABLES DIRECTLY INTO PROGRAM
IDEAS FOR MANAGEMENT (PAPERS PRESENTED AT THE 1966
INTERNATIONAL SYSTEMS MEETING)
1966, 54-90.
ALSO IN : ADT, 171-214
AND IN : ACM DETAB/65 PREPROCESSOR (ACM66).
THIS DESCRIPTION OF THE DETAB/65 PREPROCESSOR CONTAINS
SEVERAL PARTS : THE DETAB/65 LANGUAGE, TEST DECK LISTING,
PREPROCESSOR LISTING AND A DETAB/65 USER'S MANUAL.
THE SECTION ON THE DETAB/65 LANGUAGE PREFACES ITS MAIN
REMARKS BY MENTIONING THE ORIGINS OF DETAB/65 AND
SUMMARISING ITS DIFFERENCES FROM THE EARLIER DETAB-X.
CHAPTER I SETS OUT THE PURPOSE AND CONTEXT OF DETAB/65
AND GOES ON TO DESCRIBE DECISION TABLES, IN BOTH
EXTENDED AND LIMITED ENTRY FORMS. CHAPTER II DESCRIBES
THE SOURCE PROGRAM FORMAT WHILST CHAPTER III SETS OUT
THE TABLE FORMAT REQUIREMENTS ADOPTED. THE USER'S
MANUAL SUMMARISES THE MATERIAL SET OUT IN THE FIRST
SECTION, BUT IN TERMS OF A LIMITED ENTRY TABLE ONLY.
IT INCLUDES A NOTE ON OUTPUT AND THE MAIN ERRORS
ENCOUNTERED AND LISTS IN AN APPENDIX THE ERROR MESSAGES
PRODUCED. ANOTHER APPENDIX INCLUDES THE PRINT-OUT, WITH
GENERATED STATEMENTS, OF A TEST PROGRAM (FOR CHOICE OF
A BEAUTY QUEEN, THE WELL-KNOWN CHOICE-PICK DECISION
TABLE).

- POL68A POLLACK, S.L.
DECISION, NOT IMPRECISION
DATAMATION, 14(3)
MARCH 1968, 172.
THIS SHORT COMMENT ON JACKSON'S ARTICLE (JAC68A) IN THE
PREVIOUS ISSUE DISCUSSES THREE SPECIFIC CONSTRAINTS IN
THE USE OF DECISION TABLES TO WHICH IT OBJECTED : 1. THE
NEED FOR A COMPLETE AND ACCURATE STATEMENT OF PROBLEM
LOGIC, 2. THE TABLE MUST NOT CONTAIN CONTRADICTIONS AND
3. THE TABLE MUST BE CHECKED FOR REDUNDANCIES. IT FI-
NISHES WITH A SHORT NOTE ON WHAT A PREPROCESSOR SHOULD
BE DESIGNED TO DO IN EACH CASE.
- POL68B POLLACK, S.L. HICKS, H.T.
DECISION TABLES FOR SYSTEM DESIGN AND PROGRAMMING
NOTES FOR AN ACM PROFESSIONAL DEVELOPMENT SEMINAR
INFORMATION MANAGEMENT, INC.
SAN FRANCISCO, 1968.
- POL69 POLLACK, S.L. HARRISON, W.J.
DETAB, VERSION III USER'S GUIDE
IMI
JULY 1969.
- POL70 POLLACK, S.L. HICKS, H.T. HARRISON, W.J.
A DECISION TABLE APPROACH TO SYSTEM ANALYSIS
DATA BASE, 2(1)
1970, 6-12.
THIS ARTICLE DISCUSSES THE ROLE OF THE SYSTEM ANALYSIS
TABLE, SETTING UP THE TABLE, REDUCTION OF TABLE SIZE,
APPLICATION OF THE ANALYSIS TO PROGRAM FLOW AND USE OF
THE DON'T CARE AND OTHER CONDITION TYPES.
- POL71A POLLACK, S.L.
COMMENT ON THE CONVERSION OF DECISION TABLES TO COMPUTER
PROGRAMS
CACM, 14(1)
JANUARY 1971, 52.
POLLACK DISPROVES THE OPINION THAT ANY ATTEMPT AT
RESOLVING AMBIGUITIES AT COMPILE TIME WILL BE
IMPOSSIBLE. THIS OPINION WAS PROPAGATED BY MUTHUKRISHNAN
AND RAJARAMAN IN MUT70. A REPLY BY MUTHUKRISHNAN IS ADDED
TO THIS COMMENT (MUT71).
- POL71B POLLACK, S.L. HICKS, H.T. HARRISON, W.J.
DECISION TABLES : THEORY AND PRACTICE
WILEY-INTERSCIENCE
NEW YORK, 1971, 179 PP.
THIS BOOK COVERS THOROUGHLY THE CONCEPTUAL, MATHEMATICAL,
PROCEDURAL AND OPERATIONAL FOUNDATIONS OF DECISION
TABLES. THE BOOK IS DIVIDED INTO FIVE PARTS :
A) HISTORICAL DISCUSSION AND DECISION TABLE STRUCTURE ;
B) THEORETICAL FOUNDATIONS ; C) ANALYSIS AND IMPLICATIONS

OF DECISION TABLES : D) USES AND DEVELOPMENT OF DECISION TABLES ; AND E) TABLE TRANSLATION. MANY EXAMPLES ILLUSTRATE THE TREATMENT. AN EXTENSIVE BIBLIOGRAPHY IS PROVIDED.

- POM65 POMEROY, L.K.
ROAD MAPS TO DECISION
NAVY MANAGEMENT REVIEW, 10(1)
JANUARY 1965, 4-5.
- P0074 POOCH, U.W.
TRANSLATION OF DECISION TABLES
COMPUTING SURVEYS, 6(2)
JUNE 1974, 125-151.
DECOMPOSITION AND CONVERSION ALGORITHMS FOR TRANSLATING DECISION TABLES ARE SURVEYED AND CONTRASTED UNDER TWO BROAD CATEGORIES : THE RULE MASK TECHNIQUE AND THE NETWORK TECHNIQUE. ALSO, DECISION TABLE STRUCTURE IS BRIEFLY COVERED, INCLUDING CHECKS FOR REDUNDANCY, CONTRADICTIONS, AND COMPLETENESS ; DECISION TABLE NOTATIONS AND TECHNOLOGY ; AND DECISION TABLE TYPES AND APPLICATIONS.
- PRE65 PRESS, L.I.
CONVERSION OF DECISION TABLES TO COMPUTER PROGRAMS
CACM, 8(6)
JUNE 1965, 385-390.
SEVERAL TRANSLATION PROCEDURES FOR THE CONVERSION OF DECISION TABLES TO PROGRAMS ARE PRESENTED AND THEN EVALUATED IN TERMS OF STORAGE REQUIREMENTS, EXECUTION TIME AND COMPILE TIME. THE PROCEDURES ARE VALUABLE AS HAND-CODING GUIDES OR AS ALGORITHMS FOR A COMPILER. BOTH LIMITED-ENTRY AND EXTENDED-ENTRY TABLES ARE ANALYZED. IN ADDITION TO TABLE ANALYSIS, THE NATURE OF TABLE ORIENTED PROGRAMMING LANGUAGES AND FEATURES IS DISCUSSED.
- PRY76 PRYWES, N.S.
AUTOMATIC GENERATION OF PROGRAMS
PROC. SOFTWARE SYSTEMS ENGINEERING CONFERENCE
LONDON, SEPTEMBER 1976, 39-56.
IN THIS PAPER, DECISION TABLES ARE USED AS STANDARD REPORTS FOR DOCUMENTATION OF AUTOMATICALLY GENERATED COMPUTER PROGRAMS.
- QUI71 QUITTNER, P.
DONTESI TABLAZATOK
(H) DECISION TABLES
SZAMVITEL ES UGYVITELTECHNIKA, (11)
1971, 516-523.
- RAB67 RABIN, J.
LOGICAL DESIGN OF INFORMATION - PROCESSING SYSTEMS WITH DECISION TABLES
SYMPOSIUM ON DECISION TABLES
USASI AD HOC X3.6.7/X3.4.2D
ATLANTIC CITY, APRIL 1967.
- RAB71 RABIN, J.
CONVERSION OF LIMITED ENTRY DECISION TABLES INTO OPTIMAL DECISION TREES : FUNDAMENTAL CONCEPTS
SIGPLAN NOTICES, 6(8)
SEPTEMBER 1971, 68-74.
IN THIS PAPER, A SYSTEMATIC APPROACH TO THE CONVERSION OF LIMITED ENTRY DECISION TABLES INTO OPTIMAL DECISION TREES BASED ON THE DESIGN OF SO-CALLED RELAY TREES IN SWITCHING THEORY, IS DEVELOPED. BOTH DECISION TABLES WITH OR WITHOUT THE ELSE-RULE ARE CONSIDERED.
- RAG73 RAGU, G.
TABLES DE DECISION, ALGEBRE DE BOOLE, DIAGRAMMES DE VEITCH
(F) DECISION TABLES, BOOLEAN ALGEBRA, VEITCH DIAGRAMS
L'INFORMATIQUE,
PART I : (37), FEBRUARY 1973, 4-13
PART II : (38), MARCH 1973, 4-13
PART III : (39), APRIL 1973, 5-19.
IN THIS SERIES OF ARTICLES, SOME LOGICAL TECHNIQUES (DECISION TABLES, BOOLEAN ALGEBRA, VEITCH DIAGRAMS) ARE ILLUSTRATED. THEY ARE COMPARED IN THE FIELD OF PROBLEM

ANALYSIS, ORGANIZATION AND AS TOOLS FOR PROGRAM OPTIMIZATION. SOME EARLIER CONVERSION ALGORITHMS ARE DEALT WITH.

REI60 REINWALD, L.T. WILLIAMS, J.S.
DECISION TABLE STANDARDS
UNPUBLISHED PAPER
DATE UNKNOWN.

REI66A REINWALD, L.T.
AN INTRODUCTION TO TAB40 : A PROCESSOR FOR TABLE-WRITTEN
FORTRAN IV PROGRAMS
CLEARINGHOUSE REPORT NO. AD-647-418
RESEARCH ANALYSIS CORPORATION
MC LEAN, 1966.

TAB40, A PROGRAM WRITTEN FOR THE IBM 7040 COMPUTER, IS A
PREPROCESSOR FOR FORTRAN IV. SPECIFICALLY, TAB40 ACCEPTS
INPUT WRITTEN IN A PREDOMINATELY TABULAR FORMAT AND CON-
VERTS IT INTO A FREE FORM FORMAT ACCEPTABLE FOR COMPILA-
TION BY FORTRAN IV.

TAB40 ITSELF IS WRITTEN ALMOST ENTIRELY IN FORTRAN IV
AND WITH MINOR MODIFICATIONS CAN BE RUN ON ANY COMPUTER
FOR WHICH FORTRAN IV WITH A LOGICAL IF HAS BEEN IMPLE-
MENTED.

REI66B REINWALD, L.T. SOLAND, R.M.
CONVERSION OF LIMITED-ENTRY DECISION TABLES TO OPTIMAL
COMPUTER PROGRAMS I : MINIMUM AVERAGE PROCESSING TIME
JACM, 13(3)
JULY 1966, 339-358.

THIS PAPER BEGINS WITH A BRIEF DESCRIPTION OF DECISION
TABLES, AND THEN PRESENTS A DISCUSSION OF ALTERNATE
EXPRESSIONS FOR THEM AS SEQUENTIAL TESTING PROCEDURES
FOR COMPUTER IMPLEMENTATION AND AS BOOLEAN FUNCTIONS.
AN ALGORITHM IS DEVELOPED WHICH IN A FINITE NUMBER OF
STEPS, WILL CONVERT ANY GIVEN LIMITED-ENTRY DECISION
TABLE INTO AN "OPTIMAL" COMPUTER PROGRAM, I.E. ONE
WITH MINIMUM AVERAGE PROCESSING TIME. THE ALGORITHM IS
MORE GENERAL THAN PROCEDURES PREVIOUSLY DEVELOPED AND
GUARANTEES OPTIMALITY OF THE RESULTING COMPUTER PRO-
GRAM.

COMPUTER IMPLEMENTATION OF THE ALGORITHM IS ALSO DIS-
CUSSED.

REI67A REINWALD, L.T. SOLAND, R.M.
CONVERSION OF LIMITED-ENTRY DECISION TABLES TO OPTIMAL
COMPUTER PROGRAMS II : MINIMUM STORAGE REQUIREMENT
JACM, 14(4)
OCTOBER 1967, 742-755.
FIRSTLY EDITED AS : RESEARSH PAPER NO. AD-648624
RESEARCH ANALYSIS CORPORATION
JANUARY 1967, 25 PP.

GIVEN THE NUMBER OF WORDS OF COMPUTER STORAGE REQUIRED
BY INDIVIDUAL TESTS IN A LIMITED-ENTRY DECISION TABLE,
IT IS SOMETIMES DESIRABLE TO FIND AN EQUIVALENT COMPUTER
PROGRAM WITH MINIMUM TOTAL STORAGE REQUIREMENT. IN THIS
PAPER AN ALGORITHM IS DEVELOPED TO DO THIS. THE RULES IN
THE DECISION TABLE ARE GROUPED INTO ACTION SETS, SO THAT
SEVERAL RULES WITH THE SAME ACTIONS NEED NOT BE DISTIN-
GUISHED. MOREOVER, IF CERTAIN COMBINATIONS OF CONDITIONS
CAN BE EXCLUDED FROM CONSIDERATION, THE ALGORITHM WILL
TAKE ADVANTAGE OF THIS EXTRA INFORMATION. THE ALGORITHM
IS INITIALLY DEVELOPED FOR COMPUTER PROGRAMS PROCESSING
A TREELIKE FORM AND THEN EXTENDED TO A WIDER CLASS OF
PROGRAMS. THE ALGORITHM CAN BE COMBINED WITH ONE WHICH
FINDS AN EQUIVALENT COMPUTER PROGRAM WITH MINIMUM AVE-
RAGE PROCESSING TIME, AND THUS USED TO FIND AN EQUIVA-
LENT COMPUTER PROGRAM WHICH MINIMIZES A COST FUNCTION
WHICH IS NONDECREASING IN BOTH AVERAGE PROCESSING TIME
AND TOTAL STORAGE REQUIREMENT.

REI67B REINWALD, L.T. WILLIAMS, J.S.
TAB40 FOR FORTRAN IV. A SELF-INSTRUCTIONAL COURSE
RESEARCH ANALYSIS CORPORATION
MC LEAN, JULY 1967.

THE INTRODUCTION OF THIS MANUAL BRIEFLY REVIEWS ADVAN-
TAGES OF DECISION TABLES AND QUOTES SOME PERFORMANCE
FIGURES. FURTHER SECTIONS DEAL WITH FUNDAMENTALS, PROGRAM
DESIGN USING DECISION TABLES, TAB40 DECISION TABLES AND

OPERATING STRATEGY.

- REI71 REINISCH, H. SCHWAIGER, A.
AUTOMATISIERTE ARBEITSPLANERSTELLUNG MIT
ENTSCHEIDUNGSTABELLEN
(G) AUTOMATIC MAN-POWER PLANNING BY MEANS OF DECISION TABLES
IBM NACHRICHTEN, 21(208)
OCTOBER 1971, 926-930.
THE MAN-POWER PLANNING IN A FOUNDRY IS DEALT WITH. THE DESIGNER UNIQUELY HAS TO CONSTRUCT A SYSTEM OF DECISION TABLES REPRESENTING THE LOGICAL PLANNING OF PRODUCTION UNITS. THE INDIVIDUAL PLANNING SCHEDULES ARE OBTAINED ON THE BASIS OF THE DIFFERENT PRODUCTION ORDERS.
- REI78 REINWALD, L.T.
COMMENTS ON A DYNAMIC PROGRAMMING ALGORITHM FOR CONVERTING DECISION TABLES INTO OPTIMAL PROGRAMS
CACM, 21(2)
FEBRUARY 1978, 179-180.
THE AUTHOR DISCUSSES THE DYNAMIC PROGRAMMING APPROACH TO THE CONVERSION PROBLEM AS PROPOSED IN SCH76. HE POINTS OUT SOME INACCURACIES IN THE LATTER PAPER.
- REI80 REINWALD, L.T.
DECISION TABLES AND STURCTURED PROGRAMMING
U.S. BUREAU OF THE CENSUS, STATISTICAL RESEARCH DIVISION
INTERNAL REPORT
1980, 52 PP.
THIS PAPER ENDEAVORS TO MAKE DECISION TABLES MORE Viable PROGRAMMING STRUCTURES BY PROPOSING A REVISION IN TRADITIONAL DECISION TABLE THEORY. THE PAPER INTRODUCES A GROUP OF NESTED IF-THEN-ELSE CONSTRUCTS AND THE WHILE CONDITION. THE LATTER ONE PERMITS TO REPEAT PART OF A DECISION TABLE. THE USE OF DECISION TABLES IN TOP DOWN PROGRAM VERIFICATION IS INVESTIGATED.
- RIC71A RICHARDS, P.A.
PROFESSIONAL DEVELOPMENT SEMINAR ON DECISION TABLES
AUSTRALIAN COMPUTER SOCIETY (OLD BRANCH)
BRISBANE, 1971.
- RIC71B RICHLING, C.
UNTERSUCHUNG DER ANWENDBARKEIT DER ENTSCHEIDUNGSTABELLEN IN DER INFORMATION UND DOKUMENTATION ANHAND DES BEISPIELS OPTO-ELEKTRONISCHER BAUELEMENTE
(G) AN INVESTIGATION OF THE APPLICABILITY OF DECISION TABLES IN INFORMATION AND DOCUMENTATION BY MEANS OF AN EXAMPLE (OPTO-ELECTRONIC ELEMENTS)
FORSCHUNGSBERICHT DER TH ILMENAU, SEKTION INER
1971.
- RIC75 RICHTER, J.E.
MODERNE DATENAUFNAHME-ORGANISATION
(G) MODERN DATA ACQUISITION ORGANIZATION
BURO, 23(3) AND 23(4)
MARCH 1975, 239-241 AND APRIL 1975, 394, 396, 398-399.
THESE PAPERS DISCUSS THE USE OF DECISION TABLES FOR PLAUSIBILITY CHECKS.
- RIE68 RIES, C.W.
DECISION TABLES
THE COMPUTER BULLETIN (SOUTH AFRICA)
NOVEMBER 1968.
- RII72 RIIHIMAKI, O.
FROM PRECEDENCE ANALYSIS TO DECISION TABLES - A PLAN FOR IMPLEMENTATION
DIFO-REPORT, UNIV. FO TAMPERE (FINLIND)
TAMPERE, DECEMBER 1972, 21 PP.
THE PRECEDENCE GRAPH, THE RESULT OF A PRELIMINARY INFORMATION ANALYSIS, CAN BE TRANSFORMED INTO ITS EQUIVALENT DECISION TABLE. A METHOD FOR PERFORMING THIS IS EXPLAINED.
- RIP74 RIPKEN, K.
ENTSCHEIDUNGSTABELLEN - HANDWERKZEUG FUR PROBLEMLOESUNGEN
(G) DECISION TABLES - A TOOL FOR PROBLEM SOLUTION
RECHNUNGSWESEN DATENTECHNIK ORGANISATION, 20(2)
FEBRUARY 1974, 68-73.

THE BASIC CONCEPTS OF THE DECISION TABLE TECHNIQUE ARE EXPLAINED. MOREOVER, THE CONSTRUCTION OF DECISION TABLES IS DEALT WITH IN DETAIL.

- ROB70 ROBINSON, F.
PROCESSING OF DECISION TABLES IN COBOL
COMPUTER WEEKLY, (222-223)
DECEMBER 1970, 6-7.
THE BASIC CONCEPTS OF THE DECISION TABLE TECHNIQUE ARE EXPLAINED AND THE POSSIBILITY TO GENERATE COBOL STATEMENTS FROM DECISION TABLES IS INVESTIGATED.
- ROM72 ROMERA, S.
TABLES DE DECISION ET PROGRAMMATION
(F) DECISION TABLES AND PROGRAMMING
INFORMATIQUE ET GESTION, (43)
1972, 66-71.
THE AUTHOR STARTS BY DISCUSSING DECISION TABLES, FLOW-CHARTS AND SUBPROGRAMS. AFTER THIS, SHE SHOWS IN A PRACTICAL WAY HOW TO DERIVE A PROGRAM FROM A DECISION TABLE.
- ROM75 ROMERA, S.
GENERATION AUTOMATIQUE DE PROGRAMMATION A L'AIDE DE TABLE DE DECISION
(F) AUTOMATIC PROGRAM GENERATION BY MEANS OF DECISION TABLES
INFORMATIQUE ET GESTION, (69)
JULY-AUGUST 1975, 25-30.
FIRST OF ALL, THE AUTHOR OUTLINES THE BASIC CONCEPTS OF THE DECISION TABLE TECHNIQUE. AFTER THIS, SHE DESCRIBES A SYSTEM THAT GENERATES A HIERARCHICAL PROGRAM STRUCTURE STARTING FROM THE DESCRIPTION OF THE DESIRED OUTPUT AND THE DECLARATION OF THE DIFFERENT LEVELS. AUTOMATICALLY GENERATED DECISION TABLES ALLOW THE PRODUCTION OF THE MISSING PROCEDURES.
- SAU72 SAUER, S.
ENTSCHEIDUNGSTABELLEN IN THEORIE UND PRAXIS
(G) DECISION TABLES IN THEORY AND PRACTICE
ZEITSCHRIFT FUR DATENVERARBEITUNG, 10(2)
MARCH 1972, 99-105.
THE AUTHOR POINTS OUT THAT DECISION TABLES ARE EXCELLENT TOOLS FOR SATISFYING MINIMUM STORAGE REQUIREMENTS. MOREOVER, THEY ENSURE MINIMUM UPDATING EFFORTS.
- SAU73 SAUER, S.
ENTSCHEIDUNGSTABELLEN - AUFBAU UND EINSATZ
(G) DECISION TABLES - CONSTRUCTION AND USE
BURO, 21(9)
SEPTEMBER 1973, 844-847.
DECISION TABLES ARE DEFINED. THE FORMULATION IS EXPLAINED ON THE BASIS OF A FOUR-QUADRANT SCHEME INVOKING IF/THEN OPTIONS FOLLOWED BY YES/NO ACTIONS RELATING TO THE SITUATION. APPLICATIONS OF DECISION TABLES DISCUSSED INCLUDE ANALYSIS, DOCUMENTATION AND COMMUNICATION. A COMPUTER SEQUENCE MAY BE BUILT UP USING SEVERAL DECISION TABLES LINKED BY IF/GO TO ... OPTIONAL TRANSFERS. THE ARTICLE CONCLUDES WITH A SURVEY OF THE MACHINE OPERATION STATUS AND TRENDS INVOLVING DECISION TABLES.
- SAWOO SAWYER, R.B.
MECHANIZATION OF DECISION TABLES
IBM TIE Z 77-4564
DATE UNKNOWN.
- SCH64 SCHMIDT, D.T. KAVANAGH, T.F.
USING DECISION STRUCTURE TABLES
DATAMATION, 10
PART I : (2), FEBRUARY 1964, 42-49
PART II : (3), MARCH 1964, 48-54.
ALSO IN : ADT, 1-19 & 20-31.
THESE ARTICLES EMPHASIZE MANUFACTURING APPLICATIONS OF THE DECISION TABLE TECHNIQUE. IT IS ARGUED THAT DECISION TABLES COUPLED WITH COMPUTERS ARE PAYING OFF BECAUSE THEY ALLOW YOU TO : DEFINE AND THINK THROUGH MANUFACTURING PROBLEMS, OFTEN PROVIDING NEW INSIGHTS AND UNDERSTANDING WHICH HAVE LED TO IMPROVED PERFORMANCE ; FORMULATE AND RECORD DECISION SYSTEMS FOR SUBSEQUENT USE AND COMMUNICATIONS ; SIMPLIFY COMPUTER IMPLEMENTATION WHERE MECHANIZATION IS DESIRABLE ; GET MANUFACTURING TO USING COMPUTERS.

THERE IS A WEALTH OF POTENTIAL COMPUTER APPLICATIONS IN MANUFACTURING. THEY OFFER GREAT OPPORTUNITY. IT IS EASY TO LEARN HOW TO USE DECISION TABLES, AND, FURTHER, THE USER REQUIRES MINIMUM COMPUTER KNOWLEDGE AND BACKGROUND. LATER IN THESE ARTICLES A DECISION TABLE APPLICATION USING COMPUTERS IS DESCRIBED (PRONTO).

- SCH65 SCHEFF, B.H.
USING A DECISION TABLE STRUCTURE AS THE INPUT LANGUAGE FORMAT FOR PROGRAMMING AUTOMATIC TEST EQUIPMENT SYSTEMS TRANS. IEEE ELECTRONIC COMPUTERS, EC 14 (SHORT NOTES) APRIL 1965, 248-250.
THIS ARTICLE SUGGESTS THE IDEA OF USING A DECISION TABLE AS INPUT SO AS TO BE ABLE TO PERFORM AUTOMATICALLY ANY SEQUENCE OF TESTS AND CHOOSE THE NEW TESTING SEQUENCE ON THE BASIS OF THE RESULTS.
- SCH68A SCHANZ, G.
PROGRAMMIEREN MIT ENTSCHEIDUNGSTAFELN
(G) PROGRAMMING WITH DECISION TABLES
UNPUBLISHED PAPER
VW-WERK AG
WOLFSBURG, MAY 1968, 29 PP.
THIS UNPUBLISHED INTERNAL PAPER DEALS WITH THE ELEMENTARY DECISION TABLE THEORY. CONSTRUCTING DECISION TABLES IS TREATED IN DETAIL.
- SCH68B SCHWARTZ, B.M.
DECISION TABLES IN PROGRAMMING AND AUTOMATIC PROGRAM DOCUMENTATION WITH A LISP IMPLEMENTATION
COURANT INSTITUTE OF MATHEMATICAL SCIENCES
NEW YORK, 1968.
THIS MASTER THESIS IS REFERRED TO IN SCH71A, WHICH SUMMARIZES CERTAIN SECTIONS OF IT.
- SCH71A SCHWARTZ, B.M.
LISP 1.5 DECISION TABLES IMPLEMENTED FOR A SERIAL COMPUTER AND PROPOSED FOR PARALLEL COMPUTERS
SIGPLAN NOTICES, 6(8)
SEPTEMBER 1971, 93-103.
THIS PAPER REVIEWS TWO SERIAL DECISION TABLE TRANSLATORS THAT HAVE BEEN IMPLEMENTED IN LISP 1.5 AND SUGGESTS HOW SIMILAR TRANSLATORS MIGHT BE IMPLEMENTED ON HIGHLY PARALLEL COMPUTERS. THE FIRST OF THE TWO SERIAL TRANSLATORS, DEFTAB, IS A STANDARD KIRK ALGORITHM PROCESSOR THAT ACCEPTS LIMITED-ENTRY DECISION TABLES AS INPUT AND DEFINES LISP FUNCTIONS AS ITS OUTPUT. THE SECOND SERIAL TRANSLATOR DESCRIBED, DECTAB, ACCEPTS LISP 1.5 FUNCTION DEFINITIONS AND GENERATES EQUIVALENT DECISION TABLES. FINALLY, A SET OF SIMILAR BUT PARALLEL FUNCTIONS IS DESCRIBED THAT COULD FORM THE BASIS OF A COMPILER OR INTERPRETER SYSTEM, THAT PRODUCES PARALLEL CODE FROM SERIAL PROGRAMS.
- SCH71B SCHIFFERER, O.
ABLAUFTABELLE VERSUS ENTSCHEIDUNGSTABELLE ALS HILFSMITTEL DER DOKUMENTATION
(G) CONTROL TABLES VERSUS DECISION TABLES IN THE DOCUMENTATION OF SYSTEMS
ANGEWANDTE INFORMATIK, 13(10)
OCTOBER 1971, 455-460.
THE MAIN ADVANTAGE OF THE DECISION TABLE IS THE CLEARNESS RESULTING FROM LIMITATIONS IMPOSED UPON THE CONDITION COLUMN. WITH GROWING COMPLEXITY OF PROBLEMS THIS ADVANTAGE IS LOST. THIS ARTICLE SHOWS THAT DECISION TABLES ARE ALSO APPLICABLE FOR PROGRAMS WITH A GREAT NUMBER OF CONDITIONS AND ACTIVITIES BY PARTLY USING TECHNIQUES OF CONTROL TABLES.
- SCH72 SCHICK, I.
VORZUEGE UND ANWENDUNG VON ENTSCHEIDUNGSTABELLEN
(G) ADVANTAGES AND APPLICATION OF DECISION TABLES
INFORMATIK, 19(1)
1972, 40-42.
THE AUTHOR DISCUSSES THE ADVANTAGES GAINED BY USING DECISION TABLES INSTEAD OF FLOWCHARTS. THE AUTOMATIC PROBLEM SOLUTION BY MEANS OF DECISION TABLES IS MENTIONED.

- SCH73 SCHUPP, W. HAAS, P.
 STRUKTURIERTE PROGRAMMENTWICKLUNG DURCH
 ENTSCHEIDUNGSTABELLEN-TECHNIK UND NORMIERTE PROGRAMMIERUNG
 (G) STRUCTURED PROGRAM DEVELOPMENT BY MEANS OF THE DECISION
 TABLES TECHNIQUE AND STANDARDISED PROGRAMMING
 BUREO, 21(11)
 NOVEMBER 1973, 1130-1135.
- THIS ARTICLE DESCRIBES A RATIONALISED METHOD OF PROGRAMMING WHICH COMBINES THE LIMITED ENTRY DECISION TABLE TECHNIQUE WITH STANDARDISED PROGRAMMING IN ONE UNIFIED SYSTEM CALLED GOAL, A PRODUCT OF SOFTLAB, MUNICH.
- SCH74 SCHUPP, W.
 GOAL - EIN MODERNES SOFTWAREWERKZEUG ZUR STRUKTURIERTEN
 PROGRAMMENTWICKLUNG
 (G) POSSIBILITIES OF THE SOFTWARE SYSTEM GOAL FOR STRUCTURED
 PROGRAM DEVELOPMENT
 OUTPUT, 3(6)
 1974, 30-35.
- THE AUTHOR DEALS WITH THE DEVELOPMENT OF THE DECISION TABLE TECHNIQUE AND OF STRUCTURED PROGRAMMING. BOTH TECHNIQUES ARE COMBINED IN A SOFTWARE-PACKAGE CALLED GOAL. THE AUTHOR GOES ON TO DESCRIBE THIS PROGRAMMING TOOL AS FOR BENEFITS, PRACTICAL EXPERIENCES AND TECHNICAL INFORMATION.
- SCH76 SCHUMACHER, H. SEVCIK, K.C.
 THE SYNTHETIC APPROACH TO DECISION TABLE CONVERSION
 CACM, 19(6)
 JUNE 1976, 343-351.
- STARTING FROM A DECISON TABLE, DYNAMIC PROGRAMMING IS USED TO SYNTHESIZE AN OPTIMAL DECISION TREE FORM WHICH A PROGRAM CAN BE CREATED. USING THIS APPROACH, THE EFFICIENCY OF CREATING AN OPTIMAL PROGRAM IS INCREASED SUBSTANTIALLY, PERMITTING GENERATION OF OPTIMAL PROGRAMS FOR DECISION TABLES WITH UP TO TEN CONDITIONS.
- ALSO SEE : CHV76 AND REI78.
- SC071 SCOWEN, R.
 THE USE OF DECISION TABLES IN BABEL
 SIGPLAN NOTICES, 6(8)
 SEPTEMBER 1971, 54-68.
- THIS PAPER DISCUSSES AND ILLUSTRATES THE INSERTION OF DECISION TABLES IN BABEL, A PROGRAMMING LANGUAGE DEVELOPED BY THE SAME AUTHOR.
- SED72 SEDLAR, M.
 (C) SEPECIAL APPLICATIONS OF DECISION TABLES
 AUTOMATIZACE, (7)
 1972, 178-179.
- THE RELATIONS BETWEEN SYSTEMS AND THEIR SURROUNDINGS ARE DESCRIBED USING DECISION TABLES.
- SEL76 SELEBORG, C.E.
 BESLUTSTABELLER OCH STRUKTURERAD PROGRAMMERING
 (S) DECISION TABLES AND STRUCTURED PROGRAMMING
 PAPER PRESENTED AT NORDDATA 76
 HELSINKI, 1976.
- SES67A SESAGIRI, N.
 RELAY TREE NETWORK DECOMPOSITION OF DECISION TABLES
 PROC. IEEE, 55(9)
 SEPTEMBER 1967, 1648-1649.
- IN THIS LETTER, AN ANALOGY IS ESTABLISHED BETWEEN THE MINIMIZATION PROBLEM OF SOFTWARE DESIGN BASED ON DECISION TABLES AND THE PROBLEM OF MINIMIZING RELAY TREE SWITCHING NETWORKS. THE ANALOGY FACILITATES THE ADOPTION FOR SOFTWARE DESIGN OF A LARGE NUMBER OF TECHNIQUES DEVELOPED FOR THE DESIGN OF HARDWARE.
- SES67B SESAGIRI, N.
 DECISION TABLE APPROACH TO SELF-DIAGNOSTIC COMPUTERS
 PROC. IEEE, 55(12)
 DECEMBER 1967, 2180-2181.
- THE STRUCTURE OF A DECISION TABLE WHICH IMPARTS SELF-DIAGNOSTIC FEATURES TO DIGITAL COMPUTERS ALONG WITH A METHOD OF IMPLEMENTING IT IS DEVELOPED IN THIS LETTER. APART FROM CONSIDERING THE RELATIVE FREQUENCY OF FAILURE IN THE SYSTEM THE PRESENT PROCEDURE INCLUDES AN ON-LINE

ASSESSMENT OF THE SEQUENCE OF EXECUTION OF THE TESTS.

- SHA65 SHAW, C.J.
DECISION TABLES - AN ANNOTATED BIBLIOGRAPHY
SYSTEM DEVELOPMENT CORPORATION, MEMO TM-2288/000/00
SANTA MONICA, DECEMBER 1965
ALSO IN : ACM DETAB/65 PREPROCESSOR (ACM66).
THIS BIBLIOGRAPHY CONTAINS A BRIEF INTRODUCTION AND 42 REFERENCES, MOSTLY ANNOTATED, TO THE SUBJECT. MANY OF THE ANNOTATIONS ARE COPIES OF ABSTRACTS APPEARING AT THE TOP OF ARTICLES.
- SHA72 SHARMAN, G.C.H.
RECURSIVE DECISION TABLE TRANSLATOR
IBM TDB, 14(12)
MAY 1972, 3747-3748.
A SIMPLE ALGORITHM FOR TRANSLATING A DECISION TABLE INTO A NARRATIVE PROCEDURE IS GIVEN.
- SHA73A SHARMAN, G.C.H.
INITIALIZATIONS IN DECISION TABLES
IBM TDB, 16(1)
JUNE 1973, 193-194.
THE AUTOR INTRODUCES "INITIALIZATIONS", I.E. ACTIONS THAT MUST BE PERFORMED BEFORE A CONDITION TEST CAN BE MADE OR BETWEEN TWO TESTS IN THE TREE.
- SHA73B SHARMAN, G.C.H.
LOOPS IN DECISION TABLES
IBM TDB, 16(1)
JUNE 1973, 195-197.
IT IS SHOWN THAT PROGRAM LOOPS MAY BE REPRESENTED BY MEANS OF DECISION TABLES AND THAT DECISION TABLES MAY FORM ELEMENTS OF STRUCTURED PROGRAMS.
- SH066 SHOBER, J.A.H.
DECISION TABLES FOR BETTER MANAGEMENT SYSTEMS
SYSTEMS AND PROCEDURES JOURNAL, 17(2)
MARCH-APRIL 1966, 28-32.
SOME ADVANTAGES OF DECISION TABLES IN PROBLEM SOLVING AND DOCUMENTATION ARE DESCRIBED AND ILLUSTRATED BY MEANS OF EXAMPLES MAINLY TAKEN FROM MEDICAL PRODUCT MARKET RESEARCH.
- SHW71 SHWAYDER, K.
CONVERSION OF LIMITED-ENTRY DECISION TABLES TO COMPUTER PROGRAMS - A PROPOSED MODIFICATION TO POLLACK'S ALGORITHM
CACM, 14(2)
FEBRUARY 1971, 69-73.
POLLACK (POL64A) HAS PROPOSED AN ALGORITHM FOR CONVERTING DECISION TABLES INTO FLOWCHARTS WHICH MINIMIZE SUBSEQUENT EXECUTION TIME WHEN COMPILED INTO A COMPUTER PROGRAM. TWO MODIFICATIONS TO THIS ALGORITHM ARE PROPOSED. THE FIRST RELIES ON SHANNON'S NOISELESS CODING THEOREM AND THE COMMUNICATIONS CONCEPT OF ENTROPY BUT DOES NOT COMPLETELY TEST THE ELSE RULE. THE SECOND MODIFICATION COMPLETELY TESTS THE ELSE RULE BUT RESULTS IN MORE EXECUTIONS THAN THE FIRST MODIFICATION. BOTH MODIFICATIONS RESULT IN LOWER EXECUTION TIME THAN POLLACK'S ALGORITHM. HOWEVER, NEITHER MODIFICATION GUARANTEES A GLOBALLY OPTIMAL SOLUTION.
- SHW72 SHWAYDER, K. KENNEY, A.A. AINSLIE, R.U.
DECISION TABLES - A TOOL FOR TAX PRACTITIONERS
THE TAX ADVISER, 3(6)
JUNE 1972, 336-345.
- SHW74 SHWAYDER, K.
EXTENDING THE INFORMATION THEORY APPROACH TO CONVERTING LIMITED-ENTRY DECISION TABLES TO COMPUTER PROGRAMS
CACM, 17(9)
SEPTEMBER 1974, 532-537.
THIS PAPER MODIFIES AN EARLIER ALGORITHM (SHW71) FOR CONVERTING DECISION TABLES INTO FLOWCHARTS WHICH MINIMIZE SUBSEQUENT EXECUTION TIME WHEN COMPILED INTO A COMPUTER PROGRAM. THE ALGORITHMS CONSIDERED IN THIS PAPER PERFORM LIMITED SEARCH AND, ACCORDINGLY, DO NOT NECESSARILY RESULT IN GLOBALLY OPTIMAL SOLUTIONS. HOWEVER, THE GREATER SEARCH EFFORT NEEDED TO OBTAIN A GLOBALLY OPTIMAL

SOLUTION FOR COMPLEX DECISION TABLES IS USUALLY NOT JUSTIFIED BY SUFFICIENT SAVINGS IN EXECUTION TIME. THERE IS AN ANALOGY BETWEEN THE PROBLEM OF CONVERTING DECISION TABLES INTO EFFICIENT FLOWCHARTS AND THE WELL-UNDERSTOOD PROBLEM IN INFORMATION THEORY OF NOISELESS CODING. THE RESULTS OF THE NOISELESS CODING LITERATURE ARE USED TO EXPLORE THE LIMITATIONS OF ALGORITHMS USED TO SOLVE THE DECISION TABLE PROBLEM. THE ANALOGY BETWEEN THE TWO PROBLEMS IS ALSO USED TO DEVELOP IMPROVEMENTS TO THE INFORMATION ALGORITHM IN EXTENDING THE DEPTH OF SEARCH UNDER CERTAIN CONDITIONS AND IN PROPOSING ADDITIONAL CONDITIONS TO BE ADDED TO THE DECISION TABLE. FINALLY, THE INFORMATION ALGORITHM IS COMPARED WITH AN ALGORITHM PROPOSED IN A PAPER BY VERHELST (VER72A).

- SHW75 SHWAYDER, K.
COMBINING DECISION RULES IN A DECISION TABLE
CACM, 18(8)
AUGUST 1975, 476-480.
THE TECHNIQUES FOR MINIMIZING LOGIC CIRCUITS ARE APPLIED TO THE SIMPLIFICATION OF DT'S BY COMBINING DECISION RULES. THIS METHOD IS LOGICALLY EQUIVALENT TO THE QUINE-MC CLUSKEY METHOD FOR FINDING PRIME IMPLICANTS. IF SOME OF THE DECISION RULES IMPLIED IN THE ELSE RULE OCCUR WITH LOW FREQUENCY, THEN THE ELSE RULE CAN BE USED TO FURTHER SIMPLIFY THE DECISION TABLE.
- SIE00 SIEMENS
FORTET VERSION 1 (FORTRAN-ORIENTIERTE ENTSCHEIDUNGSTABELLEN-VORUEBERSETZER)
(G) FORTET VERSION 1 (FORTRAN-ORIENTED DECISION TABLE PREPROCESSOR)
REFERENCE NUMBER 643/11/1810/1
DATE UNKNOWN.
- SIL71A SILBERG, B.
SPECIAL ISSUE ON DECISION TABLES
SIGPLAN NOTICES, 6(8)
SEPTEMBER 1971, 111 PP.
THIS SPECIAL ISSUE OF SIGPLAN NOTICES, EDITED BY BRUCE SILBERG, CONTAINS 15 ARTICLES DEALING WITH VARIOUS ASPECTS OF THE SUBJECT. IN THIS BIBLIOGRAPHY, THEY ARE SEPARATELY ANNOTATED.
- SIL71B SILBERG, B.
DECISION TABLE BIBLIOGRAPHY
SIGPLAN NOTICES, 6(8)
SEPTEMBER 1971, 1-4.
THIS "CATALOG OF READINGS ON THE DECISION TABLE PROGRAMMING/ANALYSIS TECHNIQUE" COVERS ABOUT 100 TITLES. FOR EACH CITATION, THE ENTRY IN COMPUTING REVIEWS (IF ANY) IS PROVIDED.
- SIL71C SILBERG, B.
DETAB/65 IN THIRD GENERATION COBOL
SIGPLAN NOTICES, 6(8)
SEPTEMBER 1971, 4-8.
THIS PAPER INDICATES THE HISTORY OF DECISION TABLES, REVIEWS THEIR STRUCTURE, ADVANTAGES AND DISADVANTAGES, AND PRESENTS THE MODIFICATIONS TO DETAB/65 THAT ARE NEEDED TO ALLOW ITS COMPILEATION BY COBOL-65 OR ANSI COBOL.
- SIM67 SIMPSON, J.A.D.
SAPTAO AND PET
GUIDE
MAY 1967.
- SMI00 SMITH, R.G. CONROW, K.
A DECISION TABLE ALGORITHM BASED ON LIST-PROCESSING TECHNIQUES
KANSAS STATE UNIVERSITY
MANHATTAN, NO DATE, 52 PP.
A CONTROL MECHANISM UTILIZING LIST PROCESSING TECHNIQUES HAS BEEN DEVISED TO FACILITATE THE EXECUTION OF DECISION TABLES. A DIRECTORY VECTOR CONTROLS THE FLOW OF EXECUTION AMONG CONDITION AND ACTION STUBS, FROM ONE TABLE TO ANOTHER IN A BLOCK OF TABLES, AND THROUGH WHATEVER DIAGNOSTIC MODULES MIGHT BE REQUIRED IN A GIVEN EXECUTION

THE CHOICE OF A STUB FOR EXECUTION VIA THE DIRECTORY VECTOR IMPLIES THAT ANY STUB NEED BE CODED ONLY ONCE IN THE BLOCK OF DECISION TABLES, WITH ALL OTHER USES OF THAT STUB ACHIEVED VIA REFERENCE TO THE SINGLE ENCODEMENT. THIS FEATURE ENABLES DRAMATIC REDUCTION IN THE SIZE OF AN OBJECT MODULE IN FAVORABLE CASES. TRANSLATOR COMMANDS EFFECTING TRANSFER OF CONTROL AMONG THE SEVERAL TABLES OF A BLOCK OF DECISION TABLES ARE IMPLEMENTED WITHIN THE DIRECTORY VECTOR, AND GIVE MORE FREEDOM IN LINKING TABLES, THAN HAS BEEN CUSTOMARY. INTRODUCTION OF A TRACE FEATURE AND VARIOUS LEVELS OF DIAGNOSTIC ASSISTANCE ARE LIKEWISE READILY ACCOMODATED.

- SMI62 SMITH, E.H.
MANAGEMENT RULE TABLES
ADMINISTRATIVE SYSTEMS ASSOCIATION JOURNAL
SEPTEMBER 1962.
- SMI69 SMITH, B.M.
DECISION TABLES REVISITED
PROC. FOURTH AUSTRALIAN COMPUTER CONFERENCE
ADELAIDE, 1969.
- SMI75 SMILLIE, K.W. SHAVE, M.J.R.
CONVERTING DECISION TABLES TO COMPUTER PROGRAMS
THE COMPUTER JOURNAL, 18(2)
MAY 1975, 108-111.
THIS PAPER PRESENTS A METHOD OF IMPLEMENTING THE LOGIC OF A DECISION TABLE IN A COMPUTER PROGRAM WHICH IS SIMPLER AND MORE EFFICIENT THAN A PROGRAM DERIVED FROM THE RULE MASK METHOD. THE METHOD, BASED ON DECODING THE CONDITION BODY ENTRIES TO A MIXED BASE, INCORPORATES TECHNIQUES FOR MAKING CERTAIN ENTRIES IN THE DECISION TABLE MORE SPECIFIC, AND ENABLES THE LOGIC OF THE TABLE AND THE VALIDITY OF ANY STATED SET OF CONDITION ENTRIES TO BE CHECKED MORE EASILY.
- SOC67 SOCCI, R.J. FLERI, D.A.
RELAY TREE NETWORK DECOMPOSITION OF DECISION TABLES
PROC. IEEE
SEPTEMBER 1967, 1648-
- SOL64 SOLOMON, M.J.
REMARKS ON FUTURE POSSIBILITIES OF DECISION TABLES
SEMINAR ON DECISION TABLES
SEPTEMBER 1964.
- SOROO SORSET, M.J.
HOW TO WRITE A DECISION TABLE
IBM TIE Z 77-7025
DATE UNKNOWN.
- SPA67 SYSTEMS & PROCEDURES ASSOCIATION
DECISION TABLE USAGE IN THE SYSTEMS AND PROCEDURES FUNCTION
UNDATED BUT NO EARLIER THAN JANUARY 1967.
THIS IS THE REPORT OF A SURVEY APPARENTLY CARRIED OUT IN THE PERIOD 1966-1967 BY THE SPA INTERNATIONAL RESEARCH COMMITTEE. THE SURVEY RESULTS, CHIEFLY IN TABLE FORMAT, ANALYSE THE ACCEPTANCE AND USE OF THE TECHNIQUE WITHIN THE SPA MEMBERSHIP. SPECIFIC APPLICATIONS IN WHICH DECISION TABLES ARE USED FOR ANALYSIS AND DOCUMENTATION ARE DISCUSSED. THE REPORT ENDS WITH A SHORT UNANNOTATED BIBLIOGRAPHY.
- SPI75 SPIEGEL, W.
(G) DECISION TABLE GENERATORS FOR PL/I
ELEKTR. RECHENANLAGEN, 17(6)
DECEMBER 1975, 293-299.
THE DECTAT, DETAB/GT AND VORELLE PREPROCESSORS FOR GENERATING PL/I CODE ARE INVESTIGATED.
- SPR66 SPRAGUE, V.G. POLLACK, S.L.
ON STORAGE SPACE OF DECISION TABLES
CACM, 9(5)
MAY 1966, 319-320.
IN THIS LETTER TO THE EDITOR, SPRAGUE DEMONSTRATES THAT NEITHER OF THE TWO ALGORITHMS PROPOSED BY POLLACK (POL64A) ALWAYS SUCCEEDS IN MINIMIZING STORAGE SPACE OR EXECUTION TIME.

- SPR70 SPROWLS, R.C.
 THE USE OF DECISION TABLES IN PREPARING COURSE MATERIALS
 FOR CAI
 IFIP WORLD CONFERENCE ON COMPUTER EDUCATION
 AMSTERDAM, 24-28 AUGUST 1970, III/119-122.
 THE PURPOSE OF THIS PAPER IS TO PRESENT DECISION TABLES
 AS A POTENTIAL METHODOLOGY FOR PREPARING CAI (COMPUTER
 AIDED INSTRUCTION) COURSE MATERIALS, BECAUSE (1) THEY
 EXPRESS COMPLEX DECISION LOGIC, (2) THEY ARE INDEPENDENT
 OF COMPUTER EQUIPMENT AND PROGRAMMING LANGUAGE
 FACILITIES, AND (3) THEY CONSTITUTE A GOOD
 DOCUMENTATION SYSTEM PRESENTING DETAILS OF WHAT IS DONE
 IN A PARTICULAR LESSON OR LESSON SEGMENT.
- STA00A STAHL, H.
 UEBUNG : ENTSCHEIDUNGSTABELLE ZAEHLERBLOCK
 (G) EXERCISE : A DECISION TABLE COUNTING BLOCK
 FORSCHUNGSBERICHT DER SEKTION INFORMATIONSVERARBEITUNG
 TU DRESDEN, DATE UNKNOWN.
 THIS EXERCISE DEALS WITH THE CONSTRUCTION OF A DECISION
 TABLE REPRESENTING A COUNTING BLOCK. THE PROPOSED
 DECISION TABLE IS PROCESS-ORIENTED.
- STA00B STATISTISKA CENTRALBYRAN
 BESLUTSTABELLER
 (S) DECISION TABLES
 PLACE OF ISSUE AND DATE UNKNOWN.
- STA67A STATISTISKA CENTRALBYRAN
 BESLUTSTABELLER VID TABELLAGESBESKRIVNING
 (S) DECISION TABLES AS DESCRIPTIVE TOOLS IN TABLE FORMAT
 RAPPORT 21.12.67 (SVEN FELME)
 THIS IS REPORTED ELSEWHERE (WIL69) TO BE A RATHER
 PESSIMISTIC REVIEW OF AN EXPERIENCE IN USING DECISION
 TABLES, ESPECIALLY THE DETAB/65 PREPROCESSOR.
- STA67B STAUB, P.E.
 DECISION TABLES : A SYSTEM DESIGNER'S CONTROL OF
 COMPUTER PROGRAMS
 SYMPOSIUM ON DECISION TABLES AND AD HOC'S
 USASI X3.6.7/3.4.2D SPRING JOINT
 COMPUTER CONFERENCE
 APRIL 1967.
- STA74 STAHL, H.
 ENTSCHEIDUNGSTABELLEN ALS DARSTELLUNGSMITTEL ABSTRAKTER
 AUTOMATEN
 (G) DECISION TABLES AS A MEANS FOR REPRESENTING ABSTRACT AUTOMATA
 WISSENSCHAFTLICHE ZEITSCHRIFT DER TU DRESDEN, 23(1)
 1974, 159-162.
 THE AUTHOR DERIVES THE DECISION TABLE TECHNIQUE FROM
 THE AUTOMATA THEORY. THE RESULTING DECISION TABLES,
 THOUGH DIFFERENT FROM THE CLASSICAL FORMAT, ARE
 SUCCESSFULLY APPLICABLE IN PROCESS REGULATION PROBLEMS.
- STC70 ST. CLAIR, P.R.
 DECISION TABLES CLEAR THE WAY FOR SHARP SELECTION
 COMPUTER DECISIONS, 12(2)
 FEBRUARY 1970, 14-18.
- STE71 STERBENZ, R.F.
 TABSOL DECISION TABLE PREPROCESSOR
 SIGPLAN NOTICES, 6(8)
 SEPTEMBER 1971, 33-40.
 THIS PAPER DESCRIBES THE TABSOL DECISION TABLE PREPRO-
 CESSOR. THE TABSOL (TABULAR SYSTEM ORIENTED LANGUAGE)
 PREPROCESSOR, ITSELF WRITTEN IN ANSI COBOL, IS DESIGNED
 TO TRANSLATE HORIZONTAL-RULE, EXTENDED-ENTRY DECISION
 TABLES INTO EQUIVALENT COBOL STATEMENTS. DESIGN OBJEC-
 TIVES, LANGUAGE FORMAT AND IMPLEMENTATION RESULTS OF THE
 TABSOL PREPROCESSOR ARE SUMMARIZED.
- STE74 STEPANOWICZ, B.
 (P) CONVERSION OF DECISION TABLES INTO COMPUTER PROGRAMS
 INFORMATYKA, 10(3)
 MARCH 1974, 6-8.
 THE ARTICLE PRESENTS A METHOD OF WRITING COMPUTER
 PROGRAMS BASED ON LIMITED ENTRY DECISION TABLES.

- STR70A STRUNZ, H.
 EINE METHODE ZUR ZERGLIEDERUNG VON ENTSCHEIDUNGSTABELLEN
 (G) A METHOD FOR THE PARSING OF DECISION TABLES
 BIFOA-ARBEITSBERICHT, 70/12
 KOELN, 1970.
- STR70B STRUNZ, H.
 ENTSCHEIDUNGSTABELLEN UND IHRE ANWENDUNG BEI
 SYSTEMPLANUNG, -IMPLEMENTIERUNG UND -DOKUMENTATION
 (G) DECISION TABLES AND THEIR APPLICATION IN PLANNING,
 IMPLEMENTING AND DOCUMENTING DATA PROCESSING SYSTEMS
 ELEKTRONISCHE DATENVERARBEITUNG, 12(2)
 1970, 56-65.
 THIS PAPER PRESENTS THE FUNDAMENTAL PRINCIPLES OF
 DECISION TABLES AND GIVES SOME INDICATION OF THE
 APPLICABILITY OF THIS TECHNIQUE AS A TOOL FOR PLANNING,
 IMPLEMENTING AND DOCUMENTING BUSINESS DATA PROCESSING
 SYSTEMS.
- STR71A STRUNZ, H.
 HANDBUCH DER ENTSCHEIDUNGSTABELLENTECHNIK
 (G) DECISION TABLE TECHNIQUE : A MANUAL
 MATHEMATISCHER BERATUNGS- UND PROGRAMMIERUNGSDIENST
 DORTMUND, 1971.
- STR71B STRUNZ, H.
 EINE METHODE ZUR ZERGLIEDERUNG VON ENTSCHEIDUNGSTABELLEN
 (G) A METHOD FOR THE PARSING OF DECISION TABLES
 ANGEWANDTE INFORMATIK, 13(3)
 1971, 117-122.
 THIS PAPER IS CONCERNED WITH A PARTIAL PROBLEM FROM THE
 FIELD OF ANALYZING A DATA PROCESSING PROBLEM BY USING
 THE DECISION TABLE TECHNIQUE : THE PARSING OF DECISION
 TABLES. A METHOD IS DESCRIBED THAT ALLOWS THE SPLITTING
 UP OF A COMPLEX DECISION SITUATION IN SUCH A WAY THAT
 IT IS EASY TO DOCUMENT IT BY SEVERAL LINKED DECISION
 TABLES.
- STR73A STRUNZ, H.
 ENTSCHEIDUNGSTABELLENTECHNIK - EIN HILFSMITTEL DER
 GESTALTUNG VON SYSTEMEN DER AUTOMATISIERTEN
 DATENVERARBEITUNG
 (G) DECISION TABLE TECHNIQUES AS AN AID FOR SYSTEMS ANALYSIS
 IN EDP-APPLICATIONS
 ZEITSCHRIFT FUR ORGANISATION
 PART I : 42(2), 1973, 84-89
 PART II : 42(3), 1973, 172-177.
 THE AUTHOR FIRSTLY ILLUSTRATES THE BASIS OF THE DECISION
 TABLE TECHNIQUE BY MEANS OF SIMPLE EXAMPLES. IN THE
 SECOND PART, HE GOES ON TO DISCUSS SOME ALTERNATIVE
 DECISION TABLE FORMS. THE CONSTRUCTION AND THE MINIMI-
 ZATION OF DECISION TABLES ARE BRIEFLY DEALT WITH.
 IT IS INDICATED HOW DECISION TABLES CAN BE APPLIED
 BENEFICIALLY IN THE SUCCESSIVE PHASES OF THE SYSTEMS
 DESIGN PROCESS.
- STR73B STRUNZ, H.
 THE DEVELOPMENT OF DECISION TABLES VIA PARSING OF COMPLEX
 DECISION SITUATIONS
 CACM, 16(6)
 JUNE 1973, 366-369.
 A NEW PARSING TECHNIQUE IS PROPOSED WHICH ALLOWS PARSING
 BASED ONLY ON SYNTACTICAL CHARACTERISTICS OF THE DECISION
 PROBLEM. IT REQUIRES A DESCRIPTION OF THE PROBLEM IN
 DECISION GRID CHART FORMAT AND ALLOWS THE DEVELOPMENT
 OF DECISION TABLES WITHIN DEFINED LIMITS BY AVOIDING,
 OR AT LEAST MINIMIZING, REPETITION OF CONDITIONS AND
 ACTIONS IN THE RESULTING TABLES.
- STR73C STRUNZ, H.
 ENTSCHEIDUNGSTABELLEN ALS HILFSMITTEL DER BESCHREIBUNG UND
 ANALYSE VON ARBEITSLAUFEN
 (G) DECISION TABLES AS AN AID FOR ANALYSIS IN THE WORK-
 SITUATION
 HANDBUCH DER MODERNEN DATENVERARBEITUNG, 10(50)
 MARCH 1973, 1.1/2/3-8.1/2/3.
 TOPICS TREATED ARE : DECISION TABLES AS AN ALTERNATIVE
 PROBLEM STATEMENT TECHNIQUE, DECISION TABLE LOGIC,

ELEMENTS AND FORMAT, THE CONSTRUCTION AND ANALYSIS OF
DECISION TABLES AND THE AREA OF APPLICATION.

- STR75A STRUNZ, H.
ENTSCHEIDUNGSTABELLENTECHNIK ODER STRUKTURIERTE
PROGRAMMIERUNG
(G) DECISION TABLE TECHNIQUES OR STRUCTURED PROGRAMMING
ONLINE, 13(3)
1975, 114-115.
THE AUTHOR COMPARES THE DECISION TABLE TECHNIQUE WITH
STRUCTURED PROGRAMMING. FOR THIS, HE TAKES THE VIEW OF
THE SKILLED USER OF THE DECISION TABLE TECHNIQUE AS
WELL AS THE VIEW OF THE UNEXPERIENCED USER.
- STR75B STRUNZ, H.
GRUNDLAGEN UND ANWENDUNGSMOEGLICHKEITEN DER ENTSCHEIDUNGSTABELLENTECHNIK BEI DER GESTALTUNG RECHNERGESTUETZTER INFORMATIONSSYSTEME - UNTER BESONDRE BERUECKSICHTIGUNG PROGRAMMTECHNISCHER ASPEKTE
(G) FUNDAMENTALS AND APPLICATIONS OF DECISION TABLE TECHNIQUES FOR THE DEVELOPMENT OF COMPUTER-ASSISTED INFORMATION SYSTEMS - WITH SPECIAL CONSIDERATION OF PROGRAMMING ASPECTS
PH. D. DISSERTATION,
UNIVERSITY OF KOELN (W-GERMANY),
1975, 341 PP.
THIS DISSERTATION IN ITS FIRST PART DISCUSSES THE
FUNDAMENTAL THEORY OF SINGLE HIT AND MULTIPLE HIT
DECISION TABLES. A DECISION TABLE IS VIEWED AS A FORMAL
STRUCTURE, AND SYNTAX, SEMANTICS AND PRAGMATICS OF THIS
FORMAL STRUCTURE ARE ANALYZED.
IN ITS SECOND PART THE DISSERTATION RELATES DECISION
TABLES TO SYSTEMS PLANNING (ANALYSIS AND DESIGN), SYSTEMS
IMPLEMENTATION AND SYSTEMS APPLICATION. THE ISSUES
DISCUSSED INCLUDE APPLICATION OF GRAPH THEORETICAL
METHODS TO THE PROBLEM OF THE DEVELOPMENT OF DECISION
TABLES AND THE RELATION OF DECISION TABLE TECHNIQUES TO
OTHER SOFTWARE ENGINEERING TECHNIQUES SUCH AS MODULAR
PROGRAMMING AND STRUCTURED PROGRAMMING.
- STR75C STRAUCH, D.
LEITUNGSBEURTEILUNG ZUR AUSWAHL VON GENERATOREN
(G) PERFORMANCE COMPARISON FOR SELECTING PROGRAM GENERATORS
BURO, 23(9)
SEPTEMBER 1975, 878, 880, 882, 884-892.
PROGRAM GENERATORS (A.O. DECISION TABLE PREPROCESSORS)
ARE EVALUATED. A TABULAR SURVEY IS PROVIDED.
- STR76 STRUNZ, H. JORGENSEN, P.C.
ANWENDUNG GRAPHEN THEORETISCHER VERFAHREN IN DER
ENTSCHEIDUNGSTABELLENTECHNIK
(G) APPLICATION OF GRAPH-THEORETICAL METHODS IN THE DECISION
TABLES TECHNIQUE
ANGEWANDTE INFORMATIK, 18(2)
FEBRUARY 1976.
THE REPRESENTATION OF DECISION TABLES AND DECISION TABLE
NETWORKS BY LINEAR GRAPHS IS PRESENTED AND APPLIED TO
TWO PROBLEMS IN THE DECISION TABLES TECHNIQUE :
PARTITIONING OF A DECISION TABLE AND RE-ORDENING A
DECISION TABLE NETWORK TO MINIMIZE BACKWARD REFERENCES.
- STR77 STRUNXZ, H.
ENTSCHEIDUNGSTABELLENTECHNIK : GRUNDLAGEN UND ANWENDUNGSMOEGLICHKEITEN BEI DER GESTALTUNG RECHNERGESTUETZTER INFORMATIONSSYSTEME
(G) FUNDAMENTALS AND APPLICATIONS OF THE DECISION TABLE
TECHNIQUE FOR THE DEVELOPMENT OF COMPUTER-ASSISTED
INFORMATION SYSTEMS
HANSER
MUECHEN, 1977, 344 PP.
SEE STR75B.
- SUE00 SUELFLOHN, R.A.
DECISION TABLES FOR COMMUNICATING SYSTEMS LOGIC
IBM TIE 6309-03790
DATE UNKNOWN.
- SUN00 SUN OIL
DECISION TABLES FOR COMPUTER SYSTEM DESIGN AND PROGRAMMING
INSTRUCTIONAL COURSE

DATE UNKNOWN.

- SUN68 SUN OIL
SURVEY OF DECISION TABLE EFFECTIVENESS
INTER-OFFICE CORRESPONDENCE
AUGUST 1968.
- SUTOO SUTHERLAND COMPANY
MANAGEMENT DECISION STRUCTURED TABLES
SUTHERLAND COMPANY, MANAGEMENT CONSULTANTS
PEORIA, DATE UNKNOWN.
- SY500 SYSTOR
DETAB/GT-PROZESSOR PL/1 - BENUTZERHANDBUCH
(G) DETAB/GT-PROCESSOR - MANUAL
DATE UNKNOWN.
- TAN74 TANAJEW, W.S. POWARITSCH, M.P.
(R) SYNTHESIS OF GRAPHS FOR DECISIONAL ALGORITHMS
SCIENCE AND TECHNIQUE
MINSK, 1974.
IN THIS BOOK, THE DECISION TABLE FORM IS USED TO
REPRESENT SOME DISCRETE FUNCTIONS. THESE FUNCTIONS ARE
IMPORTANT IN BASIC PROGRAM OPTIMIZATION THEORY. MANY
PRACTICAL EXAMPLES ARE PROVIDED.
- TAY68 TAYLOR, H.
DECISION LOGIC TABLE TECHNIQUE FOR COMPUTER SYSTEMS
HIRSCHFELD PRESS
DENVER, 1968, 44 PP.
THIS MONOGRAPH PROPOSES DECISION TABLES AS "AN ADVANCE
IN THE ART OF COMPUTER SYSTEMS DESIGN". THE AUTHOR
WANTS TO INCREASE THE UNDERSTANDING AND TO DEMONSTRATE
THE USEFULNESS OF THE DECISION TABLE TECHNIQUE. AN
IMPORTANT PART OF THE BOOKLET IS DEVOTED TO THE STUDY OF
STRUCTURES OF DECISION TABLES.
- TEM73A TEMPLE, G.
HOW TO ACHIEVE THE PRACTICAL BENEFITS
COMPUTER WEEKLY, 15(367)
NOVEMBER 1973, 6.
THIS ARTICLE DESCRIBES THE METHOD USED TO CONSTRUCT
DECISION TABLES AS PROGRAMMING AIDS FOR COMBINED
CONDITIONAL STATEMENTS.
- TEM73B TEMPLE, G.
WHAT GOALS CAN BE GAINED USING DECISION TABLES ?
COMPUTER WEEKLY, 15(370)
DECEMBER 1973, 6-17.
THE ARTICLE DESCRIBES THE OBJECTIVES WHICH CAN BE
ACHIEVED BY WRITING PROGRAMS IN DECISION TABLES.
- TH066 THORNTON, P.S.
SAPTAD
GUIDE
NOVEMBER 1966.
- TH069 THOMAS, A.
APPLICATION DES TABLES DE DECISION
(F) APPLICATION OF DECISION TABLES
AUTOMATISME, 14(1)
JANUARY 1969, 13-17.
THIS PAPER RESTATES THE DECISION TABLE CONCEPT,
ILLUSTRATING MORE PARTICULARLY THE RELATIONSHIP BETWEEN
DECISION TABLES AND BOOLEAN ALGEBRA.
- THU71 THURNER, R.
A CASE STUDY
DECISION TABLES CONFERENCE
AMSTERDAM, 7-8 DECEMBER 1971.
- THU72 THURNER, R.
ENTSCHEIDUNGSTABELLEN : AUFBAU - ANWENDUNG - PROGRAMMIERUNG
(G) DECISION TABLES : CONSTRUCTION - APPLICATION
PROGRAMMING
VERLAG DES VEREINS DEUTSCHER INGENIEURE
DUSSeldorf, 1972.
THIS BOOK GIVES A GOOD INTRODUCTION TO THE DECISION TABLE
TECHNIQUE. SUBJECTS TREATED : FUNDAMENTAL CONCEPTS,

APPLICATIONS AND ADVANTAGES OF THE TECHNIQUE, CONVERSION,
DETAB/GT-PREPROCESSOR, STRUCTURED PROGRAMMING BY MEANS
OF DECISION TABLES. A THOROUGH BIBLIOGRAPHY IS ADDED.

- THU73 THURNER, R.
UMWANDLUNG VON ENTSCHEIDUNGSTABELLEN
(G) THE CONVERSION OF DECISION TABLES
BURO, 21(11)
NOVEMBER 1973, 1136-1140.
THIS ARTICLE DISCUSSES THE CONVERSION OF DECISION TABLES
TO COMPUTER PROGRAMS BY MEANS OF RULE MASK TECHNIQUES.
IT IS POINTED OUT THAT BECAUSE THE TECHNIQUE OF
CONVERSION OF DECISION TABLES HAS A GREAT INFLUENCE ON
THE CHARACTERISTICS OF THE GENERATED PROGRAMS WITH
RESPECT TO TIME AND STORAGE REQUIREMENTS, IT IS ONE OF
THE MOST IMPORTANT CRITERIA FOR EVALUATION OF DECISION-
TABLE GENERATORS.
- THU74 THURNER, R. BAUKNECHT, K. GUNTHER, A.
LEVRAUT, B. LIPPS, H.
PROCEDURAL DECISION TABLES AND THEIR IMPLEMENTATION
INTERNATIONAL COMPUTING SYMPOSIUM
DAVOS, 4-7 SEPTEMBER 1973
PUBL.: NORTH HOLLAND
AMSTERDAM, 1974, 259-263.
IN THIS PAPER, A "PROCEDURAL" DECISION TABLE DEFINITION
IS PROPOSED BY FORMULATING ADDITIONAL CONVENTIONS ABOUT
THE MANNER OF CONDITION PROCESSING. THE RESTRICTIVE AS-
SUMPTIONS ON THE CONDITIONS ARE DROPPED IN FAVOUR OF THE
CONDITION BLOCK CONCEPT. A NEW ALGORITHM FOR THE CON-
VERSION OF PROCEDURAL TABLES, THE LINE MASK TECHNIQUE, IS
PRESENTED. THE DETAB/GT PROCESSOR (AN IMPLEMENTATION OF
THIS ALGORITHM) SHOWS THE CHARACTERISTICS OF THIS
TECHNIQUE. IN PARTICULAR THE PROCESSING OF OR-RULES AND
THE IMPLEMENTATION OF TEST AID FACILITIES IS SHOWN.
- THU75 THURNER, R.
STRUKTURIERTE PROGRAMMIERUNG ODER ENTSCHEIDUNGSTABELLEN ?
(G) STRUCTURED PROGRAMMING OR DECISION TABLES ?
OUTPUT, 4(4)
APRIL 1975, 28-30.
THE AUTHOR OF THIS ARTICLE ARGUES IN FAVOUR OF THE
COMBINATION OF STRUCTURED PROGRAMMING AND THE DECISION
TABLE TECHNIQUE. IN A STRUCTURED SYSTEMS ANALYSIS,
DECISION TABLES CAN BE USED AS TOOLS FOR CROSS-REFERENCE
AND AS HIERARCHICAL LINKING MECHANISMS. THEY HAVE THE
ADDITIONAL ADVANTAGE OF ALLOWING A THOROUGH LOGICAL
VERIFICATION.
- TUR69 TURNER, L.J.
CAN DECISION TABLES END COMPUTER PROBLEM CRISIS
OFFICE ADMINISTRATION
FEBRUARY 1969, 32-34.
THIS IS NOTED IN ANBAR ABSTRACTS 8, (8), APRIL 1969,
REF. SC.7, AS REPRODUCING THE 18 AXIOMS AND THEOREMS OF
BOOLEAN ALGEBRA, DISCUSSING THEIR APPLICATION IN LOGIC
MANIPULATION AND DESCRIBING THE USE AND LAYOUT OF
DECISION TABLES.
- UNK71 UNKEL, C.
ENTSCHEIDUNGSTABELLEN UNTER BESONDERER BERUECKSICHTIGUNG
VON SYSTEMANALYSE, PROGRAMMIERUNG UND DOKUMENTATION
(G) DECISION TABLES VIEWED AS TOOLS FOR SYSTEMS
ANALYSIS, PROGRAMMING AND DOCUMENTATION
DISSERTATION
UNIVERSITY OF COLOGNE
APRIL 1971, 73 PP.
AFTER AN INTRODUCTION TO THE DECISION TABLE TECHNIQUE,
THE AUTHOR GOES ON TO DESCRIBE SOME APPLICATIONS IN
SYSTEMS ANALYSIS, PROGRAMMING AND DOCUMENTATION.
A DETAILED PRACTICAL EXAMPLE (TOLL PAYMENT) IS ADDED.
- URM70 URMES, N.M.
ENTSCHEIDUNGSTABELLEN UND NICHTLINEARE OPTIMIERUNG
(G) DECISION TABLES AND NON-LINEAR OPTIMIZATION
IBM DEUTSCHLAND, FORM NO. 81581-2-70
SINDELINGEN, 1970.
THE AUTHOR INVESTIGATES THE POSSIBILITY TO CONSTRUCT A
NON-LINEAR OPTIMIZATION PROGRAM WITH THE HELP OF DECISION

TABLES. THE CONSTRUCTION OF A SECONDARY TRANSFORMER WINDING IS GIVEN AS AN EXAMPLE.

- USA65 U.S. AIR FORCE
AIR FORCE PAMPHLET 5-1-1
SEPTEMBER 1965.
ALSO IN : ADT, 94-114.
CHAPTER 3 OF THIS PAMPHLET DESCRIBES A TECHNIQUE FOR CONVERTING A NARRATIVE DIRECTIVE TO DECISION TABLES. IT DEALS WITH A RATHER COMPLICATED EXAMPLE (THE ENDORSEMENT OF MILITARY REPORTS).
- USC64A U.S. CENSUS BUREAU
SEMINAR ON DECISION TABLES
WASHINGTON, SEPTEMBER 1964.
A TWO YEARS' EXPERIENCE OF DECISION TABLES AT THE U.S. BUREAU IS PRESENTED BY TEN OF ITS MEMBERS : JACKSON, L., EXPERIENCE OF THE FOREIGN TRADE DIVISION KAHN, J., REFLECTIONS ON DECISION TABLES AND FOREIGN TRADE EXPORTS. BREEN, T., USE OF DECISION TABLES IN THE 1964 CENSUS OF AGRICULTURE. KING, B.H. COMMENTS ON THE USE OF DECISION TABLES IN THE 1964 CENSUS OF AGRICULTURE. O'BRIEN, J.L., COMMENTS ON THE USE OF DECISION TABLES IN THE 1964 CENSUS OF AGRICULTURE. HORNSETH, R.A., DECISION TABLES AT THE BUREAU OF THE CENSUS : HISTORY AND PROSPECTS. SOLOMON, M.J., REMARKS ON FUTURE POSSIBILITIES OF DECISION TABLES. FLETCHER, H.R., DECISION TABLE PREPROCESSOR. SOLOMON, M.J., SUMMARY OF DISCUSSION. O'BRIEN, J.L., SOME PROMISING APPROACHES TO COMPUTERIZING ADMINISTRATIVE OPERATIONS.
- THE PAPERS BY BREEN, KING, O'BRIEN (I) AND HORNSETH ARE AL2 INCLUDED IN ADT, 62-81.
- USC64B U.S. CENSUS BUREAU
CENTAB : A DECISION TABLE PREPROCESSOR
WASHINGTON, DECEMBER 1964.
CENTAB. ANOTHER EFFORT IN THE APPROACH TO THE PROBLEM OF DEVELOPING DECISION TABLE PREPROCESSORS, IS DESCRIBED.
- USH71 USHER, T.
DECISION TABLES IN THE DESIGN OF GENERALIZED MANAGEMENT INFORMATION SYSTEMS
NORTH WESTERN UNIVERSITY, DISSERTATION, UNIVERSITY MICROFILM 1971, 244 PP.
IN THIS DISSERTATION, A CONCEPT FOR THE CONSTRUCTION OF INFORMATION SYSTEMS IS PROPOSED. FOR THIS, THE DECISION TABLE TECHNIQUE AND THE TECHNIQUE USING BINARY DATA-STRUCTURES ARE COMBINED. BOTH TECHNIQUES ARE DISCUSSED IN DETAIL. THE APPROPRIATELY CONSTRUCTED PROGRAMS ARE EXPLAINED.
- USI68 USA STANDARDS INSTITUTE
FINAL REPORT OF THE AD HOC COMMITTEE ON DECISION TABLES
USA STANDARDS INSTITUTE
NEW YORK, JULY 1968.
THE AD HOC COMMITTEE WAS SET UP AS A PRELIMINARY MOVE IN THE CONSIDERATION OF DECISION TABLES "AS A CANDIDATE FOR STANDARDIZATION". THE REPORT INCLUDES : EVALUATION OF CRITERIA TO ESTABLISH NEED FOR STANDARDIZATION, PAPERS PRODUCED BY COMMITTEE MEMBERS, A BIBLIOGRAPHY AND A PROGRAM OF A PUBLIC SYMPOSIUM HELD BY THE COMMITTEE.
- VAN68 VANAVERBEKE, C.
THE USE OF DECISION TABLES
INTERNATIONAL CONFERENCE ON PRACTICAL EXPERIENCES IN SYSTEMS ANALYSIS, IFIP ADP GROUP 1968.
A DESCRIPTION OF THE TABLE BUILDING AND OTHER CONVENTIONS USED IN PHILIPS DATA SYSTEMS (BRUSSELS) IS GIVEN ; THE PAPER DOES NOT DESCRIBE ANY APPLICATIONS.
- VAN79 VANDER SANDE, L.
HET GEBRUIK VAN BESLISSINGSTABELLEN BIJ HET TOESTAAN VAN EEN KASKREDIET

- (D) THE USE OF DECISION TABLES AND THE APPROVAL OF CASH CREDITS
 K.U.LEUVEN, DEPT. OF APPLIED ECONOMICS, THESIS
 1979, 76 PP.
 THE USE OF DECISION TABLES WHEN ANALYSING THE CASH CREDIT APPROVAL PROCEDURE IS INVESTIGATED.
- VAN80 VANTHIENEN, J.
 A DYNAMIC PROGRAMMING ALGORITHM FOR DISPLAYING DECISION TABLES
 K.U.LEUVEN, DEPT. OF APPLIED ECONOMICS
 RESEARCH REPORT NO. 8006
 1980, 30 PP.
 THIS RESEAECH REPORT PRESENTS AN ALGORITHM FOR EDITING EXTENDED-ENTRY DECISION TABLES IN A USER-FRIENDLY WAY, I.E. BY TAKING EQUAL ADJACENT CONDITION ENTRIES TOGETHER.
- VEI66A VEINOTT, C.G.
 PROGRAMMING DECISION TABLES IN FORTRAN, COBOL OR ALGOL
 CACM, 9(1)
 JANUARY 1966, 31-35.
 A SIMPLE BROAD-BASED APPROACH FOR PROGRAMMING DECISION TABLES IN FORTRAN OR COBOL IS DEVELOPED AND PRESENTED. WITH INPUTS IN STANDARD FORM, AS DEFINED IN THE PAPER, THE PROGRAMMING OF ANY DECISION TABLE CAN BE DONE WITH ONE OR TWO FORTRAN STATEMENTS, OR WITH TWO COBOL STATEMENTS, IF THE COMPUTE VERB IS AVAILABLE, IN THE COBOL PROCESSOR. IT IS SHOWN THAT THE METHOD IS APPLICABLE EVEN WHEN THERE ARE MORE THAN TWO MUTUALLY EXCLUSIVE STATES OF ONE, TWO OR MORE TABLE CONDITIONS. IT IS FURTHER SHOWN THAT MULTI-STATE CONDITIONS IN DECISION TABLES CAN OFTEN SIMPLIFY PROGRAMMING. THE METHOD OUTLINED HAS THE FURTHER ADVANTAGE THAT ALL POSSIBLE COMBINATIONS OF CONDITIONS ARE CONSIDERED. IT IS SHOWN THAT THE SUGGESTED PROCEDURE IS EASILY IMPLEMENTED IN ALGOL.
- VEI66B VEINOTT, C.G.
 MORE ON PROGRAMMING DECISION TABLES
 CACM, 9(7)
 JULY 1966, 485.
 THIS LETTER TO THE EDITOR IS A POSTSCRIPT TO THE WRITER'S "PROGRAMMING DECISION TABLES IN FORTRAN, COBOL, OR ALGOL" (VEI66A). IT EXTENDS THE METHOD DESCRIBED THEREIN TO COVER CASES WHERE ONLY A FEW OF THE POSSIBLE COMBINATIONS OF CONDITIONS ARE MEANINGFUL AND THE REST ARE MEANINGLESS WHICH SHOULD LEAD TO A SIMPLE ACTION, POSSIBLY A DIAGNOSTIC.
- VEI68 VEISMANN, A. PERTILLER, H. MATHIES, H.
 TECHNISCHE ANGEBOTS- UND AUFTRAGSBEARBEITUNG IN DER VARIANTENFERTIGUNG (ADE) MIT ENTSCHEIDUNGSTABELLEN
 (G) AUTOMATED DESIGN ENGINEERING AND MAN-POWER PLANNING BY MEANS OF DECISION TABLES
 IBM NACHRICHTEN, 19(195)
 1969, 693-700.
 AUTOMATED DESIGN ENGINEERING AND MAN-POWER PLANNING AS THEY ARE APPLIED IN ENGINEERING WORKS ARE DESCRIBED. DECISION TABLES ARE USED TO GRASP AND TO PROGRAM THE LOGIC. THE COMBINATION OF THE DECISION TABLE TECHNIQUE AND BOMP ALLOWS THE APPLICATION OF STANDARDIZED CONSTRUCTION PACKAGES.
- VER68A VERHELST, M.
 DECISION TABLES AND THEIR USES
 WORKING PAPERS OF THE IFIP SEMINAR IN ADP
 SWETS AND ZEITLINGER
 AMSTERDAM, 1968, VOL. 2, PART 4, 29 PP.
- VER68B VERHELST, M.
 PROCEDURES FOR FINDING OPTIMAL AND NEAR OPTIMAL TEST SEQUENCES FOR APPLYING RULE MASK TECHNIQUES IN OBJECT PROGRAMS DERIVED FROM DECISION TABLES
 IAG QUARTERLY JOURNAL, 1(1)
 1968, 47-65.
 IN THIS PAPER, THE AUTHOR EXPLAINS THE RULE MASK AND INTERRUPTED RULE MASK TECHNIQUES, CALCULATES A "TOTAL TEST TIME" FOR A GIVEN TABLE UNDER BOTH METHODS AND CONCLUDES THAT NEITHER CAN BE SHOWN TO RESULT IN AN ABSOLUTE MINIMAL EXECUTION TIME. HE GOES ON TO DESCRIBE

A SEARCH METHOD THAT FINDS THE BEST SEQUENCE OF TESTS TO MINIMISE "TOTAL TEST TIME" IN ALL CASES.
THE AUTHOR ALSO PROPOSES A SECOND METHOD WHICH TAKES LESS TIME BUT FOUND SEQUENCES IN SOME CASES ONLY THAT LED TO A LOWER "TOTAL TEST TIME" THAN THOSE FOUND BY THE STRATEGIES PROPOSED BY KING (KING66). HE CONCLUDES WITH A TABLE OF COMPARATIVE TIMES AND AN ORDER OF PREFERENCE BASED ON EXECUTION TIME ONLY.

- VER69A VERHELST, M.
A TECHNIQUE FOR CONSTRUCTING DECISION TABLES
IAG QUARTERLY JOURNAL, 2(1)
MARCH 1969, 27-36.
A NEW METHOD FOR THE CONSTRUCTION OF DECISION TABLES IS PROPOSED. THE TECHNIQUE USES SIMPLE BOOLEAN ALGEBRA AND IS DESIGNED TO OBTAIN IN AN AUTOMATIC WAY A FINAL DECISION TABLE STARTING FROM A TABLE CONTAINING SIMPLE RULES. IT IS SHOWN THAT THE TECHNIQUE IS VERY USEFUL FOR FORMALIZING COMPLEX DECISION SITUATIONS CORRECTLY IN A QUICK WAY AND WITH A MINIMUM OF HUMAN EFFORT. THROUGHOUT THE ARTICLE, THE VARIOUS STEPS TO BE TAKEN ARE ILLUSTRATED BY MEANS OF AN EXAMPLE.
- VER69B VERHELST, M.
HET GEBRUIK VAN BESLISSINGSTABELLEN IN DE INFORMATIEVERWERKING
(D) THE USE OF DECISION TABLES IN INFORMATION PROCESSING
HET INGENIEURSBLAD, 39(5)
1969, 123-128.
ALSO IN : INFORMATIE, 12(12)
DECEMBER 1970, 516-521.
THIS TEXT IS INTENDED TO DEMONSTRATE THE USE OF DECISION TABLES IN DATA PROCESSING. AFTER A SHORT INTRODUCTION, THE AUTHOR GOES ON TO EXPLAIN IN DETAIL THE CONSTRUCTION OF DECISION TABLES. THE ADVANTAGES GAINED BY USING DECISION TABLES ARE OUTLINED.
- VER70 VERHELST, M.
BESLISSINGSTABELLEN
(D) DECISION TABLES
ECONOMISCH EN SOCIAAL TIJDSCHRIFT, 24(6)
DECEMBER 1970, 713-719.
THE PURPOSE OF THIS ARTICLE IS TO PRESENT THE FUNDAMENTAL DEFINITIONS OF THE DECISION TABLE TECHNIQUE. SOME APPLICATION AREAS TOGETHER WITH A CONCISE BIBLIOGRAPHY ARE GIVEN.
- VER71A VERHELST, M.
DECISION TABLES
DECISION TABLES : NEW DEVELOPMENTS AND IMPLEMENTATIONS
AMSTERDAM, 7-8 DECEMBER 1971, 62 PP.
THIS TEXT IS A TRANSLATION OF AN EARLIER VERSION OF VER73.
- VER71B VERHELST, M.
THE USE OF DECISION TABLES
PROCEEDINGS OF THE IIND YUGOSLAV INTERNATIONAL ADP SEMINAR 70
ZAGREB, 1971, B1-B5.
- VER71C VERHELST, M.
THE CONVERSION OF DECISION TABLES TO COMPUTER PROGRAMS
PROCEEDINGS OF THE IIND YUGOSLAV INTERNATIONAL ADP SEMINAR 70
ZAGREB, 1971, B6-B11.
- VER72A VERHELST, M.
THE CONVERSION OF LIMITED-ENTRY DECISION TABLES TO OPTIMAL AND NEAR-OPTIMAL FLOWCHARTS : TWO NEW ALGORITHMS
CACM, 15(11)
NOVEMBER 1972, 974-980.
TWO NEW ALGORITHMS FOR DERIVING OPTIMAL AND NEAR-OPTIMAL FLOWCHARTS FROM LIMITED-ENTRY DECISION TABLES ARE PRESENTED. BOTH TAKE INTO ACCOUNT RULE FREQUENCIES AND THE TIME NEEDED TO TEST CONDITIONS. ONE OF THE ALGORITHMS, CALLED THE OPTIMUM-FINDING ALGORITHM, LEADS TO A FLOW-CHART WHICH TRULY MINIMIZES EXECUTION TIME FOR A DECISION TABLE IN WHICH SIMPLE RULES ARE ALREADY CONTRACTED TO COMPLEX RULES. THE OTHER ONE, CALLED THE OPTIMUM-AP-

PROACHING ALGORITHM, REQUIRES MANY FEWER CALCULATIONS BUT DOES NOT NECESSARILY PRODUCE THE OPTIMUM FLOWCHART. THE ALGORITHMS ARE FIRST DERIVED FOR TREATING DECISION TABLES NOT CONTAINING AN ELSE-RULE, BUT THE OPTIMUM-APPROACHING ALGORITHM IS SHOWN TO BE EQUALLY VALID FOR TABLES INCLUDING SUCH A RULE.

BOTH ALGORITHMS ARE COMPARED WITH EXISTING ONES AND ARE APPLIED TO A SOMEWHAT LARGE DECISION TABLE DERIVED FROM A REAL CASE.

FOR COMMENTS, SEE : KIN74.

- VER72B VERHELST, M. DE VRIES, P. DE WINTER, L.
BESLISSINGSTABELLEN
(D) DECISION TABLES
KONFERENTIE BESLISSINGSTABELLEN EN HUN GEBRUIK
AMSTERDAM, 6-7 JUNE 1972, 230 PP.
THESE NOTES INCLUDE CHAPTERS DEVOTED TO DEFINITIONS, METHODS FOR CONSTRUCTING DECISION TABLES, FOR CONTROLLING THE ACCURACY OF A DECISION TABLE AND FOR CONVERTING DECISION TABLES INTO COMPUTER PROGRAMS. FINALLY SOME ADVANTAGES OF THE USE OF DECISION TABLES ARE PRESENTED.
- VER72C VERBRAEKEN, W.
DE OMZETTING VAN BESLISSINGSTABELLEN IN COMPUTERPROGRAMMA'S
(D) THE CONVERSION OF DECISION TABLES INTO COMPUTER PROGRAMS
KUL, INSTITUUT VOOR TOEGEPASTE EKONOMISCHE WETENSCHAPPEN
LEUVEN, 1972, 118 PP.
IN THE INTRODUCTORY CHAPTERS, THE AUTHOR OF THIS THESIS DISCUSSES THE BASIC CONCEPTS OF THE DECISION TABLE TECHNIQUE, THE CONVERSION METHODS AND THE POSSIBLE CONTROL FUNCTIONS WHEN CONVERTING DECISION TABLES. AFTER THIS, HE GOES ON TO PRESENT A FORTRAN IV-PROGRAM BASED ON THE OPTIMUM SEARCHING ALGORITHM OF VERHELST (VER72A). AT THE END, HE INVESTIGATES THE CONVERSION METHODS AS THEY ARE APPLIED IN A FEW BELGIAN FIRMS.
- VER73 VERHELST, M.
BESLISSINGSTABELLEN
(D) DECISION TABLES
STUDIECENTRUM VOOR INFORMATICA
SAMSON
ALPHEN AAN DE RIJN
1973, 75 PP.
THIS BOOK CONTAINS A FAIRLY COMPLETE TREATMENT OF THE DECISION TABLE TECHNIQUE. IT TRACES THE FUNDAMENTAL DEFINITIONS AND GOES ON TO DESCRIBE THE EXISTING TECHNIQUES FOR THE CONSTRUCTION OF DECISION TABLES. A SECTION IS DEVOTED TO PARSING AND COUPLING, WHILST THE CONVERSION INTO COMPUTER PROGRAMS IS THOROUGHLY DEALT WITH. SOME APPLICATION AREAS ARE MENTIONED.
- VER75 VERHELST, M.
DE BESLISSINGSTABEL : EEN NUTTIGE TECHNIEK VOOR MANAGEMENT
(D) THE DECISION TABLE : A USEFUL TECHNIQUE FOR MANAGEMENT
TIJDSCHRIFT VOOR ECONOMIE EN MANAGEMENT, 20(3)
1975, 393-412.
THE PURPOSE OF THIS ARTICLE IS TO SHOW HOW THE USE OF DECISION TABLES CAN INCREASE EFFICIENCY IN ORGANIZATIONS. THE FOLLOWING ASPECTS OF THE USE OF DECISION TABLES ARE COVERED : A) FOR ANALYZING AND IMPROVING PREVIOUSLY STRUCTURED DECISION SITUATIONS ; B) FOR STUDYING NOT PREVIOUSLY STRUCTURED DECISION SITUATIONS ; C) FOR MAKING DECISIONS.
- VER77A VERHETSEL, M.
EVALUATIE EN VERVETERING VAN DRIE ALGORITMEN TOT HET OMZETTEN VAN BESLISSINGSTABELLEN IN COMPUTERPROGRAMMA'S
(D) EVALUATION AND IMPROVEMENT OF THREE ALGORITHMS FOR THE CONVERSION OF DECISION TABLES IN COMPUTER PROGRAMS
K.U.LEUVEN, DEPT. OF APPLIED ECONOMICS, THESIS
LEUVEN, 1977, 142 PP.
THE MAIN RESULTS OF THIS THESIS ARE SUMMARIZED IN VER77B. THE SAMPLE OF DECISION TABLES USED TO COMPARE THE ALGORITHMS AS WELL AS THE COMPUTER PROGRAMS ARE ADDED IN APPENDIX.
- VER77B VERHETSEL, M.
EVALUATION AND IMPROVEMENT OF THREE ALGORITHMS FOR THE CONVERSION OF LIMITED-ENTRY DECISION TABLES

RESEARCH REPORT, K.U.LEUVEN, DEPT. OF APPLIED ECONOMICS
LEUVEN, 1977, 19 PP.

IN THIS PAPER, SOME CHANGES TO THE ALGORITHMS OF POLLACK,
SHWAYDER AND VERHELST ARE PROPOSED IN ORDER TO IMPROVE
THEIR PERFORMANCE. AN ALTERNATIVE STOPPING RULE IS
INVESTIGATED AND THE CONDITION TEST TIME IS BROUGHT INTO
THE ALGORITHM OF POLLACK. THERE REMAINS LITTLE SIGNIFI-
CANT DIFFERENCE BETWEEN THE PERFORMANCE OF THE THREE
IMPROVED ALGORITHMS.

- VER80 VERHELST, M.
DE PRAKTIJK VAN BESLISSINGSTABELLEN
(D) DECISION TABLES IN PRACTICE
KLUWER, DEVENTER/ANTWERPEN, 1980, 175 PP.
THIS BOOK PRESENTS MANY NEW INSIGHTS IN THE USE AND THE
APPLICATION FIELDS OF DECISION TABLES. IT SUCCESSIVELY
DEALS WITH THE SYSTEMATIC CONSTRUCTION OF DECISION TABLES
ADDITIONAL TECHNIQUES AND CONVENTIONS, DECISION TABLES
AND STRUCTURED PROGRAMMING, DECISION TABLES AND THE LIFE-
CYCLE OF INFORMATION SYSTEMS AND THE OPTIMIZATION OF
PROCEDURES DOCUMENTED BY MEANS OF DECISION TABLES.
- VIE73 VIEWEG, W.
DIE KONSTRUKTION VON ENTSCHEIDUNGSTABELLEN
(G) CONSTRUCTION OF DECISION TABLES
BETRIEBSWIRTSCHAFTLICHER VERLAG DR. TH. GABLER
WIESBADEN, 1973, 232 PP.
AFTER AN INTRODUCTION TO THE APPLICATION AREA OF THE
DECISION TABLE TECHNIQUE, THE AUTHOR GOES INTO THE "IN-
DIRECT" METHOD FOR THE CONSTRUCTION OF DECISION TABLES AS
FIRST PROPOSED BY VERHELST IN VER69. FOR THIS, HE PRO-
POSES AN INTERACTIVE DECISION TABLE GENERATOR ; THE LOGIC
DEVELOPED IS ENTIRELY DUE TO VERHELST.
- VIV71 VIVIAN, R.L.
PROGRAM LOGIC TABLE (PLOT) PROGRAMMING METHOD
IBM TDB, 14
SEPTEMBER 1971, 1109-1110.
- WAR68 WARWICK, M.
AN EFFECTIVE COMPUTER SYSTEM
COMPUTER WEEKLY,
OCTOBER 1968, 6.
THE ARTICLE DESCRIBES THE PRINCIPLES OF A COMPUTER AIDED
"NETTING" SYSTEM, DESIGNED BY HONEYWELL FOR USE WITHIN
A CUSTOMER'S PRODUCTION CONTROL SYSTEM : THE NETTING
RULES WERE DEVELOPED IN DECISION TABLE FORM.
- WEBOO WEBB, T.A.
USES FOR DECISION TABLES IN COMPUTER PROGRAMMING
IBM TIE Z 77-7238
DATE UNKNOWN.
- WEB70 WEBSTER, C.A.G.
A BASIS FOR DECISION
DATA SYSTEMS
MAY 1970, 34-35.
DECISION TABLES ARE BASED ON THE PRINCIPLE THAT ALL
APPROPRIATE CONDITIONS MUST BE SATISFIED BEFORE THE
CONCLUSION IS DRAWN AND THE SPECIFIED ACTION TAKEN. AS
SUCH, THEY ARE USEFUL TOOLS FOR SYSTEMS ANALYSTS,
PARTICULARLY WHEN THE RULES FOR MAKING A CHOICE ARE
MORE COMPLEX THAN JUST A SPECIFIC DISCRIMINATING TEST
AND, WHEN AS IS USUALLY THE CASE, THE VARIOUS CHOICE
CRITERIA ARE PRESENTED SIMULTANEOUSLY RATHER THAN
SERIALLY.
- WEI71 WEISBARD, M.F.
DETAP/55 : A DECISION TABLE PREPROCESSOR FOR GENERATING
SINGLE-PARAGRAPH, FULLY-NESTED COBOL CODE
SIGPLAN NOTICES, 6(8)
SEPTEMBER 1971, 45-53.
THIS PAPER DESCRIBES THE DESIGN AND LOGIC OF A DECISION
TABLE PREPROCESSOR DETAP/65. THIS PROGRAM GENERATES
FULLY-NESTED COBOL STATEMENTS, FORMATTED AS A SINGLE
PARAGRAPH FOR EACH TABLE. THIS FORM OF OUTPUT
DISTINGUISHES IT FROM OTHER PREPROCESSORS, SUCH AS
DETAB/65, WHICH GENERATE MULTIPLE PARAGRAPHS CONSISTING
OF TEST-AND-BRANCH COBOL CODE.

- WEL68 WELCH, J.E.
DECISION TABLES - A USER'S GUIDE
J.E.WELCH
LONDON - NEW-YORK, 1968.
THIS GUIDE CONTAINS NOTES ON PET AND THE DETAB/65
PREPROCESSOR. MUCH OF THE MATERIAL CONSISTS OF
EXTRACTS FROM VARIOUS MANUALS.
- WEN71 WENDELIN, R. ZANKL, A.
DATENVERKNUEPFUNG IM PROZESSRECHNER MIT HILFE VON
ENTSCHEIDUNGSTAFELN
(G) LOGICAL CONNECTION OF PROCESS-DATA BY MEANS OF DECISION
TABLES
ANGEWANDTE INFORMATIK, 13(11)
1971, 493-499.
THE REPORT SHOWS, THAT THE LOGICAL CONNECTION OF PROCESS-
DATA, A CHARACTERISTIC TASK FOR REALTIME-COMPUTERS,
CAN BE FORMULATED BY ENGINEERS IN THE FORM OF DECISION-
TABLES. BY MEANS OF SIMPLE EXAMPLES, IT IS DEMONSTRATED
HOW AN UNCONVENTIONAL INTERPRETATION OF DECISION TABLES
OPENS PRACTICAL POSSIBILITIES IN THEIR USE. BESIDES.
REFERENCE IS MADE TO A PROGRAMM-STRUCTURE, WHICH FULFILS
THE DEMAND FOR LITTLE STORE-PLACE AND REALTIME.
- WEN73 WENGER, F.
ENTSCHEIDUNGSTABELLEN : ORGANISATIONSMITTEL UND STEUERLOGIK
(G) DECISION TABLES AS A TOOL FOR ORGANIZATION AND
MANAGEMENT
OUTPUT, 2(1)
FEBRUARY 1973, 40-47.
DECISION TABLES ARE DESCRIBED AS A TOOL FOR ORGANIZATION
AND MANAGEMENT. A METHOD FOR THE DIRECT CONSTRUCTION OF
DECISION TABLES IS PROPOSED.
- WER74 WERNER, R.
PROGRAMMTECHNISCHES VORGEHEN BEI DER ZEILENPRUEFUNG MIT
HILFE EINER ENTSCHEIDUNGSTABELLE
(G) PROGRAMMED HANDLING OF LINE CHECKING WITH A DECISION
TABLE
RTDV, 11(4)
APRIL 1974, 20-22.
THE PROPOSED METHOD IS APPLIED TO TELEPRINTERS OF THE
RAILWAY CONTROL SYSTEM AND EXHIBITS FLEXIBILITY THROUGH
THE SELECTION OF FUNCTION CODE FOR THE CHECK ROUTINE.
DEVIATIONS DERIVED FROM THE DECISION TABLE ARE USED IN
ERROR SIGNALLING. AN EXAMPLE OF THE APPLICATION IS GIVEN.
- WILOO WILLOUGHBY, T.C. JOHNSTON, A.L.
PROGRAMMING FROM PROSE, FLOW CHARTS OR DECISION TABLES
UNPUBLISHED PAPER
NO DATE.
THE RESEARCH REPORTED IN THIS PAPER COMPARES THE ABILITY
OF PROGRAMMERS TO WRITE PROGRAMS STATED IN PROSE, FLOW
CHART AND DECISION TABLE FORM. THREE PROGRAMS WERE
DEVELOPED TO BE PROGRAMMED IN FORTRAN ; EACH OF THE
THREE PROGRAMS WAS EXPRESSED IN THREE DIFFERENT VERSIONS
: PROSE, FLOW CHART AND DECISION TABLE. AN ANALYSIS
OF VARIANCE OF THE RESULTS IS PERFORMED WITH RESPECT TO
PROGRAMMING TIME AND ERRORS.
- WIL65 WILLIAMS, W.K.
DECISION STRUCTURE TABLES
NAA BULLETIN, 46(9)
MAY 1965, 58-62.
SOME BASIC CONCEPTS ABOUT THE STRUCTURE OF DECISION
TABLES ARE PRESENTED. THE AUTHOR GIVES SEVERAL VARIATIONS
IN THE STRUCTURE OF THE TABLES AND HE INTRODUCES THREE
FIELDS OF APPLICATIONS : PROGRAM LOGIC, SYSTEM DEFINITION
AND A COMMUNICATION TOOL. THE ARTICLE ENDS WITH A LIST OF
SOME ADVANTAGES OF THE DECISION TABLE APPROACH.
- WIL69 WILLUMSEN, B.
BESLUTSTABELLER - EN TEKNIK VARD ATT UPPMARKSAMMA
(S) DECISION TABLES - A REMARQUEABLE TECHNIQUE
REPORT
CHALMERS TEKNISKA HOGSKOLA
1969, 118 PP.
THIS THESIS DEALS WITH : DEFINITIONS AND BASIC CONCEPTS

OF THE DECISION TABLE TECHNIQUE, THE CONSTRUCTION
OF A DECISION TABLE, LOGICAL CONTROL METHODS,
DECISION TABLE STRUCTURES, THE CONVERSION PROBLEM AND
DECISION TABLES APPLICATIONS. A THOROUGH BIBLIOGRAPHY
IS ADDED.

- WIL71 WILLUMSEN, B.
BESLUTSTABELLER I ADMINISTRATIV RATIONALISERING
(S) DECISION TABLES IN RATIONALIZING THE ADMINISTRATION
SVERIGES MEKANFORBUND, REPORT
STOCKHOLM, MARCH 1971.
- WIL72 WILLOUGHBY, T.C. ARNOLD, A.D.
COMMUNICATING WITH DECISION TABLES, FLOW CHARTS AND PROSE
DATA BASE, 4(3)
FALL 1972, 13-16.
PROSE, FLOWCHARTS AND DECISION TABLES ARE THREE
TECHNIQUES WHICH CAN BE USED TO DESCRIBE PROCESS FLOWS.
SINCE THEY CAN BE USED TO COMMUNICATE ACTIONS AND
ALTERNATIVES TO PEOPLE, ONE SHOULD BE ABLE TO MEASURE
THEIR EFFECTIVENESS AS COMMUNICATION AIDS. THE STUDY
REPORTED ON COMPARES THE ABILITY OF UNSOPHISTICATED USERS
TO SOLVE SIMPLE PROBLEMS STATED IN EACH OF THE THREE
FORMS.
- WOOD66 WOODSIDE, B.
TRUVECDT : A TRUTH VECTOR DECISION TABLE COMPILER
RELIANCE ELECTRIC AND ENGINEERING
UNPUBLISHED PAPER
SEPTEMBER 1966.
THE PURPOSE OF THIS PAPER IS TO DESCRIBE TRUVECDT, A
PROGRAM WHICH CONVERTS CODING IN DECISION TABLE FORMAT
DIRECTLY INTO COBOL CODING IN A WAY WHICH GREATLY
MINIMIZES COMPUTER CORE REQUIREMENTS AS COMPARED TO
TREE METHODS. HOWEVER, THIS FEATURE IS GAINED ONLY BY
SACRIFICING SOME SPEED WITH WHICH THE TABLE COULD
THEORETICALLY BE EXECUTED. THEREFORE, THE TRADE-OFF
BETWEEN CORE AND SPEED BETWEEN THIS METHOD AND OTHER
APPROACHES IS EXAMINED.
- WOOD67 WOOD, D.R.
DECISION TABLES
IDEAS FOR MANAGEMENT (PAPERS PRESENTED AT THE 1967
INTERNATIONAL SYSTEMS MEETING)
1967, 113-120.
THIS PAPER PROVIDES A GENERAL INTRODUCTION TO THE
SUBJECT. IT OUTLINES THE BENEFITS TO BE DERIVED AND
GOES ON TO DESCRIBE AND TO ILLUSTRATE THE TYPES OF TABLES
AND METHODS OF CONSTRUCTION AND CHECKING. THE FULL
SPECIFICATION METHOD IS OUTLINED AS WELL AS ANOTHER ONE
WHICH RESEMBLES THE PROGRESSIVE RULE DEVELOPMENT APPROACH
FOR THE EXPERIENCED USER. THE AUTHOR GOES ON TO DISCUSS
SEGMENTATION, LINKAGE MECHANISMS AND THE NEED TO
INDICATE TABLE RELATIONSHIPS CLEARLY. HE ENDS WITH A
BRIEF REVIEW OF SOME COMMON PROBLEMS.
- WOOD68 WOODGATE, H.S.
PLANNING NETWORKS AND RESOURCE ALLOCATION
DATAMATION, 14(1)
JANUARY 1968, 36-43.
ALSO IN : ADT, 82-93.
SOME OF THE SALIENT FEATURES OF THE CT 1900 SERIES PERT
ARE DESCRIBED. IN THIS, DECISION TABLES ARE USED TO
SPECIFY THE MANNER IN WHICH THE ACTIVITIES HAVE TO BE
SCHEDULED. A SEGMENT OF A TYPICAL DECISION TABLE IS
EXPLAINED.
- WOOD70 WOODALL, A.D.
A RULE MASK TECHNIQUE FOR DECISION TABLE TRANSLATION
THE COMPUTER BULLETIN, 14(10)
OCTOBER 1970, 347.
A SCHEME FOR IMPLEMENTATION OF COMPLEX LIMITED ENTRY
DECISION TABLES IS SUGGESTED AS AN ALTERNATIVE TO THAT
PROPOSED BY KIRK IN KIR65.
- WOOD71 WOODS, C.G. HAWES, M.K.
OPTIMIZED CODE GENERATION FROM EXTENDED ENTRY DECISION
TABLES
SIGPLAN NOTICES, 6(8)

SEPTEMBER 1971, 74-80.

OPTIMIZED ALGORITHMS HAVE BEEN DEVELOPED FOR THE PROCESSING OF EXTENDED ENTRY DECISION TABLES, AND FOR THE DIRECT CONVERSION OF SUCH TABLES TO CODE WITHOUT TRANSLATION TO LIMITED ENTRY TABLES. ONE OF THE ALGORITHMS IS DESCRIBED AND DISCUSSED IN THIS PAPER. EXAMPLES ARE GIVEN TO ILLUSTRATE THAT DIRECT CODE CONVERSION IS BETTER AND WHY IT IS BETTER. ALTHOUGH THE AMOUNT OF IMPROVEMENT WILL VARY FROM TABLE TO TABLE, AN INDICATION OF THE SAVINGS IS INCLUDED.

- WRI62 WRIGHT, K.R.
APPROACHES TO DECISION TABLE PROCESSORS
PROC. DECISION TABLES SYMPOSIUM
SEPTEMBER 1962, 41-44.
THE AUTHOR DISCUSSES THE FOUR BASIC TYPES OF PROCESSORS OR METHODS OF CONVERTING DECISION TABLES TO A MACHINE LANGUAGE. THESE ARE (1) THE MANUAL PROCESSOR, (2) THE INTERPRETIVE PROCESSOR, (3) THE TRANSLATOR, AND (4) THE COMPILER.
- WYL68A WYLD, M.T. MC PARTLAND, K.M.
A VALIDATION PACKAGE
UNPUBLISHED
NOVEMBER 1968.
THIS IS A DESCRIPTION, WRITTEN FOR INTERNAL USE, OF THE FACILITIES AVAILABLE IN THE ENGLISH CALICO VALIDATION PACKAGE BASED ON DECISION TABLES.
- WYL68B WYLD, M.T.
AN UPDATE PACKAGE
UNPUBLISHED
NOVEMBER 1968, SUBSEQUENTLY REVISED.
ANNOTATION : SEE WYL68A.
- YAS71 YASUI, T.
SOME ASPECTS OF DECISION TABLE TECHNIQUES
SIGPLAN NOTICES, 6(8)
SEPTEMBER 1971, 104.
BY CONSIDERING A SIMPLIFIED MATHEMATICAL MODEL OF DECISION TABLES, SOME BASIC THEORETICAL RESULTS CONCERNING DECISION TABLE OPTIMIZATION PROBLEMS ARE DERIVED. AN ALGORITHM CALLED ITERATED LOCAL MINIMIZATION IS PROPOSED AND QUANTITATIVELY COMPARED WITH THE OPTIMAL CONVERSION ALGORITHM OF REINWALD AND SOLAND AND WITH POLLACK'S FIRST ALGORITHM. ONLY DECISION TABLES WITH NO ELSE RULE AND NO REDUNDANCY ARE DEALT WITH.
- YAS72 YASUI, T.
CONVERSION OF DECISION TABLES INTO DECISION TREES
UNIVERSITY OF ILLINOIS, REF. AD-740345
URBANA, FEBRUARY 1972, 139 PP.
KNOWN MANUAL METHODS OF CONVERTING DECISION TABLES INTO DECISION TREES ARE BASED MAINLY ON PLAUSIBLE ARGUMENTS WITH LITTLE THEORETICAL BACKGROUND. THE INTENTION OF THE AUTHOR IS TO ESTABLISH A NEW THEORY IN THIS FIELD. BY CONSIDERING A SPECIAL KIND OF PARTITIONS OF VERTICES OF AN N-CUBE AS A MODEL OF DECISION TABLES, THE AUTHOR PUTS THE CONVERSION PROBLEM INTO A SIMPLIFIED AND ABSTRACT FORM. HE DERIVES SOME THEORETICAL RESULTS CONCERNING THE OPTIMIZATION PROBLEM, AND THEN AN ALGORITHM CALLED ITERATED LOCAL MINIMIZATION IS PROPOSED AND COMPARED QUANTITATIVELY WITH OTHER ALGORITHMS. ALSO, A NEW TOPIC, THE DECOMPOSITION THEORY OF DECISION TABLES AND DECISION TREES IS PRESENTED. THE AUTHOR CONSIDERS DECOMPOSING DECISION TABLES AND DECISION TREES INTO SMALLER ONES SO THAT THEY CAN BE PROCESSED EFFECTIVELY IN PARALLEL.
- ZMI73 ZMITROVIC, A.I.
(R) TABLE METHODS AND ALGORITHMS
EKONOMIKA I MATEMATICHESKIE METODY, (5)
1973.
- ZOB70 ZOBEL, H. KROEMER, H. BODE, H.
LANGER, H.U.
ENTSCHEIDUNGSTABELLEN ZUR AUTOMATISCHEN ERSTELLUNG VON FERTIGUNGSAUFTRAEGEN IN VERBINDUNG MIT DEM TERMINIERUNGSPROGRAMM CLASS

(G) DECISION TABLES EMPLOYED FOR THE AUTOMATIC PREPARATION OF PRODUCTION ORDERS IN CONNECTION WITH CLASS IBM NACHRICHTEN, 20(201) JUNE 1970, 239-246.

THIS ARTICLE DESCRIBES AND DISCUSSES THE PRACTICAL IMPLEMENTATION OF DECISION TABLES IN THE PROCESS OF AUTOMATIC PREPARATION OF PRODUCTION ORDERS. AT EACH MOMENT, THESE ORDERS TAKE INTO ACCOUNT THE MOST RECENT CONSTRUCTION METHODS ; AT THE SAME TIME, THEY LOOK AFTER OPTIMAL SPREADING OF WORKING-HOURS AND OPTIMAL PLANT SELECTION.

- ZZZ00A ZZZ
AGENTA GENERATOR AUF DER GRUNDLAGE VON ENTSCHEIDUNGSTABELLEN (G) THE AGENTA-GENERATOR BASED ON DECISION TABLES SOFTWARE AG DARMSTADT, DATE UNKNOWN.
- ZZZ00B ZZZ
DECISION TABLES CAN IMPROVE DOCUMENTATION COMPUTERWORLD, A RESEARCH REPORT DATE UNKNOWN.
DECISION TABLES ARE PRESENTED AS MORE USEFUL TOOLS THAN OTHER TECHNIQUES CURRENTLY AVAILABLE FOR COMMUNICATION AND DOCUMENTATION PURPOSES.
- ZZZ66B ZZZ
DECISION TABLES AS A PROGRAMMING TOOL
IN : ADT, 150-154
REPRINT OF : EDP ANALYZER, 4(5)
MAY 1966, 8-10 (REF. CANG6).
THERE ARE FOUR WAYS TO CONVERT DECISION TABLES INTO COMPUTER PROGRAMS : MANUAL CODING, THE USE OF AN INTERPRETER, INSERTING A PREPROCESSOR OR THE DEVELOPMENT OF A DECISION TABLE COMPILER. THE AUTHOR EXPLAINS WHY ONLY MANUAL CODING AND PREPROCESSORS WERE WIDELY USED IN THE PAST. HE GOES ON TO DISCUSS BOTH METHODS.
- ZZZ66A ZZZ
OVER 400 REQUESTS RECEIVED FOR ACM'S DETAB/65 PREPROCESSOR CACM, 9(2)
FEBRUARY 1966, 125.
- ZZZ68 ZZZ
DEFINITIONS AND DECISIONS IN SYSTEMS AND PROGRAMMING MECHANISED ACCOUNTING & COMPUTER MANAGEMENT, 3(11)
NOVEMBER 1968, 10 AND 13-14.
THE ARTICLE REVIEWS THE TECHNIQUES OF AUTOMATIC FLOW-CHARTING, ADS AND DECISION TABLES. IT DISCUSSES THE DEFICIENCIES OF DETAB.
- ZZZ71A ZZZ
COBOL SUPPORT PACKAGES... PROGRAMMING AND PRODUCTIVITY AIDS, DECISION TABLE TRANSLATORS... GENERATING COBOL LOGIC STATEMENTS
DATA PROCESSING DIGEST, 17(11)
NOVEMBER 1971, 19 PP.
THIS ARTICLE FIRST INTRODUCES THE BASIC CONCEPTS OF DECISION TABLES BY MEANS OF A SERIES OF ELEMENTARY EXAMPLES, AND THEN SHOWS, BY A MORE COMPLEX EXAMPLE, HOW A DECISION TABLE PROCESSOR WORKS. FOUR REPRESENTATIVE DECISION TABLE PROCESSORS ARE DISCUSSED, AND THEIR IMPORTANT CHARACTERISTICS TABULATED. A LIST OF AVAILABLE PROCESSORS AND THEIR SUPPLIERS IS ALSO PROVIDED.
- ZZZ71B ZZZ
TABLEMASTER
DATA PROCESSING, 13(5)
SEPTEMBER 1971, 344-349.
IN THIS ARTICLE THE COBOL VERSION OF THE HOSKYN'S SYSTEMS' TABLEMASTER PREPROCESSOR IS EXAMINED. ALTHOUGH THE BASIC CONCEPTS AND OPERATION OF TABLEMASTER ARE COMMON TO ALL VERSIONS (SYSTEM/360 BASIC ASSEMBLER, COBOL AND PL/I). TABLEMASTER (COBOL VERSION) ALLOWS A PROGRAMMER TO SPECIFY A COBOL PROCEDURE DIVISION IN TABULAR FORM USING TABLEMASTER SOURCE LANGUAGE. THE LANGUAGE USED IS COBOL BUT INCLUDES ADDITIONAL ELEMENTS TO ALLOW FOR THE CONSTRUCTION OF PROGRAM TABLES.

- ZZZ73A ZZZ
PROBLEME LOGISCH AUFBEREITEN
(G) THE LOGICAL PREPARATION OF PROBLEMS
MANAGER MAGAZIN, (5)
1973, 82-84.
IN THIS ARTICLE, THE PRACTICAL ADVANTAGES OF USING
DECISION TABLES IN EDP-APPLICATIONS ARE MENTIONED.
- ZZZ73B ZZZ
(G) PROGRAM ORGANISATION. IV. SYSTEMATIZATION OF PROGRAMMING
ADL-NACHRICHTEN, 18(81)
JULY-AUGUST 1973, 46-48 AND 50-51.
THIS ARTICLE DISCUSSES THE PROBLEM OF SYSTEMATISING THE
PROCESS OF PROGRAMMING BASED ON THE USE OF DECISION
TABLES. THE BASIC CONSTRUCTION OF DECISION TABLES IS
FULLY EXPLAINED AND THE VARIOUS TYPES OF TABLES ARE
DISCUSSED LEADING TO AN EXPLANATION OF THE METHOD OF
PROGRAMMING FROM THE TABLES.
- ZZZ74 ZZZ
BIBLIOGRAPHIE ENTSCHEIDUNGSTABELLEN
(G) DECISION TABLES : A BIBLIOGRAPHY
VEB CARL ZEISS
JENA, 1974, 56 PP.
THIS MOSTLY ANNOTATED BIBLIOGRAPHY (133 TITLES) IS SUB-
DIVIDED IN 5 PARTS : INTRODUCTORY TEXTS, CONVERSIONS,
ALGORITHMS, SOFTWARE DESCRIPTIONS, PRACTICAL EXPERIENCES
AND MONOGRAPHS. THE GERMAN LITERATURE ON DECISION TABLES
IS THOROUGHLY TREATED ; MANY OTHER IMPORTANT ARTICLES
AND BOOKS HOWEVER ARE MISSING. SOME OF THE REFERENCES
ARE INCOMPLETE OR WRONG.