MICROFILMED
BY
CENTRAL MICROFILM UNIT
PUBLIC ARCHIVES
OF
CANADA
OTTAWA, ONTARIO

MICROFILMÉ
PAR LE
SERVICE CENTRAL DU MICROFILM
ARCHIVES PUBLIQUES
DU
CANADA
OTTAWA, ONTARIO

24/4/74 DATE

REDUCTION 15%

A. L. ... OPERATOR
OPERATEUR

EXPOSURE
EXPOSITION 2.2

**NATIONAL LIBRARY
OF CANADA**

**CANADIAN THESES
ON MICROFILM**

**BIBLIOTHÈQUE
NATIONALE
DU CANADA**

**THÈSES CANADIENNES
SUR MICROFILM**

1 6 5 5 2

16552

NATIONAL LIBRARY
OTTAWA

BIBLIOTHÈQUE NATIONALE
OTTAWA

NAME OF AUTHOR..... Mehmet Esin Ulug.....
TITLE OF THESIS.. A Unified Theory of the Algebraic Topological
... Methods for the Synthesis of Switching Systems
.....
UNIVERSITY..... Carleton University.....
DEGREE FOR WHICH THESIS WAS PRESENTED..... Ph.D.....
YEAR THIS DEGREE GRANTED..... 1973.....

Permission is hereby granted to THE NATIONAL LIBRARY
OF CANADA to microfilm this thesis and to lend or sell copies
of the film.

The author reserves other publication rights, and
neither the thesis nor extensive extracts from it may be
printed or otherwise reproduced without the author's
written permission.

(Signed)..... *M. Esin Ulug*

PERMANENT ADDRESS:

..2326..WHITEHAVEN
..Ottawa.. K2B5H4
.....

DATED..... *Nov. 30* ..1972

NL-91 (10-68)

A UNIFIED THEORY OF THE ALGEBRAIC TOPOLOGICAL

METHODS FOR THE SYNTHESIS OF SWITCHING SYSTEMS

by

Mehmet Esin Ulug, M.Sc., B.Sc.

A thesis submitted to the Faculty of
Graduate Studies in partial fulfilment
of the requirements for the degree of
Doctor of Philosophy

Division of Systems Engineering

Carleton University
Ottawa, Ontario.

October 11, 1972

The undersigned recommend to the Faculty
of Graduate Studies acceptance of the thesis

"A Unified Theory of the Algebraic
Topological Methods for the Synthesis
of Switching Systems"

submitted by Mehmet Esin Ulug, M.Sc., B.Sc.,
in partial fulfilment of the requirements for
the degree of Doctor of Philosophy.

.....

.....
Thesis Supervisor

.....

.....
Chairman, Division of Systems Engineering

Carleton University
May 8, 1973

ABSTRACT

A set of Logic Operators, first proposed by Roth [16], based on an algebraic topological interpretation of a function table have provided the basis for the subsequent development of algorithms for the computer aided synthesis of combinatorial networks.

This thesis presents the theoretical basis of the development of a unified theory which condenses four different operator tables into a groupoid of order six. It is shown that this groupoid structure, upon examination, can be further refined to yield a four element Boolean ring with unity, the quotient ring of which is isomorphic to Z_2 field.

It is shown that this ring structure, with its "and" and "exclusive-or" binary operations, makes use of quartal coding and relates to a four state Boolean algebra.

By presenting several computer programmes, it is also shown that the unified theory in this new space yields simplicity and greater ease in programming, and results in approximately 20% improvement in storage space and execution time.

ACKNOWLEDGEMENTS

The author wishes to express his gratitude to his supervisor, Professor B.A. Bowen, for his guidance, advice and criticism during research and assistance in preparing this thesis. The author is also indebted to Professor A.R. Boothroyd for his early encouragement and for providing the opportunity to embark on this project.

The research was carried out at Microsystems International Limited, Ottawa, Ontario, as part of the "Combinational Logic" case. The permission given by Microsystems International Limited to use these results as part of this thesis is gratefully acknowledged. Special thanks are expressed to J.M. Beddoes, Vice-President - Technical Development, for his understanding and encouragement during the Ph.D programme.

The author is grateful to Professor D.L. Dietmeyer of Wisconsin University for supplying his computer programme and to Mr. P. Humbert-Droz for modifying and re-writing certain subroutines of this programme in accordance with the results of the unified theory.

The skiffful typing of Mrs. L. Guilfoyle is very much appreciated.

Finally the author thanks his wife Doreen for her support and patience during this difficult period.

<u>TABLE OF CONTENTS</u>	<u>PAGE</u>
ABSTRACT	i
ACKNOWLEDGEMENTS	ii
TABLE OF CONTENTS	iii
LIST OF PRINCIPLE SYMBOLS	vii
LOCATION OF TABLES	ix
LOCATION OF FIGURES	xi
 <u>I. PRELIMINARY</u>	
1.1 CUBICAL COMPLEXES	1
1.2 CALCULUS OF CUBICAL COMPLEXES	4
 <u>II. INTRODUCTION</u>	
2.1 MOTIVATION	6
2.2 DEVELOPMENT OF THE THEORY	7
 <u>III. GROUPOID STRUCTURE</u>	
3.1 QUATERNIONS OF HAMILTON	10
3.2 IDEMPOTENT SCALAR SEMIGROUP	11
3.3 COMMUTATIVE VECTOR GROUP	13
3.4 COMMUTATIVE VECTOR SEMIGROUP	14
3.5 "HYBRID" GROUPOID OF ORDER SIX	16
3.6 THE SECOND MAPPING FUNCTION	18
3.7 DEFINITION OF THE SECOND MAPPING FUNCTION	22
3.7.1 SUBSUME OPERATOR	22
3.7.2 CUBE INTERSECTION OPERATOR	22
3.7.3 STAR PRODUCT OPERATOR	23
3.7.4 SHARP OPERATOR	23
3.8 APPLICATION OF THE UNIFIED THEORY	26
3.8.1 SUBSUME OPERATOR	26
3.8.2 CUBE INTERSECTION OPERATOR	27

	<u>PAGE</u>
3.8.3 STAR PRODUCT OPERATOR	28
3.8.4 SHARP OPERATOR	29
3.9 UNIFIED THEORY AND THE CUBICAL ARRAYS	31
3.9.1 ABSORB	31
3.9.2 CUBE UNION	32
3.9.3 CUBE INTERSECTION	35
3.9.4 SHARP PRODUCT	38

IV. RING STRUCTURE

4.1 ANOTHER APPROACH TO THE UNIFIED THEORY	42
4.2 SIMPLIFICATION OF SUBSUME ALGORITHM	43
4.3 SIMPLIFICATION OF SHARP ALGORITHM	48
4.4 REDUNDANCIES IN THE ALGORITHM	55
4.5 FOUR ELEMENT BOOLEAN RING WITH UNITY	60
4.6 REDEFINITION OF THE SECOND MAPPING FUNCTION	66
4.7 DEFINITION OF THE THIRD MAPPING FUNCTION	70
4.8 BOOLEAN RING AND THE CUBICAL ARRAYS	73
4.8.1 CUBE UNION	74
4.8.2 CUBE INTERSECTION	75
4.8.3 SHARP PRODUCT	76

V. FIELD STRUCTURE

5.1 STILL ANOTHER APPROACH TO THE UNIFIED THEORY	80
5.2 IDEALS AND QUOTIENT RINGS	81
5.3 HOMOMORPHISMS AND ISOMORPHISMS	88
5.4 REDEFINITION OF THE THIRD MAPPING FUNCTION	91

VI. COMPARISONS AND CONCLUSIONS

6.1 COMPARISON OF ALGEBRAIC STRUCTURES	94
6.1.1 GROUPOID STRUCTURE	94
6.1.2 RING STRUCTURE	95
6.1.3 FIELD STRUCTURE	96
6.2 CONCLUSIONS	97

PAGE

VII. COMPUTER PROGRAMMES

7.1	COMPUTER PROGRAMMES BASED ON THE UNIFIED THEORY	98
7.2	ASSIGNMENT OF OCTAL CODE WORDS	99
7.3	SUBROUTINES FOR MANIPULATING CUBICAL ARRAYS	104
7.3.1	RHEAD	104
7.3.2	WHEAD	104
7.3.3	ERRORS	105
7.3.4	READAR	105
7.3.5	PRNTAR	105
7.3.6	COMBINE & COPYAR	105
7.3.7	CPYCUB & MOVCUB	106
7.4	A COMPUTER PROGRAMME USING QUARTAL CODING	106
7.4.1	SUBROUTINE SUBSUM	107
7.4.2	SUBROUTINE INTERX	107
7.4.3	SUBROUTINE SHARPS	109
7.4.4	SUBROUTINE CNSSUS	110
7.5	TWO COMPUTER PROGRAMMES USING OCTAL CODING	110
7.6	A COMPUTER PROGRAMME USING INTERRELATED SUBROUTINES	112

VIII ALGORITHMIC EFFICIENCY

8.1	IMPLEMENTATION AND MEASUREMENT OF ALGORITHMIC EFFICIENCY..	113
8.2	MEMORY STORAGE ALLOCATION	116
8.3	LINKAGES OF SUBROUTINES	120
8.4	EXECUTION TIME	125
8.5	CONCLUSIONS	128

XI CLOSING REMARKS

9.1	SUMMARY OF THE WORK PRESENTED IN THE THESIS	129
9.2	VIEWS ON THE FUTURE WORK	131

	<u>PAGE</u>
APPENDIX A	133
A Computer Programme Using Quartal Coding	
APPENDIX B	151
A Computer Programme Using Octal Coding With Improved Sharps Subroutine	
APPENDIX C	166
A Computer Programme Using Octal Coding	
APPENDIX D	183
A Computer Programme Using Interrelated Subroutines	
APPENDIX E	201
Idempotent Scalar Quaternians	
APPENDIX F	213
Boolean Algebra of the Unified Theory	
BIBLIOGRAPHY	217

LIST OF PRINCIPLE SYMBOLS

SYMBOL

Z_2	= The field of integer modulo 2, {0, 1}
$R^{(n)}$	= n dimensional Euclidian space
X_j	= n dimensional input vector
Y_i	= m dimensional output vector
C_{ij}	= n x m transformation matrix over the field Z_2 .
Q	= The set {0, 1, x}
x	= "don't care" symbol
N	= {0, 1, 2, ... n} The set of first n positive integers
ρ	= Mapping function from N to Q
x_i	= Co-ordinates of a cube in Q domain
$K(f)$	= Cubical complex
$\langle Q, +, \cdot \rangle$	= Quaternions of Hamilton
S	= The set {0, 1, i, j, k} derived from Quaternions of Hamilton
$\langle S, \cdot \rangle$	= Idempotent Scalar semigroup
H	= {1, i, j, k} The subset of S
$\langle H, x \rangle$	= Commutative Vector group.
$\langle S, x \rangle$	= Commutative Vector semigroup.
U	= The set {0, 1, i, j, k, K}
$\langle U, * \rangle$	= Hybrid groupoid
ψ	= Mapping function from Q to U
y_i	= Co-ordinates of a cube in U domain.
$\emptyset$	= Empty element
e	= Identity element

SYMBOL

$\sqsubseteq$	=	Subsume Operator
$\sqcap$	=	Cube Intersection Operator
$*$	=	Star Product Operator
$\#$	=	Sharp Operator
$\sqcup$	=	Cube Union Operator
X	=	The set $\{k, e, K\}$
Y	=	The set $\{k, e, \emptyset, K\}$
Σ	=	Absorption Operator
B	=	$\{0, 1, i, j\}$ The subset of S
B'	=	The set $\{\emptyset, e, i, j\}$
B''	=	The set $\{e, K, i, j\}$
L	=	The set $\{\emptyset, e, i, j, K\}$
ξ	=	The mapping function from Q to L domain
z_i	=	Co-ordinates of cubes in L domain
W	=	The set $\{e, K\}$
V	=	The set $\{e, \emptyset, K\}$
ζ	=	Mapping function from L to B domain.
Z_i	=	Co-ordinate of a cube in B domain.
R/N	=	Residue class ring R modulo N
$\langle B, \wedge, \oplus \rangle$	=	Boolean ring B with unity
$[G : H]$	=	Index of H in G .
$\times_{i=1}^4 Z_2^{(i)}$	=	External direct product of Z_2 under addition with itself four times.

LOCATION OF TABLES

<u>TABLE</u>	<u>TITLE</u>	<u>PAGE</u>
I	IDEMPOTENT SCALAR SEMIGROUP	11
II	COMMUTATIVE VECTOR GROUP	13
III	COMMUTATIVE VECTOR SEMIGROUP	14
IV	HYBRID GROUPOID OF ORDER SIX	16
V	HYBRID GROUPOID WITH REGENERATIVE REGIONS ...	19
VI	SUBSUME ALGORITHM	22
VII	CUBE INTERSECTION ALGORITHM	22, 5
VIII	STAR PRODUCT ALGORITHM	23
IX	SHARP ALGORITHM	23
X	OPERATOR DEPENDENCE OF x_i^a	24
XI	UNIFIED IMAGES UNDER ψ	24
XII	OPERATOR DEPENDENCE OF y_i^c	25
XIII	UNIFIED IMAGES UNDER ψ^{-1}	25
XIV	CUBE UNION OF ARRAYS	34
XV	$\langle B', \cdot \rangle$ SEMIGROUP	58
XVI	$\langle B'', + \rangle$ GROUP	58
XVII	$\langle B, \wedge \rangle$ SEMIGROUP	61
XVIII	$\langle B, \oplus \rangle$ GROUP	63
XIX	OPERATOR DEPENDENCE OF x_i^a IN ξ MAPPING	67
XX	UNIFIED IMAGES UNDER ξ	68
XXI	OPERATOR DEPENDENCE OF z_i^c	69
XXII	UNIFIED IMAGES UNDER ξ^{-1}	69
XXIII	SUBSETS & OPERATIONS IN ζ	70
XXIV	OPERATOR DEPENDENCE OF ζ	71
XXV	SUBGROUP $\langle N, \oplus \rangle$	82
XXVI	SEMIGROUP $\langle N, \wedge \rangle$	83
XXVII	SPLITTING OF $\langle B, \oplus, \wedge \rangle$ UNDER $\oplus$	85
XXVIII	SPLITTING OF $\langle B, \oplus, \wedge \rangle$ UNDER $\wedge$	85
XXIX	Z_2 FIELD STRUCTURE - $\oplus$ GROUP	87
XXX	Z_2 FIELD STRUCTURE - $\wedge$ GROUP	87
XXXI	ABELIAN GROUP $\langle x_{i=1}^4 z_2^{(i)}, + \rangle$	205
XXXII	ABELIAN SEMIGROUP $\langle x_{i=1}^4 z_2^{(i)}, \cdot \rangle$	206

LOCATION OF TABLES

<u>TABLE</u>	<u>TITLE</u>	<u>PAGE</u>
XXXIII	EVOLUTION OF PROGRAMME A	117
XXXIV	SYSTEM AND BOOKKEEPING SUBROUTINES	119
XXXV	COMPARISON OF MEMORY STORAGE ALLOCATIONS	121
XXXVI	LINKAGES OF SUBROUTINES IN PROGRAMME A	123
XXXVII	LINKAGES OF SUBROUTINES IN DIETMEYER'S PROGRAMME ..	124
XXXVIII	COMPARISON OF EXECUTION TIMES	126
XXXIX	<K, +> GROUP	210
XXXX	<K, ·> SEMIGROUP	210
XXXXI	<B, V> SEMIGROUP	215

LOCATION OF FIGURES

<u>FIGURE</u>	<u>TITLE</u>	<u>PAGE</u>
1	DEVELOPMENT OF THE UNIFIED THEORY.....	9
2	HOMOMORPHISM $\emptyset : R \rightarrow R'$	89
3	ISOMORPHISM $\psi = R \cdot \emptyset \rightarrow B/\text{Ker } \emptyset$	89
4	TOPOLOGY OF CODING	103
5	COSETS OF $\langle X_{i=1}^4 Z_2^{(i)}, + \rangle$	207
6	SUMMARY OF MAPPING FUNCTIONS	130

CHAPTER I

PRELIMINARY

1.1 CUBICAL COMPLEXES

For a single variable x of a combinatorial logic function f , the field Z_2 is a subset of a real line, or the one dimensional Euclidean space R .

For n variables $(x_1, x_2, \dots, x_n)$ the n 'th Cartesian product $Z_2^{(n)}$ can be considered as a subset of an n dimensional Euclidean space $R^{(n)} = R \times R \dots \times R$. Obviously $Z_2^{(n)}$ consists of 2^n elements and each element can be considered as an ordered n -tuple, or a vector x_i , or a point in n dimensional space.

Let f be a mapping of $Z_2^{(n)}$ into Z_2 of integer modulo 2

$$f : Z_2^{(n)} \longrightarrow Z_2$$

$Z_2^{(n)}$ can be thought of as an n -dimensional real unit cube and its 2^n elements as 0-cubes or vertices.

The set $(1).f^{-1}$ is called the on-set of the switching function and the set $(0).f^{-1}$ is called the off-set of the switching function.

In general, a combinatorial logic circuit can be described by a time invariant relationship such as

$$Y_i = C_{ij} X_j$$

where X_j is an n dimensional vector

Y_i is an m dimensional vector

C_{ij} is an $n \times m$ transformation matrix over the field Z_2 .

This relationship implies that Y_i is true only for those X_j 's which are related to Y_i by the coefficients of the transformation matrix $C_{ij} = 1$.

When a zero cube is transformed to another zero cube by the change of a single co-ordinate, two such zero cubes are termed a 1-cube. For example, 111 is transformed into 101 by changing one co-ordinate. The resulting 1-cube is designated as $1x1$, where x is generally referred to as the "don't care" symbol. The use of the term "don't care" in this particular application is somewhat ambiguous. As it will be shown later, x plays a key role in the unified theory of the algebraic topological methods. It is convenient to think of x as a variable taking on the values of 0 and 1, and call it a free component.

Let us consider the set $Q = \{0, 1, x\}$, where x is the "don't care" term as defined above. Also let $N = \{1, 2, 3, \dots, n\}$ denote the set of first n positive integers. Then the mapping

$$\rho : N \longrightarrow Q$$

determines a subset C of $Z_2^{(n)}$. A vertex $\{x_1, x_2, \dots, x_n\}$ of $Z_2^{(n)}$ is in the subset C , if and only if $x_i = i.\rho$ for all $i \in N$ such that $i.\rho \neq x$. This subset C of $Z_2^{(n)}$ is a cube in $Z_2^{(n)}$ defined by the mapping function ρ . For every positive integer $i \in N$, the element $i.\rho \in Q$ is called the i 'th co-ordinate of the cube C . Hence the cube C is completely determined by its co-ordinates, i.e.

$$\{(1).\rho, (2).\rho, \dots, (n).\rho\}$$

The dimension of the cube is determined by the cardinality of the subset $x.\rho^{-1}$ of N . The 0-cubes have already been described as the vertices of the $Z_2^{(n)}$. The 1-cubes are called the edges of $Z_2^{(n)}$. Similarly the 2-cubes are called the faces of $Z_2^{(n)}$. Two 1-cubes are said to form opposite faces of a 2-cube if their free elements, i.e. x , occupy the same co-ordinate, and if exactly one of their bound co-ordinates, i.e. 0 or 1, disagree. For example, $0x1$ and $0x0$ are the opposite edges of the face $0xx$.

By induction, if the dimension of a cube is r , then C is called an r -cube. It can be shown that the number of r cubes in $Z_2^{(n)}$ is given by:

$$N(r) = \frac{n! 2^{n-r}}{r!(n-r)!} \quad \text{where } 0 < r \leq n$$

In the algebra of groups, the complex is defined as follows:

A non-empty subset S of a group $\langle G, * \rangle$ is called a complex of G ; S is then called a subgroup of G , if S is itself a group under the induced binary operation $*$.

The cubical complex is defined as follows:

Let f be arbitrarily given subset of $Z_2^{(n)}$. The collection of $K(f)$ of all cubes in $Z_2^{(n)}$ contained in the set f is called the cubical complex. For each non-negative integer, let $K_r(f)$ denote the collection of all r -cubes in $Z_2^{(n)}$ contained in the set f . Then $K(f)$ is the union of the following collections:

$$K(f) = K_0(f) \cup K_1(f) \cup \dots \cup K_n(f)$$

If the cubical complex contains the vertices of a cube, then it contains the cube itself.

1.2 CALCULUS OF CUBICAL COMPLEXES

Geometrical cubes have long been used by the switching theorist. However, their representation as an ordered set of symbols 0, 1, or x made possible the development of a calculus of cubical complexes.

Calculus of cubical complexes concentrates on the algorithmic synthesis of switching systems by the use of algebraic topological methods. Such algorithms are specifically designed for execution by the digital computer.

There are several logic operators associated with each algorithm. The following four operators are the basic building blocks used in the calculus of cubical complexes:

1. Subsume Operator
2. Cube Intersection Operator
3. Star Product Operator
4. Sharp Operator

Other operators can be derived from these four as the need arises.

These operators are either applied individually to a pair of cubes, say "a" cube and "b" cube, or a few of them are combined together in a broader algorithm and applied to two arrays of cubes, say A array with the cubes $a_1, a_2, \dots a_n$, and B array with the cubes $b_1, b_2, \dots b_m$. It is assumed that all the cubes to be operated on at one time have the same co-ordinate dimensions.

The algorithms associated with each operator are defined in a 3 x 3 table. The rows are assigned to the 0, 1 and x of the "a" cube and the columns to 0, 1 and x of the "b" cubes. For example, the Subsume Operator is defined in the following table.

$\subseteq$		b		
		0	1	x
a	0	ϵ	$\emptyset$	ϵ
	1	$\emptyset$	ϵ	ϵ
	x	$\emptyset$	$\emptyset$	ϵ

TABLE VI
SUBSUME ALGORITHM (also see page 22)

The table is interpreted as follows:

$a \subseteq b$ if $a_i \subseteq b_i = \epsilon$ for all i

$a \not\subseteq b$ if $a_i \subseteq b_i = \emptyset$ for any i

where a_i and b_i represent the co-ordinates of "a" and "b" cubes respectively. The notation $a \subseteq b$ is read as "a" cube subsumes "b" cube or "b" cube covers "a" cube.

Since not all the operators have symmetrical tables, the correct assignment of the rows and the columns to the "a" and "b" cubes is very important.

When one of these operators is applied to a pair of cubes, the outcome is always one of the following:

1. A "yes" or "no" answer depending on whether or not the algebraic operations encounter one " $\emptyset$ ". (Subsume Operator)
2. A new cube or cubes. (Sharp Operator)
3. A new cube allowing one " $\emptyset$ " and placing an "x" in place of the empty literal. (Star Product Operator)
4. Combination of (1) and (2). (Intersection Operator)

CHAPTER II

INTRODUCTION

2.1 MOTIVATION

The advantages of the algebraic topological methods, as far as providing mechanization and visualization of the combinatorial logic problems, can be greatly improved if one can develop a unified theory which links all the operators together on a sound mathematical basis. If such a unified theory could be developed, it would be desirable to relate the mathematical basis of this theory to the fundamental concepts of logic, so that it would yield still simpler algorithms.

As the operators become more complex and their numbers increase, the use of tables becomes very cumbersome. From the software point of view, it is necessary to write a separate computer programme for each operator. In handling cube arrays using more than one operator, the problems associated with programming become more evident. From the pedagogical point of view, a unified theory can be a definite improvement in presentation of the algebraic topological methods.

Careful examination of the algorithms used for the computer aided synthesis of the switching systems reveals that a unified theory can be founded on the following algebraic structures:

- (1) Abelian groupoid of order six having an identity element.
- (2) Four element Boolean ring with unity.
- (3) Field of Z_2 .

It will be shown that as the unified structure moves up in the algebraic hierarchy from a groupoid to a field, the theory provides a better insight to how the various algorithms are used, how they are linked together, and how they are related to the basic concepts of logic, such as Boolean algebra, Boolean rings, and $Z_2 = \{0, 1\}$ field.

2.2 DEVELOPMENT OF THE THEORY

The theory will be developed as follows:

- (a) A set S having five elements will be derived from the Quaternions of Hamilton.
- (b) A binary operation called "Idempotent Scalar Product" will be defined and applied to the set S . It will be shown that the resulting algebraic structure is an Abelian Semigroup having an identity element.
- (c) A second binary operation called "Commutative Vector Product" will be defined and applied to a subset H of the set S . It will be shown that the resulting algebraic structure is the Klein 4-group (Viergruppe), isomorphic to $Z_2 \times Z_2$.
- (d) The Commutative Vector Product operation will then be applied to the set H in an attempt to unify the two binary operations. It will be shown that the resulting algebraic structure is a semigroup having the same identity element as the semigroup generated by the Idempotent Scalar Product operation.
- (e) The algebraic properties of the two semigroups will be combined into an Abelian groupoid of order six having an identity element. This groupoid and its binary operation will be called "Hybrid". It will be shown that when the co-ordinates 0, 1 and x of "a" and "b" cubes are mapped into the elements of the Hybrid groupoid, the images of all the co-ordinates, except one, are invariant to the choice of operators. In other words, only the x co-ordinates of the "a" cube are operator dependent. Hence, the unified theory of algebraic topological methods will condense four different tables of algorithms into a single groupoid structure of order six using octal coding.

- (f) Certain redundancies in subsume and sharp operations will be discovered. Elimination of these redundancies will lead the way towards the establishment of a more compact theory based on a four element Boolean ring with unity using bit by bit "and" and "exclusive-or" operations. These binary operations will link the unified theory of the algebraic topological methods to the Boolean algebra and Boolean rings. Moreover it will be possible to replace the octal coding with quartal coding, which requires less storage space and execution time in the digital computer.
- (g) It will be shown that four element Boolean ring with unity has a maximal ideal. This ideal and its coset will split the Boolean ring into blocks with induced operations well defined. As will be expected, the resultant quotient ring is a field isomorphic to Z_2 .
- (h) Furthermore, it will be shown that two basic concepts which are the foundations of the unified theory, namely "idempotent scalar" and "commutative vector", are contained in the extremely simple field structure of Z_2 .
- (i) The unified theory based on the field of Z_2 , using binary coding with a sign convention, will produce results which are very similar to the ones obtained using a Boolean ring structure.

The development of the unified theory starting from a set of logic operators first proposed by Roth [16] is shown in Figure 1 on page 9 .

FIGURE 1. DEVELOPMENT OF THE UNIFIED THEORY

CHAPTER III
GROUPOID STRUCTURE

3.1 QUATERNIONS OF HAMILTON

Examination of algorithms used for the computer aided synthesis of switching systems reveals that geometrical cubes, represented by an ordered set of symbols 0, 1, or x, exhibit properties of algebraic structures with binary operations resembling scalar and cross product of Cartesian vectors. This is rather unusual because the scalar vector multiplication on a vector set, for example, is not a closed binary operation.

It would appear therefore that Quaternions of Hamilton, $\langle Q, +, \cdot \rangle$, would be a good starting point in the development of the theory, because it is one of those rare algebraic structures, a skew field over the real numbers, using addition by components and multiplicative rules resembling the cross product of vectors.

We shall present here another type of quaternion and derive from it certain sets which will be used in building algebraic structures.

Let $X_{i=1}^4 Z_2^{(i)}$ denote the external direct product of Z_2 under addition with itself four times. We shall let

$1 = (1, 0, 0, 0)$, $i = (0, 1, 0, 0)$, $j = (0, 0, 1, 0)$, $k = (0, 0, 0, 1)$.

Also we let

$a_1 = (a_1, 0, 0, 0)$, $a_2 i = (0, a_2, 0, 0)$, $a_3 j = (0, 0, a_3, 0)$, $a_4 k = (0, 0, 0, a_4)$.

In view of our definition of addition we have:

$$(a_1, a_2, a_3, a_4) = a_1 + a_2 i + a_3 j + a_4 k \text{ where } a_i \in Z_2.$$

If we define the multiplication using "Idempotent Scalar" operation given in section 3.2, it can be shown that $\langle X_{i=1}^4 Z_2^{(i)}, +, \cdot \rangle$ is a sixteen element Boolean ring with unity (See Appendix E). The study of this algebraic structure reveals that one of its quotient rings is isomorphic to Z_2 field and the other one is a four element Boolean ring with unity. Hence the relationships introduced by this new quaternion map out an effective approach which will be used in the development of the unified theory.

3.2 IDEMPOTENT SCALAR SEMIGROUP

One of the most common algebraic operations used in the topological methods is based on the following postulates:

$$G1 \quad \delta_i^j = \begin{cases} 0 & \text{if } i \neq j \\ u_i & \text{if } i = j \end{cases}$$

$$G2 \quad 1 \cdot u_i = u_i \cdot 1 = u_i$$

$$G3 \quad 0 \cdot u_i = u_i \cdot 0 = 0$$

$$G4 \quad 0 \cdot 1 = 1 \cdot 0 = 0$$

where u_i represent u_x, u_y, u_z and δ_i^j is similar to Kronocker's delta, a mixed tensor of rank two. It can be easily seen that if we use the symbols i, j, k instead of u_x, u_y, u_z , the same postulates can be specified by defining a binary operation on the set H as shown in Table I.

.	0	1	i	j	k
0	0	0	0	0	0
1	0	1	i	j	k
i	0	i	i	0	0
j	0	j	0	j	0
k	0	k	0	0	k

TABLE I

IDEMPOTENT SCALAR SEMIGROUP

This binary operation will be called "Idempotent Scalar Product".

By inspecting the table, it can be seen that:

- 1.) Binary operation is closed.
- 2.) There is an identity element $e = 1$.
- 3.) Groupoid is Abelian.
- 4.) Elements of the groupoid are idempotent.

In addition, it can be proved that the binary operation is associative.

For example:

a.) $(i.j).k = 0.k = 0$	d.) $i.(i.1) = i.i = i$
$i(j.k) = i.0 = 0$	$(i.i).1 = i.1 = i$
b.) $(i.i).k = i.k = 0$	e.) $(0.i).j = 0.j = 0$
$i.(i.k) = i.0 = 0$	$0.(i.j) = 0.0 = 0$
c.) $1.(i.j) = 1.0 = 0$	f.) $0.(k.1) = 0.k = 0$
$(1.i).j = i.j = 0$	$(0.k).1 = 0.1 = 0$

Hence, the algebraic structure $\langle S, . \rangle$ given by Table I is an Abelian semigroup having an identity element. Since all of its elements are idempotent, each element is its own additive inverse. This raises the question of $\langle S, +, . \rangle$ being a Boolean Ring. Particularly the theorem which states that "a Boolean Ring with more than two elements is a ring with divisors of zero" supports this possibility since $\langle S, . \rangle$ does have divisors of zero. Unfortunately, however, since five is a prime number, there is only one group of order five up to isomorphism and that group is cyclic. Hence the group $\langle S, + \rangle$ cannot have its additive identity zeros placed on the diagonal of the table and therefore "Idempotent Scalar" semigroup is not a multiplicative part of a Boolean Ring.

3.3 COMMUTATIVE VECTOR GROUP

Let us now define a second algebraic operation based on the following postulates:

- E1 . $i \times j = j \times i = k$ $j \times k = k \times j = i$ $k \times i = i \times k = j$
 E2 . $i \times i = 1$, $j \times j = 1$, $k \times k = 1$
 E3 . $1 \times i = i \times 1 = i$ $1 \times j = j \times 1 = j$, $1 \times k = k \times 1 = k$

The same postulates can be specified by defining a binary operation on the subset $H = \{1, i, j, k\}$ of the set S as shown in Table II:

X	1	i	j	k
1	1	i	j	k
i	i	1	k	j
j	j	k	1	i
k	k	j	i	1

TABLE II
COMMUTATIVE VECTOR GROUP

This binary operation will be called "Commutative Vector Product". By inspecting the table, it can be seen that the algebraic structure $\langle H, X \rangle$ given by the table II is the Klein 4-group (Viergruppe), isomorphic to $Z_2 \times Z_2$.

3.4 COMMUTATIVE VECTOR SEMIGROUP

It has been shown that one of the most common algebraic operations used in the topological methods can be represented by a semigroup of order five. In addition, a group of order four has been defined. The unification of these two structures into one could be greatly simplified if they were both of order five. To achieve this, let us apply the "Commutative Vector Product" operation to the set $S = \{0, 1, i, j, k\}$ by introducing a fourth postulate, namely

$$\begin{aligned} E4. \quad 0 \times 1 &= 1 \times 0 = 0 & 0 \times i &= i \times 0 = 0 & 0 \times j &= j \times 0 = 0 \\ & & 0 \times k &= k \times 0 = 0 \end{aligned}$$

The results are shown in Table III:

x	0	1	i	j	k
0	0	0	0	0	0
1	0	1	i	j	k
i	0	i	1	k	j
j	0	j	k	1	i
k	0	k	j	i	1

TABLE III

COMMUTATIVE VECTOR SEMIGROUP

By inspecting the table, it can be seen that

- 1.) Binary operation is closed
- 2.) There is an identity element $e = 1$
- 3.) Groupoid is Abelian

The associativity can be proven as follows:

a.) $(i \times j) \times k = k \times k = 1$
 $i \times (j \times k) = i \times i = 1$

b.) $(i \times i) \times k = 1 \times k = k$
 $i \times (i \times k) = i \times j = k$

c.) $1 \times (i \times j) = 1 \times k = k$
 $(1 \times i) \times j = i \times j = k$

d.) $i \times (i \times 1) = i \times i = 1$
 $(i \times i) \times 1 = 1 \times 1 = 1$

e.) $(0 \times i) \times j = 0 \times j = 0$
 $0 \times (i \times j) = 0 \times k = 0$

f.) $0 \times (k \times 1) = 0 \times k = 0$
 $(0 \times k) \times 1 = 0 \times 1 = 0$

Hence the algebraic structure $\langle H, \times \rangle$ given by table III is an Abelian semigroup having an identity element.

3.5 "HYBRID" GROUPOID OF ORDER SIX

In order to combine "Idempotent Scalar" and Commutative Vector" semigroups, let us introduce a new set U as follows:

$$U = \{0, 1, i, j, k, K\}$$

Obviously the set U contains both H and S sets. Let u_i represent u_x , u_y , and u_z where $u_x = i$, $u_y = j$ and $u_z = k$. The following postulates can then be specified in a compact form:

$$F1. \quad \delta_i^j = \begin{cases} 0 & \text{if } i \neq j \\ u_i & \text{if } i = j \end{cases}$$

$$F2. \quad K * i = i * K = j, \quad K * j = j * K = i, \quad K * k = k * K = 1$$

$$F3. \quad 0 * 0 = 0, \quad 1 * 1 = 1, \quad u_i * u_i = u_i, \quad K * K = K$$

$$F4. \quad 0 * 1 = 1 * 0 = 0, \quad 0 * u_i = u_i * 0 = 0, \quad 0 * K = K * 0 = 0$$

$$F5. \quad 1 * u_i = u_i * 1 = u_i, \quad 1 * K = K * 1 = K$$

The same postulates can be specified by defining a binary operation $*$ on the set $U = \{0, 1, i, j, k, K\}$ as shown in Table IV

*	0	1	i	j	k	K
0	0	0	0	0	0	0
1	0	1	i	j	k	K
i	0	i	i	0	0	j
j	0	j	0	j	0	i
k	0	k	0	0	k	1
K	0	K	j	i	1	K

TABLE IV

HYBRID GROUPOID OF ORDER SIX

This binary operation will be called "Hybrid Product". By inspecting the table it can be seen that:

- 1.) Binary operation is closed.
- 2.) There is an identity element $e = 1$.
- 3.) Groupoid is Abelian.
- 4.) Elements of the groupoid are idempotent.

Further examination of the table reveals that whereas the interaction of the elements i, j, k with each other is "Idempotent Scalar", the interaction of K with the elements i and j is "Commutative Vector". Because of this property of the "Hybrid" operation, the associativity breaks down. For example:

$$\begin{aligned} \text{a.) } (j * k) * K &= 0 * K = 0 \\ j * (k * K) &= j * 1 = j \end{aligned}$$

$$\begin{aligned} \text{b.) } (i * i) * K &= i * K = j \\ i * (i * K) &= i * j = 0 \end{aligned}$$

$$\begin{aligned} \text{c.) } (K * K) * i &= K * i = j \\ K * (K * i) &= K * j = i \end{aligned}$$

Hence the algebraic structure $\langle U, * \rangle$ given by the Table IV is an Abelian groupoid having an identity element. Moreover, since all of its elements are idempotent, each element is its own additive inverse.

3.6 THE SECOND MAPPING FUNCTION

The first mapping function was defined as

$$\rho : N \longrightarrow Q$$

where $N = \{1, 2, \dots, n\}$ was the set of first "n" positive integers and the set $Q = \{0, 1, x\}$ contained $\{0, 1\}$, the bound elements of the field Z_2 and the free component x taking on the values of 0 and 1.

Then for the "a" cube

$$\rho^a : N \longrightarrow Q \text{ such that } x_i^a = i \cdot \rho^a, \text{ where } i \in N.$$

x_i^a will be called the i'th co-ordinate of the "a" cube in Q domain.

Let us now define a second mapping function ψ as follows:

$$\psi : Q \longrightarrow U \text{ such that } y_i^a = x_i^a \cdot \psi, \text{ where } i \in N.$$

y_i^a will be called the i'th co-ordinate of the "a" cube in U domain.

Unfortunately, two elements in each of the sets $Q = \{0, 1, x\}$ and $U = \{0, 1, i, j, k, K\}$ have been denoted by 0 and 1. Since these elements are in no way related, let us replace 0 and 1 of the set U by $\emptyset$, meaning empty element, and by e , meaning identity element, respectively. Hence the set U now becomes as follows:

$$U = \{\emptyset, e, i, j, k, K\}$$

Notice that only the names of the two elements have been changed. The "Hybrid" groupoid $\langle U, * \rangle$ still has the same algebraic structure. This is shown in Table V.

*	$\emptyset$	e	i	j	k	K
$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$
e	$\emptyset$	e	i	j	k	K
i	$\emptyset$	i	i	$\emptyset$	$\emptyset$	$\emptyset$
j	$\emptyset$	j	$\emptyset$	j	$\emptyset$	$\emptyset$
k	$\emptyset$	k	$\emptyset$	$\emptyset$	k	e
K	$\emptyset$	K	$\emptyset$	$\emptyset$	e	K

TABLE V

HYBRID GROUPOID WITH REGENERATIVE REGIONS

Let us consider the application of one of the operators of the topological methods to a pair of cubes, say "a" cube and "b" cube.

The i'th co-ordinates of these cubes in Q domain will be given by:

$$x_i^a = i \cdot \rho^a \quad \text{and} \quad x_i^b = i \cdot \rho^b, \text{ where } i \in \mathbb{N}.$$

Using the composite function $\rho \circ \psi$, the i'th co-ordinates in U domain will be given by:

$$y_i^a = i \cdot \rho^a \circ \psi \quad \text{and} \quad y_i^b = i \cdot \rho^b \circ \psi, \text{ where } i \in \mathbb{N}.$$

When the y co-ordinates of the "a" and "b" cubes have been obtained, it is just a matter of applying the algebra of the "Hybrid" groupoid $\langle U, * \rangle$ as per Table V. Hence the operation in U domain will be as follows:

$$(i \rho^a \circ \psi) * (i \rho^b \circ \psi)$$

Since $\langle U, * \rangle$ is an Abelian groupoid, the binary operation $*$ is commutative.

Mapping back to the Q domain, we have

$$[(i.\rho^a \circ \psi) * (i.\rho^b \circ \psi)] . \psi^{-1}, \text{ where } i \in N.$$

The result may be the co-ordinates of another cube in the Q domain, say x_i^c of "c" cube, or simply " $\emptyset$ " indicating that for example "a" cube does not subsume "b" cube, or "a" cube does not intersect "b" cube.

In the sharp algorithm the results of the binary operation may represent the co-ordinates of more than one cube. The expansion of $a * b$ into an array of cubes takes place when there is more than one confrontation of the type $K * i$ or $K * j$. Four squares of Table $\bar{V}$ have been shaded to indicate the "regenerative areas" of the groupoid. Whenever $y_i^a * y_i^b$ falls into one of these squares, the result of the binary operation will be called regenerative and accompanied by an oblique sign. For example $K * i = j$ will be written as $K * i = j/$ and $K * j = i/$. The oblique sign will be used to partition the set of co-ordinates $(i.\rho^a \circ \psi) * (i.\rho^b \circ \psi)$. The partitioned set will be interpreted as follows:

(a) If the set of co-ordinates have been partitioned only once or not partitioned at all, the results of the binary operation will represent a single cube. For example the following sets $\{i,j,K,i,i,K,j\}$, $\{K,i/,i,j,K,i,K,j\}$ cannot be expanded into an array of cubes.

(b) If the set of co-ordinates has been partitioned more than once, say n times, then the result of the binary operation represents n different cubes. For example if $n = 4$, and the resulting set representing the co-ordinates of four different cubes is as shown below, then the co-ordinates of the individual cubes will be obtained from the partitioned set as follows:

Partitioned Cube	$\{i,j/,K,i/,i,j,j/,i,K,i/,j,K\}$
1st Cube	$\{i,j,K,e,i,j,e,i,K,e,j,K\}$
2nd Cube	$\{i,e,K,i,i,j,e,i,K,e,j,K\}$
3rd Cube	$\{i,e,K,e,i,j,j,i,K,e,j,K\}$
4th Cube	$\{i,e,K,e,i,j,e,i,K,i,j,K\}$

Namely, each regenerative element $i/$ or $j/$ appears once and only once in each cube in a location which is aligned with its original location in the partitioned set. The other regenerative elements are replaced with e 's to complete the set. Hence the number of co-ordinates of each cube is the same as in the partitioned set.

Now e_i appearing in the i 'th column will map back to Q domain as $i.p^a$. Since " e " does not appear among the co-ordinates of either " a " or " $a * b$ " cubes, there is no ambiguity in this representation.

This method of expanding the results of a sharp operation into an array of cubes will be called "partitioning" method and will be used extensively in the proceeding chapters.

3.7 DEFINITION OF THE SECOND MAPPING FUNCTION

Let us define the second mapping function ψ , with the help of the tables of algorithm of the four basic operators used in algebraic topological methods.

(a) Subsume Operator

$\sqsubseteq$		b		
		0	1	x
a	0	ϵ	$\emptyset$	ϵ
	1	$\emptyset$	ϵ	ϵ
	x	$\emptyset$	$\emptyset$	ϵ

TABLE VI - SUBSUME
ALGORITHM

$a \sqsubseteq b$ if $a_i \sqsubseteq b_i = \epsilon$ for all i
 $a \not\sqsubseteq b$ if $a_i \sqsubseteq b_i = \emptyset$ for any i

$\sqsubseteq$	Q domain	U domain
"a" cube	0 $\rightarrow$	i
	1 $\rightarrow$	j
	x $\rightarrow$	k
"b" cube	0 $\rightarrow$	i
	1 $\rightarrow$	j
	x $\rightarrow$	e
"a * b" cube	0 $\leftarrow$	i
	1 $\leftarrow$	j
	x $\leftarrow$	k
	$\emptyset \leftarrow$	$\emptyset$

(b) Cube Intersection Operator

$\sqcap$		b		
		0	1	x
a	0	0	$\emptyset$	0
	1	$\emptyset$	1	1
	x	0	1	x

TABLE VII - CUBE INTERSECT.
ALGORITHM

$a \sqcap b$ if $a_i \sqcap b_i = \emptyset$ for any i

$\sqcap$	Q domain	U domain
"a" or "b" cube	0 $\rightarrow$	i
	1 $\rightarrow$	j
	x $\rightarrow$	e
a * b cube	0 $\leftarrow$	i
	1 $\leftarrow$	j
	x $\leftarrow$	e
	$\emptyset \leftarrow$	$\emptyset$

(c) Star Product

*		b		
		0	1	x
a	0	0	$\emptyset$	0
	1	$\emptyset$	1	1
	x	0	1	x

TABLE VIII - STAR PRODUCT
ALGORITHM

$a * b = \emptyset$ if $a_i * b_i = \emptyset$

for more than one i.

*	Q domain	U domain
"a" or "b" cube	0 $\rightarrow$	i
	1 $\rightarrow$	j
	x $\rightarrow$	e
a * b cube	0 $\leftarrow$	i
	1 $\leftarrow$	j
	x $\leftarrow$	e
	x $\leftarrow$	1st $\emptyset$
	$\emptyset \leftarrow$	2nd $\emptyset$

(d) Sharp Operator

#		b		
		0	1	x
a	0	ϵ	$\emptyset$	ϵ
	1	$\emptyset$	ϵ	ϵ
	x	1	0	ϵ

TABLE IX - SHARP ALGORITHM

$a \# b = a$ if $a_i \# b_i = \emptyset$ for any i, which implies

$a \nsubseteq b$ and $a \# b = \emptyset$ if $a \subseteq b$.

Otherwise $a \# b =$

$$\bigcup_i (a_1, a_2, \dots, \bar{b}_i, \dots, a_n)$$

where the union is for all

i for which $a_i \# b_i = \alpha_i \in \{0,1\}$

#	Q domain	U domain
"a" cube	0 $\rightarrow$	i
	1 $\rightarrow$	j
	x $\rightarrow$	K
"b" cube	0 $\rightarrow$	i
	1 $\rightarrow$	j
	x $\rightarrow$	e
a * b cube or cubes	0 $\leftarrow$	i
	1 $\leftarrow$	j
	x $\leftarrow$	K
	$\emptyset \leftarrow$	$\emptyset$
	$i.p^a$	e_i

Review of these tables reveals the following:

1. ψ is invariant to the choice of operators for all the co-ordinates of the "b" cube. Namely:

<u>Q domain</u>		<u>U domain</u>
0	→	i
1	→	j
x	→	e

2. ψ is invariant to the choice of operators for the co-ordinates of the "a" cube having 0's and 1's. Namely:

<u>Q domain</u>		<u>U domain</u>
0	→	i
1	→	j

3. ψ is operator dependent for the co-ordinates of the "a" cube having x's. This dependence is shown in Table X:

$\sqsubseteq$	$\sqcap$ or *	#
k	e	K

TABLE X- OPERATOR DEPENDENCE OF x_i^a

Let us define a subset X of the set U to represent operator dependence of ψ as follows:

$$X = \{k, e, K\}$$

The images of 0, 1 and x can now be unified under ψ for all four operators as shown in Table XI:

ψ	0	1	x
"a" cube	i	j	{X}
"b" cube	i	j	e

TABLE XI

UNIFIED IMAGES UNDER ψ

4. ψ^{-1} is invariant to the choice of operators for the co-ordinates of the $a * b$ cube in U domain having i's, j's and $\emptyset$'s.

<u>U domain</u>		<u>Q domain</u>
i	$\longrightarrow$	0
j	$\longrightarrow$	1
$\emptyset$	$\longrightarrow$	$\emptyset$

(Note that in Star Product Operator, one $\emptyset$ is allowed.)

5. ψ^{-1} is operator dependent for certain co-ordinates of the $a * b$ cube when it is mapping back to x in Q domain. This dependence is shown in Table XII.

$\sqsubset$	*	$\sqcap$	#
k	e or 1st $\emptyset$	e	K.

TABLE XII - OPERATOR DEPENDENCE OF y_i^c

Let us define a subset Y of the set U to represent the operator dependence of ψ^{-1} as follows:

$$Y = \{k, e, \emptyset, K\}$$

The images of i, j, $\emptyset$, and $\{Y\}$ can now be unified under ψ^{-1} for all four operators as shown in Table XIII.

ψ^{-1}	i	j	$\{Y\}$	$\emptyset$
a * b cube	0	1	x	$\emptyset$

TABLE XIII

UNIFIED IMAGES UNDER ψ^{-1}

3.8 APPLICATION OF THE UNIFIED THEORY

The following examples will illustrate the application of the unified theory to the solution of problems involving a pair of cubes. Let us define each operator by assigning to it an element of the set $X = \{k, e, K\}$. Let the assigned element be denoted by X_a .

3.8.1 Subsume Operator ($\sqsubseteq$)

Definition: $X_a = k$

Operation: $a \sqsubseteq b$ if $a_i \sqsubseteq b_i = a_i$ for all i

$a \not\sqsubseteq b$ if $a_i \sqsubseteq b_i = \emptyset$ for any i

Purpose: Absorption of a cube array by deletion of those that subsume one or more cubes of the array.

Examples:

$$1.) \quad a = \{1, x, 0, 1, x\} \longrightarrow \{j, k, i, j, k\}$$

$$b = \{1, 0, x, 1, 1\} \longrightarrow \{j, i, e, j, j\}$$

* _____

$$\text{Therefore } a \not\sqsubseteq b \quad \{j, \emptyset, i, j, \emptyset\}$$

$$2.) \quad a = \{1, 0, 0, 1, 1\} \longrightarrow \{j, i, i, j, j\}$$

$$b = \{1, 0, x, 1, 1\} \longrightarrow \{j, i, e, j, j\}$$

* _____

$$c = \{1, 0, 0, 1, 1\} \longleftarrow \{j, i, i, j, j\}$$

Therefore $a \sqsubseteq b$. Moreover $c = a$, indicating that, "a" should be deleted in the absorption process.

3.8.2 Cube Intersection ($\cap$)

Definition: $X_a = e$

Operation: $a \cap b$ if $a_i \cap b_i = \emptyset$ for any i

Purpose: To find the largest single cube "c" common to the cubes "a" and "b".

Examples:

$$\begin{array}{lcl}
 1.) & a = \{1, 0, x\} & \longrightarrow \{j, i, e\} \\
 & b = \{1, 1, 1\} & \longrightarrow \{j, j, j\} \\
 & & * \hline
 & \text{Therefore } a \cap b & \{j, \emptyset, j\}
 \end{array}$$

$$\begin{array}{lcl}
 2.) & a = \{1, 1, x\} & \longrightarrow \{j, j, e\} \\
 & b = \{1, x, 0\} & \longrightarrow \{j, e, i\} \\
 & & * \hline
 & c = \{1, 1, 0\} & \longleftarrow \{j, j, i\}
 \end{array}$$

"c" is the largest single cube common to the cubes "a" and "b"

3.8.3 Star Product (*)

Definition: $X_a = e$

Operation: The same as in the cube intersection except one $\emptyset$ is allowed and the empty literal is replaced by an "x" in Q domain.

Purpose: To find the "consensus" cube which is the largest cube between the cubes "a" and "b".

Example:

$$a = \{1, 0, x\} \longrightarrow \{j, i, e\}$$

$$b = \{x, 1, 1\} \longrightarrow \{e, j, j\}$$

*

$$c = \{1, x, 1\} \longleftarrow \{j, \emptyset, j\}$$

3.8.4 Sharp Operator (#)

Definition: $X_a = K$

Operation: $a \# b = a$ if $a_i \# b_i = \emptyset$ for any i , i.e. $a \not\sqsubseteq b$.

$a \# b = \emptyset$ if $a \sqsubseteq b$

Purpose: The sharp operator is analogous to the "relative complement" of the set theory. In other words if

$$f_3 = f_1 \cdot \bar{f}_2$$

then the function f_3 is the complement of the function f_2 relative to the function f_1 . Hence f_3 can be thought as the remaining vertices of f_1 , when the vertices of f_2 are removed from f_1 . Therefore the outcome of the binary operation $a \# b$ can be more than one "c" cube. If, in U domain, $y_i^a = K$ while $y_i^b = i$ or $y_i^b = j$, then $y_i^a \# y_i^b$ generates say q different cubes if $K * i$ or $K * j$ occurs q times. For example:

1st cube		$K * i$		$K * e$		$K * e$	
2nd cube		$K * e$		$K * j$		$K * e$	
3rd cube		$K * e$		$K * e$		$K * j$	

$$1.) \quad a = \{1, 0, x\} \longrightarrow \{j, i, K\}$$

$$b = \{1, 1, 1\} \longrightarrow \{j, j, j\}$$

*

$$a \# b = a \text{ since } a \not\sqsubseteq b \quad \{j, \emptyset, i\}$$

$$2.) \quad a = \{1, 1, 0\} \longrightarrow \{j, j, i\}$$

$$b = \{1, 1, x\} \longrightarrow \{j, j, e\}$$

*

$$c = a = \{1, 1, 0\} \longleftarrow \{j, j, i\}$$

$a \# b = \emptyset$ since $a \sqsubseteq b$.

$$\begin{aligned}
 3.) \quad a &= \{1, 1, x\} \longrightarrow \{j, j, K\} \\
 b &= \{1, 1, 0\} \longrightarrow \{j, j, i\} \\
 &\quad \quad \quad * \underline{\hspace{1cm}} \\
 c &= \{1, 1, 1\} \longleftarrow \{j, j, j\}
 \end{aligned}$$

When the vertex 110 is removed from the edge 11x, only the vertex 111 is left.

$$\begin{aligned}
 4.) \quad a &= \{x, 1, x\} \longrightarrow \{K, j, K\} & \{K, j, K\} \\
 b &= \{0, 1, 0\} \longrightarrow \{i, j, e\} & \{e, j, i\} \\
 &\quad \quad \quad * \underline{\hspace{1cm}} & \underline{\hspace{1cm}} \\
 c_1 &= \{1, 1, x\} \longleftarrow \{j, j, K\} & \{K, j, j\} \\
 c_2 &= \{x, 1, 1\} \longleftarrow \{K, j, j\}
 \end{aligned}$$

When the vertex 010 is removed from the face x1x, only the edges x11 and 11x are left.

The same results can be obtained by using the partitioning method as follows:

$$\begin{aligned}
 a &= \{x, 1, x\} \longrightarrow \{K, j, K\} \\
 b &= \{0, 1, 0\} \longrightarrow \{i, j, i\} \\
 &\quad \quad \quad * \underline{\hspace{1cm}} \\
 c_1 &= \{1, 1, x\} \longleftarrow \{j/, j, j/\} \\
 c_2 &= \{x, 1, 1\} \longleftarrow \{j/, j, j/\}
 \end{aligned}$$

In this particular case, using the partitioning method one binary operation has been eliminated. Similarly $n-1$ binary operations would have been eliminated if there had been " n " $K * i$ or $K * j$ confrontations.

3.9 UNIFIED THEORY AND THE CUBICAL ARRAYS

In solving combinatorial logic problems, quite often a few of the basic operators of the algebraic topological methods are combined together in a broader algorithm and applied to two arrays of cubes, say A array with the cubes $\{a^1, a^2, \dots a^n\}$ and B array with the cubes $\{b^1, b^2, \dots b^m\}$.

In operations involving cubical arrays when the co-ordinates are mapped from Q domain to U domain, it becomes very easy to switch from one operator to another by simply replacing X_a with another element of the set $X = \{k, e, K\}$.

The following are some of the basic operations carried out on cubical arrays:

3.9.1 ABSORB (Σ)

When the array A is absorbed by deletion of those cubes that subsume one or more cubes of A, the resulting B array is given by the equation.

$$B = \Sigma A \quad \text{where } \Sigma = \text{Absorption Operator.}$$

In this case, each and every cube is subjected to the subsume operation with all other cubes of the array. For example, if we consider two cubes, say a^p and a^q , we have

$$a^p \sqsubseteq a^q \quad \text{with mappings } x^p . \psi = k \quad \text{and} \quad x^q . \psi = e$$

and

$$a^q \sqsubseteq a^p \quad \text{with mappings } x^q . \psi = k \quad \text{and} \quad x^p . \psi = e$$

where $x \in Q = \{0, 1, x\}$ and $k, e \in X = \{k, e, K\}$.

Hence, as we reverse the order of the subsume operation, the images of the x's of the cubes a^p and a^q in U domain are also reversed.

On the other hand, the binary operation involving the images of 0 and 1 in U domain, namely i and j, are commutative. In other words, if a " $\emptyset$ " is obtained as a result of $(i * j)$ in $a^p \sqsubseteq a^q$, there is no reason to check $a^q \sqsubseteq a^p$. The latter will also give a " $\emptyset$ " or "does not subsume" answer.

3.9.2 CUBE UNION (LD)

The cube union of two arrays A and B is the absorbed set {AUB}. Namely:

$$AUB = \Sigma \{AUB\} = \Sigma \{a^1, a^2, \dots a^n, b^1, b^2, \dots b^m\}.$$

The necessary subsume operations can be shown in the form of an $(n+m) \times (n+m)$ matrix as follows:

$$\begin{array}{cccc} & & & \text{where} \\ \begin{array}{c} c_{11} \quad c_{12} \quad \dots \quad c_{1(n+m)} \\ c_{21} \quad c_{22} \quad \dots \quad c_{2(n+m)} \\ \vdots \\ c_{(n+m)1} \quad c_{(n+m)2} \dots c_{(n+m)(n+m)} \end{array} & & \begin{array}{l} c_{pq} = a^p \sqsubseteq a^q \text{ if } 1 \leq (p, q) \leq n \\ c_{pq} = b^{p-n} \sqsubseteq b^{q-n} \text{ if } n+1 \leq (p, q) \leq n+m \\ c_{pq} = a^p \sqsubseteq b^{q-n} \text{ if } 1 \leq p \leq n \text{ and } \\ n+1 \leq q \leq n+m \end{array} \end{array}$$

Elements of this matrix which are not equal to $\emptyset$ will be deleted from the set {AUB}. In other words, $c_{pq} \neq \emptyset$ implies $c_{pq} = a^p$ if $1 \leq p \leq n$ or $c_{pq} = b^{p-n}$ if $n+1 \leq p \leq n+m$.

Example:

Let $A = \{0\ 0\ 0, x\ 1\ 1\}$ and $B = \{0\ x\ 0, x\ 1\ x\}$

Now:

$$\begin{array}{lcl} a^1 = 0\ 0\ 0 & \longrightarrow & i\ i\ i \\ a^2 = x\ 1\ 1 & \longrightarrow & e\ j\ j \\ & * & \hline & & i\ \emptyset\ \emptyset \end{array}$$

Therefore

$$\begin{array}{l} a^1 \not\sqsubseteq a^2 \text{ and } a^2 \not\sqsubseteq a^1 \\ \text{since } \emptyset = i * j \end{array}$$

$$\begin{array}{lcl} b^1 = 0\ x\ 0 & \longrightarrow & i\ k\ i \\ b^2 = x\ 1\ x & \longrightarrow & e\ j\ e \\ & * & \hline & & i\ \emptyset\ i \end{array}$$

Therefore

$$\begin{array}{l} b^1 \not\sqsubseteq b^2 \\ \text{Also } b^2 \not\sqsubseteq b^1 \text{ since the replacement} \\ \text{of } e\text{'s in } b^2 \text{ by } k\text{'s will result in } \emptyset\text{'s.} \end{array}$$

$$\begin{array}{lcl} a^1 = 0\ 0\ 0 & \longrightarrow & i\ i\ i \\ b^1 = 0\ x\ 0 & \longrightarrow & i\ e\ i \\ & * & \hline a^1 = 0\ 0\ 0 & \longleftarrow & i\ i\ i \\ a^1 \sqsubseteq b^1 \text{ therefore delete } a^1 \\ \text{Also } b^1 \not\sqsubseteq a^1 \end{array}$$

$$\begin{array}{lcl} a^2 = x\ 1\ 1 & \longrightarrow & k\ j\ j \\ b^2 = x\ 1\ x & \longrightarrow & e\ j\ e \\ & * & \hline a^2 = x\ 1\ 1 & \longleftarrow & k\ j\ j \\ a^2 \sqsubseteq b^2 \text{ therefore delete } a^2 \\ \text{Also } b^2 \not\sqsubseteq a^2 \end{array}$$

$$\begin{array}{lcl} a^1 = 0\ 0\ 0 & \longrightarrow & i\ i\ i \\ b^2 = x\ 1\ x & \longrightarrow & e\ j\ e \\ & * & \hline & & i\ \emptyset\ i \end{array}$$

$$\begin{array}{l} \text{Therefore } a^1 \not\sqsubseteq b^2 \\ \text{and } b^2 \not\sqsubseteq a^1 \text{ since } \emptyset = i * j \end{array}$$

$$\begin{array}{lcl} a^2 = x\ 1\ 1 & = & k\ j\ j \\ b^1 = 0\ x\ 0 & = & i\ e\ i \\ & * & \hline & & \emptyset\ j\ \emptyset \end{array}$$

$$\begin{array}{l} \text{Therefore } a^2 \not\sqsubseteq b^1 \\ \text{and } b^1 \not\sqsubseteq a^2 \text{ since } \emptyset = j * i \end{array}$$

The results are shown in Table XIV.

Σ	a^1	a^2	b^1	b^2
a^1	$\emptyset$	$\emptyset$	a^1	$\emptyset$
a^2	$\emptyset$	$\emptyset$	$\emptyset$	a^2
b^1	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$
b^2	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$

TABLE XIV

CUBE UNION OF ARRAYS

Hence,

$$A \sqcup B = B = \{0 \times 0, x \ 1 \ x\}$$

3.9.3 CUBE INTERSECTION ($\cap$)

The intersection of a cube b^q and an array $A = \{a^1, a^2, \dots a^n\}$ is defined as:

$$A \cap b^q = \{(a^1 \cap b^q) \cup (a^2 \cap b^q) \cup \dots (a^n \cap b^q)\}$$

Then the intersection of array A with the array $B = \{b^1, b^2, \dots b^m\}$ is defined as

$$A \cap B = \sum_{q=1}^m \sum_{p=1}^n a^p \cap b^q \quad \text{where } \Sigma \text{ is the absorption operator}$$

The necessary intersection operations can be shown in an $n \times m$ matrix as follows:

$$\begin{pmatrix} c_{11} & c_{12} & \dots & c_{1m} \\ c_{21} & c_{22} & \dots & c_{2m} \\ \vdots & \vdots & & \vdots \\ c_{n1} & c_{n2} & \dots & c_{nm} \end{pmatrix}$$

$$\text{where } c_{pq} = a^p \cap b^q$$

$$\text{and } 1 \leq p \leq n$$

$$1 \leq q \leq m$$

If the cube arrays A and B are considered as vectors with the components $\{a^1, a^2, \dots a^n\}$ and $\{b^1, b^2, \dots b^m\}$ respectively, the above matrix can also be written as the "Dyadic Product" or "Open Product" of two vectors as in tensor analysis. Namely:

$$A B = \begin{pmatrix} a^1 b^1 & a^1 b^2 & \dots & a^1 b^m \\ a^2 b^1 & a^2 b^2 & \dots & a^2 b^m \\ \vdots & \vdots & & \vdots \\ a^n b^1 & a^n b^2 & \dots & a^n b^m \end{pmatrix}$$

Some elements of this $n \times m$ matrix may be equal to " $\emptyset$ " indicating "no intersection". The absorption operator is then applied to the remaining terms of the matrix by switching X_a from e to k or e as required.

Example:

Let $A = (x \ 1 \ x, \ 0 \ 0 \ 0)$ and $B = (0 \ x \ 0, \ x \ 1 \ 1)$

$$\begin{array}{lcl} a^1 & = x \ 1 \ x & \rightarrow e \ j \ e \\ b^1 & = 0 \ x \ 0 & \rightarrow i \ e \ i \\ & * & \text{-----} \\ c_{11} & = 0 \ 1 \ 0 & \leftarrow i \ j \ i \end{array}$$

$$\begin{array}{lcl} a^1 & = x \ 1 \ x & \rightarrow e \ j \ e \\ b^2 & = x \ 1 \ 1 & \rightarrow e \ j \ j \\ & * & \text{-----} \\ c_{12} & = x \ 1 \ 1 & \leftarrow e \ j \ j \end{array}$$

$$\begin{array}{lcl} a^2 & = 0 \ 0 \ 0 & \rightarrow i \ i \ i \\ b^1 & = 0 \ x \ 0 & \rightarrow i \ e \ i \\ & * & \text{-----} \\ c_{21} & = 0 \ 0 \ 0 & \leftarrow i \ i \ i \end{array}$$

$$\begin{array}{lcl} a^2 & = 0 \ 0 \ 0 & \rightarrow i \ i \ i \\ b^2 & = x \ 1 \ 1 & \rightarrow e \ j \ j \\ & * & \text{-----} \\ c_{22} & = \emptyset & \leftarrow i \ \emptyset \ \emptyset \end{array}$$

Hence:

$$A \sqcap B = \Sigma \begin{pmatrix} c_{11} & c_{12} \\ c_{21} & \emptyset \end{pmatrix}$$

X_a will now be switched from e to k or e to apply the subsume operator.

$$c_{11} \rightarrow i j i$$

$$c_{12} \rightarrow e j j$$

$$* \frac{\quad}{i j \emptyset}$$

Therefore $c_{11} \not\sqsubseteq c_{12}$

and since $\emptyset = i * j$

$$c_{12} \not\sqsubseteq c_{11}$$

$$c_{11} \rightarrow i j i$$

$$c_{21} \rightarrow i i i$$

$$* \frac{\quad}{i \emptyset i}$$

Therefore $c_{11} \not\sqsubseteq c_{21}$

and since $\emptyset = j * i$

$$c_{21} \not\sqsubseteq c_{11}$$

$$c_{12} \rightarrow k j j$$

$$c_{21} \rightarrow i i i$$

$$* \frac{\quad}{\emptyset \emptyset \emptyset}$$

Therefore $c_{12} \not\sqsubseteq c_{21}$

and since $\emptyset = j * i$

$$c_{21} \not\sqsubseteq c_{12}$$

Therefore, there is no absorption and

$$A \sqcap B = (0 0 0, 0 1 0, x 1 1)$$

3.9.4 SHARP PRODUCT (#)

The sharp product of an array $A = \{a^1, a^2, \dots, a^n\}$ and a cube b^q is defined as:

$$A \# b^q = \{(a^1 \# b^q) \sqcup (a^2 \# b^q) \sqcup \dots (a^n \# b^q)\}$$

Similarly the sharp product of a cube a^p with an array $B = \{b^1, b^2, \dots, b^m\}$ is defined as:

$$a^p \# B = \{\dots [(a^p \# b^1) \# b^2] \# \dots b^m\}$$

Hence, the sharp product of two arrays is defined as

$$A \# B = \sum_{p=1}^n \prod_{q=1}^m a^p \# b^q$$

where Σ represents the absorption operator.

After the sharp product is taken, the absorption operator is applied by switching X_a from K to k or e .

Example:

Let $A = (1 \ x \ x \ x, \ x \ 0 \ 1 \ x)$ and $B = (1 \ 0 \ 1 \ x, \ 1 \ 1 \ 1 \ 1)$

Now

$$A \# B = \sum_{p=1}^2 \prod_{q=1}^2 a_p b_q = [(a^1 \# b^1) \# b^2] \sqcup [(a^2 \# b^1) \# b^2]$$

$$a^1 = \{1, x, x, x\} \longrightarrow \{j, K, K, K\}$$

$$b^1 = \{1, 0, 1, x\} \longrightarrow \underbrace{\{j, i, j, e\}}_*$$

$$c^1 = \{1, 1, x, x\} \longleftarrow \{j, j/, i/, K\}$$

$$c^2 = \{1, x, 0, x\} \longleftarrow$$

$$c^1 = \{1, 1, x, x\} \longrightarrow \{j, j, K, K\}$$

$$b^2 = \{1, 1, 1, 1\} \longrightarrow \underbrace{*\{j, j, j, j\}}$$

$$d^1 = \{1, 1, 0, x\} \longleftarrow \{j, j, i/, i/\}$$

$$d^2 = \{1, 1, x, 0\} \longleftarrow$$

$$c^2 = \{1, x, 0, x\} \longrightarrow \{j, K, i, K\}$$

$$b^2 = \{1, 1, 1, 1\} \longrightarrow \underbrace{*\{j, j, j, j\}}$$

$$c^2 \sqcap b^2 \longleftarrow \{j, i/, \emptyset, i/\}$$

$$\therefore d^3 = c^2 = \{1, x, 0, x\}$$

$$\text{Therefore } [(a^1 \# b^1) \# b^2] = \{1, 1, 0, x, \quad 1, 1, x, 0, \quad 1, x, 0, x\}$$

Similarly

$$a^2 = x, 0, 1, x \longrightarrow K, i, j, K$$

$$b^1 = 1, 0, 1, x \longrightarrow \underbrace{*\{j, i, j, e\}}$$

$$c^3 = 0, 0, 1, x \longleftarrow i/, i, j, K$$

$$c^3 \longrightarrow i, i, j, K$$

$$b^2 \longrightarrow \underbrace{*\{j, j, j, j\}}$$

$$\emptyset, \emptyset, j, i/$$

$$\text{Since } c^3 \sqcap b^2 \quad d^4 = c^3 = 0, 0, 1, x$$

$$\text{Therefore } [(a_2 \# b_1) \# b_2] = 0, 0, 1, x \quad \text{and}$$

$$A \# B = (1, 1, 0, x, \quad 1, 1, x, 0, \quad 1, x, 0, x) \sqcup (0, 0, 1, x)$$

X_a will now be switched from K to k or e to apply the subsume operator.

$$\begin{array}{lcl} d^1 = 110x & \longrightarrow & jjik \\ d^2 = 11x0 & \longrightarrow & jjei \\ & * & \hline & & jjio \end{array}$$

Therefore

$d^1 \not\sqsubseteq d^2$ also $d^2 \not\sqsubseteq d^1$ since replacement of e in d^2 by k will result in a $\emptyset$.

$$\begin{array}{lcl} d^1 = 110x & \longrightarrow & jjik \\ d^3 = 1x0x & \longrightarrow & jeei \\ & * & \hline d^1 = 110x & \longleftarrow & jjik \end{array}$$

Therefore

$d^1 \sqsubseteq d^3$ and d^1 will be deleted.
Also $d^3 \not\sqsubseteq d^1$

$$\begin{array}{lcl} d^1 = 110x & \longrightarrow & jjik \\ d^4 = 001x & \longrightarrow & iije \\ & * & \hline & & \emptyset\emptyset\emptyset k \end{array}$$

Therefore

$d^1 \not\sqsubseteq d^4$ and $d^4 \not\sqsubseteq d^1$
since $\emptyset = i * j$

$$\begin{array}{lcl} d^2 = 11x0 & \longrightarrow & jjki \\ d^3 = 1x0x & \longrightarrow & jeei \\ & * & \hline & & jj\emptyset i \end{array}$$

Therefore

$d^2 \not\sqsubseteq d^3$. Also $d^3 \not\sqsubseteq d^2$ since replacement of e's in d^3 by k's will result in $\emptyset$'s.

$$\begin{array}{lcl} d^2 = 11x0 & \longrightarrow & jjki \\ d^4 = 001x & \longrightarrow & iije \\ & * & \hline & & \emptyset\emptyset\emptyset i \end{array}$$

Therefore $d^2 \not\sqsubseteq d^4$ and $d^4 \not\sqsubseteq d^2$ since $\emptyset = i * j$

$$d^3 = 1 \times 0 \times \rightarrow j \ k \ i \ k$$

$$d^4 = 0 \ 0 \ 1 \ x \rightarrow i \ i \ j \ e$$

$$* \text{ --- }$$

$$\emptyset \ \emptyset \ \emptyset \ k$$

Therefore $d^3 \not\subseteq d^4$ and $d^4 \not\subseteq d^3$ since $\emptyset = i * j$

Hence deleting d^1 which subsumes d^3 we have

$$A \# B = \{1 \ 1 \ x \ 0, 1 \ x \ 0 \ x, 0 \ 0 \ 1 \ x\}$$

CHAPTER IV
RING STRUCTURE

4.1 ANOTHER APPROACH TO THE UNIFIED THEORY

Further investigation of the algorithms used in the topological methods reveals certain redundancies in subsume and sharp operations. Elimination of these redundancies not only results in more efficient computer programmes, but also opens the way towards a complex but more elegant theory using the Boolean rings.

The advantages and disadvantages of this new approach have been discussed in detail in the proceeding section. Naturally there is a trade-off between the simplicity of the theory and the efficiency of the software. However, the most important point about the new approach is that it ties the unified theory to the Boolean algebra and Boolean rings. Furthermore, it makes it possible to apply the highly developed algebraic theory of rings to the study of topological methods.

The application of the ring theory, however, takes place in "B space". In this space a single variable x of a combinatorial logic function is represented by $Z_2 \times Z_2$ group, which is a subset of the two dimensional Enclidian space R^2 .

Similarly " n " variables $(x_1, x_2 \dots x_n)$ are represented by $(Z_2 \times Z_2)^n$, which can be considered a subset of " $2n$ " dimensional Enclidian space R^{2n} .

The Boolean algebra associated with this ring structure is a four state Boolean algebra. (See Appendix F).

4.2 SIMPLIFICATION OF SUBSUME ALGORITHM

The subsume algorithm was given in Table VI on page 22. The second mapping function was also defined on this page for this operator.

When the subsume operator is applied to a pair of cubes, the outcome is always a "yes" or "no" answer, depending whether or not the binary operation encounters one " $\emptyset$ ".

Let us consider two cubes, say a^P and a^Q . Although the binary operation applied to the co-ordinates of these cubes is commutative, as the order of the subsume operation is reversed, the images of the x's of the a^P and a^Q cubes in U domain are reversed.

For example:

$a^P \sqsubseteq a^Q$ with mappings $x^P.\psi = k$ and $x^Q.\psi = e$

and

$a^Q \sqsubseteq a^P$ with mappings $x^Q.\psi = k$ and $x^P.\psi = e$

where $x \in Q = \{0, 1, x\}$ and $k, e \in X = \{k, e, K\}$.

Let us look at the various encounters likely to occur in a subsume operation between the co-ordinates of a^P and a^Q cubes. These are as follows:

a^P cube		i	i	i	j	j	j	k	k	k
a^Q cube	*	i	j	e	j	i	e	i	j	e
$a^P * a^Q$ cube		i	$\emptyset$	i	j	$\emptyset$	j	$\emptyset$	$\emptyset$	k

It is clear that first and second $\emptyset$ resulted from $i * j$ or $j * i$ confrontations will not change, if the order of the subsume operation is changed. On the other hand, third and fourth $\emptyset$ resulted from $k * i$ and $k * j$ will not occur, if the order of the operation is changed. These binary operations will become $i * e = i$ and $j * e = j$, when the order is reversed.

Furthermore, $i * e = i$ and $j * e = j$ will produce $\emptyset$'s if the order of operation is changed, since $k * i = \emptyset$ and $k * j = \emptyset$. Finally, $i * i = i$, $j * j = j$, and $k * e = k$ will not change as the order of the operation is changed.

The binary operations can now be grouped as follows:

a^P cube	i	j	k	{	invariant to the order of operation
a^Q cube	*	i	j		
$a^P * a^Q$ cube		i	j		
a^P cube		i	j	{	will produce $\emptyset$'s if the order is reversed
a^Q cube	*	e	e		
$a^P * a^Q$ cube		i	j		
a^P cube		i	j	{	invariant to the order
a^Q cube	*	j	i		
$a^P * a^Q$ cube		$\emptyset$	$\emptyset$		
a^P cube		k	k	{	will not produce $\emptyset$'s if the order is reversed.
a^Q cube	*	i	j		
$a^P * a^Q$ cube		$\emptyset$	$\emptyset$		

From this, the following conclusions can be drawn:

- (1) If a $\emptyset$ occurs as a result of $i * j$ or $j * i$,
 $a^p \not\sqsubseteq a^q$ and $a^q \not\sqsubseteq a^p$
- (2) $i * e, j * e$ and $k * i, k * j$ are not invariant to the order of the operator.

Now, if the second mapping function ψ is changed to ξ so that

$$x^p. \xi = e \text{ and } x^q. \xi = e$$

the binary operation groupings shown above will be as follows:

a^p cube	$i \quad j \quad e$	{	identical to the co-ordinates of both a^p and a^q and invariant to the order of operation.
a^q cube	$* \quad i \quad j \quad e$		
$a^p * a^q$ cube	$i \quad j \quad e$		

a^p cube	$i \quad j$	{	identical to the co-ordinates of a^p or a^q depending on the order of operation
a^q cube	$* \quad e \quad e$		
$a^p * a^q$ cube	$i \quad j$		

a^p cube	$i \quad j$	{	invariant to the order of operation. It will always produce $\emptyset$'s.
a^q cube	$* \quad j \quad i$		
$a^p * a^q$ cube	$\emptyset \quad \emptyset$		

a^p cube	$e \quad e$	{	identical to the co-ordinates of a^q or a^p depending on the order of operation
a^q cube	$* \quad i \quad j$		
$a^p * a^q$ cube	$i \quad j$		

It is clear that if ψ is used, $i * e$, $j * e$ and $k * i$, $k * j$ may or may not produce $\emptyset$'s depending on the order of operation. On the other hand, if ξ is used, $i * e$, $j * e$ and $e * i$, $e * j$ produce identical co-ordinates to a^p or a^q cube depending on the order of the operation.

Hence, by using the criteria of $a^p * a^q$ being equal to a^p or a^q , the same "yes" or "no" answer can be produced by the new algorithm, while the set $S = \{0, 1, i, j, k\}$ is reduced to its subset $B = \{0, 1, i, j\}$.

Examples:

$$\begin{aligned}
 (1) \quad a^p &= \{1, x, 0, 1, x\} \longrightarrow \{j, e, i, j, e\} \\
 a^q &= \{1, 0, x, 1, 1\} \longrightarrow \underline{* \{j, i, e, j, j\}} \\
 &\qquad\qquad\qquad j \quad i \quad i \quad j \quad j
 \end{aligned}$$

Since $a^p * a^q \neq a^q \neq a^p$

$a^p \not\subseteq a^q$ and $a^q \not\subseteq a^p$

$$\begin{aligned}
 (2) \quad a^p &= \{1, 0, 0, 1, 1\} \longrightarrow \{j, i, i, j, j\} \\
 a^q &= \{1, 0, x, 1, 1\} \longrightarrow \underline{* \{j, i, e, j, j\}} \\
 &\qquad\qquad\qquad j, i, i, j, j
 \end{aligned}$$

Since $a^p * a^q = a^p$

$a^p \subseteq a^q$

$$\begin{array}{llll}
 (3) & a^p = \{1, 0, x\} & \longrightarrow & \{j, i, e\} \\
 & a^q = \{1, 0, 1\} & \longrightarrow & \underbrace{* \{j, i, j\}}_{j \quad i, \quad j}
 \end{array}$$

$$\text{Since } a^p * a^q = a^q$$

$$a^q \subseteq a^p$$

These examples illustrate the elimination of a redundancy in the subsume operator, namely the existence of k , by the use of the second mapping function ξ , which will be further discussed in section 4.6. This simplified version of the subsume algorithm will be called double-ended subsume operation.

4.3 SIMPLIFICATION OF SHARP ALGORITHM

The sharp algorithm was given in Table IX on page 23. The second mapping function was also defined on that page for the sharp operator.

The sharp algorithm is the most complicated of all four algorithms which are used as the basic building blocks in the calculus of cubical complexes.

Furthermore, the sharp operator includes the subsume and intersection operations within its own algorithm. For example, considering "a" and "b" cubes, we have:

$$\begin{aligned} a \# b &= \emptyset \quad \text{if } a \not\subseteq b \\ a \# b &= a \quad \text{if } a \sqsubset b \end{aligned}$$

For this reason, it will be assumed that the sharp operator is always preceded by subsume and intersection operators. In other words, the sharp algorithm will be defined as a specific part of the total algorithm given in Table IX on page 23, excluding the subsume and intersection algorithms.

Moreover, the algorithm given in Table IX may only be applicable when the dimension, i.e. the cardinality of the subset $x.\rho^{-1}$ of N , of the "a" or upper cube is not higher than one. If the dimension of the "a" cube is higher than one, then $a \# b$ may generate as many cubes as there are $y_i^a * y_i^b$ operations in U domain, where $y_i^a = K$ and $y_i^b = i$ or $y_i^b = j$. This was pointed out on page 20 of section 3.6. The expansion of $a \# b$ into an array of cubes will be expressed in a mathematical form in this section.

Let us look at the various binary operations, which are likely to occur in the execution of sharp algorithm between the co-ordinates of "a" and "b" cubes in U domain. These are as follows:

"a" cube	i	i	j	j	K	K or K
"b" cube	* i	e	j	e	e	i or j
"a * b" cube	i	i	j	j	K	j or i

It has been assumed that the binary operations $i * j$ or $j * i$ have been taken care of by either subsume or intersection operators which preceded the sharp operator.

As can be seen above, the binary operations involving the co-ordinates of "a" and "b" cubes have been divided up into two groups. The first group contains five different types of encounters. These binary operations result in the co-ordinates of "a * b" cube being identical to the co-ordinates of "a" cube. Because of this, since "e" is not one of the co-ordinates of the "a" cube, it cannot be one of the co-ordinates of a * b cube. This is an important result and will be used later.

The second group contains $K * i = j$ or $K * j = i$. If there was more than one of these binary operations, $a \# b$ will expand into an array of cubes.

Another important observation is that neither "k" nor "Ø" appears in the possible encounters shown above. Therefore it can be concluded that both "k" and "Ø" are redundant elements in the sharp algorithm.

In listing the possible encounters in sharp algorithm and producing the co-ordinates of $a * b$ cube in U domain, the binary operation of the "Hybrid" groupoid of order six was used. Now let us use the "exclusive-or" operation on the individual elements of the quartal coding with the following assignments:

$$\begin{aligned} i &= 01 = 1_4 \\ j &= 10 = 2_4 \\ K &= 11 = 3_4 \\ e &= 00 = 0_4 \end{aligned}$$

Then the following are the possible encounters:

"a" cube	i	i	j	j	K		K or K
"b" cube	$\oplus$	i	e	j	e	e	i or j
"a * b" cube		e	i	e	j	K	j or i

since

$$\begin{array}{cccc} i = 01 & i = 01 & j = 10 & j = 10 \\ \oplus i = \oplus 01 & \oplus e = \oplus 00 & \oplus j = \oplus 10 & \oplus e = \oplus 00 \\ \hline e = 00 & i = 01 & e = 00 & j = 10 \end{array}$$

$$\begin{array}{ccc} K = 11 & K = 11 & K = 11 \\ \oplus e = \oplus 00 & \oplus i = \oplus 01 & \oplus j = \oplus 10 \\ \hline K = 11 & j = 10 & i = 01 \end{array}$$

As can be seen, there is no change in the second group involving $K * i = j$ and $K * j = i$. The first group, however, looks different. In this group, the co-ordinates of $a * b$ cube is supposed to be identical to the co-ordinates of "a" cube. Because of this, when using "exclusive-or" operation, the identity element, "e", will be interpreted as representing the corresponding co-ordinate of "a" cube when it appears as a co-ordinate of " $a * b$ " cube. There is no ambiguity in this representation, since "e" is a redundant element both in "a" and " $a * b$ " cubes. This was pointed out as an important result earlier in this section.

Now let us consider what happens to the second group if K appears, say n times. Let the "a" and "b" cubes each have N co-ordinates. Let us form two $N \times N$ matrices and designate them $\underline{A}$ and $\underline{B}$. Let $\underline{A}$ have identical rows which consist of the co-ordinates of the "a" cube. Similarly, let $\underline{B}$ have identical rows which consist of the co-ordinates of the "b" cube. Hence, we have:

$$\underline{A} = \begin{pmatrix} a_1 & a_2 & a_3 & \dots & a_N \\ a_1 & a_2 & a_3 & \dots & a_N \\ \vdots & \vdots & \vdots & & \vdots \\ a_1 & a_2 & a_3 & \dots & a_N \end{pmatrix}$$

$$\underline{B} = \begin{pmatrix} b_1 & b_2 & b_3 & \dots & b_N \\ b_1 & b_2 & b_3 & \dots & b_N \\ \vdots & \vdots & \vdots & & \vdots \\ b_1 & b_2 & b_3 & \dots & b_N \end{pmatrix}$$

Let us now form a third matrix, $\underline{C}$, which has the complement operator $C = K \oplus$ placed on its diagonal. Let all other elements of $\underline{C}$ be equal to zero. Hence:

$$\underline{C} = \begin{pmatrix} C & 0 & 0 & \dots & 0 \\ 0 & C & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & & \vdots \\ 0 & 0 & 0 & \dots & C \end{pmatrix}$$

Now if there are n positions (columns), where $K * i$ or $K * j$ confrontations occur, then $\underline{C}$ matrix has only n C 's placed in those particular columns on its diagonal. If every element of the diagonal was equal to C , the following results would be obtained:

$$\underline{A} \cdot \underline{C} = \begin{pmatrix} 0 & a_2 & a_3 & \dots & a_N \\ a_1 & 0 & a_3 & \dots & a_N \\ \vdots & \vdots & \vdots & & \vdots \\ a_1 & a_2 & a_3 & \dots & 0 \end{pmatrix} \quad \begin{array}{l} \text{since } K = 1 \ 1 \\ \oplus \ K = 1 \ 1 \\ \hline \emptyset = 0 \ 0 \end{array}$$

and

$$\underline{B} \cdot \underline{C} = \begin{pmatrix} \bar{b}_1 & b_2 & b_3 & \dots & b_N \\ b_1 & \bar{b}_2 & b_3 & \dots & b_N \\ \vdots & \vdots & \vdots & & \vdots \\ b_1 & b_2 & b_3 & \dots & \bar{b}_N \end{pmatrix} \quad \begin{array}{l} \text{since} \\ K \oplus i = j = \bar{i} \\ \text{and} \\ K \oplus j = i = \bar{j} \end{array}$$

Now if all the elements of $\underline{B} \cdot \underline{C}$ matrix, except the one's on its diagonal, are made equal to zero, we have:

$$(\underline{B} \cdot \underline{C})^* = \begin{pmatrix} \bar{b}_1 & 0 & 0 & \dots & 0 \\ 0 & \bar{b}_2 & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & & \vdots \\ 0 & 0 & 0 & \dots & \bar{b}_N \end{pmatrix}$$

Using matrix addition, a new expansion matrix $\underline{E}$ can be generated as follows:

$$\underline{A} \cdot \underline{C} + (\underline{B} \cdot \underline{C})^* = \begin{pmatrix} \bar{b}_1 & a_2 & a_3 & \dots & a_N \\ a_1 & \bar{b}_2 & a_3 & \dots & a_N \\ \vdots & \vdots & \vdots & & \vdots \\ a_1 & a_2 & a_n & \dots & \bar{b}_N \end{pmatrix}$$

If "C" only appears n times, say on the second, fourth ... and m'th columns of the diagonal matrix, then $\underline{E} = \underline{A} \cdot \underline{C} + (\underline{B} \cdot \underline{C})^*$ will be a $N \times n$ matrix as follows:

$$\underline{E} = \begin{pmatrix} a_1 & \bar{b}_2 & a_3 & a_4 & \dots & a_m & \dots & a_N \\ a_1 & a_2 & a_3 & \bar{b}_4 & \dots & a_m & \dots & a_N \\ \vdots & \vdots & \vdots & \vdots & & \vdots & & \vdots \\ a_1 & a_2 & a_3 & a_4 & \dots & \bar{b}_m & \dots & a_N \end{pmatrix}$$

The rows of E matrix represent the expansion of $a \# b$ into an array of cubes. This matrix method of expressing $a \# b$ is very similar to the "partitioning" method described on pages 20 and 21 in Section 3.6 in connection with the regenerative regions of Hybrid groupoid of order six. Both methods are very useful in writing computer programmes for the sharp algorithm.

It is shown that it is possible to simplify the sharp algorithm so that the set $U = \{0, 1, i, j, k, K\}$ can be reduced to its subset $B = \{0, 1, i, j\}$. This will be further discussed in Section 6.1.

4.4 REDUNDANCIES IN THE ALGORITHM

The unified theory was founded on an Abelian groupoid of order six having an identity element. This algebraic structure was given in Table IV on page 16 and in a slightly different form in Table V on page 19.

This groupoid was generated by combining the "Idempotent Scalar" and "Commutative Vector" semigroups together. The binary operation used was called "Hybrid" because the interaction of the elements of the set $U = \{0, 1, i, j, k, K\}$ with each other fell into two distinctly different categories. The elements i, j, k interacted with each other in a manner similar to the scalar product of the unit cartesian vectors $\{\underline{i}, \underline{j}, \underline{k}\}$, whereas the interaction of K with i and j was similar to the cross product of the same vector set. In effect, the element K of the set U carried a cross product sign with it. In other words, carrying the similarity between the binary operation called "Hybrid" and the scalar and cross products of the cartesian vector set one step further, the element K could be considered as being equivalent to $k \times$.

Because of this dual nature of the "Hybrid" binary operation, the associativity did not hold and the combination of two semigroups resulted in a groupoid which was still another step lower in the algebraic hierarchy.

It is indeed very convenient to specify the algebraic structures associated with the algorithms of the four basic operators of the topological methods in a groupoid of order six. However, it would also be desirable to work with a more advanced structure, such as a ring, integral domain or a field, so that their well developed theories can be readily applied to the solution of the problems.

One way of achieving this desirable result is to use two algebraic structures having additive and multiplicative binary operations. Of course it is necessary that these two structures and their relations with each other should at least satisfy the axioms of a ring.

"Idempotent Scalar" semigroup of order five, given by Table I on page 11, completely defines the algebraic structures of the three operators, namely subsume, intersection and star product. If one looks for an additive group of order five which is related to this semigroup, one immediately finds that there is only one group of order five up to isomorphism and that group is cyclic. Since the idempotency in the multiplicative semigroup implies that each element is its own additive inverse, the cyclic group cannot be accepted as a useful structure. Moreover, it can be stated that unless the order of the "Idempotent Scalar" semigroup is reduced to four, it is not possible to find an additive group which has zeros placed on its diagonal.

Now it has already been shown in section 4.2 that the subsume operation can be simplified by replacing k's with e's in the upper or "a" cube. Since the intersection and star product operators do not use the element k, the set $S = \{0, 1, i, j, k\}$ can be reduced to a set $B = \{0, 1, i, j\}$ or stating it in a slightly different form $B' = \{\emptyset, e, i, j\}$. Hence the redundancy in the subsume operation can be eliminated and the order of the "Idempotent Scalar" semigroup reduced to four.

Now it is well known that there are two groups of order four up to isomorphisms. These are cyclic group Z_4 and the Klein 4-group. From these two groups, the Klein 4-group, given by Table II on page 13, has the identity element placed on its diagonal, and therefore can be used to form a ring with the "Idempotent Scalar" semigroup of order four.

The problem now becomes how to represent the algebraic structure of the sharp algorithm using the set $B = \{0, 1, i, j\}$ or $B' = \{\emptyset, e, i, j\}$ with an additive binary operation.

Since the subsume and intersection operations always precede the sharp operation, it is not possible to encounter a " $\emptyset$ ". Therefore $\emptyset$ is a redundant element in the sharp algorithm. In addition, the zero is the identity element in an additive group. Therefore 0, the first element of the subset B, will be interpreted as the identity e. This makes 1, the second element of the subset B, which was previously assigned to the identity element e, redundant.

In addition, an element K is required to produce the following binary operations:

$$\begin{aligned}K * i &= j \\K * j &= i \\K * e &= K\end{aligned}$$

Since 1 is now redundant, it will be assigned to K and the set $B = \{0, 1, i, j\}$ will be interpreted as $B'' = \{e, K, i, j\}$ in the sharp operation.

It is very important to point out at this stage that Klein-4 groups formed using the set B or B'' are isomorphic, just like "Idempotent Scalar" semigroups formed using the sets B and B' are isomorphic. However, in forming a ring, only the set $B = \{0, 1, i, j\}$ will be used. This will be further discussed in section 4.5.

Let us now review the algebraic structures which have been formed using this new approach.

Firstly, "Idempotent Scalar" semigroup of order four using the set $B' = \{\emptyset, e, i, j\}$ will be as follows:

.	$\emptyset$	e	i	j
$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$
e	$\emptyset$	e	i	j
i	$\emptyset$	i	i	$\emptyset$
j	$\emptyset$	j	$\emptyset$	j

TABLE XV
 $\langle B', \cdot \rangle$ SEMIGROUP

Secondly, Klein-4 group using the subset $B'' = \{e, K, i, j\}$ will be as follows:

+	e	K	i	j
e	e	K	i	j
K	K	e
i	i		e	K
j	j		K	e

TABLE XVI
 $\langle B'', + \rangle$ GROUP

It is shown that by eliminating a redundancy in each of the subsume and sharp operations, it is possible to reduce the order of the algebraic structures from five to four. This opens the way towards forming a four element Boolean ring with unity. In forming this ring the set $B = \{0, 1, i, j\}$ will be used, since Klein 4 groups formed using the sets B and B' are isomorphic just like "Idempotent Scalar" semigroups formed using the sets B and B' are isomorphic.

The encoding of the elements of B can be best accomplished using the elements of the group $Z_2 \times Z_2$ since this group is isomorphic to Klein 4 group. The elements of $Z_2 \times Z_2$ are as follows:

$$\{(0,0), (0,1), (1,0), (1,1)\}$$

$Z_2 \times Z_2$ is not isomorphic to Z_4 , since all four elements are of order two.

The topology of these elements is very interesting.

As it will be further discussed on page 93, the subsets of B $\{1, j\}$ and $\{0, i\}$ will be called on-set and off-set of logic operators respectively. It is also interesting to note that treating the elements of $Z_2 \times Z_2$ as the vertices of a cube and assuming that there are no D.C. terms we have:

$$ON = U_n \# OFF = x x \# 0 x = 1 x \text{ which is the set } \{1, j\}$$

$$OFF = U_n \# ON = x x \# 1 x = 0 x \text{ which is the set } \{0, i\}$$

Because of the topology of B , the set $\{0, i, 1, j\}$ form the dihedral group D_4 of permutations, the OCTIC GROUP.

4.5 FOUR ELEMENT BOOLEAN RING WITH UNITY

Let us consider a subset $B = \{0, 1, i, j\}$ of the set S . Moreover let us assign the quartal coding to the elements of the set B as follows:

$$0 = 00 = 0_4$$

$$1 = 11 = 3_4$$

$$i = 01 = 1_4$$

$$j = 10 = 2_4$$

Now it is well known that if a ring $\langle R, +, \cdot \rangle$ is a Boolean ring with unity, then $B = \langle R, \wedge, \vee, \bar{}, 0, 1 \rangle$ is a Boolean algebra if we define:

$$\bar{x} = 1 + x$$

$$x \wedge y = x \cdot y$$

$$x \vee y = x + y + (x \cdot y)$$

where $x + y = (x \wedge \bar{y}) \vee (\bar{x} \wedge y)$ is the exclusive-or (symmetric difference) operation [31].

Using this very important relationship between a Boolean ring with unity and the two-value logic under $\oplus$ and $\wedge$ operations, let us build two algebraic structures with the elements of the subset $B = \{0, 1, i, j\}$.

Now, using the bit by bit "and", $\wedge$, operation we have

$0 = 00$	$0 = 00$	$0 = 00$	$0 = 00$
$1 = 11$	$i = 01$	$j = 10$	$0 = 00$
$\wedge$	$\wedge$	$\wedge$	$\wedge$
$0 = 00$	$0 = 00$	$0 = 00$	$0 = 00$

$$\begin{array}{l} 1 = 11 \\ \underline{\Lambda} \quad 1 = \underline{\Lambda} \quad 11 \\ 1 = 11 \end{array}$$

$$\begin{array}{l} 1 = 11 \\ \underline{\Lambda} \quad i = \underline{\Lambda} \quad 01 \\ i = 01 \end{array}$$

$$\begin{array}{l} 1 = 11 \\ \underline{\Lambda} \quad j = \underline{\Lambda} \quad 10 \\ j = 10 \end{array}$$

$$\begin{array}{l} i = 01 \\ \underline{\Lambda} \quad i = \underline{\Lambda} \quad 01 \\ i = 01 \end{array}$$

$$\begin{array}{l} i = 01 \\ \underline{\Lambda} \quad j = \underline{\Lambda} \quad 10 \\ 0 = 00 \end{array}$$

$$\begin{array}{l} j = 10 \\ \underline{\Lambda} \quad j = \underline{\Lambda} \quad 10 \\ j = 10 \end{array}$$

The same results can be specified by defining a binary operation Λ on the subset $B = \{0, 1, i, j\}$ of the set S as shown in Table XVII.

Λ	0	1	i	j
0	0	0	0	0
1	0	1	i	j
i	0	i	i	0
j	0	j	0	j

TABLE XVII

$\langle B, \Lambda \rangle$ SEMIGROUP

By inspecting the table, it can be seen that:

1. Binary operation is closed
2. There is a unity $u = 1$
3. Groupoid is commutative
4. Elements of the groupoid are idempotent

In addition it can be shown that the binary operation is associative.

For example:

$$\begin{aligned} \text{(a)} \quad (i \wedge j) \wedge 1 &= 0 \wedge 1 = 0 \\ i \wedge (j \wedge 1) &= i \wedge j = 0 \end{aligned}$$

$$\begin{aligned} \text{(b)} \quad (i \wedge i) \wedge j &= i \wedge j = 0 \\ i \wedge (i \wedge j) &= i \wedge 0 = 0 \end{aligned}$$

$$\begin{aligned} \text{(c)} \quad (0 \wedge i) \wedge j &= 0 \wedge j = 0 \\ 0 \wedge (i \wedge j) &= 0 \wedge 0 = 0 \end{aligned}$$

$$\begin{aligned} \text{(d)} \quad (1 \wedge i) \wedge 0 &= i \wedge 0 = 0 \\ 1 \wedge (i \wedge 0) &= 1 \wedge 0 = 0 \end{aligned}$$

$$\begin{aligned} \text{(e)} \quad (1 \wedge 1) \wedge 0 &= 1 \wedge 0 = 0 \\ 1 \wedge (1 \wedge 0) &= 1 \wedge 0 = 0 \end{aligned}$$

Hence it can be seen that the algebraic structure $\langle B, \wedge \rangle$ given by Table XVII is a commutative semigroup with unity $u = 1$. Furthermore all of its elements are idempotent.

Similarly, using the bit by bit "exclusive-or", $\oplus$, operation we have

$$\begin{array}{rcl} 0 & = & 00 \\ \oplus 1 & = & \oplus 11 \\ \hline 1 & = & 11 \end{array} \quad \begin{array}{rcl} 0 & = & 00 \\ \oplus i & = & \oplus 01 \\ \hline i & = & 01 \end{array} \quad \begin{array}{rcl} 0 & = & 00 \\ \oplus j & = & \oplus 10 \\ \hline j & = & 10 \end{array} \quad \begin{array}{rcl} 0 & = & 00 \\ \oplus 0 & = & \oplus 00 \\ \hline 0 & = & 00 \end{array}$$

$$\begin{array}{rcl} 1 & = & 11 \\ \oplus 1 & = & \oplus 11 \\ \hline 0 & = & 00 \end{array} \quad \begin{array}{rcl} 1 & = & 11 \\ \oplus i & = & \oplus 01 \\ \hline j & = & 10 \end{array} \quad \begin{array}{rcl} 1 & = & 11 \\ \oplus j & = & \oplus 10 \\ \hline i & = & 01 \end{array}$$

$$\begin{array}{rcl} i & = & 01 \\ \oplus i & = & \oplus 01 \\ \hline 0 & = & 00 \end{array} \quad \begin{array}{rcl} i & = & 01 \\ \oplus j & = & \oplus 10 \\ \hline 1 & = & 11 \end{array} \quad \begin{array}{rcl} j & = & 10 \\ \oplus j & = & \oplus 10 \\ \hline 0 & = & 00 \end{array}$$

The same results can be specified by defining a binary operation $\oplus$ on the subset $B = \{0, 1, i, j\}$ of the set S as shown in Table XVIII.

$\oplus$	0	1	i	j
0	0	1	i	j
1	1	0
i	i		0	1
j	j		1	0

TABLE XVIII

$\langle B, \oplus \rangle$ GROUP

By inspecting Table XVIII, it can be seen that the algebraic structure $\langle B, \oplus \rangle$ is the Klein 4-group (Viergruppe), isomorphic to $Z_2 \times Z_2$.

At this point, the only axiom which is left to be proven to show that $\langle B, \oplus, \wedge \rangle$ is a Boolean ring with unity, is the right and left distributivity under exclusive-or, $\oplus$, namely:

$$\begin{aligned} a \wedge (b \oplus c) &= a \wedge b \oplus a \wedge c, \text{ and} \\ (b \oplus c) \wedge a &= b \wedge a \oplus c \wedge a \end{aligned}$$

The fact that the left distributive law holds can be demonstrated as follows:

$$\begin{aligned} \text{(a)} \quad i \wedge (j \oplus 1) &= i \wedge i = i \\ i \wedge j \oplus i \wedge 1 &= 0 \oplus i = i \end{aligned}$$

$$\begin{aligned} \text{(b)} \quad j \wedge (j \oplus i) &= j \wedge 1 = j \\ j \wedge j \oplus j \wedge i &= j \oplus 0 = j \end{aligned}$$

$$\begin{aligned} \text{(c)} \quad 1 \wedge (j \oplus 0) &= 1 \wedge j = j \\ 1 \wedge j \oplus 1 \wedge 0 &= j \oplus 0 = j \end{aligned}$$

$$\begin{aligned} \text{(d)} \quad 1 \wedge (i \oplus 1) &= 1 \wedge j = j \\ 1 \wedge i \oplus 1 \wedge 1 &= i \oplus 1 = j \end{aligned}$$

$$\begin{aligned} \text{(e)} \quad j \wedge (i \oplus j) &= j \wedge 1 = j \\ j \wedge i \oplus j \wedge j &= 0 \oplus j = j \end{aligned}$$

Similarly the right distributive law holds as shown below:

$$(a) \quad (j \oplus 1) \wedge i = i \wedge i = i \\ j \wedge i \oplus 1 \wedge i = 0 \oplus i = i$$

$$(b) \quad (j \oplus i) \wedge j = 1 \wedge j = j \\ j \wedge j \oplus i \wedge j = j \oplus 0 = j$$

$$(c) \quad (j \oplus 0) \wedge 1 = j \wedge 1 = j \\ j \wedge 1 \oplus 0 \wedge 1 = j \oplus 0 = j$$

$$(d) \quad (i \oplus 1) \wedge 1 = j \wedge 1 = j \\ i \wedge 1 \oplus 1 \wedge 1 = i \oplus 1 = j$$

$$(e) \quad (i \oplus j) \wedge j = 1 \wedge j = j \\ i \wedge j \oplus j \wedge j = 0 \oplus j = j$$

4.6 REDEFINITION OF THE SECOND MAPPING FUNCTION

On section 3.6, the second mapping function ψ was defined as follows:

$$\psi : Q \longrightarrow U \text{ such that } y_i^a = x_i^a \cdot \psi \text{ where } i \in N$$

y_i^a was called the i'th co-ordinate of the "a" cube in U domain.

The elimination of the redundant elements in the subsume and sharp algorithms makes it necessary to redefine the second mapping function as follows:

$$\xi : Q \longrightarrow L \text{ such that } z_i^a = x_i^a \cdot \xi \text{ where } i \in N$$

z_i^a is called the i'th co-ordinate of the "a" cube in L domain

where $N = \{0, 1, \dots, n\}$ is the set of the first n positive integers

the set $Q = \{0, 1, x\}$

the subset $L = \{\emptyset, e, i, j, K\}$ of the set U.

The i'th co-ordinates of the cubes in Q domain were given by

$$x_i^a = i \cdot \rho^a \text{ and } x_i^b = i \cdot \rho^b \text{ where } i \in N$$

Using the composite function $\rho \circ \xi$, the i'th co-ordinates in L domain will now be given by

$$z_i^a = i \cdot \rho^a \circ \xi \quad z_i^b = i \cdot \rho^b \circ \xi \text{ where } i \in N$$

Reviewing the tables given on pages 22 and 23, with the elimination of the redundancies in mind, the following conclusions can be drawn:

1. ξ is invariant to the choice of operators for all the co-ordinates of the "b" cube. Namely:

<u>Q domain</u>		<u>L domain</u>
0	→	i
1	→	j
x	→	e

2. ξ is invariant to the choice of operators for the co-ordinates of the "a" cube having 0's and 1's. Namely:

<u>Q domain</u>		<u>L domain</u>
0	→	i
1	→	j

3. ξ is operator dependent for the co-ordinates of the "a" cube having x's. This dependence is shown in Table XIX:

$\sqsubseteq, \sqcap, *$	#
e	K

TABLE XIX

OPERATOR DEPENDENCE OF x_i^a

IN ξ MAPPING

Let us now define a subset W of the set U to represent operator dependence of ψ as follows:

$$W = \{e, K\}$$

The images of 0, 1, and x can now be unified under ξ for all four operators as shown in Table XX:

ξ	0	1	x
"a" cube	i	j	{W}
"b" cube	i	j	e

TABLE XX
UNIFIED IMAGES UNDER ξ

4. ξ^{-1} is invariant to the choice of operators for the co-ordinates of the $a * b$ cube in L domain having i's, j's and $\emptyset$'s.

<u>L domain</u>		<u>Q domain</u>
i	$\longrightarrow$	0
j	$\longrightarrow$	1
$\emptyset$	$\longrightarrow$	$\emptyset$

(Note that in Star Product Operator, one $\emptyset$ is allowed)

5. ξ^{-1} is operator dependent for certain co-ordinates of the $a * b$ cube when it is mapping back to x in Q domain. This dependence is shown in Table XXI.

$\subseteq, \cap$	*	#
e	e or 1st $\emptyset$	K

TABLE XXI
OPERATOR DEPENDENCE OF z_i^c

Let us now define a subset V of the set U to represent the operator dependence of ξ^{-1} as follows:

$$V = \{e, \emptyset, K\}$$

The images of i, j, $\emptyset$ and {V} can now be unified under ξ^{-1} for all four operators as shown in Table XXII.

ξ^{-1}	i	j	{v}	$\emptyset$
a * b cube	0	1	x	$\emptyset$

TABLE XXII
UNIFIED IMAGES UNDER ξ^{-1}

- (6) In sharp operation when e_i appears in the i'th column of the a * b cube in L domain, it is mapped back to Q domain by ξ^{-1} as $i.p^a$.

4.7 DEFINITION OF THE THIRD MAPPING FUNCTION

The second mapping function ξ maps the co-ordinates of "a" and "b" cubes from Q domain to L domain. Before the binary operations "and", $\wedge$, and "exclusive-or", $\oplus$, of the Boolean ring can be applied to the elements of the quartal coding, one more mapping, ζ , must be carried out. This is necessary because five elements of the subset $L = \{\emptyset, e, i, j, K\}$ will be represented by four elements of the set $B = \{0, 1, i, j\}$, or using quartal coding $B = \{00, 11, 01, 10\}$ as per the assignments specified in section 4.5.

To make this possible, the subset L forms two additional subsets $B' = \{\emptyset, e, i, j\}$ and $B'' = \{e, K, i, j\}$. The operator dependence of these subsets and the choice of binary operation is shown in Table XXIII.

ζ	$\sqsubset, \sqsupset, *$	#
Subset of L	B'	B''
Binary Operation	$\wedge$	$\oplus$

TABLE XXIII
SUBSETS AND OPERATIONS IN ζ

Let us define the third mapping function as follows:

$$\zeta : L \longrightarrow B \text{ such that } Z_i^a = z_i^a \cdot \zeta \text{ where } i \in \mathbb{N}.$$

Z_i^a will be called the i'th co-ordinate of the "a" cube in the B domain.

Using the composite function $\rho \circ \xi \circ \zeta$, the i 'th co-ordinates in B domain will be given by:

$$Z_i^a = i.\rho^a \circ \xi \circ \zeta \quad \text{and} \quad Z_i^b = i.\rho^b \circ \xi \circ \zeta \quad \text{where } i \in N.$$

The third mapping function ζ has the following properties:

1. ζ is invariant to the choice of operators for i and j co-ordinates of the "a" and "b" cubes in L domain.
2. ζ is operator dependent for $\emptyset$, e, K co-ordinates of the "a" and "b" cubes in L domain. This dependence is shown in Table XXIV:

ζ	$\emptyset$	e	K
$\sqsubseteq, \sqsupset, *$	00	11	
#		00	11

TABLE XXIV
OPERATOR DEPENDENCE OF ζ

3. Mapping back into L domain from B domain, ζ^{-1} has the same properties as ζ .

When the Z co-ordinates of the "a" and "b" cubes have been obtained, it is just a matter of applying the algebra of the Boolean ring $\langle B, \oplus, \wedge \rangle$ given in Tables XVII and XVIII. Hence the operation in B domain will be as follows:

$$(i.\rho^a \circ \xi \circ \zeta) * (i.\rho^b \circ \xi \circ \zeta)$$

Since $\langle B, \oplus, \wedge \rangle$ is commutative, the binary operations represented by $*$ are also commutative. Mapping back into Q domain, we have:

$$\left[(i.\rho^a \circ \xi \circ \zeta) * (i.\rho^b \circ \xi \circ \zeta) \right] . \zeta^{-1} \circ \xi^{-1} \quad \text{where } i \in \mathbb{N}$$

4.8 BOOLEAN RING AND THE CUBICAL ARRAYS

In chapter four, the application of the unified theory to the solution of combinatorial problems involving cubical arrays was reviewed. It was shown that when the co-ordinates of the cubes were mapped from Q domain to U domain, it was necessary to replace X_a with another element of the set $X = \{k, e, K\}$ in order to switch from one operator to another.

In using the Boolean ring approach, the second mapping function ξ maps the co-ordinates of the cubes from Q domain to L domain. When switching operators, it is necessary to ~~replace~~ X_a with another element of the set $W = \{e, K\}$. As shown in table XIX, on page 67. the replacement of e by K takes place only when switching to the sharp operator.

In addition, there is a third mapping function ζ which maps the co-ordinates from L to B domain. In this case, when switching to the sharp operator, both the binary operation and the quartal coding must be changed as shown in tables XXIII and XXIV on pages 70 and 71.

In the examples given below, we shall use the elements of the Boolean ring $B = \{0, 1, i, j\}$ in place of quartal coding with the understanding that:

$$0 = 00 = 0_4$$

$$i = 01 = 1_4$$

$$j = 10 = 2_4$$

$$1 = 11 = 3_4$$

4.8.1 CUBE UNION (L)

$$A \sqcup B = \Sigma A \cup B = \Sigma (a^1, a^2, \dots a^n, b^1, b^2, \dots b^m)$$

where Σ is the absorption operator.

Example:

Let $A = \{000, x11\}$ and $B = \{0x0, x1x\}$

$$\begin{array}{ll} a^1 = 000 \rightarrow iii \rightarrow iii & b^1 = 0x0 \rightarrow iei \rightarrow ili \\ a^2 = x11 \rightarrow ejj \rightarrow \underline{1jj} & b^2 = x1x \rightarrow eje \rightarrow \underline{1jl} \\ & i\emptyset\emptyset \leftarrow i00 \qquad \qquad \qquad iji \end{array}$$

Therefore

$$a^1 \not\sqsubseteq a^2 \quad a^2 \not\sqsubseteq a_1$$

Therefore

$$b^1 \not\sqsubseteq b_2 \quad \text{and} \quad b^2 \not\sqsubseteq b_1$$

$$\begin{array}{ll} a^1 = 000 \rightarrow iii \rightarrow iii & a^2 = x11 \rightarrow ejj \rightarrow \underline{1jj} \\ b^1 = 0x0 \rightarrow iei \rightarrow \underline{ili} & b^2 = x1x \rightarrow eje \rightarrow \underline{1jl} \\ & iji \end{array}$$

$a^1 \sqsubseteq b^1$ Therefore delete a^1 ,

also $b^1 \not\sqsubseteq a^1$

$a^2 \sqsubseteq b^2$ Therefore delete a^2 ,

also $b^2 \not\sqsubseteq a^2$

$$\begin{array}{ll} a^1 = 000 \rightarrow iii \rightarrow iii & a^2 = x11 \rightarrow ejj \rightarrow \underline{1jj} \\ b^2 = x1x \rightarrow eie \rightarrow \underline{1il} & b^1 = 0x0 \rightarrow iei \rightarrow \underline{ili} \\ & iji \end{array}$$

$a^1 \sqsubseteq b_2$ Therefore delete a_1

also $b_2 \not\sqsubseteq a_1$

Therefore $a^2 \not\sqsubseteq b^1$ $b^1 \not\sqsubseteq a^2$

Hence: $A \sqcup B = \{0x0, x1x\}$ since a^1, a^2 have been deleted.

4.8.2 CUBE INTERSECTION ($\cap$)

$$A \cap B = \sum_{q=1}^m \sum_{p=1}^n a^p b^q \text{ where } \Sigma \text{ is the absorption operator.}$$

Let $A = \{x 1 x, 0 0 0\}$ and $B = \{0 x 0, x 1 1\}$

$$\begin{aligned} a^1 &= x 1 x \rightarrow e j e \rightarrow l j l \\ b^1 &= 0 x 0 \rightarrow i e i \rightarrow i l i \\ c_{11} &= 0 1 0 \leftarrow i j i \leftarrow i j i \end{aligned}$$

$$\begin{aligned} a^1 &= x 1 x \rightarrow e j e \rightarrow l j l \\ b^2 &= x 1 1 \rightarrow e j j \rightarrow l j j \\ c_{12} &= x 1 1 \leftarrow e j j \leftarrow l j j \end{aligned}$$

$$\begin{aligned} a^2 &= 0 0 0 \rightarrow i i i \rightarrow i i i \\ b^1 &= 0 x 0 \rightarrow i e i \rightarrow i l i \\ c_{21} &= 0 0 0 \leftarrow i i i \leftarrow i i i \end{aligned}$$

$$\begin{aligned} a^2 &= 0 0 0 \rightarrow i i i \rightarrow i i i \\ b^2 &= x 1 1 \rightarrow e j j \rightarrow l j j \\ i \emptyset \emptyset &\leftarrow i 0 0 \end{aligned}$$

Hence: $A \cap B = \Sigma \begin{pmatrix} c_{11} & c_{12} \\ c_{21} & \emptyset \end{pmatrix}$

$$\begin{aligned} c_{11} &\rightarrow i j i \\ c_{12} &\rightarrow l j j \\ i j \emptyset &\leftarrow i j 0 \end{aligned}$$

$$\begin{aligned} c_{11} &\rightarrow i j i \\ c_{21} &\rightarrow i i i \\ i \emptyset i &\leftarrow i 0 i \end{aligned}$$

$$\begin{aligned} c_{12} &\rightarrow l j j \\ c_{21} &\rightarrow i i i \\ i \emptyset \emptyset &\leftarrow i 0 0 \end{aligned}$$

Therefore $c_{11} \nsubseteq c_{12}$
and $c_{12} \nsubseteq c_{11}$

Therefore $c_{11} \nsubseteq c_{21}$
and $c_{21} \nsubseteq c_{11}$

Therefore $c_{12} \nsubseteq c_{21}$
and $c_{21} \nsubseteq c_{12}$

Since there is no absorption:

$$A \cap B = \{0 0 0, 0 1 0, x 1 1\}$$

$$\begin{array}{lcl}
 c^3 & = & 001x \rightarrow iije \rightarrow iijl \\
 b^2 & = & 1111 \rightarrow jjjj \rightarrow jjjj \\
 & & \Lambda \\
 & & \emptyset\emptyset jj \leftarrow 00jj
 \end{array}$$

Therefore $c^3 \not\sqsubseteq b^2$, $b^2 \not\sqsubseteq c^3$ and $b^2 \not\sqsubseteq c^3$

$$d^4 = c^3 = 001x$$

Hence we have:

$$[(a^2 \# b^1) \# b^2] = \{001x\} \text{ and}$$

$$A \# B = \{110x, 11x0, 1x0x\} \sqcup \{001x\}$$

$$\begin{array}{lcl}
 d^1 & = & 110x \rightarrow jjie \rightarrow jjil \\
 d^2 & = & 11x0 \rightarrow jjei \rightarrow jjli \\
 & & \Lambda \\
 & & jjii
 \end{array}$$

Therefore $d^1 \not\sqsubseteq d^2$, $d^2 \not\sqsubseteq d^1$

$$\begin{array}{lcl}
 d^1 & = & 110x \rightarrow jjie \rightarrow jjil \\
 d^3 & = & 1x0x \rightarrow jeei \rightarrow jlii \\
 & & \Lambda \\
 & & jjil
 \end{array}$$

$d^1 \sqsubseteq d^3$ Therefore

delete d^1

$$\begin{array}{lcl}
 d^1 & = & 110x \rightarrow jjie \rightarrow jjil \\
 d^4 & = & 001x \rightarrow iije \rightarrow iijl \\
 & & \Lambda \\
 & & \emptyset\emptyset\emptyset e \leftarrow 0001
 \end{array}$$

Therefore $d^1 \not\sqsubseteq d^2$ and $d^2 \not\sqsubseteq d^1$

CHAPTER V

FIELD STRUCTURE *

5.1 STILL ANOTHER APPROACH TO THE UNIFIED THEORY

Elimination of certain redundancies in the subsume and sharp algorithms made it possible to move from an Abelian groupoid of order six to a four element Boolean ring with unity. In addition, a third mapping function had to be used to map the elements of the sets $B' = \{\emptyset, e, i, j\}$ and $B'' = \{e, K, i, j\}$ to $B = \{0, 1, i, j\}$. However, as a result of these manipulations, it was possible to replace the octal coding by quartal coding which required less storage space and execution time. Furthermore by the use of "and" and "exclusive-or" operations, the unified theory has been linked to Boolean algebra and Boolean rings.

At this point, one may wonder whether it is possible to base the unified theory on a structure which is still higher in the algebraic hierarchy. There are no more redundancies to be eliminated in the algorithms, and the four element Boolean ring with unity has already been congested by the double assignments.

However, it may be worthwhile to investigate the "ideals" and the "quotient ring" or "the residue class ring R modulo N ", " R/N ", of the four element Boolean ring with unity upon which the unified theory is founded.

A well-known theorem of the abstract algebra states the following: "Let R be a commutative ring with unity. Then N is a maximal ideal of R if, and only if, R/N is a field".

Another well-known theorem states the following: "Let R be a commutative ring with unity, and let $N \neq R$ be an ideal in R . Then " R/N " is an integral domain if, and only if, N is a prime ideal in R ."

A corollary to this theorem is as follows: Every maximal ideal in a commutative ring R with unity is a prime ideal.

* [5] [31] [32]

Now a Boolean ring satisfies the eight basic axioms of the rings and an additional (9th) axiom, which is idempotency. On the other hand idempotency, $a \cdot a = a$ for every $a \in B$, implies that Boolean ring is a commutative ring. Therefore the theorems stated above apply to Boolean rings since they are commutative.

Since both the integral domain and the field are higher structures that the rings in the algebraic hierarchy, the ideals of the Boolean ring with unity $B = \{0, 1, i, j\}$ will be investigated.

5.2 IDEALS AND QUOTIENT RINGS

The ideals in the theory of rings are defined as follows:

A subring N of a ring R satisfying $r \cdot N \subseteq N$ and $N \cdot r \subseteq N$ for all $r \in R$ is an "ideal" (or two sided ideal) of R .

Similarly a "quotient ring" is defined as follows:

If N is an ideal in a ring R , then the ring of cosets $r + N$ under the induced operations is the "quotient ring" or "residue class ring of R modulo N ", and is denoted by " R/N ". The cosets are "residue classes modulo N ".

It will now be shown that $N = \{0, i\}$ is an ideal of the Boolean ring with unity $B = \{0, 1, i, j\}$. For this, let us first show that N is a subring of B . A "subring of a ring" is a non-empty subset of a ring which is a ring under induced operations from the whole ring. The proof that N is a subring of B will be based on the following theorem: "A non-empty subset C of a ring R is a subring if, and only if, $a - b \in C$ and $a \cdot b = b \cdot a \in C$ for every $a, b \in C$." The first condition $a - b \in C$ for every $a, b \in C$ implies that $\langle N, \oplus \rangle$ is an additive subgroup of $\langle B, \oplus, \wedge \rangle$. This can be shown with the help of Table XXV.

$\oplus$	0	i
0	0	i
i	i	0

TABLE XXV

SUBGROUP $\langle N, \oplus \rangle$

using quartal coding as before, where

$$0 = 00 = 0_4$$

$$1 = 11 = 3_4$$

$$i = 01 = 1_4$$

$$j = 10 = 2_4$$

we have

$$\begin{array}{lll}
 0 = 0 \cdot 0 & 0 = 00 & i = 01 \\
 \oplus i = \oplus 01 & \oplus 0 = \oplus 00 & \oplus i = \oplus 01 \\
 \hline
 i = 01 & 0 = 00 & 0 = 00
 \end{array}$$

It can be seen that the four axioms of a group are satisfied

- (1) Binary operation $\oplus$ is closed on N
- (2) There is an identity element $e = 0 \in N$ such that $0 + i = i + 0 = i$
- (3) Every element is its own additive inverse.
- (4) Associativity holds since

$$\begin{array}{ll}
 (0 \oplus i) \oplus i = 0 & (i \oplus i) \oplus i = i \\
 0 \oplus (i \oplus i) = 0 & i \oplus (i \oplus i) = i \\
 \\
 (i \oplus i) \oplus 0 = 0 & (0 \oplus 0) \oplus 0 = 0 \\
 i \oplus (i \oplus 0) = 0 & 0 \oplus (0 \oplus 0) = 0
 \end{array}$$

Therefore $\langle N, \oplus \rangle$ is an additive subgroup of $\langle B, \oplus, \wedge \rangle$.

To show that $\langle N, \oplus, \wedge \rangle$ is a subring of B , we have to prove that $a \wedge b = b \wedge a \in N$ for every $a, b \in N$. Indeed this is so since

$$\begin{array}{cccc}
 0 = 0 \ 0 & i = 0 \ 1 & i = 0 \ 1 & 0 = 0 \ 0 \\
 \wedge & \wedge & \wedge & \wedge \\
 i = 0 \ 1 & 0 = 0 \ 0 & i = 0 \ 1 & 0 = 0 \ 0 \\
 \wedge & \wedge & \wedge & \wedge \\
 0 = 0 \ 0 & 0 = 0 \ 0 & i = 0 \ 1 & 0 = 0 \ 0
 \end{array}$$

This is shown in Table XXVI.

$\wedge$	0	i
0	0	0
i	0	i

TABLE XXVI

SEMIGROUP $\langle N, \wedge \rangle$

Hence it has been shown that $N = \{0, i\}$ is a subring of $B = \{0, 1, i, j\}$.

For N to be an ideal of B , it must be shown that $b \cdot N \subseteq N$ and $N \cdot b \subseteq N$ for every $b \in B$. This is indeed so because

$$\begin{array}{cccc}
 1 = 1 \ 1 & 1 = 1 \ 1 & j = 1 \ 0 & j = 1 \ 0 \\
 0 = 0 \ 0 & i = 0 \ 1 & 0 = 0 \ 0 & i = 0 \ 1 \\
 \wedge & \wedge & \wedge & \wedge \\
 0 = 0 \ 0 & i = 0 \ 1 & 0 = 0 \ 0 & 0 = 0 \ 0
 \end{array}$$

The cosets of N are $N \oplus 1$ and $N \oplus j$. It will be shown that these two cosets are identical.

$$\begin{array}{rcl}
 0 & = & 0 \ 0 \\
 \oplus \frac{1}{1} & = & \oplus \frac{1 \ 1}{1 \ 1} \\
 \oplus \frac{1}{j} & = & \oplus \frac{1 \ 1}{1 \ 0}
 \end{array}
 \qquad
 \begin{array}{rcl}
 i & = & 0 \ 1 \\
 \oplus \frac{1}{j} & = & \oplus \frac{1 \ 1}{1 \ 0}
 \end{array}
 \qquad
 \begin{array}{rcl}
 0 & = & 0 \ 0 \\
 \oplus \frac{j}{j} & = & \oplus \frac{1 \ 0}{1 \ 0} \\
 \oplus \frac{j}{1} & = & \oplus \frac{1 \ 0}{1 \ 1}
 \end{array}$$

Therefore $N \oplus 1 = N \oplus j = \{1, j\} = M$

This is expected from Lagrange theorem which states the following:
 "Let G be a group of finite order " n " and H a subgroup of G . The order of H divides the order of G ." A corollary to this theorem is as follows: "If G is a finite group then

$$|G| = |H| \cdot [G : H]$$

where $[G : H]$ is the number of right (left) cosets of H in G and called the "index" of H in G ."

In our case $|N| = 2$, $|B| = 4$, therefore $[B : N] = |B| / |N| = 4 / 2 = 2$
 Hence the only two cosets of $B = \{0, 1, i, j\}$ are $N = \{0, i\}$ and $M = \{1, j\}$.

Before investigating the quotient ring of B , let us review the following theorems which are pertinent to our problem.

Theorem: "If a group G does split up into blocks with the induced operation well defined, and if the blocks form a group under this induced operation, then the block containing the identity " e " of G must be a subgroup of G ."

In our case, it has already been shown that $\langle N, \oplus \rangle$ does contain additive identity $e = 0$, and is, in fact, an additive subgroup of $\langle B, \oplus, \wedge \rangle$

Theorem: "If a group G splits up into blocks with the induced operation well defined and giving a group, then the blocks must be precisely the left (and also the right) cosets of a subgroup of G . In particular, every left coset must be a right coset."

In our case, we have two cosets, the ideal $N = \{0, i\}$ and its coset $M = \{1, j\}$. In addition, left cosets and right cosets are identical. Hence it is possible that N and M may be the blocks which form a group. Unfortunately, it is not always true that the cosets of a subgroup of a group do form a group under the induced operation. The difficulty lies in the fact that the induced operation may not be well defined.

To see whether or not the induced operations $\oplus$ and $\wedge$ are well defined, let us split our Boolean ring with unity $B = \{0, 1, i, j\}$ into blocks using $N = \{0, i\}$ and $M = \{1, j\}$. This is shown in Tables XXVII and XXVIII.

$\oplus$	0	i	1	j
0	0	i	1	j
i	i	0	j	1
1	1	j	0	i
j	j	1	i	0

TABLE XXVII

$\wedge$	0	i	1	j
0	0	0	0	0
i	0	i	i	0
1	0	i	1	j
j	0	0	j	i

TABLE XXVIII

SPLITTING OF $\langle B, \oplus, \wedge \rangle$ INTO BLOCKS

One glance at these tables XXVII and XXVIII shows that the induced operations $\oplus$ and $\wedge$ are well defined because the products of two blocks always produce elements which are in just one coset. Some of the blocks in Tables XXVII and XXVIII have been shaded to demonstrate this very important property.

After splitting Z_6 into three cosets with the induced operation well defined, Prof. J. B. Fraleigh very aptly wrote in his book [5] "It works. Isn't that lovely? Such a delightful display of symmetry should make shivers of joy run up and down the spine of anyone with any mathematical sensitivity."

Now that the quotient ring " B/N " has been formed, its algebraic structure will have to be determined. As stated in Section 5.1, this will depend on N being a maximal or prime ideal. If N is a maximal ideal, then " B/N " is a field. On the other hand, if N is a prime ideal, " B/N " is an integral domain. By inspecting Tables XXVII and XXVIII, one can immediately say " B/N " is a field isomorphic to Z_2 . However let us prove that N is a maximal and also a prime ideal.

In abstract algebra, a "maximal ideal" of a ring R is defined as follows: "An ideal M , different from R , such that there is no proper ideal N of R properly containing M ." In our case $N = \{0, 1\}$ is a maximal ideal because $B = \{0, 1, i, j\}$ does not have any other ideals, and therefore does not have any ideals which can properly contain N .

In Boolean algebra, the maximal ideal is defined as follows: "Given a proper ideal N of a Boolean algebra B . Then N is maximal if, and only if, for any y in B either $y \in N$ or $\bar{y} \in N$."

In our case $0, i \in N$ since $N = \{0, i\}$. Also $\bar{1} = \bar{1} \bar{1} = 0 \cdot 0 = 0 \in N$ and $\bar{j} = \bar{1} \bar{0} = 0 \cdot 1 = i \in N$. Hence N is a maximal ideal.

Let us now show that $N = \{0, i\}$ is also a prime ideal. In abstract algebra, a prime ideal is defined as follows: "An ideal $N \neq R$ in a commutative ring R is a prime ideal if $a \cdot b \in N$ implies that either $a \in N$ or $b \in N$ for all $a, b \in R$."

Now it can be seen from Table XXVIII that the elements of the three unshaded blocks belong to N . Again from the same table it can be seen $a \wedge b \in N$ implies either $a \in N$ or $b \in N$ for all a, b belonging to those three blocks.

In Boolean algebra a prime ideal is defined as follows: "An ideal N in a Boolean algebra B is prime if, and only if, for any x and y in B , $x \notin N$ and $y \notin N$ imply $x \wedge y \notin N$."

Again it can be seen from Table XXVIII that the elements of the shaded block do not belong to N . It is also clear that from Table XXVIII that $a \wedge b \notin N$ implies $a \notin N$ and $b \notin N$ in this shaded block.

Hence it has been shown that $N = \{0, i\}$ is a maximal and prime ideal of the four element Boolean ring with unity $B = \{0, 1, i, j\}$. Therefore, from the theorem stated in section 5.1, the quotient ring " B/N " is a field. By comparing the individual blocks shown in Tables XXVII and XXVIII with the Tables XXIX and XXX, it can be very easily seen that " B/N " is isomorphic to Z_2 .

$\oplus$	0	1
0	0	1
1	1	0

TABLE XXIX

$\wedge$	0	1
0	0	0
1	0	1

TABLE XXX

Z_2 FIELD STRUCTURES

5.3 HOMOMORPHISM AND ISOMORPHISM

Let us now investigate the properties of the function ϕ mapping the Boolean ring with unity $B = \{0, 1, i, j\}$ into its quotient ring " B/N " where $N = \{0, i\}$.

The fundamental homomorphism theorem states the following: "Let ϕ be a homomorphism of a ring R into a ring R' with kernel K . Then $R\phi$ is a ring and there is a canonical isomorphism of $R\phi$ with " R/K "."

Now from the definition, the kernel of homomorphism ϕ of a ring R into a ring R' is the set of all elements of R mapped into the additive identity $0'$ of R' .

Another well-known theorem states the following: "The set S of elements mapped onto zero in any homomorphism of a ring R is an ideal of R ."

In our case the quotient ring " B/N " is isomorphic to $Z_2 = \{0', 1'\}$. From the above theorem it is clear that the ideal $N = \{0, i\}$ will map onto $0'$ of Z_2 . This is confirmed by another theorem which states the following: "Let ϕ be a homomorphism of a ring R into R' . If 0 is the additive identity in R , then $0\phi = 0'$ is the additive identity in R' ."

Since $N = \{0, i\}$ maps onto $0'$ of Z_2 , from the fundamental homomorphism theorem $M = \{1, j\}$ has to map onto $1'$ of Z_2 .

The Figures 2 and 3 illustrate the application of the fundamental homomorphism theorem to the Boolean ring with unity and its quotient ring.

FIGURE 2

FIGURE 3

Let us look at the mapping function $\phi : R \rightarrow R'$ in greater detail.
From Figures 2 and 3 the following equations can be written down:

$$0 \cdot \phi = 0' \quad i \cdot \phi = 0' \quad 1 \cdot \phi = 1' \quad j \cdot \phi = 1' \quad \text{where } 0', 1' \in \mathbb{Z}_2$$

Now a map ϕ of a ring R into a ring R' is homomorphism if

$$(a + b) \phi = a \cdot \phi + b \cdot \phi$$

$$(a \cdot b) \phi = (a \cdot \phi) \cdot (b \cdot \phi)$$

Using the Boolean ring with unity $B = \{0, 1, i, j\}$, we can verify that ϑ is homomorphism of B to Z_2 as follows:

$$(i) \quad (0 \oplus i) \vartheta = (0 \cdot \vartheta) \oplus (i \cdot \vartheta) = 0' \oplus 0' = 0' \\ i \cdot \vartheta = 0'$$

$$(0 \wedge i) \vartheta = (0 \cdot \vartheta) \wedge (i \cdot \vartheta) = 0' \wedge 0' = 0' \\ 0 \cdot \vartheta = 0'$$

$$(ii) \quad (i \oplus j) \vartheta = (i \cdot \vartheta) \oplus (j \cdot \vartheta) = 0' \oplus 1' = 1' \\ 1 \cdot \vartheta = 1'$$

$$(i \wedge j) \vartheta = (i \cdot \vartheta) \wedge (j \cdot \vartheta) = 0' \wedge 1' = 0' \\ 0 \cdot \vartheta = 0'$$

$$(iii) \quad (0 \oplus j) \vartheta = (0 \cdot \vartheta) \oplus (j \cdot \vartheta) = 0' \oplus 1' = 1' \\ j \cdot \vartheta = 1'$$

$$(0 \wedge j) \vartheta = (0 \cdot \vartheta) \wedge (j \cdot \vartheta) = 0' \wedge 1' = 0' \\ 0 \cdot \vartheta = 0'$$

$$(iv) \quad (1 \oplus j) \vartheta = (1 \cdot \vartheta) \oplus (j \cdot \vartheta) = 1' \oplus 1' = 0' \\ i \cdot \vartheta = 0'$$

$$(1 \wedge j) \vartheta = (1 \cdot \vartheta) \wedge (j \cdot \vartheta) = 1' \wedge 1' = 1' \\ j \cdot \vartheta = 1'$$

$$(v) \quad (i \oplus i) \vartheta = (i \cdot \vartheta) \oplus (i \cdot \vartheta) = 0' \oplus 0' = 0' \\ 0 \cdot \vartheta = 0'$$

$$(i \wedge i) \vartheta = (i \cdot \vartheta) \wedge (i \cdot \vartheta) = 0' \wedge 0' = 0' \\ i \cdot \vartheta = 0'$$

$$\begin{aligned} \text{(vi)} \quad (0 \oplus 1) \emptyset &= (0 \cdot \emptyset) \oplus (1 \cdot \emptyset) = 0' \oplus 1' = 1' \\ 1 \cdot \emptyset &= 1' \end{aligned}$$

$$\begin{aligned} (0 \wedge 1) \emptyset &= (0 \cdot \emptyset) \wedge (1 \cdot \emptyset) = 0' \wedge 1' = 0' \\ 0 \cdot \emptyset &= 0' \end{aligned}$$

5.4 REDEFINITION OF THE THIRD MAPPING FUNCTION

The second mapping function ξ was used to map the co-ordinates of "a" and "b" cubes from Q domain to L domain. When using a Boolean ring with unity, a third mapping function ζ was used to map five elements of the subset $L = \{\emptyset, e, i, j, K\}$ into four elements of the set $B = \{0, 1, i, j\}$, or using quartal coding $B = \{00, 11, 01, 10\}$.

To make this mapping possible, two additional subsets $B' = \{\emptyset, e, i, j\}$ and $B'' = \{e, K, i, j\}$ were formed. From these, B' represented the subsume, intersection and star product operations and the other, B'' , was used with sharp algorithm.

Using a field structure, the third mapping function is expected to map five elements of the subset L into two elements of Z_2 . Let us assume that B' and B'' subsets will be used for this purpose. Even then four elements of these subsets will have to be mapped into $\{0', 1'\}$ in an orderly way so that there will not be any ambiguity.

It has been shown that the maximal ideal $N = \{0, i\}$ maps into $0'$ and its coset $M = \{1, j\}$ maps into $1'$. It is obvious that the elements of Z_2 must be given additional identifications or signs to accomplish one to one mapping.

For this purpose, the following sign convention is proposed for the elements of Z_2 :

$$\begin{array}{cc}
 N = \{ 0, i \} & M = \{ 1, j \} \\
 \downarrow \quad \downarrow & \downarrow \quad \downarrow \\
 0' \quad 0' & 1' \quad 1' \\
 + \quad - & - \quad +
 \end{array}$$

Although the coding is binary, every element is still represented by two bits. As per the convention, the left most bit represents the sign and the second bit represents the coding. Hence:

$$0 \rightarrow 0' \circ 0' \quad i \rightarrow 1' \circ 0' \quad 1 \rightarrow 1' \circ 1' \quad j \rightarrow 0' \circ 1'$$

Again as per the convention, $0'$ represents positive sign and $1'$ represents the negative sign.

Let us compare the quartal coding which was used with ring structure and the binary coding with sign convention proposed for the field structure:

	0	1	i	j
Quartal Coding of Ring Structure	0 0	1 1	0 1	1 0
Binary Coding with Sign Convention	0' 0'	1' 0'	1' 1'	0' 1'

As can be seen, encoding of 0 and 1 are identical in both cases and encoding of i and j are complements of each other. In choosing the quartal coding, we could have just as easily adopted the coding $i = 1\ 0$ and $j = 0\ 1$ instead of $i = 0\ 1$ and $j = 1\ 0$ without changing the algebraic structure of the Boolean ring with unity. In other words, the unified theory based on the field Z_2 is leading us back to the theory previously developed using a ring structure. However, the field approach provides

us with a better understanding of the theory and gives us an insight to the algorithms used. In particular, quartal coding used in ring structure can be looked upon as binary coding with positive or negative sign. In other words, the left most bit represents the sign and the next bit the coding.

In section 1.1 the n'th Cartesian product $Z_2^{(n)}$ was mapped into Z_2 as follows: $f : Z_2^{(n)} \rightarrow Z_2$
The set $(1).f^{-1}$ was called the on-set of the switching function and the set $(0).f^{-1}$ was called the off-set of the switching function.

Similarly the ideal N and its coset M will be called the off-set and on-set of the logic operators respectively. Hence:
ON-SET $(1').\emptyset^{-1} = \{1, j\} = M$ OFF-SET $(0').\emptyset^{-1} = \{0, i\} = N$
Now in subsume, cube intersection, and star product operations "1" of the on-set maps into $e \in B' = \{\emptyset, e, i, j\}$ and "0" of the off-set maps into $\emptyset \in B'$. In the previous sections, it was shown that the identity element "e" always contributed towards an affirmative result, whereas $\emptyset$ indicated a negative outcome, such as $a \nsubseteq b$, $a \nabla b$, etc.

In sharp algorithm "1" of the on-set maps into $K \in B'' = \{e, K, i, j\}$ and "0" of the off-set maps into $e \in B''$. Again it was shown that the existence of K in the upper cube was necessary to generate a new cube or cubes, whereas "e" simply mapped back to the co-ordinates of the "a" (upper) cube. Hence the binary operations $K * i$ and $K * j$ resulted in an affirmative action by generating a new cube or cubes, while the existence of "e" contributed towards a negative outcome such as $a \# b = a$ since $a \nabla b$.

It was also shown that "i" mapped into "0" and "j" mapped into "1" in Q domain. Therefore one would expect to find "j" in the on-set and "i" in the off-set. This is indeed so.

Definition of ON and OFF sets of the logic operators and understanding of their functions provides additional visibility into unified theory.

CHAPTER VI

COMPARISONS AND CONCLUSIONS

6.1 COMPARISON OF ALGEBRAIC STRUCTURES

The theory of algebraic topological methods for the synthesis of switching systems has been unified using:

- (a) Abelian groupoid of order six having an identity element.
- (b) Four element Boolean ring with unity.
- (c) Field of Z_2 .

All three structures have advantages and disadvantages. These will now be discussed in some detail.

6.1.1 GROUPOID STRUCTURE

Advantages:

- (1) Algebra associated with the algorithms of all four operators can be shown in one table using "Hybrid" binary operations.
- (2) Only the x co-ordinate of the "a" cube (upper cube) is operator dependent.
- (3) It uses two mapping functions. This makes the theory relatively simple.
- (4) It provides greater ease in programming. There is almost one to one correspondence between the theory and software.

Disadvantages:

- (1) Redundancies in the algorithms result in a groupoid structure which is very low in the algebraic hierarchy.
- (2) Associativity does not hold.
- (3) The second mapping function has different properties in forward and reverse directions.
- (4) Octal coding is necessary in programming.
- (5) When octal coding is used, Hybrid binary operation splits into bit by bit "and" and "exclusive-or" operations.
- (6) It is necessary to change the coding of "x" in the upper or "a" cube as the operators are changed.

6.1.2 RING STRUCTURE

Advantages:

- (1) It makes it possible to apply highly developed theory of the rings to the solution of algebraic topological problems.
- (2) Bit by bit "and" and "exclusive-or" binary operations used in this approach relate the unified theory to the Boolean Algebra and the two element Boolean Ring with unity.
- (3) Only the x co-ordinate of the "a" cube is operator dependent.
- (4) It uses quartal coding instead of octal, resulting in reduced storage space and execution time in the computer.
- (5) It is not necessary to change the coding of any element as the operators are changed, although their meaning may be interpreted differently.

Disadvantages:

- (1) Because of ring structure, the algebra associated with the algorithms of the operators is shown in two different tables.
- (2) It has three mapping functions. This makes the theory somewhat more complex.
- (3) As the theory becomes more complex, the programming demands higher skills.

6.1.3 FIELD STRUCTURE

Advantages:

- (1) The extreme simplicity of the field structure of Z_2 gives additional beauty and elegance to the unified theory.
- (2) It links the algebraic-topological operators with the basic concepts of logic, such as Boolean algebra, Boolean rings and $Z_2 = \{0, 1\}$ field.
- (3) It gives an insight to the algorithms used in the synthesis of the switching systems and provides additional visibility into unified theory.
- (4) It has all the advantages of the unified theory developed using a ring structure.

Disadvantages:

- (1) It requires a sign convention in addition to the binary coding to achieve one to one mapping.

- (2) Because of the fact that two bits are still required to represent an element, from the software point of view, the unified theory based on the field of Z_2 has no advantages over the theory developed using a ring structure.

6.2 CONCLUSIONS

The Groupoid approach, with its redundancies, results in less complex theory and provides greater ease in programming, whereas the Boolean Ring with its difficult and elegant theory bridges the gap between the unified theory and the Boolean algebra, and results in considerably more efficient software.

Unified theory, based on the field structure of Z_2 , provides additional visibility and insight to the algorithms used. Moreover, it links these algorithms on a sound mathematical basis with the basic concepts of logic.

CHAPTER VII

COMPUTER PROGRAMMES

7.1 COMPUTER PROGRAMMES BASED ON THE UNIFIED THEORY

The computer programmes, which will be described in this chapter, were built into the framework of the software complex developed by Professor D. L. Dietmeyer [27] [28] [29] .

Since the objectives of this thesis were to develop a unified theory for computer aided synthesis of digital systems, no attempts were made to duplicate the monumental work done by Professor Dietmeyer and his students, which represents many man-years of research. For this reason, considerable help was obtained from the subroutines developed at Wisconsin University for manipulating cubical arrays. Specifically the following subroutines were used:

- ERRORS, READAR, PRNTAR for the input/output
- COMBIN, COPYCUB for the book-keeping of arrays.

It has been noted that the programmes developed by Professor Dietmeyer had little resemblance to the algorithms presented in his book [4] and also shown in Tables VI, VII, VIII, IX of this thesis. On the other hand, there is almost one to one correspondence between the unified theory and the computer programmes which will be described in this chapter. Perhaps the two approaches can be best described as the explicit and implicit way of programming.

It will be shown that unified theory yields simplicity and greater ease of programming. Moreover the use of quartal coding results in reduced storage and execution time.

7.2 ASSIGNMENT OF OCTAL CODE WORDS

Each element of a cube was encoded by Professor D. L. Dietmeyer as follows:

$$0 = 001 = 1_8$$

$$1 = 110 = 6_8$$

$$x = 111 = 7_8$$

In addition, $0_8 = 000$ was assigned to " $\emptyset$ ". He also used the following special features of FORTRAN $\mathcal{V}$ extensively.

AND (a^1, b^1) : bit by bit "and" operation on a^1 and b^1 .

$\begin{array}{r} 0 = 001 \\ 1 = 110 \\ \hline \emptyset = 000 \end{array}$	$\begin{array}{r} 0 = 001 \\ x = 111 \\ \hline 0 = 001 \end{array}$	$\begin{array}{r} 1 = 110 \\ x = 111 \\ \hline 1 = 110 \end{array}$
---	---	---

OR (a^1, b^1) : bit by bit "or" operation on a^1 and b^1 .

$$\begin{array}{r} 0 = 001 \\ 1 = 110 \\ \hline 1 = 111 \end{array}$$

COMPL (a^1) : all bits of a^1 are complemented

$$\text{COMPL}(0) = \overline{0}\overline{0}\overline{1} = 110 = 1$$

Professor Dietmeyer used "AND" operation in subsume and intersection algorithms to generate a "Ø" since $0 \wedge 1 = \emptyset$. When a "Ø" occurred, it was located with the help of the mask $K1 = \{0\ 0\ 1, 0\ 0\ 1, 0\ 0\ 1, \dots, 0\ 0\ 1\}$ and using the following algorithm:

$$[(\bar{a}_i \wedge K1) \times 2] \wedge \bar{a}_i = \begin{cases} 0\ 0\ 0 = 0_8 & \text{for } a_i \neq 0_8 \text{ and } a_i \neq 4_8 \\ 0\ 1\ 0 = 2_8 & \text{for } a_i = 0_8 \text{ and } a_i = 4_8 \end{cases}$$

Since $4_8 = 1\ 0\ 0$ produced the same result as $0_8 = 0\ 0\ 0$, $4_8 = 1\ 0\ 0$ could not be assigned to another element. Also $2_8 = 0\ 1\ 0$ occurred every time there was a "Ø". Therefore, the assignment of $2_8 = 0\ 1\ 0$ was also restricted.

Professor D. L. Dietmeyer used $x = 7_8 = 1\ 1\ 1$ as a logic window and generated a new mask "I12"or"MV" = $\{1\ 1\ 1, 0\ 0\ 0, 0\ 0\ 0, \dots, 0\ 0\ 0\}$ in which the position of $1\ 1\ 1 = 7_8$ was indexed to 12 positions in a 36 bit word. From this, two more indexing masks were generated. Namely:

$$(MV) \wedge (K1) = M1 = \{0\ 0\ 1, 0\ 0\ 0, 0\ 0\ 0, 0\ 0\ 0 \dots, 0\ 0\ 0\}$$

$$\text{and } (MV) \wedge (K6) = M6 = \{1\ 1\ 0, 0\ 0\ 0, 0\ 0\ 0, 0\ 0\ 0 \dots, 0\ 0\ 0\}$$

These masks were used to locate the positions of 0's and 1's in a given column of a cubical array before the generalized consensus algorithm was applied. This was done by producing $a^n \wedge MV$ and then comparing the result with M1 and M6 masks. For example, if $a^n = \{1, 0, x, 0, 0\}$

a^n	=	1 1 0	0 0 1	1 1 1	0 0 1	0 0 1
MV	=	1 1 1	0 0 0	0 0 0	0 0 0	0 0 0
$\wedge$	$\wedge$					
$a^n \wedge MV$	=	1 1 0	0 0 0	0 0 0	0 0 0	0 0 0

This is compared with

M6 = 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0
 and M1 = 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0

In this algorithm $7_8 = 1 1 1$ simply acted as a logic window in "AND" operation.

Professor D. L. Dietmeyer used the same "112" or "MV" mask in locating the positions of x's in the upper or "a" cube. This was done by producing $a_i \wedge MV$, comparing the result with MV and indexing the logic window. The same technique was used for the lower or "b" cube. When, say on the n'th position, $a_n \wedge MV = MV$, i.e. there was an x, and $b_n \wedge MV \neq MV$, i.e. there was 0 or 1, the co-ordinate of the new cube was generated by producing:

$$(a_n \wedge MV) \wedge \text{COMPL}(b_n \wedge MV) = \bar{b}_n$$

It would appear that Professor D. L. Dietmeyer decided on the assignment of the octal code words for the following reasons:

(a) $1 1 0 \wedge 0 0 1 = 0 0 0$ since $0.1 = \emptyset$

This technique is used in subsume and intersection operations.

(b) Both $1 1 0$, $0 0 1$ and $1 1 1$ results in $0 0 0$ in $[(\bar{a}_i \wedge K1) \times 2] \wedge \bar{a}_i$ algorithm. However, this restricts the use of the octal words $4_8 = 1 0 0$ and $2_8 = 0 1 0$ for other purposes.

This technique is used to locate $\emptyset$'s in the star product operation.

- (c) $7_8 = 111$ acts like a logic window in "AND" operation.

This technique is used in generalized consensus and sharp algorithms.

- (d) $6_8 = 110$ and $1_8 = 001$ are the complements of each other.

This technique is used in the sharp algorithm.

In the assignment of octal code words for the unified theory, the following points were taken into consideration:

- (a) "Idempotent Scalar" semigroup, which represented subsume, intersection and star product operations, could only be produced by the following octal assignment using bit by bit "AND" operation:

$$\begin{aligned} i &= 001 = 1_8 \\ j &= 010 = 2_8 \\ k &= 100 = 4_8 \\ e &= 111 = 7_8 \\ \emptyset &= 000 = 0_8 \end{aligned}$$

- (b) While the first five rows and columns of the "Hybrid" groupoid were identical to the "Idempotent Scalar" semigroup, its sixth column and row was representing the sharp operator. Most of the binary operations involved in the sixth row and column can be achieved by assigning $3_8 = 011$ to K, and using bit by bit "exclusive-or" operation. For example:

$$\begin{array}{ccc} K = 011 & K = 011 & K = 011 \\ \oplus i = \oplus 001 & \oplus j = \oplus 010 & \oplus k = \oplus 100 \\ \hline j = 010 & i = 001 & e = 111 \end{array}$$

(c) $e = 111 = 7_8$ provided a logic window for the elements $\{\emptyset, i, j, k, K\}$ in "AND" operation. In addition, $K = 011 = 3_8$ provided a logic window for $i = 001 = 1_8$ and $j = 010 = 2_8$.

(d) In the sharp algorithm, it is required to distinguish between $K * i$, $K * j$ and $K * e$. This can be done with the octal code word assignments specified in (a) and (b) and using bit by bit "OR" operation. For example:

$$\begin{array}{rcl} K & = & 011 \\ \underline{v_i} & = & \underline{v_{001}} \\ K & = & 011 \end{array} \quad \begin{array}{rcl} K & = & 011 \\ \underline{v_j} & = & \underline{v_{010}} \\ K & = & 011 \end{array} \quad \begin{array}{rcl} K & = & 011 \\ \underline{v_e} & = & \underline{v_{111}} \\ e & = & 111 \end{array}$$

(e) If $k = 100$ could be declared redundant and e and K combined together, it would be possible to go from octal to quartal coding as follows:

<u>Element</u>	<u>Octal</u>	<u>Quartal</u>
i	001	01
j	010	10
K	011	11
e	111	11
$\emptyset$	000	00
k	100	---

The following is the topology of coding used by

Fig 4 Topology of Coding

7.3 SUBROUTINES FOR MANIPULATING CUBICAL ARRAYS

The following subroutines will be used extensively in manipulating cubical arrays [4]:

7.3.1 RHEAD

The first word of each cubical array written in the Univac 1106 FORTRAN V language is referred to as the "header". Using octal coding up to 12 cube elements may be placed in each of the 36-bit Univac 1106 words. With quartal coding, the maximum number of elements in a word may be increased to 18. The "header" consists of the following:

- (a) Key Field (0 - 5 th bit)
- (b) NWPC Field (6 - 9 th bit). This indicates the number of words used to store each cube of the logic array.
 $NWPC \leq 15$.
- (c) NV Field (10 - 17 th bit). This indicates the number of variables or elements in each cube. $NV \leq 256$.
- (d) NC Field (18 - 26 th bit). This indicates the number of cubes in each array. $NC \leq 512$.
- (e) NW Field (27 - 35 th bit). This indicates the number of words used to store the array and header. $NW \leq 512$.

The leftmost bit is the 0'th bit of a 36 bit Univac word. The subroutine RHEAD reads the "header" record for an array. If some field is not correct, the subroutine prints the error message: INVALID ARRAY HEADER RECORD and calls the ERRORS subroutine.

7.3.2 WHEAD

This subroutine writes the header record associated with each logic array.

7.3.3 ERRORS

This subroutine is called whenever the other subroutines encounter illegal data.

7.3.4 READAR CALL READAR (A, NV, MXA)

An array is transferred from punched cards to array A, with dimensioned size MXA. One cube must be punched per card. Only 0, 1, 2, X and D are taken as valid characters (2, X and D are all logically equivalent). Blank columns of punched cards are ignored. A card with other than one of these characters in the leftmost position is an end of array indicator. Only the NV leftmost characters are read from each card. MXA indicates the amount of storage available.

7.3.5 PRNTAR CALL PRNTAR (A, N)

Array A is printed with a blank placed between the N'th and (N + 1)th character. If N is outside the range $0 \leq N \leq NV$, where NV is the number of variables or elements of an array A, then N is taken as NV. If illegal characters are encountered in an array, they are printed as question marks (?). 0's, 1's and X's are printed in the N leftmost columns and 0's, 1's and -'s are printed in the right columns.

7.3.6 COMBINE A $\leftarrow$ AUB CALL COMBINE (A, B, MXB)

The unabsorbed union of arrays A and B replaces array A, if this union requires less than MXA storage locations, and if the arrays are of the same number of variables.

COPYAR A $\leftarrow$ B CALL COPYAR (A, B, MXA)

Array A is replaced with a copy of array B if array B occupies less than MXA locations.

7.3.7 CPYCUB B $\leftarrow$ aⁱ CALL CPYCUB (A, B, ICUBE)

MOVCUB B $\leftarrow$ aⁱ A $\leftarrow$ A - aⁱ CALL MOVCUB (A, B, ICUBE)

The ICUBE'th cube of an array A is copied as array B, or moved from array A to array B, if the cube existed in the array. Array B consists of a single cube in both cases.

7.4 A COMPUTER PROGRAMME USING QUARTAL CODING

After the second mapping, the cube elements are encoded in quartal as follows:

$$\emptyset = 00 \quad e = K = 11 \quad i = 01 \quad j = 10$$

Even the sharp operation is carried out using this encoding. Throughout the programme FLD function of FORTRAN V is used. FLD (i, j, V) is typed INTEGER whether it appears on the left or right of the equal sign in an arithmetic statement. In the statement:

$$\text{FLD (i, j, V) = e}$$

where i and j are integer expressions, V is INTEGER, REAL, or LOGICAL, and e is an arithmetic expression. e is evaluated and converted to type integer; the rightmost j bits of this converted expression are inserted into variable V starting at bit position i. In the statement

$$N = \text{Field (i, j, v)}$$

a field of j bits starting with the i'th bit is taken and right justified in word N with the left bits of N set to 0.

The listings of logic subroutines of this programme are given in Appendix A. An adequate number of comments have been provided between the statements so that there will not be any difficulties following it through. In the following sections, the highlights of each subroutine will be discussed. The remainder is self-explanatory.

7.4.1 SUBROUTINE SUBSUM (A)

This is a relatively simple subroutine involving a single array A. Some complication results from the fact that NWPC may be greater than one. For this the following variable L is introduced and used in DO loops as shown below:

```
L = NW - NWPC
DO 4 I = 2, L, NWPC
```

Hence the cube index I starting from the first cube in word 2 (first word is always the header record) goes to the last cube starting in word L in steps of NWPC.

The subsume operation is carried out in the following manner:

```
K = 0
IO(1) = 0
IO(2) = 0
1  M = AND (A (I + K), A(J + K))
   IF (M.NE.A (I + K)) IO(1) = 1
   IF (M.NE.A (J + K)) IO(2) = 1
   IF (IO(1) + IO(2) . EQ . 2) GO TO 2
   K = K + 1
   IF (K.LT.NWPC) GO TO 1
```

DO loops are arranged so that if $A(I) \not\subseteq A(J)$ and $A(J) \not\subseteq A(I)$ or if $A(J) \subseteq A(I)$, then J is indexed. If $A(I) \subseteq A(J)$, then I is indexed. K is used as an index when and'ing the cubes which are stored in more than one word. The subsumed cubes are removed from the array A by declaring $A(I) = 0$ or $A(J) = 0$

7.4.2 SUBROUTINE INTERX (A, B, C, MXC, NCC)

This subroutine is a little bit more involved, because it produces the intersections of two arrays A and B. The cube index IA or IB is related to the word index JA or JB (index of the first word minus one) as follows:

$$JA = IA * NWPC - NWPC + 1$$

$$JB = IB * NWPC - NWPC + 1$$

For example, if $NWPC = 2$, the 4th cube of array A starts at the 8th word, namely:

$$JA = 4 * 2 - 2 + 1 = 7 = 8 - 1$$

If $NWPC$ is greater than one, it is necessary to know the position of the first bit of the last byte in the last word of any cube. This is calculated as follows:

$$ILAST = (NV - 1 - (NWPC - 1) * I18) * I2$$

Where $I18$ = number of bytes in a word and $I2$ = number of bits in a byte.

For example if $NWPC = 2$ and $NV = 2 * 18 = 36$, then

$$(36 - 1 - 18) * 2 = 17 * 2 = 34$$

Hence the 34th bit is the position of the first bit of the last byte in the last (second) word. (The leftmost bit is the 0'th bit of the 36 bit Univac word.)

First of all, the following "and" operation is performed on the first words of IA and IB cubes

$$L = \text{AND} (A (N + JA), B (N + JB))$$

Before carrying out the "and" operation using the other words of the cubes, the result is checked for an empty intersection as follows:

$$\text{IF (FLD (I, 2, L).EQ.FLD (0, 2, 0)) GO TO 20}$$

DO loops are arranged so that if the intersection of the first words of the cubes is not empty, N is indexed and the "and" operation is carried out on the other words of the same cubes, one at a time. If there is still no intersection, COMBIN subroutine is called and the intersection cube, which is stored in

temporary array CC, is combined with C array. When the indexing of IB and IA is finished, the array C is absorbed by calling the SUBSUM(C) subroutine.

7.4.3 SUBROUTINE SHARPS (A, B, C, D, MXC, NCC)

This is the most complicated of all subroutines. Both the subsume and intersection operations are carried out before the sharp operation using the techniques described in the previous sections. This is due to the fact that if $a^i \subseteq b^i$, then $a^i \# b^i = \emptyset$ and if $a^i \cap b^i = \emptyset$, then $a^i \# b^i = a^i$. Because of this, it is desirable to use the same encoding throughout this subroutine.

The sharp operation is carried out by testing the elements of the cube A (N + JA) for K as follows:

```
IF ( (FLD (I, 2, A (N + JA)) . EQ . FLD (34, 2, 3) . AND.
FLD (I, 2, B (N + JB)) . EQ . FLD (34, 2, 3)) . OR . FLD
(I, 2, A (N + JA)) . NE . FLD (34, 2, 3)) GO TO 100
```

This expression is false only if A (N + JA) = K and B (N + JA) ≠ K. In this case a new D cube is generated as follows:

```
DO 300 NN= 1, NWPC
300 D (NN + 1 + K) = A (NN + JA)
IF (FLD (I, 2, B (N + JB)) . EQ . FLD (34, 2, 1)) GO TO 303
FLD (I, 2, D (N + 1 + K)) = FLD (34, 2, 1)
GO TO 302
303 FLD (I, 2, D (N + 1 + K)) = FLD (34, 2, 2)
302 K = K + NWPC
100 CONTINUE
```

In other words, when a K is detected in A (N + JA) cube, the following binary operations are carried out on the A (N + JA) and B (N + JB) cubes:

$$K * i = j \quad \text{and} \quad K * j = i$$

Then K is indexed and the next byte of A (N + JA) is tested.

7.4.4 SUBROUTINE CNSSUS (A, B, C, MXA, NCC)

In star product operation, one $\emptyset$ is allowed. Because of this, a $\emptyset$ counter is required. This is provided as follows:

```
NPFI = 0
DO 50 N = 1, NWPC
L = AND (A (N + JII), A (N + JJJ))
IF (N . EQ . NWPC) GO TO 60
IMAX = 34
GO TO 70
60  IMAX = ILAST
70  DO 80 I = 0, IMAX, 2
    IF (FLD (I, 2, L) . NE . FLD (0, 2, 0)) GO TO 80
    NPFI = NPFI + 1
    IF (NPFI . GT . 1) GO TO 10
```

Then $\emptyset$ is replaced by $e = 1\ 1 = 3_4$ as follows:

```
FLD (I, 2, L) = FLD (34, 2, 3)
80  CONTINUE
    B (N + 1) = L
50  CONTINUE
    CALL COMBIN (A, B, MXA)
10  CONTINUE
    CALL SUBSUM(A)
```

7.5 TWO COMPUTER PROGRAMMES USING OCTAL CODING

After the second mapping, the cube elements are encoded in octal as follows:

$\emptyset = 0\ 0\ 0$ $e = 1\ 1\ 1$ $K = 0\ 1\ 1$ $i = 0\ 0\ 1$ $j = 0\ 1\ 0$

Two different programmes are presented in Appendix B and C. The first one is almost identical to the one presented in section 7.4 except that it uses octal coding.

The second one presented in Appendix C differs only in the execution of the sharp operation. In the previous two programmes, there was no change made in encoding when moving from subsume or intersection into the sharp operation. In this programme, $e = 111 = 7_8$ will be changed to $K = 011$ in IATH cube so that $K * i$ and $K * j$ can be distinguished from $K * e$ using "OR" operation. For example,

$\begin{array}{r} K = 011 \\ i = 001 \\ \hline \end{array}$	$\begin{array}{r} K = 011 \\ j = 010 \\ \hline \end{array}$	$\begin{array}{r} K = 011 \\ e = 111 \\ \hline \end{array}$
$K = 011$	$K = 011$	$e = 111$

The conversion is done as follows:

IF (FLD (I, 3, A (NENC + JA)) . EQ . FLD (33, 3, 7)) FLD (1, 3, A (NENC + JA)) = FLD (33, 3, 3)

Then A (NENC + JA) "OR" B (NENC + JB) is computed and stored in AORB (NENC) as follows:

AORB (NENC) = OR (A (NENC + JA), B (NENC + JB))

Then AORB is scanned and a new cube is produced only if the byte in position I is a $K = 011$. This is done as follows:

IF (FLD (I, 3, AORB(N)) . NE . FLD (33, 3, 3) GO TO 100

IF (FLD (I, 3, B (N + JB)) . EQ . FLD (33, 3, 1) GO TO 303

FLD (I, 3, D (N + 1 + K)) = FLD (33, 3, 1)

GO TO 302

303 FLD (I, 3, D (N + 1 + K)) = FLD (33, 3, 2)

302 K = K + NWPC

100 CONTINUE

In order to satisfy the convention adopted by PRINTAR, $K = 011 = 3_8$ must now be changed back to $e = 111 = 7_8$ as follows:

IF (FLD (I, 3, D(N)) . EQ . FLD (33, 3, 3)) FLD (I, 3, D(N)) = FLD (33, 3, 7).

7.6 A COMPUTER PROGRAMME USING INTERRELATED SUBROUTINES

In this programme, the encoding of a pair of cubes a^i and b^i is executed using a subroutine called ENC. This subroutine realizes the second mapping function $\psi : Q \rightarrow U$ with the following octal coding:

$\emptyset = 000$, $i = 001$, $j = 010$, $K = 011$, $e = 111$, $k = 100$

The binary operation on these cubes $c^i = a^i * b^i$ and subsequent inverse mapping $\psi^{-1} : U \rightarrow Q$ is executed by a subroutine called OPDEC. The switch variable ISW1 is used to identify each logic operator as follows:

ISW1 = 1	Subsume Operator
ISW1 = 2	Cube Intersection Operator
ISW1 = 3	Star Product Operator
ISW1 = 4	Sharp Operator

Another switch variable ISW2 is used to identify the upper and lower cubes as follows:

ISW2 = 1	Upper Cube
ISW2 = 2	Lower Cube

These two subroutines ENC and OPDEC are called repeatedly by the individual subroutines. Whenever it is convenient and unambiguous, the octal code assignments of the cube elements are left unchanged in U domain increasing the effectiveness of this software approach.

Since the subroutines of subsume, intersection, star product and sharp are now all interrelated through the repeated use of ENC and OPDEC, they are considerably simpler and shorter.

The same cube manipulation techniques described in the previous sections of this chapter have also been used in this programme.

CHAPTER VIII

ALGORITHMIC EFFICIENCY

8.1 IMPLEMENTATION AND MEASUREMENT OF ALGORITHMIC EFFICIENCY

In chapter VII, four different computer programmes based on the unified theory and written in the Univac 1106 FORTRAN V language were discussed. The listings of these programmes were given in Appendix A,B,C,and D. These programmes represent the evolution of a set of algorithms starting with the earliest version given in Appendix D, which used octal coding and interrelated subroutines, to the latest ones given in Appendix A using quartal coding and independent subroutines. In the forthcoming discussions these programmes will be referred to as programme A, programme B, etc according to their respective appendices.

In section 7.2 the algorithms used by Professor D.L. Dietmeyer [27] [28] [29] were also described. This is an example of where several algorithms may exist for the same problem. In such cases one must decide on the relative merits of each approach, thus leading into the field of "algorithmic analysis"[30].

In comparing the algorithms used by Professor Dietmeyer and the ones evolved from the unified theory of array operators, let us first use the five important guidelines suggested by D.E. Knuth[30] of the California Institute of Technology namely:

1-) Finiteness

An algorithm must always terminate after a finite number of steps .

2-) Definiteness

Each step of the algorithm must be precisely defined.

3-) Input

An algorithm has zero or more inputs .

4-) Output

An algorithm has one or more outputs which have a specified relation

to the inputs.

5-) Effectiveness

An algorithm is also generally expected to be "effective". This means that all of the operations to be performed in the algorithm must be sufficiently basic so that they can in principle be done exactly and in a finite length of time by a man using a paper and pencil.

With respect to finiteness, definiteness, input and output both algorithms are approximately equivalent, since after a finite number of steps, for a given example, both lead to a unique and identical result. The effectiveness, however, is increased considerably in the new approach, since it consists of a unified treatment of the array operators based on a four element Boolean ring using $\oplus$ and $\wedge$ operations instead of many ingenious software techniques. This improvement in effectiveness yields simplicity and greater ease in programming. Hence it reduces the demands made on the programmer's special skills, experience, and inventiveness.

Let us next carry out the "algorithmic analysis" by reviewing and discussing the following performance characteristics of each programme:

- a-) Memory Storage Allocation
- b-) Linkages of Subroutines
- c-) Execution Time

In programme A the total memory space has been reduced by approximately 22% over Professor Dietmeyer's programme for the four basic array operations, namely consensus, sharps, intersection and subsume. This reduction has been made possible by

- a-) Using a set of algorithms evolved from the unified theory (12.5%)
- b-) Adapting quartal coding instead of octal (6.5%)
- c-) Calling the existing subroutines and entries whenever possible (3%)

The subroutines are very useful in saving space in a programme. They do not save any time other than the time implicitly saved by having less space (e.g. less time to load the programme, or fewer passes necessary over the programme.) Subroutines have several other advantages. They make it easier to visualize the structure of a complex programme; they form a logical segmentation of the entire problem and this usually makes debugging of the programme easier.

Although the frequent calling of the subroutines is often desirable to minimize the memory storage space, this is always done at the expense of execution time. Another approach is to have a trade off between the storage space and the execution time by building redundancies into the programme and thereby reducing the number of times certain subroutines or enteries are called. This latter approach will be called logical redundancy.

In programmes written for array operations, where large number of variables are exchanged between subroutines, an increase in storage space of 5-6% due to logical redundancy can result typically in a 20 - 25% improvement in execution time.

This is the approach taken by Professor Dietmeyer in his sharp subroutine. In other words instead of reducing his memory storage to a minimum, he traded off storage space for execution time. As a result he achieved in the sharp operation a 34.8% improvement in execution time over programme A.

The recent price reductions in dynamic random access memories using silicon gate technology will undoubtedly make Professor Dietmeyer's approach more attractive. On the other hand from the data, which is presented in the sections 8.2, 8.3 and 8.4, one can predict that in programme A by creating appropriate redundancies using 3 - 5% additional storage space, an overall improvement of 15-20% in execution times can be accomplished. This will still result in a

17 - 19% reduction in memory storage space over Professor Dietmeyer's programme, while achieving as good or better execution times.

The use of quartal coding instead of octal reduces the number of binary bits handled by the computer in the loading and unloading of the data by 33.3%. Therefore a similar improvement in execution time of Input/Output processing can be expected. However, when writing a programme in a high level language such as FORTRAN it is not possible to predict how much improvement, if any, will result from the use of quartal coding when executing a particular operation. Any improvement in execution time will depend on the compiler and the machine architecture.

8.2 MEMORY STORAGE ALLOCATION

Table XXXIII gives the memory storage allocations of programmes A,B,C, and D and shows the evolution of programme A from programme D. As can be seen the total reduction in storage space is 41.5% or 1127_{10} words.

Programme D had two additional subroutines, namely OPDEC (633_{10}) and ENC (148_{10}). The array operators consensus, sharp, intersection, and subsume were all inter-related through these two subroutines. This early approach resulted in extremely simple programmes for the array operators. However the total storage space, linkages of subroutines, and the execution times were all high in comparison with the programmes written by Professor Dietmeyer.

The programmes B and C are almost identical. They both have independent subroutines for the array operators and use octal coding. Programme C differs only in the execution of the sharp operation. In this earlier approach, after executing the subsume or intersection operation the encoding of K was changed from 7_8 to 3_8 before moving into sharp, and changed back again at the end of this operation. These additional steps were removed in programme B and a saving of 92_{10} words was realized.

Type of Programm	A			B			C			D		
	(Quartal Coding)			(Octal Coding)			(Octal with modified Sharp)			(Octal with interrelated Subroutines)		
Type of Storage	Code	data	total	code	data	total	code	data	total	code	data	total
Main Program	130	362	492	130	494	624	129	493	622	141	693	834
Cnssus	213	41	254	214	41	255	213	40	253	152	222	374
Sharps	383	52	435	383	52	435	466	69	535	185	117	302
Interx	166	40	206	167	40	207	166	39	205	149	29	178
Subsum	179	22	201	179	22	201	178	21	199	218	28	246
Opdec	///	///	///	///	///	///	///	///	///	571	62	633
Enc	///	///	///	///	///	///	///	///	///	123	25	148
Totals	1071	517	1588	1073	649	1722	1152	662	1814	1539	1176	2715

Table XXXIII
Evolution of Programme A

The programme A differs from B only in the coding scheme. It uses quartal instead of octal coding. This results in a saving of 134_{10} words. The reason for this is as follows: Each Univac word in programme A can store up to $36/2 = 18$ variables in comparison to $36/3 = 12$ in programme B. In other words with quartal coding it is possible to accomodate 50% more variables in a 36 bit word.

In the main programme of B the data storage of the arrays is set up as follows:

A (100), B (100), C (100), D (100).

The first word of each array is the header. This leaves room for $99 \times 12 = 1188$ variables. Using quartal coding the same number of variables can be stored in 66 words, since $66 \times 18 = 1188$. Thus allowing one additional word for the header, the data storage of the arrays in programme A is set up as follows:

A (67), B (67), C (67), D (67)

Of course, if each cube contained twelve or less variables, i.e. $NWPC = 1$, then the programme B can still handle 99 cubes whereas programme A is limited to 66. Fortunately this situation does not happen in practice. Usually if $NWPC = 1$ the number of cubes in an array is considerably less than 66. As the combinatorial problems become more complex $NWPC$ increases rather than the number of cubes in an array.

Table XXXIV gives the memory storage allocations of the "system" and "bookkeeping" subroutines. With some minor difference these subroutines are identical with the ones used by Professor Dietmeyer. Note that Cpycub subroutine is only used in programme D.

Subroutine	Code	Data	Total
Prntar	135	24	159
Readar	208	45	253
Errors	134	19	153
Copycub	145	24	169
Combin	146	29	175
Totals	768	141	909

Table XXXIV

System and Bookkeeping Subroutines

Table XXXV compares the memory storage allocation of programme A with Professor Dietmeyer's programme. In order to make the comparison realistic the total storage space of Professor Dietmeyer's main programme has been reduced from 730_{10} to 629_{10} words. This was done because his main programme called three additional subroutines, namely factor product, cube union, and split. The figure 629_{10} is obtained by adding five more words for the mask array to the storage allocation of programme B, which uses octal coding and calls the absorb, consensus, intersection and sharp subroutines.

The only other change which has been made to the storage allocation of Professor Dietmeyer's programme is in the subsume subroutine. Here it was discovered that the compiler of Univac 1106 was unable to differentiate between $+0 = 000...0_8$ and $-0 = 777...7_8$. In order to overcome this problem the statement

IF (A (I) . EQ.0) GO TO 6

in Professor Dietmeyers' programme has been replaced by the following statement.

IF (FLD (0, 18, A(I)). EQ . FLD (0, 18, 0). AND. FLD (17, 18, A(I)). EQ.
\$FLD (0, 18, 0)) GO TO 6

Table XXXV shows that a 22.2% reduction in storage space has been accomplished in programme A over Professor Dietmeyer's programme for the four basic array operators. When the "system" and "bookkeeping" subroutines, which are common to both programmes, are added the total overall improvement is 16.3%.

8.3 LINKAGES OF SUBROUTINES

Tables XXXVI and XXXVII show the linkages of the subroutines in programme A and in the programme written by Professor Dietmeyer. In addition these tables give the total code (programme) memory space in words travelled during the execution of any subroutine. For example the sharp operator in programme A has 383_{10} code words allocated to it. During its execution it calls the rhead entry twice, whead entry five times, combin subroutine once,

TYPE OF PROGRAMME		PROGRAMME A				DIETMEYER			
TYPE OF STORAGE	CODE	% Δ	DATA	% Δ	TOTAL	% Δ	CODE	DATA	TOTAL
MAIN P.	130	0%	362	+27.4%	492	+21.8%	130	499	629
CNSSUS	213	+48%	41	+14.6%	254	+44.3%	408	48	456
SHARPS	383	+16.4%	52	+ 1.9%	435	+14.9%	458	53	511
INTERX	166	+21.6%	40	-25%	206	+15.6%	212	32	244
SUBSUM	179	0%	22	0%	201	0%	179	22	201
SUB TOTAL	1071	+22.8%	517	+20.9%	1588	+22.2%	1387	654	2041
PRNTAR	135	0%	24	0%	159	0%	135	24	159
READAR	208	+0.48%	45	0%	253	+0.39%	209	45	254
ERRORS	134	0%	19	0%	153	0%	134	19	153
COMBIN	146	0%	29	0%	175	0%	146	29	175
SUB TOTAL	623	+0.16%	117	0%	740	+0.13%	624	117	741
TOTAL	1694	+15.8%	634	+17.8%	2328	+16.3%	2011	771	2782

Table XXXV

Comparison of Memory Storage Allocations

copyar entry twice, subsum subroutine twice. In turn combin subroutine calls the errors subroutine twice, rhead entry three times and whead entry twice, etc, etc. The total number of code memory space thus travelled during the execution of sharp operation adds up to 2196_{10} words.

Similarly the sharp subroutine in Professor Dietmeyer's programme has 458_{10} code words allocated to it. However, the total number of code words travelled during its execution is only 1635_{10} words.

Comparing the two sharp subroutines it can be seen that the one in programme A uses 16.4% less code words. However, during the execution of the same subroutine 34.3% more code words are travelled.

In Professor Dietmeyer's sharp subroutine two calls to the combin subroutine and one call to copyar entry were avoided using the following DO loops:

```
      DO 14 I = L, IFF
      NWF = NWF + 1
14   F (NWF) = F(I)
      -----
16   DO 17 I = 2, NWF
      C (NWC) = F(I)
17   NWC = NWC + 1
      -----
      DO 22 I = 1, NWA
22   C(I) = A(I)
      -----
```

This is an excellent example of building logical redundancy to reduce execution time. The DO loops increased the code storage of the subroutine by approximately 36_{10} words or 8.3%, while reducing the code words travelled by 1415_{10} . However, one additional call to each of the subroutines errors and subsum reduced the margin gained by the three DO loops to 561_{10} words. This represents an

Subroutines	Code Words	134 Errors	Number of Calls				251 Copyar	247 Subsum	Total Code Words Travelled	% change over Dietmeyer
			34 Rhead	34 Whead	582 Combin	34				
Cnssus	213		2	1	1		1	1	1393	-22.6%
Sharps	383		2	5	1		2	2	2196	-34.3%
Interx	166		2	2	1			1	1129	- 1.9%
Subsum	179		1	1					247	0%
Prtar	135		1						169	0%
Readar	208			2					276	+0.362%
Cpycub	145		1	3					279	0%
Combin	146	2	3	2					582	0%
Copyar	49	1	1	1					251	0%
Errors	134	1							134	0%

Table XXXVI

Linkages of Subroutines in Programme A

Subroutines	Code Words	Number of Calls					Total Code Words Travelled
		134 Errors	34 Rhead	34 Whead	582 Combin	251 Copyar	
Cnssus	408	1	1	2		2	1136
Sharps	458	2	2	3		3	1635
Interx	212	2	2	2		2	1108
Subsum	179		1	1			247
Prntar	135		1				169
Readar	209			2			277
Cpycub	145		1	3			279
Combin	146	2	3	2			582
Copyar	49	1	1	1			251
Errors	134	1					134

Table XXXVII

Linkages of Subroutines in Dietmeyer's Programme

improvement of 34.3% over programme A. As will be pointed out in the next section, there is a definite relationship between the code words travelled and the execution time.

8.4 EXECUTION TIME

Table XXXVIII compares the C.P.U. times and C.P.U. costs obtained in the execution of three identical Absorb, Intersection and Sharp operations by programme A, programme B and the programme written by Professor Dietmeyer.

The absorb operation has been carried out on an array having 20 cubes with 9 variables each. The resultant array consisted of 15 cubes.

The intersection operation was carried out on two arrays having 10 cubes each with nine variables. The resultant array consisted of 21 cubes.

The sharp operation was carried out on two arrays having 10 cubes each with 9 variables. The resultant array consisted of 39 cubes.

It was found that the C.P.U. costs in cents were not accurate enough to be used for comparison purposes. The operations cost either 9 cents or 12 cents regardless of what they were.

Difficulties were also experienced in the measurement of C.P.U. times. For example, when a particular programme was run consecutively, it gave execution times which varied $\pm 5\%$ from the average value which was in the range of 0.5 - 1.25 seconds.

Table XXXVIII also compares the performance of different programmes using words/seconds (W/S). The words represent the code storage space travelled during the execution of a subroutine. These words were given in Tables XXXVI and XXXVII.

	Absorb	Interx	Sharp
Dietmeyer (Octal)	0.545 secs 455 w/s	0.755 secs. 1480 w/s	0.910 secs. 1800 w/s
Program A (Quartal)	0.554 secs. 454 w/s	0.653 secs. 1730 w/s	1.227 secs. 1790 w.s
Program B (Octal)	0.738 secs. 355 w/s	0.707 secs. 1600 w/s	1.170 sécs. 1880 w/s
Improvements over Dietmeyer	0% in secs. 0% in w/s	+13.5% in secs. +14.5% in w/s	-34.8% in secs. -0.55% in w/s
Improvement over Program B	+24.5% in secs. +27.9% in w/s	+7.65% in secs. +8.12% in w/s	-4.85% in secs. -5.05% in w/s

Table XXXVIII

Comparison of Execution Times

Comparing the programmes A and B the following conclusions can be drawn:

- a-) There is an excellent correlation between the execution times and words/second. This is expected since both programmes are identical except for the coding scheme.
- b-) Quartal coding results in an improvement of approximately $25 \pm 5\%$ in the Absorb operation, and $8 \pm 5\%$ in the Intersection operation. It is expected that this improvement occurs mainly in the I/O processing. As the programme size increases from 247_{10} code words in Absorb to 1129_{10} in Intersection the overall improvement is reduced from $25 \pm 5\%$ to $8 \pm 5\%$. Finally in the Sharp operation with 2196 code words the improvement is lost in the measurement error and we have $-5 \pm 5\%$.

Comparing the programme A with the programme written by Professor Dietmeyer the following conclusion can be drawn:

- a-) The absorb operation shows no improvement. With $\pm 5\%$ fluctuation in the execution time either programme can be as much as 10% faster than the other. The anticipated improvement due to quartal coding is lost in the error.
- b-) Programme A shows an improvement of approximately $14 \pm 5\%$ over the programme written by Professor D.L. Dietmeyer in the Intersection operation. Since the total code words of both programmes only differ by 1.9% this improvement is due to a more efficient algorithm used in programme A.
- c-) The sharp subroutine written by Professor Dietmeyer shows an improvement of $34.8 \pm 5\%$ in seconds but only 0.55% in words/seconds over programme A. Since the total code words travelled of both subroutines differ by 34.3% it is believed that the improvement in execution time is due to the logical redundancies built into the programme which used additional 8% storage space. As was pointed out in section 8.3, the same technique can be applied to the sharp subroutine of programme A to improve its execution time.

8.5 CONCLUSIONS:

The Unified theory improves the algorithmic efficiency in the following areas:

- a-) Increased effectiveness through the introduction of the ring structure yields simplicity and greater ease in programming.
- b-) The elements of the Klein-4 group, introduced as quartal coding, allow a 36 bit Univac word to accomodate 50% more variables than was possible with octal coding.
- c-) The total storage space required by the four basic array operations has been reduced by 22% in comparison with the state of the art.
- d-) Comparable or better execution times have been achieved for all array operations except sharps. It has been shown, however, that by creating of some redundancy in the sharp subroutine, it is possible to bring its execution time in line with the state of the art.

CHAPTER IX

CLOSING REMARKS

9.1 SUMMARY OF THE WORK PRESENTED IN THE THESIS

The algebraic topological methods, first proposed by Roth [16] in 1958, came into prominence in the mid sixties and have provided the basis for the subsequent development of algorithms for the computer aided synthesis of combinatorial networks.

The cubical complex and a set of array operators defined by Roth permitted a direct visualization of Boolean algebra and a clearer picture of the process of synthesis. However his array operators and the resulting algebra remained unstructured and isolated from any encompassing body of mathematics.

This thesis presented a theory which continued the array modelling started by Roth. By a further sequence of mappings the unified theory yielded

a-) Abelian groupoid of order six having an identity element.

b-) Four element Boolean Ring with unity.

c-) Field of Z_2

and related the algebraic topological methods back into the basic concepts of logic such as Boolean algebra, Boolean rings, and the Z_2 field.

The unified theory has been developed using various mapping functions. These have been summarized in Fig. 6. While Roth only used the $Q = \{0, 1, x\}$ space, the unified theory used the U space for the groupoid, B space for the ring and Z_2 space for the field structure.

As the proposed structure is moved up in the algebraic hierarchy from a groupoid to a field the number of mapping functions is increased from two to five. In chapter VI different merits of these structures have been discussed in detail. In summary, it can be stated that the Boolean ring is the one which is most suited for the computer aided synthesis of switching systems. Moreover the elements of the Klein-4 group (additive part of the ring) are used as quartal coding in B space. The resulting simplifications

Spaces of the Unified Theory

ROTH's Space

Fig. 6

Summary of Mapping Functions

in array coding and processing are then presented by means of four sets of computer programmes in chapter VII. The improvements in algorithmic efficiency have been discussed in chapter VIII.

9.2 VIEWS ON THE FUTURE WORK

The way is now clear for the professional programmers to become familiar with the results of the unified theory and apply their skills and ingenuity to create more efficient and more general software systems for the synthesis of the combinatorial networks.

As far as the application of the unified theory is concerned the surface is barely scratched. The journey from Roth's Q space to our B space made it possible to apply the well developed theory of the rings to the synthesis of switching systems.

Without abandoning the cubical complex of Roth, future researchers will be thinking and working in B space using the theory of groups, rings and fields. It is expected that a number of new array operators will be discovered in B space. These will probably have different algebraic structures with different complexities. However, they will be linked together and produce considerably simpler algorithms. The elements of these algebraic structures will provide optimized encoding schemes for the computer programmes. The problem of finding prime implicants, extremals, the minimization of multiple output switching functions and multilevel network synthesis will be done much more efficiently in B space, or in some more advanced space, using modern algebra.

The unified theory with its mapping functions, special spaces and algebraic structures is somewhat analogous to the application of quantum mechanics to the crystal structure of matter using Brillion zones, momentum spaces, transformations and group theory.

Roth's topological approach provides as clear a picture of the cubical complex as the one provided for the crystal lattice by X-ray or electron diffraction procedures. However, it does not yield simplicity in solving logic problems comparable to that allowed by the group theory in determining the movement of wave packets in crystal.

The theory presented in this thesis united the array operators and removed the mystery from their algorithms by using modern algebra while retaining the clear view provided by Roth's topological approach. However, this is only the beginning of an exciting era yet to come in the logic design of digital systems.

The future of large scale automated design of complex digital systems will become inextricably bound to the concepts of, and insights provided by the modern algebra.

APPENDIX A

A COMPUTER PROGRAMME USING QUARTAL CODING

-IR M E ULUG-PH D THESIS-CARLETON UNIVERSITY-

```

JEL      3
          INTEGER A( 67 ),B( 67 ),C( 67 ),D( 67 )
C
C      THIS PROGRAM IS AN EXAMPLE ILLUSTRATING THE USE OF THE MAIN LOGIC
C      OPERATORS. A VALUE FROM 1 TO 4 IS ASSIGNED TO ISW1,I.E.:
C
C          ISW1 = 1 ABSORB                      - UNARY OPERATOR
C
C          = 2 CONSENSUS                      - UNARY OPERATOR
C
C          = 3 CUBE INTERSECTION              - BINARY OPERATOR
C
C          = 4 SHARP PRODUCT                  - BINARY OPERATOR
C
C      READ INPUT PARAMETERS
C
C          N      = NUMBER OF INPUT VARIABLES
C
C          M      = NUMBER OF OUTPUT VARIABLES
C
C          NV     = TOTAL NUMBER OF VARIABLES
C
C          ISW1   = CODE OF THE LOGIC OPERATORS
C
C      READ 1,N,M,ISW1
C      1      FORMAT(3I2)
C          MX    = DIMENSIONED SIZE OF ARRAY A,B,C,D
C
C          MX=100
C          NV=N+M
C          PRINT 2,N,M,ISW1,NV,MX
C      2      FORMAT('1PROGRAM LOGOP IS A DEMONSTRATION PROGRAM OF THE LOGIC OPE
$RATORS'// ' INPUT PARAMETERS : ' N = ',I2,' M = ',I2,' ISW1 = ',I2
$, ' NV = ',I2,' MX = ',I4// ' INPUT A - ARRAY'//)
C
C      ARRAY A IS READ BY READAR IN ORDER TO BE OPERATED BY AN UNARY
C      OPERATOR (ISW1 = 1 TO 2) OR TO BE OPERATED AS THE LEFT SIDE ARRAY
C      BY A BINARY OPERATOR (ISW1 = 3 TO 4).ARRAY A IS PRINTED BY PRNTAR
C
C          CALL READAR(A,NV,MX)
C          CALL PRNTAR(A,N)
C
C      ACCORDING TO ISW1 ,THE RIGHT SIDE ARRAY B OPERATED BY A BINARY
C      OPERATOR IS READ BY READAR AND PRINTED BY PRNTAR
C          IF(ISW1.LE.2) GO TO 3
C          PRINT 13
C      13      FORMAT('/' INPUT B - ARRAY'//)
C          CALL READAR(B,NV,MX)
C          CALL PRNTAR(B,N)
C          GO TO(4,5,8,9),ISW1
C
C      ARRAY A IS ABSORBED AND REPLACED IN A ' A <-- A(A) '
C
C      4      CALL SUBSUM(A)
C          PRINT 15

```

-MR M E ULUG-PH D THESIS-CARLETON UNIVERSITY-

```

15  FORMAT(/' ABSORB - UNARY OPERATOR : A <-- A(A)'/
    $' ABSORBED ARRAY A'/)
    GO TO 16

C
C  THE CUBES OF ARRAY A ARE CONVERTED INTO PRIME IMPLICANTS USING
C  THE ITERATIVE CONSENSUS ALGORITHM, ASSUMING BARTEE CODING, I.E.
C  FOR M>L EACH '1' IN THE M-UPLES PART OF EACH CUBE OF A FUNCTION
C  ARRAY IS REPLACED WITH AN 'X' ' A <-- *(A) '
C
5   CALL CNSSUS(A, B, C, MX, NV)
    PRINT 17
17  FORMAT(/' CONSENSUS - UNARY OPERATOR : A <-- *(A)'/
    $' PRIME IMPLICANTS ARRAY'/)
    GO TO 16

C
C  THE INTERSECTION OF ARRAYS A AND B IS PLACED IN ARRAY C ' C <-- A\B
C  NCC RETURNS THE NUMBER OF CUBES IN ARRAY C
C
8   CALL INTERX(A, B, C, MX, NCC)
    PRINT 20, NCC
20  FORMAT(/' CUBE INTERSECTION - BINARY OPERATOR : C <-- A\B'/
    $' NCC = ', I3/)
    GO TO 24

C
C  THE SHARP PRODUCT OF ARRAY A AND B IS PLACED IN ARRAY C ' C <-- A#B '
C  NCC RETURNS THE NUMBER OF CUBES IN ARRAY C
C
9   CALL SHARPS(A, B, C, D, MX, NCC)
    PRINT 26, NCC
26  FORMAT(/' SHARP PRODUCT - BINARY OPERATOR : C <-- A#B'/
    $' NCC = ', I3/)
    GO TO 24
16  CALL PRNTAR(A, N)
    GO TO 23
24  CALL PRNTAR(C, N)
23  CALL EXIT
    END

```

-MR M F ULUG-PH D THESIS-CARLETON UNIVERSITY-

```

/EL      3
SUBROUTINE CNSSUS(A, B, C, MXA, NCC)
COMMON /QZ/I2,I6,I18,I23,I26,IBK,K1,K6,K7,KEY,IPCL(8),IRCL(7),
1'MASK(12),I,J,K,L,M,N,NWPC,NV,NC,NW,IA,IB,IC,NWA,NWB,NWC,MSK,IO(80)
INTEGER A(1),B(1),C(1)
C
C      TEST IF A IS EMPTY THEN RETURN, OTHERWISE SET NCA
C      (NCA = NUMBER OF CUBES IN A )
C
C      CALL RHEAD(A)
C      IF(NC.LE.1) RETURN
C
C      INITIALIZE D THE TEMPORARY ARRAY
C
C      NCA=NC
C      NC=1
C      CALL WHEAD(B)
C
C      ILAST = THE POSITION OF THE FIRST BIT OF THE LAST BYTE
C
C      ILAST=(NV-1-(NWPC-1)*I18)*I2
C      NCAM1=NCA-1
C
C      DO LOOP 10 THE CUBES OF A
C      (II = INDEX OF THE CUBES OF A FOR 1.LE.II.LE.NCAM1)
C      (JJ = INDEX OF THE CUBES OF A FOR NN.LE.JJ.LE.NCA)
C
C      DO 10 II=1,NCAM1
C
C      JII = INDEX OF THE FIRST WORD MINUS ONE OF THE II TH CUBE OF A
C
C      JII=II*NWPC-NWPC+1
C      NN=II+1
C      DO 10 JJ=NN,NCA
C
C      JJJ = INDEX OF THE FIRST WORD MINUS ONE OF THE JJ TH CUBE OF A
C
C      JJJ=JJ*NWPC-NWPC+1
C
C      INITIALIZE NPHI, THE COUNTER OF THE OCCURENCE OF PHI
C
C      NPHI=0
C
C      DO LOOP 50 CONTROL THE CONSENSUS ACCORDING TO THE FOLLOWING
C      BINARY OPERATIONS I*J=PHI J*I=PHI ETC...
C      CONVENTION K IS 3 QUA. IN CNSSUS, THEREFORE NO DECODING OR
C      ENCODING OF K IS NEEDED
C
C      DO 50 N=1,NWPC
C      L=AND(A(N+JII),A(N+JJJ))
C      IF(N.EQ.NWPC) GO TO 60
C      IMAX=34
C      GO TO 70
C      IMAX=ILAST
C      DO 80 I=0,IMAX,2

```


-MR M E ULUG-PH D THESIS-CARLETON UNIVERSITY-

```
C  TEST IF A PHI WAS ENCOUNTEED AND INCREMENT NPHI BY ONE, IF YES
C
      IF(FLD(I,2,L).NE.FLD(0,2,0)) GO TO 80
      NPHI=NPHI+1
C
C  IF NPHI.GT.1 PICKS THE NEXT CUBE, OTHERWISE REPLACE PHI BY 3 QUA.
C
      IF(NPHI.GT.1) GO TO 10
      FLD(I,2,L)=FLD(34,2,3)
80  CONTINUE
      R(N+1)=L
50  CONTINUE
C
C  COMBIN THE TEMPORARY CUBE AT THE BOTTOM END OF ARRAY A
C
      CALL COMBIN(A, B, MXA)
10  CONTINUE
      CALL SUBSUM(A)
      CALL RHEAD(A)
      NCA=NC
47
C
C  DO LOOP 30 TEST IF NEW CONSENSUS TERMS WERE FOUND IN THE
C  LAST ITERATION, IF NOT THEN TERMINATE
C
      DO 30 KK=1,NW
      IF(A(KK).NE.C(KK)) GO TO 40
30  CONTINUE
      RETURN
40  CALL COPYAR(C, A, MXA)
      GO TO 20
      END
```

-R E ULUG-PH D THESIS-CARLETON UNIVERSITY-

```

3
SUBROUTINE SHARPS(A,B,C,D,MXC,NCC)
COMMON /QZ/I2,I6,I18,I23,I26,IBK,K1,K6,K7,KEY,IPCL(8),IRCL(7),
1MASK(12),I,J,K,L,M,N,NWPC,NV,NC,NW,IA,IB,IC,NWA,NWB,NWC,MSK,IO(80)
INTEGER A(1),B(1),C(1),D(1)

C
C SHARPS COMPUTES A # B = C ( A,B,C BEING ARRAYS OF CUBES )
C INITIALIZE C ( NC = NUMBER OF CUBES IN C )
C
CALL WHEAD(C)
NC=0

C
C
C TEST IF A IS EMPTY THEN C IS EMPTY (NC = NUMBER OF CUBES IN A )
CALL RHEAD(A)
IF(NC.EQ.0) GO TO 19
NCA=NC

C
C TEST IF B IS EMPTY THEN C = A ( NC = NUMBER OF CUBES IN B )
C
CALL RHEAD(B)
IF(NC.EQ.0) GO TO 21
NCB=NC

C
C ILAST = THE POSITION OF THE FIRST BIT OF THE LAST BYTE
C OF THE LAST WORD OF ANY CUBES OF ANY ARRAYS
C ( NV = NUMBER OF ELEMENTS IN A CUBE )
C ( NWPC = NUMBER OF WORDS IN A CUBE )
C ( I18 = 12 = NUMBER OF BYTES IN A WORD )
C ( I2 = 3 = NUMBER OF BITS IN A BYTE )
C
ILAST=(NV-1-(NWPC-1)*I18)*I2

C
C DO LOOP 20 CONTROL THE CUBES OF B
C (IB = INDEX OF CUBES OF B )
C
(NCB = NUMBER OF CUBES IN B )
DO 20 IB=1,NCB

C
C JB = INDEX OF THE FIRST WORD MINUS ONE OF THE IB TH CUBE OF B
C
JB=IB*NWPC-NWPC+1

C
C DO LOOP 30 CONTROL THE CUBES OF A
C (IA = INDEX OF CUBES OF A )
C (NCA = NUMBER OF CUBES IN B )
C
DO 30 IA=1,NCA

C
C JA = INDEX OF THE FIRST WORD MINUS ONE OF THE IA TH CUBE OF A
C
JA=IA*NWPC-NWPC+1

C
C INITIALIZE THE SUBSUMING SWITCH VARIABLE ISW2
C
ISW2 = 0

```

-PROGRAM E ULUG-PH D THESIS-CARLETON UNIVERSITY-

```

C
C DO LOOP 50 CONTROL THE SUBSUM AND
C INTERSECTION TEST
C      ( N = THE RELATIVE INDEX OF THE WORDS TESTED )
C
C      DO 50 N=1,NWPC
C      L=AND(A(N+JA),B(N+JB))
C
C TEST IF A(N+JA) SUBSUMS B(N+JB) AND SET ISW2 = 1 IF NOT,
C OTHERWISE PICKS THE NEXT PAIR OF WORDS
C
C      IF(L.NE.A(N+JA)) ISW2=1
C      IF(N.EQ.NWPC) GO TO 51
C      IMAX=34
C      GO TO 52
51      IMAX=ILAST
52      DO 50 I=0,IMAX,2
C
C TEST IF A(N+JA) INTERSECTION B(N+JB) PRODUCES A PHI
C IF THE FOLLOWING BINARY OPERATIONS HOLD, I.E.
C I*J=PHI      J*I=PHI
C
C      IF(FLD(I,2,L).EQ.FLD(0,2,0)) GO TO 70
50      CONTINUE
C
C TEST IF ISW2 = 0 , MEANING IA TH CUBE OF A SUBSUMS IB TH
C CUBE OF B, THEN CUBE C IS EMPTY
C
C      IF(ISW2.EQ.0) GO TO 30
C      GO TO 60
C
C CUBE D IS GENERATED KNOWING THAT INTERSECTION IS EMPTY
C
C      DO 80 N=1,NWPC
80      D(N+1) =A(N+JA)
C
C INITIALIZE THE INDEX OF THE WORDS OF THE TEMPORARY ARRAY CC, I.E. K
C THE HEADER OF THE TEMPORARY ARRAY D (FOR THE CUBE C ) IS SET
C
C      NC=1
C      GO TO 90
C
C DO LOOP 100 CONTROL THE NEW CUBES GENERATED AND TEST FOR
C EMPTY INTERSECTION
C      ( N = THE RELATIVE INDEX OF THE WORDS )
C INITIALIZE THE INDEX OF THE WORDS OF THE TEMPORARY ARRAY D
C
C      K=0      (SEE CORRECTIONS)
C      IF(N.EQ.NWPC) GO TO 101
C      GO TO 102
101      IMAX=ILAST
102      DO 100 I=0,IMAX,2
C
C TEST IF A(N+JA) CONTAINS A K ,IF YES GENERATES A NEW CUBE ,
C OTHERWISE TEST THE NEXT BYTE
C
C      IF((FLD(I,2,A(N+JA)).EQ.FLD(34,2,3).AND.FLD(I,2,B(N+JB)).EQ.FLD(34

```

PROGRAM F OLUG-PH D THESIS-CARLETON UNIVERSITY-

```

      3,2,3)).OP.FLD(I,2,A(N+JA)).NE.FLD(34,2,3)) GO TO 100
C
C DO LOOP 300 CONTROL THE BINARY OPERATIONS, I.E.
C K*I=J      K*J=I
C      ( NN = THE RELATIVE INDEX OF THE WORDS )
C
      DO 300 NN=1,NWPC
300  D(NN+1+K)=A(NN+JA)
      IF(FLD(I,2,B(N+JB)).EQ.FLD(34,2,1)) GO TO 303
      FLD(I,2,D(NN+1+K))=FLD(34,2,1)
      GO TO 302
303  FLD(I,2,D(NN+1+K))=FLD(34,2,2)
302  K=K+NWPC
100  CONTINUE
      NC=K/NWPC
90   CALL WHEAD(D)
      CALL COMPIN(C,D,MXC)
30   CONTINUE
C
C SINCE EVERY CUBES OF A WERE PICKED, ABSORB ARRAY C AND IF EVERY
C CUBES OF B WERE PICKED THEN C = A # B, OTHERWISE COPY C INTO A
C AND SHARPS THE IB+1 TH CUBE
C
      CALL SUBSUM(C)
      NCA=FLD(18,9,C(1))
      IF(1B.EQ.NCB) GO TO 20
      CALL COPYAR(A,C,MXC)
      NC=0
      CALL WHEAD(C)
20   CONTINUE
C
C NCC = THE NUMBER OF CUBES IN ARRAY C
C
      NCC=NCA
      RETURN
19   CALL WHEAD(C)
22   NCC=NC
      RETURN
21   CALL WHEAD(C)
      CALL SUBSUM(A)
      CALL COPYAR(C,A,MXC)
      GO TO 22
      END

```

-F.R. M. E. ULUG-PH.D. THESIS-CARLETON UNIVERSITY-

```

EL      3
        SUBROUTINE INTERX(A, B, C, MXC, NCC)
        COMMON /QZ/I2,I6,I18,I23,I26,IBK,K1,K6,K7,KEY,IPCL(8),IRCL(7),
        1MASK(12),I,J,K,L,M,N,NWPC,NV,NC,NW,IA,IB,IC,NWA,NWR,NWC,MSK,IO(8)
        INTEGER A(1),B(1),C(1),CC(10)

C
C      SUBROUTINE INTERX SETS IN ARRAY C THE INTERSECTION
C      OF ARRAY A AND B
C      SET THE HEADER OF ARRAY C ASSUMING C EMPTY
C      ( NCC = NC = NUMBER OF CUBES IN C )
C
        NC=0
        NCC=0
        CALL WHEAD(C)
        CALL RHEAD(A)
        NCA=NC

C
C      TEST IF ARRAY A IS EMPTY, IF YES C IS EMPTY
C      ( NCA = NUMBER OF CUBES IN A )
C
        IF(NCA.EQ.0) GO TO 30
        CALL RHEAD(B)
        NCB=NC

C
C      TEST IF ARRAY B IS EMPTY, IF YES C IS EMPTY
C      ( NCB = NUMBER OF CUBES IN B )
C
        IF(NCB.EQ.0) GO TO 30

C
C      SET THE HEADER OF THE TEMPORARY ARRAY CC
C      ( NC = NUMBER OF CUBES IN ARRAY CC )
C
        NC=1
        CALL WHEAD(CC)

C
C      ILAST = THE POSITION OF THE FIRST BIT OF THE LAST BYTE
C      OF THE LAST WORD OF ANY CUBES OF ANY ARRAYS
C
        ILAST=(NV-1-(NWPC-1)*I18)*I2

C
C      DO LOOP 20 CONTROLS THE CUBES OF ARRAY A
C      ( IA = INDEX OF THE CUBES OF ARRAY A )
C      ( NCA = NUMBER OF CUBES IN ARRAY A )
C
        DO 20 IA=1,NCA

C
C      JA = THE INDEX OF THE FIRST WORD MINUS ONE OF THE IA TH CUBE OF A
C
        JA=IA*NWPC-NWPC+1

C
C      DO LOOP 20 CONTROLS THE CUBES OF ARRAY B
C      ( IB = INDEX OF THE CUBES OF ARRAY B )
C      ( NCB = NUMBER OF CUBES IN ARRAY B )
C
        DO 20 IB=1,NCB

C
C      JB = THE INDEX OF THE FIRST WORD MINUS ONE OF THE IB TH CUBE OF B

```

-R F ULUG-PH D THESIS-CARLETON UNIVERSITY-

```
C
      JB=IB*NWPC-NWPC+1
C
C      DO LOOP 50 CONTROLS THE WORDS OF THE PAIR OF CUBES INTERSECTED
C      A(N+JA) AND B(N+JB) TO BE INTERSECTED
C      ( N = THE RELATIVE INDEX OF THE WORDS )
C      ( NWPC = NUMBER OF WORDS BY CUBE )
C
      DO 50 N=1,NWPC
      L=AND(A(N+JA),B(N+JB))
      IF(N.EQ.NWPC) GO TO 60
      IMAX=34
      GO TO 70
60    IMAX=ILAST
C      DO LOOP 80 CONTROLS THE SEARCH FOR EMPTY INTERSECTION
C      ( I = POSITION OF THE FIRST HIT OF A BYTE )
C
70    DO 80 I=0,IMAX,2
C
C      TEST IF THE BYTES OF L = PHI , IF YES INTERSECTION IS EMPTY
C      OTHERWISE STORE L IN THE TEMPORARY ARRAY CC
C
      IF(FLD(I,2,L).EQ.FLD(0,2,0)) GO TO 20
80    CONTINUE
      CC(N+1)=I
50    CONTINUE
C
C      COMBIN THE INTERSECTION CUBE STORED IN CC WITH ARRAY C
C
      CALL COMPIN(C,CC,MXC)
      NCC=FLD(18,9,C(1))
20    CONTINUE
C
C      ABSORB THE INTERSECTION ARRAY C
C
30    CALL SUBSUM(C)
      RETURN
      END
```

-MR M E ULUG-PH D THESIS-CARLETON UNIVERSITY-

VEL

```

3
SUBROUTINE SUBSUM(A)
COMMON /QZ/I3,I6,I12,I33,I36,IBK,K1,K6,K7,KEY,IPCL(8),IRCL(7),
IMASK(12),I,J,K,L,M,N,NWPC,NV,NC,NW,IA,IB,IC,NWA,NWR,NWC,MSK,IO(80)
INTEGER A(1)
CALL RHEAD(A)
IF(NC .LE. 1) RETURN
L = NW - NWPC
DO 4 I = 2,L,NWPC
IF(A(I) .EQ. 0) GO TO 4
N = I + NWPC
DO 2 J = N, NW, NWPC
IF(A(J) .EQ. 0) GO TO 2
K = 0
IO(1) = 0
IO(2) = 0
1 M = AND(A(I+K), A(J+K))
IF(M .NE. A(I+K)) IO(1) = 1
IF(M .NE. A(J+K)) IO(2) = 1
IF(IO(1)+IO(2) .EQ. 2) GO TO 2
K = K + 1
IF(K .LT. NWPC) GO TO 1
IF(IO(1) .EQ. 0) GO TO 3
A(J) = 0
2 CONTINUE
GO TO 4
3 A(I) = 0
4 CONTINUE
NC = 0
J = 2
DO 6 I = 2,NW, NWPC
IF(FLD(0,18,A(I)).EQ.FLD(0,18,0).AND.
$FLD(17,18,A(I)).EQ.FLD(0,18,0)) GO TO 6
DO 5 K = 1,NWPC
5 A(J+K-1) = A(I+K-1)
NC = NC + 1
J = J + NWPC
6 CONTINUE
CALL WHEAD(A)
RETURN
END

```

1
1
1
1
1
1
1
1
1
1
2
2
2
2
2
2
2
2
2
2
2
3

3
3
3
3
3
3

-MR M E ULUG-PH D THESIS-CARLETON UNIVERSITY-

[illegible]

-R M E ULUG-PH D THESIS-CARLETON UNIVERSITY-

'EL	3		
		SUBROUTINE READAR(A, NVAR, MAX)	
		COMMON /OZ/I3,I6,I12,I33,I36,IBK,K1,K6,K7,KEY,IPCL(8),IRCL(7),	
		1MASK(12),I,J,K,L,M,N,NWPC,NV,NC,NW,IA,IB,IC,NWA,NWR,NWC,MSK,IO(80)	
		INTEGER A(1)	
		NV = NVAR	
		DO 1 I = 1,MAX	
1		A(I) = 0	
		NW = 1	
		NC = -1	
2		READ 3, IO	1
3		FORMAT(80A1)	1
		I = 0	1
		J = 0	1
		NC = NC + 1	1
		NW = NW + 1	1
		IF(NW .GT. MAX) GO TO 19	1
		DO 9 N = 1,80	1
		IF(IO(N) .EQ. IBK) GO TO 9	1
		K = 3	1
		DO 4 L = 1,5	2
		IF(IO(N) .EQ. IRCL(L)) GO TO (6, 7, 8, 8, 8), L	2
4		CONTINUE	2
		IF(J .EQ. 0) GO TO 21	2
		PRINT 5	2
5		FORMAT('OREADAR--ILLEGAL CHARACTER ON CARD')	2
		GO TO 11	2
6		K=1	
		GO TO 8	
7		K=2	
8		FLD(I, I3, A(NW)) = K	2
		J = J + 1	3
		IF(J .EQ. NV) GO TO 2	3
		I = I + I3	3
		IF(I .LT. I36) GO TO 9	3
		I = 0	3
		NW = NW + 1	3
		IF(NW .GT. MAX) GO TO 19	3
9		CONTINUE	3
		PRINT 10	3
10		FORMAT('OREADAR--INSUFFICIENT VARIABLES ON CARD')	3
11		PRINT 12, IO	4
12		FORMAT(' ---', 80A1, ' ---')	4
13		CALL WHEAD(A)	4
		J = 0	4
14		READ 3, IO	4
		J = J + 1	4
		DO 15 N = 1,80	4
		IF(IO(N) .NE. IBK) GO TO 16	4
15		CONTINUE	4
		GO TO 14	4
16		DO 17 K = 1,5	5
		IF(IO(N) .EQ. IRCL(K)) GO TO 14	5
17		CONTINUE	5
		PRINT 18, J	5
18		FORMAT(I4, ' CARDS SKIPPED')	5

-MR M F ULUG-PH D THESIS-CARLETON UNIVERSITY-

```
      RETURN  
19 PRINT 20  
20 FORMAT('DREADAR--ARRAY OVERFLOW')  
      GO TO 13  
21 CALL XHEAD(A)  
      RETURN  
      END
```

5
5
5
5
6
6

-MR THE ULUG-PH D THESIS-CARLETON UNIVERSITY-

```

EL      3
SUBROUTINE ERRORS
C      I12 = 18 QUADRAL ENCODED LOGIC ARRAY ELEMENTS PER WORDS
C      I6 = 6 ALPHANUMERIC CHARACTERS PER WORD
C      I36 = 36 BITS PER WORD
C      I3 = 2 BITS PER BYTE OF LENGTH 2
C      IBK BLANK CHARACTER
C      IPCL PRINTER CHARACTER LIST
C      IRCL READER CHARACTER LIST
C      NWPC NUMBER OF WORDS PER CUBE
C      NV NUMBER OF VARIABLES
C      NC NUMBER OF CUBES IN ARRAY
C      NW NUMBER OF WORDS IN ARRAY
COMMON /QZ/I3,I6,I12,I33,I36,IBK,K1,K6,K7,KEY,IPCL(8),IRCL(7),
1MASK(12),I,J,K,L,M,N,NWPC,NV,NC,NW,IA,IB,IC,NWA,NWR,NWC,MSK,IO(80)
DATA I3/2/,I6/6/,I12/18/,I33/34/,I36/36/,Z/00/,IBK/' ',KEY/052/
DATA K1/0111111111111111/, K6/0666666666666666/, K7/0777777777777777/
DATA IPCL/'0','0','0','0','0','0','0','1','X'/
DATA IRCL/'0','1','X','X','D','Q','X'/
DATA MASK/07,070,0700,07000,070000,0700000,07000000,070000000,
1 0700000000,07000000000,070000000000,0700000000000/
INTEGER A(1)
GO TO 3
ENTRY WHFAD(A)
A(1) = 0
FLD(0, 6, A(1)) = KEY
IF(NV .EQ. 0) RETURN
IF(NC .EQ. 0) RETURN
IX = NV - 1
IX = IX/I12
NWPC = IX + 1
IX = NWPC*NC
NW = IX + 1
FLD(6, 4, A(1)) = NWPC
FLD(10, 8, A(1)) = NV
FLD(18, 9, A(1)) = NC
FLD(27, 9, A(1)) = NW
RETURN
ENTRY RHEAD(A)
IX = FLD(0, 6, A(1))
IF(IX .NE. KEY) GO TO 1
NWPC = FLD(6, 4, A(1))
NV = FLD(10, 8, A(1))
NC = FLD(18, 9, A(1))
NW = FLD(27, 9, A(1))
IF(NV .EQ. 0) RETURN
IF(NW .EQ. NWPC*NC+1) RETURN
1 PRINT 2
2 FORMAT('INVALID ARRAY HEADER RECORD')
3 I = 0
GO TO (1, 1), I
END

```

-MR E ULUG-PH D THE SIS-CARLETON UNIVERSITY-

```

/EL      3
          SUBROUTINE CPYCUB(A, B, NCUBE)
          COMMON /OZ/I3,I6,I12,I33,I36,IBK,K1,K6,K7,KEY,IPCL(8),IRCL(7),
1 MASK(12),I,J,K,L,M,N,NWPC,NV,NC,NW,IA,IB,IC,NWA,NWB,NWC,MSK,IO(80)
          INTEGER A(1), B(1)
          MM = 0
          GO TO 1
          ENTRY MOVUCUB(A, B, NCUBE)
          MM = 1
1 CALL RHEAD(A)
          IF(NCUBE .LE. 0) GO TO 5
          IF(NCUBE .GT. NC) GO TO 5
          J = NCUBF*NWPC - NWPC + 1
          DO 2 N = 1,NWPC
2 R(N+1) = A(N+J)
          K = NC
          L = NW
          NC = 1
          CALL WHEAD(B)
          IF(MM .EQ. 0) RETURN
          NW = L - NWPC
          NC = K - 1
          IF(NC .EQ. 0) GO TO 4
          IF(NCUBE .GT. NC) GO TO 4
          J = J + 1
          DO 3 N = J,NW
2 A(N) = A(N+NWPC)
4 CALL WHEAD(A)
          RETURN
5 NC = 0
          CALL WHEAD(R)
          PRINT 6
6 FORMAT('0CPYCUB--NONEXISTENT CUBE, R ARRAY SET EMPTY')
          RETURN
          END

```

-MR E ULUG-PH D THESIS-CARLETON UNIVERSITY-

```

VEL      3
          SUBROUTINE COMBIN(A, B, MAX)
          COMMON /QZ/I3,I6,I12,I33,I36,IBK,K1,K6,K7,KEY,IPCL(8),IRC(7),
1 MASK(12),I,J,K,L,M,N,NWPC,NV,NC,NW,IA,IB,IC,NWA,NWB,NWC,MSK,IO(8)
          INTEGER A(1), B(1)
          CALL RHEAD(B)
          J = NV
          K = NC
          L = NW
          CALL RHEAD(A)
          IF(L .EQ. 0) RETURN
          IF(NW .EQ. 0) GO TO 5
          IF(J .EQ. NV) GO TO 2
          PRINT 1
1  FORMAT('COMBIN--UNEQUAL NUMBER OF VARIABLES')
          CALL ERRORS
2  NC = NC + K
          J = NW - 1
          NW = I + L
3  IF(NW .GT. MAX) GO TO 7
          DO 4 K = 2,L
4  A(K+I) = B(K)
          CALL WHEAD(A)
          RETURN
          ENTRY COPYAR(A, B, MAX)
5  CALL RHEAD(B)
          IF(NW .GT. MAX) GO TO 7
          DO 6 I = 1,NW
6  A(I) = B(I)
          CALL WHEAD(A)
          RETURN
7  PRINT 8
8  FORMAT('COMBIN--ARRAY OVERFLOW')
          CALL ERRORS
          END

```

1
1
1
1
1
1
1
2
2
2
2
2
2
2
2
2
3
3
3
3

ACCOUNT: D8015

PROJECT: CADMDN

02.796 IN: 14 OUT: 0 PAGES: 17 COST: \$ 0.69

TIME: 13:13:51-MAR 2,1972

TIME: 13:15:40-MAR 2,1972

Corrections on Appendix A

SHARPS subroutine

```
60      K=0
        DO 100 N=1, NWPC
        IF (N.EQ.NWPC) GO TO 101
        I MAX = 34
        GO TO 102
101     I MAX = I LAST
102     DO 100 I=0, I MAX, 2
```

-PROGRAM LOGOP-PH D THESIS-CARLETON UNIVERSITY-

```

3
INTEGER A(100),B(100),C(100),D(100)

THIS PROGRAM IS AN EXAMPLE ILLUSTRATING THE USE OF THE MAIN LOGIC
OPERATORS. A VALUE FROM 1 TO 4 IS ASSIGNED TO ISW1,I.E.:

      ISW1 = 1 ABSORB                      - UNARY OPERATOR
              = 2 CONSENSUS                - UNARY OPERATOR
              = 3 CUBE INTERSECTION        - BINARY OPERATOR
              = 4 SHARP PRODUCT            - BINARY OPERATOR

READ INPUT PARAMETERS

      N      = NUMBER OF INPUT VARIABLES
      M      = NUMBER OF OUTPUT VARIABLES
      NV     = TOTAL NUMBER OF VARIABLES
      ISW1   = CODE OF THE LOGIC OPERATORS

READ 1,N,M,ISW1
FORMAT(3I2)
      MX     = DIMENSIONED SIZE OF ARRAY A,B,C,D

      MX=100
      NV=N+M
      PRINT 2,N,M,ISW1,NV,MX
2      FORMAT('PROGRAM LOGOP IS A DEMONSTRATION PROGRAM OF THE LOGIC OPE
$RATORS'// ' INPUT PARAMETERS : ' N = ',I2,' M = ',I2,' ISW1 = ',I2
$, ' NV = ',I2,' MX = ',I4// ' INPUT A - ARRAY'//)

      ARRAY A IS READ BY READAR IN ORDER TO BE OPERATED BY AN UNARY
      OPERATOR (ISW1 = 1 TO 2) OR TO BE OPERATED AS THE LEFT SIDE ARRAY
      BY A BINARY OPERATOR (ISW1 = 3 TO 4).ARRAY A IS PRINTED BY PRNTAR

      CALL READAR(A,NV,MX)
      CALL PRNTAR(A,N)

      ACCORDING TO ISW1 ,THE RIGHT SIDE ARRAY B OPERATED BY A BINARY
      OPERATOR IS READ BY READAR AND PRINTED BY PRNTAR
      IF(ISW1.IE.2) GO TO 3
      PRINT 13
13      FORMAT(' INPUT B - ARRAY'//)
      CALL READAR(B,NV,MX)
      CALL PRNTAR(B,N)
      GO TO(4,5,8,9),ISW1

      ARRAY A IS ABSORBED AND REPLACED IN A * A <-- A(A) *

4      CALL SUBSUM(A)
      PRINT 15

```

APPENDIX B

A COMPUTER PROGRAMME USING OCTAL CODING WITH IMPROVED SHARPS
SUBROUTINE

WORKING DRAFT - PH.D. THESIS - CARLETON UNIVERSITY -

```

15  FORMAT(/' ABSORB - UNARY OPERATOR : A <-- A(A)'/
    $' ABSORBED ARRAY A'/)
    GO TO 16

C
C  THE CUBES OF ARRAY A ARE CONVERTED INTO PRIME IMPLICANTS USING
C  THE ITERATIVE CONSENSUS ALGORITHM, ASSUMING BARTEE CODING, I.E.
C  FOR EACH '1' IN THE M-UPLES PART OF EACH CUBE OF A FUNCTION
C  ARRAY IS REPLACED WITH AN 'X' : A <-- *(A)
C
5   CALL CNSSUS(A, B, C, MX, NV)
    PRINT 17
17  FORMAT(/' CONSENSUS - UNARY OPERATOR : A <-- *(A)'/
    $' PRIME IMPLICANTS ARRAY'/)
    GO TO 16

C
C  THE INTERSECTION OF ARRAYS A AND B IS PLACED IN ARRAY C : C <-- A\B
C  NCC RETURNS THE NUMBER OF CUBES IN ARRAY C
C
8   CALL INTERX(A, B, C, MX, NCC)
    PRINT 20, NCC
20  FORMAT(/' CUBE INTERSECTION - BINARY OPERATOR : C <-- A\B'/
    $' NCC = ', I3/)
    GO TO 24

C
C  THE SHARP PRODUCT OF ARRAY A AND B IS PLACED IN ARRAY C : C <-- A#B
C  NCC RETURNS THE NUMBER OF CUBES IN ARRAY C
C
9   CALL SHAPPS(A, B, C, D, MX, NCC)
    PRINT 26, NCC
26  FORMAT(/' SHARP PRODUCT - BINARY OPERATOR : C <-- A#B'/
    $' NCC = ', I3/)
    GO TO 24

16  CALL PRNTAR(A, N)
    GO TO 23
24  CALL PRNTAR(C, N)
23  CALL EXIT
    END

```

-MR M E ULUG-PH D THESIS-CARLETON UNIVERSITY-

```

1L      3
        SUBROUTINE SUBSUM(A)
        COMMON /QZ/I3,I6,I12,I33,I36,IBK,K1,K6,K7,KEY,IPCL(8),IRCL(7),
1 MASK(12),I,J,K,L,M,N,NWPC,NV,NC,NW,IA,IB,IC,NWA,NWR,NWC,MSK,IO(80)
        INTEGER A(1)
        CALL RHEAD(A)
        IF(NC .LE. 1) RETURN
        L = NW - NWPC
        DO 4 I = 2,L,NWPC
        IF(A(I) .EQ. 0) GO TO 4
        N = I + NWPC
        DO 2 J = N, NW, NWPC
        IF(A(J) .EQ. 0) GO TO 2
        K = 0
        IO(1) = 0
        IO(2) = 0
1 M = AND(A(I+K), A(J+K))
        IF(M .NE. A(I+K)) IO(1) = 1
        IF(M .NE. A(J+K)) IO(2) = 1
        IF(IO(1)+IO(2) .EQ. 2) GO TO 2
        K = K + 1
        IF(K .LT. NWPC) GO TO 1
        IF(IO(1) .EQ. 0) GO TO 3
        A(J) = 0
2 CONTINUE
        GO TO 4
3 A(I) = 0
4 CONTINUE
        NC = 0
        J = 2
        DO 6 I = 2,NW, NWPC
        IF(FLD(0,18,A(I)).EQ.FLD(0,18,0)).AND.
$FLD(17,18,A(I)).EQ.FLD(0,18,0)) GO TO 6
        DO 5 K = 1,NWPC
5 A(J+K-1) = A(I+K-1)
        NC = NC + 1
        J = J + NWPC
6 CONTINUE
        CALL WHEAD(A)
        RETURN
        END

```

-MR M E ULUG-PH D THESIS-CARLETON UNIVERSITY-

```

3
SUBROUTINE CNSSUS(A, B, C, MXA, NCC)
COMMON /QZ/I3,I6,I12,I33,I36,IBK,K1,K6,K7,KEY,IPCL(8),IRCL(7),
1MASK(12),I,J,K,L,M,N,NWPC,NV,NC,NW,IA,IB,IC,NWA,NWR,NWC,MSK,IO(80)
INTEGER A(1),B(1),C(1)

C
C TEST IF A IS EMPTY THEN RETURN, OTHERWISE SET NCA
C      (NCA = NUMBER OF CUBES IN A )
C
C      CALL RHEAD(A)
C      IF(NC.LE.1) RETURN

C
C INITIALIZE D THE TEMPORARY ARRAY
C
C      NCA=NC
C      NC=1
C      CALL WHEAD(R)

C
C ILAST = THE POSITION OF THE FIRST BIT OF THE LAST BYTE
C
C      ILAST=(NV-1-(NWPC-1)*I12)*I3
20      NCAM1=NCA-1
C
C DO LOOP 10 THE CUBES OF A
C      (II = INDEX OF THE CUBES OF A FOR 1.LE.II.LE.NCAM1)
C      (JJ = INDEX OF THE CUBES OF A FOR NN.LE.JJ.LE.NCA)
C
C      DO 10 II=1,NCAM1
C
C      JII = INDEX OF THE FIRST WORD MINUS ONE OF THE II TH CUBE OF A
C
C      JII=II*NWPC-NWPC+1
C      NN=II+1
C      DO 10 JJ=NN,NCA
C
C      JJJ = INDEX OF THE FIRST WORD MINUS ONE OF THE JJ TH CUBE OF A
C
C      JJJ=JJ*NWPC-NWPC+1
C
C INITIALIZE NPHI, THE COUNTER OF THE OCCURENCE OF PHI
C
C      NPHI=0
C
C DO LOOP 50 CONTROL THE CONSENSUS ACCORDING TO THE FOLLOWING
C BINARY OPERATIONS I*J=PHI J*I=PHI K*I=I ETC...
C CONVENTION K IS 7 OCT. IN CNSSUS, THEREFORE NO DECODING OR
C ENCODING OF K IS NEEDED
C
C      DO 50 N=1,NWPC
C      L=AND(A(N+JII),A(N+JJJ))
C      IF(N.EQ.NWPC) GO TO 60
C      IMAX=33
C      GO TO 70
60      IMAX=ILAST
70      DO 80 I=0,IMAX,3
C

```

-MR M E ULUG-PH D THESIS-CARLETON UNIVERSITY-

```
C   TEST IF A PHI WAS ENCOUNTEED AND INCREMENT NPHI BY ONE, IF YES
C
      IF(FLD(I,3,L).NE.FLD(0,3,0)) GO TO 80
      NPHI=NPHI+1
C
C   IF NPHI.GT.1 PICKS THE NEXT CUBE, OTHERWISE REPLACE PHI BY 7 OCT.
C
      IF(NPHI.GT.1) GO TO 10
      FLD(I,3,L)=FLD(33,3,7)
30   CONTINUE
      B(N+1)=L
50   CONTINUE
C
C   COMBIN THE TEMPORARY CUBE AT THE BOTTOM END OF ARRAY A
C
      CALL COMBIN(A, B, MXA)
10   CONTINUE
      CALL SUBSUM(A)
      CALL RHEAD(A)
      NCA=NC
47
C
C   DO LOOP 30 TEST IF NEW CONSENSUS TERMS WERE FOUND IN THE
C   LAST ITERATION, IF NOT THEN TERMINATE
C
      DO 30 KK=1,NW
      IF(A(KK).NE.C(KK)) GO TO 40
30   CONTINUE
      RETURN
40   CALL COPYAR(C, A, MXA)
      GO TO 20
      END
```

-MR M E ULUG-PH D THESIS-CARLETON UNIVERSITY-

```

EL      3
        SUBROUTINE SHARPS(A,B,C,D,MXC,NCC)
        COMMON /QZ/I3,I6,I12,I33,I36,IBK,K1,K6,K7,KEY,IPCL(8),IRCL(7),
1MASK(12),I,J,K,L,M,N,NWPC,NV,NC,NW,IA,IB,IC,NWA,NWB,NWC,MSK,IO(80)
        INTEGER A(1),B(1),C(1),D(1)

C
C      SHARPS COMPUTES A # B = C ( A,B,C BEING ARRAYS OF CUBES )
C      INITIALIZE C ( NC = NUMBER OF CUBES IN C )
C
        CALL WHEAD(C)
        NC=0

C
C      TEST IF A IS EMPTY THEN C IS EMPTY (NC = NUMBER OF CUBES IN A )
C      CALL RHEAD(A)
        IF(NC.EQ.0) GO TO 19
        NCA=NC

C
C      TEST IF B IS EMPTY THEN C = A ( NC = NUMBER OF CUB IN B )
C
        CALL RHEAD(B)
        IF(NC.EQ.0) GO TO 21
        NCB=NC

C
C      ILAST = THE POSITION OF THE FIRST BIT OF THE LAST BYTE
C      OF THE LAST WORD OF ANY CUBES OF ANY ARRAYS
C      ( NV = NUMBER OF ELEMENTS IN A CUBE )
C      ( NWPC = NUMBER OF WORDS IN A CUBE )
C      ( I12 = 12 = NUMBER OF BYTES IN A WORD )
C      ( I3 = 3 = NUMBER OF BITS IN A BYTE )
C
        ILAST=(NV-1-(NWPC-1)*I12)*I3

C
C      DO LOOP 20 CONTROL THE CUBES OF B
C      (IB = INDEX OF CUBES OF B )
C
        (NCB = NUMBER OF CUBES IN B )
        DO 20 IB=1,NCB

C
C      JB = INDEX OF THE FIRST WORD MINUS ONE OF THE IB TH CUBE OF B
C
        JB=IB*NWPC-NWPC+1

C
C      DO LOOP 30 CONTROL THE CUBES OF A
C      (IA = INDEX OF CUBES OF A )
C      (NCA = NUMBER OF CUBES IN B )
C
        DO 30 IA=1,NCA

C
C      JA = INDEX OF THE FIRST WORD MINUS ONE OF THE IA TH CUBE OF A
C
        JA=IA*NWPC-NWPC+1

C
C      INITIALIZE THE SUBSUMING SWITCH VARIABLE ISW2
C
        ISW2 = 0

```

-MR M. E. ULUG-Ph.D. THESIS-CARLETON UNIVERSITY-

```

C
C DO LOOP 50 CONTROL THE SUBSUM ( ACCORDING TO DIETMEYER ) AND
C INTERSECTION TEST
C      ( N = THE RELATIVE INDEX OF THE WORDS TESTED )
C
C      DO 50 N=1,NWPC
C      L=AND(A(N+JA),B(N+JB))
C
C TEST IF A(N+JA) SUBSUMS B(N+JB) AND SET ISW2 = 1 IF NOT,
C OTHERWISE PICKS THE NEXT PAIR OF WORDS
C
C      IF(L.NE.A(N+JA)) ISW2=1
C      IF(N.EQ.NWPC) GO TO 51
C      IMAX=33
C      GO TO 52
51      IMAX=ILAST
52      DO 50 I=0,IMAX,3
C
C TEST IF A(N+JA) INTERSECTION B(N+JB) PRODUCES A PHI
C IF THE FOLLOWING BINARY OPERATIONS HOLD, I.E.
C I*J=PHI      J*I=PHI
C
C      IF(FLD(I,3,L).EQ.FLD(0,3,0)) GO TO 70
50      CONTINUE
C
C TEST IF ISW2 = 0 , MEANING IA TH CUBE OF A SUBSUMS IB TH
C CUBE OF B, THEN CUBE C IS EMPTY
C
C      IF(ISW2.EQ.0) GO TO 30
C      GO TO 60
C
C CUBE D IS GENERATED KNOWING THAT INTERSECTION IS EMPTY
C
70      DO 80 N=1,NWPC
80      D(N+1) =A(N+JA)
C
C INITIALIZE THE INDEX OF THE WORDS OF THE TEMPORARY ARRAY CC,I.F.
C THE HEADER OF THE TEMPORARY ARRAY D (FOR THE CUBE C ) IS SET
C
C      NC=1
C      GO TO 90
C
C DO LOOP 100 CONTROL THE NEW CUBES GENERATED AND TEST FOR
C EMPTY INTERSECTION
C      ( N = THE RELATIVE INDEX OF THE WORDS )
C INITIALIZE THE INDEX OF THE WORDS OF THE TEMPORARY ARRAY D
C
60      K=0
C      DO 100 N=1,NWPC
C      IF(N.EQ.NWPC) GO TO 101
C      IMAX=33
C      GO TO 102
101      IMAX=ILAST
102      DO 100 I=0,IMAX,3
C
C TEST IF A(N+JA) CONTAINS A K ,IF YES GENERATES A NEW CUBE ,

```

-PR M E ULUG-PH D THESIS-CARLETON UNIVERSITY-

```

C      IF((FLD(I,3,A(N+JA)).EQ.FLD(33,3,7).AND.FLD(I,3,B(N+JB)).EQ.FLD(33
      $,3,7)).OR.FLD(I,3,A(N+JA)).NE.FLD(33,3,7)) GO TO 100
C
C      DO LOOP 300 CONTROL THE BINARY OPERATIONS, I.E.
C      K*I=J      K*J=I      K+E=E
C      ( NN = THE RELATIVE INDEX OF THE WORDS )
C
      DO 300 NN=1,NWPC
300    D(NN+1+K)=A(NN+JA)
      IF(FLD(I,3,D(N+JB)).EQ.FLD(33,3,1)) GO TO 303
      FLD(I,3,D(N+1+K))=FLD(33,3,1)
      GO TO 302
303    FLD(I,3,D(N+1+K))=FLD(33,3,2)
302    K=K+NWPC
100    CONTINUE
      NC=K/NWPC
90     CALL WHEAD(D)
      CALL COMBIN(C,D,MXC)
30     CONTINUE
C
C      SINCE EVERY CUBES OF A WERE PICKED, ABSORB ARRAY C AND IF EVERY
C      CUBES OF B WERE PICKED THEN C = A * B, OTHERWISE COPY C INTO A
C      AND SHARPS THE IB+1 TH CUBE
C
      CALL SUBSUM(C)
      NCA=FLD(18,9,C(1))
      IF(IB.EQ.NCB) GO TO 20
      CALL COPYAR(A,C,MXC)
      NC=0
      CALL WHEAD(C)
20     CONTINUE
C
C      NCC = THE NUMBER OF CUBES IN ARRAY C
C
      NCC=NCA
      RETURN
19     CALL WHEAD(C)
22     NCC=NC
      RETURN
21     CALL WHEAD(C)
      CALL SUBSUM(A)
      CALL COPYAR(C,A,MXC)
      GO TO 22
      END

```

-R 4 F DLUG-PU D THE SIS-CARLETON UNIVERSITY-

```

TEL      3
          SUBROUTINE COMBIN(A, B, MAX)
          COMMON /QZ/ I3, I6, I12, I33, I36, IBK, K1, K6, K7, KEY, IPCL(8), IRCL(7),
1 MASK(12), I, J, K, L, M, N, NWPC, NV, NC, NW, IA, IB, IC, NWA, NWB, NWC, MSK, IO(80)
          INTEGER A(1), B(1)
          CALL RHEAD(B)
          J = NV
          K = NC
          L = NW
          CALL RHEAD(A)
          IF(L .EQ. 0) RETURN
          IF(NW .EQ. 0) GO TO 5
          IF(J .EQ. NV) GO TO 2
          PRINT 1
1  FORMAT('COMBIN--UNEQUAL NUMBER OF VARIABLES')
          CALL ERRORS
2  NC = NC + K
          I = NW - 1
          NW = I + L
3  IF(NW .GT. MAX) GO TO 7
          DO 4 K = 2, L
4  A(K+I) = B(K)
          CALL WHEAD(A)
          RETURN
          ENTRY COPYAR(A, B, MAX)
5  CALL RHEAD(B)
          IF(NW .GT. MAX) GO TO 7
          DO 6 I = 1, NW
6  A(I) = B(I)
          CALL WHEAD(A)
          RETURN
7  PRINT 8
8  FORMAT('COMBIN--ARRAY OVERFLOW')
          CALL ERRORS
          END

```


-MR M F ULUG-PH D THESIS-CARLETON UNIVERSITY-

```

3
SUBROUTINE INTERX(A, B, C, MXC, NCC)
COMMON /OZ/I3,I6,I12,I33,I36,IBK,K1,K6,K7,KEY,IPCL(8),IRCL(7),
1MASK(12),I,J,K,L,M,N,NWPC,NV,NC,NW,IA,IB,IC,NWA,NWB,NWC,MSK,IO(80)
INTEGER A(1),B(1),C(1),CC(10)

C
C SUBROUTINE INTERX SETS IN ARRAY C THE INTERSECTION
C OF ARRAY A AND B
C SET THE HEADER OF ARRAY C ASSUMING C EMPTY
C ( NCC = NC = NUMBER OF CUBES IN C )
C
C NC=0
C NCC=0
C CALL WHEAD(C)
C CALL RHEAD(A)
C NCA=NC

C
C TEST IF ARRAY A IS EMPTY, IF YES C IS EMPTY
C ( NCA = NUMBER OF CUBES IN A )
C
C IF(NCA.EQ.0) GO TO 30
C CALL RHEAD(B)
C NCB=NC

C
C TEST IF ARRAY B IS EMPTY, IF YES C IS EMPTY
C ( NCB = NUMBER OF CUBES IN B )
C
C IF(NCB.EQ.0) GO TO 30

C
C SET THE HEADER OF THE TEMPORARY ARRAY CC
C ( NC = NUMBER OF CUBES IN ARRAY CC )
C
C NC=1
C CALL WHEAD(CC)

C
C ILAST = THE POSITION OF THE FIRST BIT OF THE LAST BYTE
C OF THE LAST WORD OF ANY CUBES OF ANY ARRAYS
C
C ILAST=(NV-1-(NWPC-1)*I12)*I3

C
C DO LOOP 20 CONTROLS THE CUBES OF ARRAY A
C ( IA = INDEX OF THE CUBES OF ARRAY A )
C ( NCA = NUMBER OF CUBES IN ARRAY A )
C
C DO 20 IA=1,NCA

C
C JA = THE INDEX OF THE FIRST WORD MINUS ONE OF THE IA TH CUBE OF A
C
C JA=IA*NWPC-NWPC+1

C
C DO LOOP 20 CONTROLS THE CUBES OF ARRAY B
C ( IB = INDEX OF THE CUBES OF ARRAY B )
C ( NCB = NUMBER OF CUBES IN ARRAY B )
C
C DO 20 IB=1,NCB

```

-PROGRAM ULUG-PH D THESIS-CARLETON UNIVERSITY-

```
C
      JB=IB*NWPC-NWPC+1
C
C      DO LOOP 50 CONTROLS THE WORDS OF THE PAIR OF CUBES INTERSECTED
C      A(N+JA) AND B(N+JB) TO BE INTERSECTED
C      ( N = THE RELATIVE INDEX OF THE WORDS )
C      ( NWPC = NUMBER OF WORDS BY CUBE )
C
      DO 50 N=1,NWPC
      L=AND(A(N+JA),B(N+JB))
      IF(N.EQ.NWPC) GO TO 60
      IMAX=33
      GO TO 70
60      IMAX=ILAST
C      DO LOOP 80 CONTROLS THE SEARCH FOR EMPTY INTERSECTION
C      ( I = POSITION OF THE FIRST BIT OF A BYTE )
C
      DO 80 I=0,IMAX,3
C
      TEST IF THE BYTES OF L = PHI , IF YES INTERSECTION IS EMPTY
C      OTHERWISE STORE L IN THE TEMPORARY ARRAY CC
C
      IF(FLD(I,3,L).EQ.FLD(0,3,0)) GO TO 20
80      CONTINUE
      CC(N+1)=L
50      CONTINUE
C
      COMBIN THE INTERSECTION CUBE STORED IN CC WITH ARRAY C
C
      CALL COMBIN(C,CC,MXC)
      MCC=FLD(18,9,C(1))
20      CONTINUE
C
      ABSORB THE INTERSECTION ARRAY C
C
30      CALL SUBSUM(C)
      RETURN
      END
```

- P F LOG-PR D THESIS-CARLETON UNIVERSITY-

```

ALL      3
SUBROUTINE ERRORS
C      I12 = 12 OCTAL ENCODED LOGIC ARRAY ELEMENTS (TRITS) PER WORD
C      I6 = 6 ALPHANUMERIC CHARACTERS PER WORD
C      I36 = 36 BITS PER WORD
C      I3 = 3 BITS PER TRIT
C      IBK BLANK CHARACTER
C      IPCL PRINTER CHARACTER LIST
C      IRCL READER CHARACTER LIST
C      NWPC NUMBER OF WORDS PER CUBE
C      NV NUMBER OF VARIABLES
C      NC NUMBER OF CUBES IN ARRAY
C      NW NUMBER OF WORDS IN ARRAY
COMMON /QZ/I3,I6,I12,I33,I36,IBK,K1,K6,K7,KEY,IPCL(8),IRCL(7),
1MASK(12),I,J,K,L,M,N,NWPC,NV,NC,NW,IA,IB,IC,NWA,NWB,NWC,MSK,IO(8),
DATA I3/3/,I6/6/,I12/12/,I33/33/,I36/36/,Z/00/,IBK/' '/,KEY/052/
DATA K1/011111111111/, K6/066666666666/, K7/077777777777/
DATA IPCL/'0','0','0','0','0','0','0','1','X'/
DATA IRCL/'0','1','2','X','D','Q','X'/
DATA MASK/07,070,0700,07000,070000,0700000,07000000,070000000,
1 0700000000,07000000000,070000000000,0700000000000/
INTEGER A(1)
GO TO 3
ENTRY WHFAD(A)
A(1) = 0
FLD(0, 6, A(1)) = KEY
IF(NV .EQ. 0) RETURN
IF(NC .EQ. 0) RETURN
IX = NV - 1
IX = IX/I12
NWPC = IX + 1
IX = NWPC*NC
NW = IX + 1
FLD(6, 4, A(1)) = NWPC
FLD(10, 8, A(1)) = NV
FLD(18, 9, A(1)) = NC
FLD(27, 9, A(1)) = NW
RETURN
ENTRY RHEAD(A)
IX = FLD(0, 6, A(1))
IF(IX .NE. KEY) GO TO 1
NWPC = FLD(6, 4, A(1))
NV = FLD(10, 8, A(1))
NC = FLD(18, 9, A(1))
NW = FLD(27, 9, A(1))
IF(NV .EQ. 0) RETURN
IF(NW .EQ. NWPC*NC+1) RETURN
1 PRINT 2
2 FORMAT('INVALID ARRAY HEADER RECORD')
3 I = 0
GO TO (1, 1), I
END

```

THESIS-CARLETON UNIVERSITY-

```

L      3
      SUBROUTINE PRNTAR(A, NIV)
      COMMON /QZ/I3,I6,I12,I33,I36,IBK,K1,K6,K7,KEY,IPCL(8),IRCL(7),
1 MASK(12),I,J,K,L,M,N,NWPC,NV,NC,NW,IA,IB,IC,NWA,NWR,NWC,MSK,IO(80)
      INTEGER A(1)
      N = NIV
      CALL RHEAD(A)
      IF(NC .EQ. 0) GO TO 5
      IF(N .LE. 0) N = NV
      IF(N .GT. NV) N = NV
      N = N + 1
      IO(N) = IBK
      M = NV + 1
      NW = 1
      DO 3 J = 1,NC
      I = 0
      NW = NW + 1
      DO 2 K = 1,M
      IF(K .EQ. N) GO TO 2
      L = FLD(I, I3, A(NW))
      IO(K) = IRCL(L)
      IF(K .LT. N) GO TO 1
      IF(L .EQ. 7) IO(K) = '- '
1 I = I + I3
      IF(I .LT. I36) GO TO 2
      IF(K .EQ. M) GO TO 2
      I = 0
      NW = NW + 1
2 CONTINUE
3 PRINT 4, (IO(K), K = 1,M)
4 FORMAT(1X, 80A1)
      RETURN
5 PRINT 6
6 FORMAT(/' EMPTY ARRAY'/)
      RETURN
      END

```

-R A E ULUG-PH D THESIS-CARLETON UNIVERSITY-

```

1000 3
1001 SUBROUTINE READAR(A, NVAR, MAX)
1002 COMMON /OZ/I3,I6,I12,I33,I36,IBK,K1,K6,K7,KEY,IPCL(8),IRCL(7),
1003 1 MASK(12),I,J,K,L,M,N,NWPC,NV,NC,NW,IA,IB,IC,NWA,NWB,NWC,MSK,IO(80)
1004 INTEGER A(1)
1005 NV = NVAR
1006 DO 1 I = 1,MAX
1007 1 A(I) = 0
1008   NW = 1
1009   NC = -1
1010 2 READ 3, IO
1011 3 FORMAT(80A1)
1012   I = 0
1013   J = 0
1014   NC = NC + 1
1015   NW = NW + 1
1016   IF(NW .GT. MAX) GO TO 19
1017   DO 9 N = 1,80
1018   IF(IO(N) .EQ. IBK) GO TO 9
1019   K = 7
1020   DO 4 L = 1,5
1021   IF(IO(N) .EQ. IRCL(L)) GO TO (6, 7, 8, 8, 8), L
1022 4 CONTINUE
1023   IF(J .EQ. 0) GO TO 21
1024   PRINT 5
1025 5 FORMAT('READAR--ILLEGAL CHARACTER ON CARD')
1026   GO TO 11
1027 6 K=1
1028   GO TO 8
1029 7 K=2
1030 8 FLD(I, I3, A(NW)) = K
1031   J = J + 1
1032   IF(J .EQ. NV) GO TO 2
1033   I = I + I3
1034   IF(I .LT. I36) GO TO 9
1035   I = 0
1036   NW = NW + 1
1037   IF(NW .GT. MAX) GO TO 19
1038 9 CONTINUE
1039   PRINT 10
1040 10 FORMAT('READAR--INSUFFICIENT VARIABLES ON CARD')
1041 11 PRINT 12, IO
1042 12 FORMAT(' ---', 80A1, ' ---')
1043 13 CALL WHEAD(A)
1044   J = 0
1045 14 READ 3, IO
1046   J = J + 1
1047   DO 15 N = 1,80
1048   IF(IO(N) .NE. IBK) GO TO 16
1049 15 CONTINUE
1050   GO TO 14
1051 16 DO 17 K = 1,5
1052   IF(IO(N) .EQ. IRCL(K)) GO TO 14
1053 17 CONTINUE
1054   PRINT 18, J
1055 18 FORMAT(I4, ' CARDS SKIPPED')

```

-MR M F ULUG-Ph D THESIS-CARLETON UNIVERSITY-

```
      RETURN
19 PRINT 20
20 FORMAT('DREADAR--ARRAY OVERFLOW')
      GO TO 13
21 CALL WHEAD(1)
      RETURN
      END
```

5!
5!
5'
5!
5'
6!
6:

- R F OLUG-PH D THESIS-CARLETON UNIVERSITY-

```

EL      3
        SUBROUTINE CPYCUB(A, B, NCUBE)
        COMMON /QZ/I3,I6,I12,I33,I36,IBK,K1,K6,K7,KEY,IPCL(8),IRCL(7),
1*ASK(12),I,J,K,L,M,N,NWPC,NV,NC,NW,IA,IB,IC,NWA,NWB,NWC,MSK,IO(80)
        INTEGER A(1), B(1)
        MM = 0
        GO TO 1
        ENTRY MOVCUR(A, B, NCUBE)
        MM = 1
1      CALL RHEAD(A)
        IF(NCUBE .LE. 0) GO TO 5
        IF(NCUBE .GT. NC) GO TO 5
        J = NCUBE*NWPC - NWPC + 1
        DO 2 N = 1,NWPC
2      B(N+1) = A(N+J)
        K = NC
        L = NW
        NC = 1
        CALL WHEAD(B)
        IF(MM .EQ. 0) RETURN
        NW = L - NWPC
        NC = K - 1
        IF(NC .EQ. 0) GO TO 4
        IF(NCUBE .GT. NC) GO TO 4
        J = J + 1
        DO 3 N = J,NW
3      A(N) = A(N+NWPC)
4      CALL WHEAD(A)
        RETURN
5      NC = 0
        CALL WHEAD(B)
        PRINT 6
6      FORMAT('0CPYCUB--NONEXISTENT CUBE, R ARRAY SET EMPTY')
        RETURN
        END

```

ACCOUNT: D8015

PROJECT: CADMDN

2.806 IN: 14 OUT: 0 PAGES: 17 COST: \$ 0.73

IME: 12:59:37-MAR 2,1972

IME: 13:02:08-MAR 2,1972

APPENDIX C

A COMPUTER PROGRAMME USING OCTAL CODING

-PROGRAM LOGOP- D THESIS-CARLETON UNIVERSITY-

```

C      3
C      INTEGER A(100),B(100),C(100),D(100)
C
C      THIS PROGRAM IS AN EXAMPLE ILLUSTRATING THE USE OF THE MAIN LOGIC
C      OPERATORS. A VALUE FROM 1 TO 4 IS ASSIGNED TO ISW1, I.E.:
C
C          ISW1 = 1 ABSORB                - UNARY OPERATOR
C          = 2 CONSENSUS                 - UNARY OPERATOR
C          = 3 CUBE INTERSECTION         - BINARY OPERATOR
C          = 4 SHARP PRODUCT              - BINARY OPERATOR
C
C      READ INPUT PARAMETERS
C
C          N      = NUMBER OF INPUT VARIABLES
C          M      = NUMBER OF OUTPUT VARIABLES
C          NV     = TOTAL NUMBER OF VARIABLES
C          ISW1   = CODE OF THE LOGIC OPERATORS
C
C      READ 1,N,M,ISW1
C      FORMAT(3I2)
C          MX     = DIMENSIONED SIZE OF ARRAY A,B,C,D
C
C      MX=100
C      NV=N+M
C      PRINT 2,N,M,ISW1,NV,MX
C      FORMAT('PROGRAM LOGOP IS A DEMONSTRATION PROGRAM OF THE LOGIC OPE
2      RATORS'// ' INPUT PARAMETERS : ' N = ',I2,' M = ',I2,' ISW1 = ',I2
C      3,' NV = ',I2,' MX = ',I4// ' INPUT A - ARRAY'//)
C
C      ARRAY A IS READ BY READAR IN ORDER TO BE OPERATED BY AN UNARY
C      OPERATOR (ISW1 = 1 TO 2) OR TO BE OPERATED AS THE LEFT SIDE ARRAY
C      BY A BINARY OPERATOR (ISW1 = 3 TO 4). ARRAY A IS PRINTED BY PRNTAR
C
C          CALL READAR(A,NV,MX)
C          CALL PRNTAR(A,N)
C
C      ACCORDING TO ISW1 ,THE RIGHT SIDE ARRAY B OPERATED BY A BINARY
C      OPERATOR IS READ BY READAR AND PRINTED BY PRNTAR
C          IF (ISW1.LE.2) GO TO 3
C          PRINT 13
C      13  FORMAT(' INPUT B - ARRAY'//)
C          CALL READAR(B,NV,MX)
C          CALL PRNTAR(B,N)
C          GO TO (4,5,8,9),ISW1
C
C      ARRAY A IS ABSORBED AND REPLACED IN A ' A <-- A(A) '
C
C      4      CALL SUBSUM(A)
C          PRINT 15

```

-MR. E. ULUG-PH.D. THESIS-CARLETON UNIVERSITY-

```

15  FORMAT(/' ABSORB - UNARY OPERATOR : A <-- A(A)'/
    $' ABSORBED ARRAY A'/)
    GO TO 16

C
C  THE CUBES OF ARRAY A ARE CONVERTED INTO PRIME IMPLICANTS USING
C  THE ITERATIVE CONSENSUS ALGORITHM, ASSUMING BARTEE CODING, I.E.
C  FOR >L EACH '1' IN THE M-UPLES PART OF EACH CUBE OF A FUNCTION
C  ARRAY IS REPLACED WITH AN 'X' : A <-- *(A)
C
5    CALL CNSSUS(A, B, C, MX, NV)
    PRINT 17
17  FORMAT(/' CONSENSUS - UNARY OPERATOR : A <-- *(A)'/
    $' PRIME IMPLICANTS ARRAY'/)
    GO TO 16

C
C  THE INTERSECTION OF ARRAYS A AND B IS PLACED IN ARRAY C : C <-- A\B
C  NCC RETURNS THE NUMBER OF CUBES IN ARRAY C
C
8    CALL INTERX(A, B, C, MX, NCC)
    PRINT 20, NCC
20  FORMAT(/' CUBE INTERSECTION - BINARY OPERATOR : C <-- A\B'/
    $' NCC = ', I3/)
    GO TO 24

C
C  THE SHARP PRODUCT OF ARRAY A AND B IS PLACED IN ARRAY C : C <-- A#B
C  NCC RETURNS THE NUMBER OF CUBES IN ARRAY C
C
9    CALL SHAPPS(A, B, C, D, MX, NCC)
    PRINT 26, NCC
26  FORMAT(/' SHARP PRODUCT - BINARY OPERATOR : C <-- A#B'/
    $' NCC = ', I3/)
    GO TO 24

16  CALL PRNTAR(A, N)
    GO TO 23
24  CALL PRNTAR(C, N)
25  CALL EXIT
    END

```

-NR M F ULUG-PH D THESIS-CARLETON UNIVERSITY-

```

EL      3
        SUBROUTINE SUBSUM(A)
        COMMON /QZ/I3,I6,I12,I33,I36,IBK,K1,K6,K7,KEY,IPCL(8),IRCL(7),
1MASK(12),I,J,K,L,M,N,NWPC,NV,NC,NW,IA,IB,IC,NWA,NWR,NWC,MSK,IO(80)
        INTEGER A(1)
        CALL RHEAD(A)
        IF(NC .LE. 1) RETURN
        L = NW - NWPC
        DO 4 I = 2,L,NWPC
        IF(A(I) .EQ. 0) GO TO 4
        M = I + NWPC
        DO 2 J = N, NW, NWPC
        IF(A(J) .EQ. 0) GO TO 2
        K = 0
        IO(1) = 0
        IO(2) = 0
1 M = AND(A(I+K), A(J+K))
        IF(M .NE. A(I+K)) IO(1) = 1
        IF(M .NE. A(J+K)) IO(2) = 1
        IF(IO(1)+IO(2) .EQ. 2) GO TO 2
        K = K + 1
        IF(K .LT. NWPC) GO TO 1
        IF(IO(1) .EQ. 0) GO TO 3
        A(J) = 0
2 CONTINUE
        GO TO 4
3 A(I) = 0
4 CONTINUE
        NC = 0
        J = 2
        DO 6 I = 2,NW, NWPC
        IF(FLD(0,18,A(I)).EQ.FLD(0,18,0).AND.
5FLD(17,18,A(I)).EQ.FLD(0,18,0)) GO TO 6
        DO 5 K = 1,NWPC
        A(J+K-1) = A(I+K-1)
        NC = NC + 1
        J = J + NWPC
6 CONTINUE
        CALL WHEAD(A)
        RETURN
        END

```

-PR F ULUG-PH D THESIS-CARLETON UNIVERSITY-

```

3
SUBROUTINE CNSSUS(A, B, C, MXA, NCC)
COMMON /QZ/I3,I6,I12,I33,I36,IBK,K1,K6,K7,KEY,IPCL(8),IRCL(7),
1MASK(12),I,J,K,L,M,N,NWPC,NV,NC,NW,IA,IB,IC,NWA,NWR,NWC,MSK,IO(80)
INTEGER A(1),B(1),C(1)

C
C TEST IF A IS EMPTY THEN RETURN, OTHERWISE SET NCA
C      (NCA = NUMBER OF CUBES IN A )
C
C      CALL RHEAD(A)
C      IF(NC.LE.1) RETURN

C
C INITIALIZE D THE TEMPORARY ARRAY
C
C      NCA=NC
C      NC=1
C      CALL WHEAD(R)

C
C ILAST = THE POSITION OF THE FIRST BITE OF THE LAST BYTE
C
C      ILAST=(NV-1-(NWPC-1)*I12)*I3
20  NCAM1=NCA-1
C
C DO LOOP 10 THE CUBES OF A
C      (II = INDEX OF THE CUBES OF A FOR 1.LE.II.LE.NCAM1)
C      (JJ = INDEX OF THE CUBES OF A FOR NN.LE.JJ.LE.NCA)
C
C      DO 10 II=1,NCAM1
C
C      JII = INDEX OF THE FIRST WORD MINUS ONE OF THE II TH CUPE OF A
C
C      JII=II*NWPC-NWPC+1
C      NN=II+1
C      DO 10 JJ=NN,NCA
C
C      JJJ = INDEX OF THE FIRST WORD MINUS ONE OF THE JJ TH CUPE OF A
C
C      JJJ=JJ*NWPC-NWPC+1
C
C      INITIALIZE NPHI, THE COUNTER OF THE OCCURENCE OF PHI
C
C      NPHI=0
C
C DO LOOP 50 CONTROL THE CONSENSUS ACCORDING TO THE FOLLOWING
C BINARY OPERATIONS I*J=PHI J*I=PHI K*I=I ETC...
C CONVENTION K IS 7 OCT. IN CNSSUS, THEREFORE NO DECODING OR
C ENCODING OF K IS NEEDED
C
C      DO 50 N=1,NWPC
C      L=AND(A(N+JII),A(N+JJJ))
C      IF(N.EQ.NWPC) GO TO 60
C      IMAX=33
C      GO TO 70
60  IMAX=ILAST
70  DO 80 I=0,IMAX,3
C

```

-MR M F ULUG-PH D THESIS-CARLETON UNIVERSITY-

```
C  TEST IF A PHI WAS ENCOUNTERED AND INCREMENT NPHI BY ONE, IF YES
C
      IF(FLD(1,3,L).NE.FLD(0,3,0)) GO TO 80
      NPHI=NPHI+1
C
C  IF NPHI.GT.1 PICKS THE NEXT CUBE, OTHERWISE REPLACE PHI BY 7 OCT.
C
      IF(NPHI.GT.1) GO TO 10
      FLD(1,3,L)=FLD(33,3,7)
80  CONTINUE
      R(N+1)=L
50  CONTINUE
C
C  COMBIN THE TEMPORARY CUBE AT THE BOTTOM END OF ARRAY A
C
      CALL COMBIN(A, B, MXA)
10  CONTINUE
      CALL SUBSUM(A)
      CALL RHEAD(A)
      NCA=NC
47
C
C  DO LOOP 30 TEST IF NEW CONSENSUS TERMS WERE FOUND IN THE
C  LAST ITERATION, IF NOT THEN TERMINATE
C
      DO 30 KK=1,NW
      IF(A(KK).NE.C(KK)) GO TO 40
30  CONTINUE
      RETURN
40  CALL COPYAR(C, A, MXA)
      GO TO 20
      END
```

-PR E ULUG-PH D THESIS-CARLETON UNIVERSITY-

```

JEL      3
SUBROUTINE SHARPS(A,B,C,D,MXC,NCC)
COMMON /QZ/I3,I6,I12,I33,I36,IBK,K1,K6,K7,KEY,IPCL(8),IRCL(7),
1MASK(12),I,J,K,L,M,N,NWPC,NV,NC,NW,IA,IB,IC,NWA,NWB,NWC,MSK,IO(80)
INTEGER A(1),B(1),C(1),D(1),AORB(10)

C
C SHARPS COMPUTES A # B = C ( A,B,C BEING ARRAYS OF CUBES )
C INITIALIZE C ( NC = NUMBER OF CUBES IN C )
C
NC=0
CALL WHEAD(C)

C
C TEST IF A IS EMPTY THEN C IS EMPTY (NC = NUMBER OF CUBES IN A )
C
CALL RHEAD(A)
IF(NC.EQ.0) GO TO 19
ICA=NC

C
C TEST IF B IS EMPTY THEN C = A ( NC = NUMBER OF CUBES IN B )
C
CALL RHEAD(B)
IF(NC.EQ.0) GO TO 21
NCB=NC

C
C ILAST = THE POSITION OF THE FIRST BIT OF THE LAST BYTE
C OF THE LAST WORD OF ANY CUBES OF ANY ARRAYS
C ( NV = NUMBER OF ELEMENTS IN A CUBE )
C ( NWPC = NUMBER OF WORDS IN A CUBE )
C ( I12 = 12 = NUMBER OF BYTES IN A WORD )
C ( I3 = 3 = NUMBER OF BITS IN A BYTE )
C
ILAST=(NV-1-(NWPC-1)*I12)*I3

C
C DO LOOP 20 CONTROL THE CUBES OF B
C ( IB = INDEX OF CUBES OF B )
C ( NCB = NUMBER OF CUBES IN B )
C
DO 20 IB=1,NCB

C
C JB = INDEX OF THE FIRST WORD MINUS ONE OF THE IB TH CUBE OF B
C
JB=IB*NWPC-NWPC+1

C
C DO LOOP 30 CONTROL THE CUBES OF A
C ( IA = INDEX OF CUBES OF A )
C ( NCA = NUMBER OF CUBES IN B )
C
DO 30 IA=1,NCA

C
C JA = INDEX OF THE FIRST WORD MINUS ONE OF THE IA TH CUPE OF A
C
JA=IA*NWPC-NWPC+1

C
C INITIALIZE THE SUBSUMING SWITCH VARIABLE ISW2
C
ISW2 = 0

```

-PROGRAM FULUG-PH.D. THESIS-CARLETON UNIVERSITY-

```

C
C DO LOOP 50 CONTROL THE SUBSUM ( ACCORDING TO DIETMEYER ) AND
C INTERSECTION TEST
C ( N = THE RELATIVE INDEX OF THE WORDS TESTED )
C
C DO 50 N=1,NWPC
C L=AND(A(N+JA),B(N+JB))
C
C TEST IF A(N+JA) SUBSUMS B(N+JB) AND SET ISW2 = 1 IF NOT,
C OTHERWISE PICKS THE NEXT PAIR OF WORDS
C
C IF(L.NE.A(N+JA)) ISW2=1
C IF(N.EQ.NWPC) GO TO 51
C IMAX=33
C GO TO 52
C IMAX=ILAST
51 DO 50 I=0,IMAX,3
52
C TEST IF A(N+JA) INTERSECTION B(N+JB) PRODUCES A PHI
C IF THE FOLLOWING BINARY OPERATIONS HOLD, I.E.
C I*J=PHI J*I=PHI
C
C IF(AND(FLD(I,3,A(N+JA)),FLD(I,3,B(N+JB))).EQ.FLD(0,3,0)) GO TO 70
50 CONTINUE
C
C TEST IF ISW2 = 0 , MEANING IA TH CUBE OF A SUBSUMS IB TH
C CUBE OF B, THEN CUBE C IS EMPTY
C
C IF(ISW2.EQ.0) GO TO 30
C GO TO 60
C
C CUBE D IS GENERATED KNOWING THAT INTERSECTION IS EMPTY
C
C DO 80 N=1,NWPC
80 D(N+1) =A(N+JA)
C
C INITIALIZE THE INDEX OF THE WORDS OF THE TEMPORARY ARRAY CC, I.E. K
C THE HEADER OF THE TEMPORARY ARRAY D (FOR THE CUBE C ) IS SET
C
C NC=1
C GO TO 90
C
C DO LOOP 100 CONTROL THE NEW CUBES GENERATED AND TEST FOR
C EMPTY INTERSECTION
C ( N = THE RELATIVE INDEX OF THE WORDS )
C INITIALIZE THE INDEX OF THE WORDS OF THE TEMPORARY ARRAY D
C
C COPY IN ARRAY D NV TIMES THE IA TH CUBE OF A BY MEANS OF LOOP 399
C ( NV = NUMBER OF ELEMENTS IN CUBE OF A )
C ( ID = INDEX OF CUBES OF ARRAY D )
C ( JD = INDEX OF THE FIRST WORD MINUS ONE
C OF THE ID TH CUBE OF ARRAY D )
C
60 DO 399 ID=1,NV
C JD=ID*NWPC-NWPC+1
C DO 399 ND=1,NWPC
399 D(ND+JD)=A(ND+JA)

```

-MR M E ULUG-PH D THESIS-CARLETON UNIVERSITY-

```

C
C CHANGE THE CODE OF K IN THE IA TH CUBE OF ARRAY A
C I.E., K=OCT. 7 IS ENCODED AS K=OCT. 3 BY MEANS OF LOOP 400
C ( NENC = RELATIVE INDEX OF THE WORDS
C OF THE IA TH CUBE OF ARRAY )
C
DO 400 NENC=1,NWPC
IF(NENC.EQ.NWPC) GO TO 401
IMAX=33
GO TO 402
401 IMAX=ILAST
402 DO 403 I=0,IMAX,3
IF(FLD(I,3,A(NENC+JA)).EQ.FLD(33,3,7)) FLD(I,3,A(NENC+JA))=FLD(33,
$3,3)
403 CONTINUE
C
C COMPUTE THE A(NENC+JA) OR B(NENC+JB) AND STORE THE RESULT
C IN AORB(NENC)
AORB(NENC)=OR(A(NENC+JA),B(NENC+JB))
400 CONTINUE
C
C SCAN AORB AND PRODUCE A NEW CUBE ONLY IF THE BYTE AT POSITION I
C IS A K,I.F. K=OCT.3, OTHERWISE TEST THE NEXT BYTE
C
K=0
DO 100 N=1,NWPC
IF(N.EQ.NWPC) GO TO 101
IMAX=33
GO TO 102
101 IMAX=ILAST
102 DO 100 I=0,IMAX,3
IF(FLD(I,3,AORB(N)).NE.FLD(33,3,3)) GO TO 100
IF(FLD(I,3,B(N+JB)).EQ.FLD(33,3,1)) GO TO 303
FLD(I,3,D(N+1+K))=FLD(33,3,1)
GO TO 302
303 FLD(I,3,D(N+1+K))=FLD(33,3,2)
302 K=K+NWPC
100 CONTINUE
C
C THE HEADER OF THE TEMPORARY ARRAY D IS SET AND ARRAYS C AND COMPTN
C
PC=K/NWPC
90 CALL WHEAD(D)
C
C DO LOOP 350 CONTROL THE DECODING OPERATION
C ( N = INDEX OF THE WORDS DECODED )
C ( NW = NUMBER OF WORDS IN ARRAY D )
C
DO 350 N=2,NW
DO 350 I=0,33,3
C
C IN ORDER TO SATISFIE THE CONVENTION ADOPTED FOR K BY PRINTAR
C THE OCTAL CODE WORD ASSIGNED BY SHARPS FOR K MUST BE CHANGED
C INTO 7 OCT. INSTEAD OF 3 OCT.
C
IF(FLD(I,3,D(N)).EQ.FLD(33,3,3)) FLD(I,3,D(N))=FLD(33,3,7)
350 CONTINUE

```


-MR M E ULUG-PH D THESIS-CARLETON UNIVERSITY-

```
      CALL COMPIN(C,D,MXC)
30    CONTINUE
C
C    SINCE EVERY CUBES OF A WERE PICKED, ABSORB ARRAY C AND IF EVERY
C    CUBES OF B WERE PICKED THEN C = A # B, OTHERWISE COPY C INTO A
C    AND SHARPS THE IB+1 TH CUBE
C
      CALL SUBSUM(C)
      NCA=FLD(18,9,C(1))
      IF(1B.EQ.NCB) GO TO 20
      CALL COPYAR(A,C,MXC)
      NC=0
      CALL WHEAD(C)
20    CONTINUE
C
C    NCC = THE NUMBER OF CUBES IN ARRAY C
C
      NCC=NCA
      RETURN
19    CALL WHEAD(C)
22    NCC=NC
      RETURN
21    CALL WHEAD(C)
      CALL SUBSUM(A)
      CALL COPYAR(C,A,MXC)
      GO TO 22
      END
```

MR M E ULUG-PH D THESIS-CARLETON UNIVERSITY-

```

/EL      3
SUBROUTINE COMBIN(A, B, MAX)
COMMON /QZ/I3,I6,I12,I33,I36,IBK,K1,K6,K7,KEY,IPCL(8),IRCL(7),
1 MASK(12),I,J,K,L,M,N,NWPC,NV,NC,NW,IA,IB,IC,NWA,NWB,NWC,MSK,IO(8)
INTEGER A(1), B(1)
CALL RHEAD(0)
J = NV
K = NC
L = NW
CALL RHEAD(A)
IF(L .EQ. 0) RETURN
IF(NW .EQ. 0) GO TO 5
IF(J .EQ. NV) GO TO 2
PRINT 1
1 FORMAT('COMBIN--UNEQUAL NUMBER OF VARIABLES')
CALL ERRORS
2 NC = NC + K
I = NW - 1
NW = I + L
3 IF(NW .GT. MAX) GO TO 7
DO 4 K = 2,L
4 A(K+I) = B(K)
CALL RHEAD(A)
RETURN
ENTRY COPYAR(A, B, MAX)
5 CALL RHEAD(0)
IF(NW .GT. MAX) GO TO 7
DO 6 I = 1,NW
6 A(I) = B(I)
CALL RHEAD(A)
RETURN
7 PRINT 8
8 FORMAT('COMBIN--ARRAY OVERFLOW.')
CALL ERRORS
END

```

-MR M E ULUG-PH D THESIS-CARLETON UNIVERSITY-

```

EL      3
        SUBROUTINE INTERX(A, B, C, MXC, NCC)
        COMMON /OZ/I3,I6,I12,I33,I36,IBK,K1,K6,K7,KEY,IPCL(8),IRCL(7),
1MASK(12),I,J,K,L,M,N,NWPC,NV,NC,NW,IA,IB,IC,NWA,NWR,NWC,MSK,IO(80)
        INTEGER A(1),B(1),C(1),CC(10)

C
C      SUBROUTINE INTERX SETS IN ARRAY C THE INTERSECTION
C      OF ARRAY A AND B
C      SET THE HEADER OF ARRAY C ASSUMING C EMPTY
C      ( NCC = NC = NUMBER OF CUBES IN C )
C
        NC=0
        NCC=0
        CALL WHEAD(C)
        CALL RHEAD(A)
        NCA=NC

C
C      TEST IF ARRAY A IS EMPTY, IF YES C IS EMPTY
C      ( NCA = NUMBER OF CUBES IN A )
C
        IF(NCA.EQ.0) GO TO 30
        CALL RHEAD(B)
        NCB=NC

C
C      TEST IF ARRAY B IS EMPTY, IF YES C IS EMPTY
C      ( NCB = NUMBER OF CUBES IN B )
C
        IF(NCB.EQ.0) GO TO 30

C
C      SET THE HEADER OF THE TEMPORARY ARRAY CC
C      ( NC = NUMBER OF CUBES IN ARRAY CC )
C
        NC=1
        CALL WHEAD(CC)

C
C      ILAST = THE POSITION OF THE FIRST BIT OF THE LAST BYTE
C      OF THE LAST WORD OF ANY CUBES OF ANY ARRAYS
C
        ILAST=(NV-1-(NWPC-1)*I12)*I3

C
C      DO LOOP 20 CONTROLS THE CUBES OF ARRAY A
C      ( IA = INDEX OF THE CUBES OF ARRAY A )
C      ( NCA = NUMBER OF CUBES IN ARRAY A )
C
        DO 20 IA=1,NCA

C
C      JA = THE INDEX OF THE FIRST WORD MINUS ONE OF THE IA TH CUBE OF A
C
        JA=IA*NWPC-NWPC+1

C
C      DO LOOP 20 CONTROLS THE CUBES OF ARRAY B
C      ( IB = INDEX OF THE CUBES OF ARRAY B )
C      ( NCB = NUMBER OF CUBES IN ARRAY B )
C
        DO 20 IB=1,NCB

C
C      JB = THE INDEX OF THE FIRST WORD MINUS ONE OF THE IB TH CUBE OF B

```

-MR M F ULUG-PH D THESIS-CARLETON UNIVERSITY-

```
C
      JB=IB*NWPC-NWPC+1
C
C      DO LOOP 50 CONTROLS THE WORDS OF THE PAIR OF CUBES INTERSECTED
C      A(N+JA) AND B(N+JB) TO BE INTERSECTED
C      ( N = THE RELATIVE INDEX OF THE WORDS )
C      ( NWPC = NUMBER OF WORDS BY CUBE )
C
      DO 50 N=1,NWPC
      L=AND(A(N+JA),B(N+JB))
      IF(N.EQ.NWPC) GO TO 60
      IMAX=33
      GO TO 70
60      IMAX=ILAST
C      DO LOOP 80 CONTROLS THE SEARCH FOR EMPTY INTERSECTION
C      ( I = POSITION OF THE FIRST BIT OF A BYTE )
C
70      DO 80 I=0,IMAX,3
C
C      TEST IF THE BYTES OF L = PHI , IF YES INTERSECTION IS EMPTY
C      OTHERWISE STORE L IN THE TEMPORARY ARRAY CC
C
      IF(FLD(I,3,L).EQ.FLD(0,3,0)) GO TO 90
80      CONTINUE
      CC(N+1)=L
50      CONTINUE
C
C      COMBIN THE INTERSECTION CUBE STORED IN CC WITH ARRAY C
C
      CALL COMBIN(C,CC,MXC)
      NCC=FLD(18,9,C(1))
20      CONTINUE
C
C      ABSORB THE INTERSECTION ARRAY C
C
30      CALL SUBSUM(C)
      RETURN
      END
```

-PRK 1 F ULUG-PU D THESIS-CARLETON UNIVERSITY-

```

1L      3
        SUBROUTINE ERRORS
C      I12 = 12 OCTAL ENCODED LOGIC ARRAY ELEMENTS (TRITS) PER WORD
C      I6 = 6 ALPHANUMERIC CHARACTERS PER WORD
C      I36 = 36 BITS PER WORD
C      I3 = 3 BITS PER TRIT
C      IBK BLANK CHARACTER
C      IPCL PRINTER CHARACTER LIST
C      IRCL READER CHARACTER LIST
C      NWPC NUMBER OF WORDS PER CUBE
C      NV NUMBER OF VARIABLES
C      NC NUMBER OF CUBES IN ARRAY
C      NW NUMBER OF WORDS IN ARRAY
COMMON /OZ/I3,I6,I12,I33,I36,IB,K1,K6,K7,KEY,IPCL(8),IRCL(7),
1MASK(12),I,J,K,L,M,N,NWPC,NV,NC,NW,IA,IB,IC,NWA,NWB,NWC,MSK,IO(80)
DATA I3/3/,I6/6/,I12/12/,I33/33/,I36/36/,Z/00/,IBK/' '/,KEY/052/
DATA K1/011111111111/, K6/066666666666/, K7/077777777777/
DATA IPCL/'0','0','0','0','0','0','1','X'/
DATA IRCL/'0','1','2','X','0','0','X'/
DATA MASK/07,070,0700,07000,070000,0700000,07000000,070000000,
1 0700000000,07000000000,070000000000,0700000000000/
        INTEGER A(1)
        GO TO 3
        ENTRY WHFAD(A)
        A(1) = 0
        FLD(0, 6, A(1)) = KEY
        IF(NV.EQ. 0) RETURN
        IF(NC.EQ. 0) RETURN
        IX = NV - 1
        IX = IX/I12
        NWPC = IX + 1
        IX = NWPC*NC
        NW = IX + 1
        FLD(6, 4, A(1)) = NWPC
        FLD(10, 8, A(1)) = NV
        FLD(18, 9, A(1)) = NC
        FLD(27, 9, A(1)) = NW
        RETURN
        ENTRY RHFAD(A)
        IX = FLD(0, 6, A(1))
        IF(IX.NE. KEY) GO TO 1
        NWPC = FLD(6, 4, A(1))
        NV = FLD(10, 8, A(1))
        NC = FLD(18, 9, A(1))
        NW = FLD(27, 9, A(1))
        IF(NV.EQ. 0) RETURN
        IF(NW.EQ. NWPC*NC+1) RETURN
1 PRINT 2
2 FORMAT('OINVALID ARRAY HEADER RECORD')
3 I = 0
  GO TO (1, 1), 1
  END

```

-MR M F ULUG-PH D THESIS-CARLETON UNIVERSITY-

```

EL      3
        SUBROUTINE PRNTAR(A, NIV)
        COMMON /OZ/I3,I6,I12,I33,I36,IBK,K1,K6,K7,KEY,IPCL(8),IRCL(7),
        IMASK(12),I,J,K,L,M,N,NWPC,NV,NC,NW,IA,IB,IC,NWA,NWR,NWC,MSK,IO(8)
        INTEGER A(1)
        N = NIV
        CALL RHEAD(A)
        IF(NC .EQ. 0) GO TO 5
        IF(N .LE. 0) N = NV
        IF(N .GT. NV) N = NV
        N = N + 1
        IO(N) = IBK
        M = NV + 1
        NW = 1
        DO 3 J = 1,NC
        T = 0
        NW = NW + 1
        DO 2 K = 1,M
        IF(K .EQ. N) GO TO 2
        L = FLD(I, I3, A(NW))
        IO(K) = IRCL(L)
        IF(K .LT. N) GO TO 1
        IF(L .EQ. 7) IO(K) = '--'
1      T = I + I3
        IF(I .LT. I36) GO TO 2
        IF(K .EQ. M) GO TO 2
        I = 0
        NW = NW + 1
2      CONTINUE
3      PRINT 4, (IO(K), K = 1,M)
4      FORMAT(1X, 80A1)
        RETURN
5      PRINT 6
6      FORMAT(/' EMPTY ARRAY'/)
        RETURN
        END

```

-IR M F ULUG-PH D THESIS-CARLETON UNIVERSITY-

```

EL      3
        SUBROUTINE READAR(A, NVAR, MAX)
        COMMON /QZ/I3,I6,I12,I33,I36,IBK,K1,K6,K7,KEY,IPCL(8),IRCL(7),
1 MASK(12),I,J,K,L,M,N,NWPC,NV,NC,NW,IA,IB,IC,NWA,NWR,NWC,MSK,IO(80)
        INTEGER A(1)
        NV = NVAR
        DO 1 I = 1,MAX
1  A(I) = 0
        NW = 1
        NC = -1
2  READ 3, IO
3  FORMAT(80A1)
        I = 0
        J = 0
        NC = NC + 1
        NW = NW + 1
        IF(NW .GT. MAX) GO TO 19
        DO 9 N = 1,80
        IF(IO(N) .EQ. IBK) GO TO 9
        K = 7
        DO 4 L = 1,5
        IF(IO(N) .EQ. IRCL(L)) GO TO (6, 7, 8, 8, 8), L
4  CONTINUE
        IF(J .EQ. 0) GO TO 21
        PRINT 5
5  FORMAT('READAR--ILLEGAL CHARACTER ON CARD')
        GO TO 11
6      K=1
        GO TO 8
7      K=2
8  FLD(I, I3, A(NW)) = K
        J = J + 1
        IF(J .EQ. NV) GO TO 2
        I = I + I3
        IF(I .LT. I36) GO TO 9
        I = 0
        NW = NW + 1
        IF(NW .GT. MAX) GO TO 19
9  CONTINUE
        PRINT 10
10  FORMAT('READAR--INSUFFICIENT VARIABLES ON CARD')
11  PRINT 12, IO
12  FORMAT(' ---', 80A1, ' ---')
13  CALL WHEAD(A)
        J = 0
14  READ 3, IO
        J = J + 1
        DO 15 N = 1,80
        IF(IO(N) .NE. IBK) GO TO 16
15  CONTINUE
        GO TO 14
16  DO 17 K = 1,5
        IF(IO(N) .EQ. IRCL(K)) GO TO 14
17  CONTINUE
        PRINT 18, J
18  FORMAT(I4, ' CARDS SKIPPED')

```

-IR M F ULUG-PH D THESIS-CARLETON UNIVERSITY-

```
      RETURN
19 PRINT 20
20 FORMAT('READER--ARRAY OVERFLOW')
      GO TO 13
21 CALL WHEAD(A)
      RETURN
      END
```

5!
5!
5!
5!
6!
6

-R 4 F ULUG-PH D THESIS-CARLETON UNIVERSITY-

```

EL      3
        SUBROUTINE CPYCUR(A, B, NCUBE)
        COMMON /QZ/I3,I6,I12,I33,I36,IBK,K1,K6,K7,KEY,IPCL(8),IRCL(7),
1 MASK(12),I,J,K,L,M,N,NWPC,NV,NC,NW,IA,IB,IC,NWA,NWB,NWC,MSK,IO(80)
        INTEGER A(1), B(1)
        MM = 0
        GO TO 1
        ENTRY MOVCUR(A, B, NCUBE)
        MM = 1
1 CALL RHEAD(A)
        IF(NCUBE .LE. 0) GO TO 5
        IF(NCUBE .GT. NC) GO TO 5
        J = NCUBE*NWPC - NWPC + 1
        DO 2 N = 1,NWPC
2 B(N+1) = A(N+J)
        K = NC
        L = NW
        NC = 1
        CALL WHEAD(B)
        IF(MM .EQ. 0) RETURN
        NW = L - NWPC
        NC = K - 1
        IF(NC .EQ. 0) GO TO 4
        IF(NCUBE .GT. NC) GO TO 4
        J = J + 1
        DO 3 N = J,NW
3 A(N) = A(N+NWPC)
4 CALL WHEAD(A)
        RETURN
5 NC = 0
        CALL WHEAD(B)
        PRINT 6
6 FORMAT('COPYCUR--NONEXISTENT CURVE, R ARRAY SET EMPTY')
        RETURN
        END

```

ACCOUNT: D8015

PROJECT: CADMDN

3.066 IN: I4 OUT: 0 PAGES: 18 COST: \$ 0.74

IME: 13:01:51-MAR 2,1972

IME: 13:03:25-MAR 2,1972

APPENDIX D

A COMPUTER PROGRAMME USING INTERRELATED SUBROUTINES

— F R M E ULUG-PH D THESIS-CARLETON UNIVERSITY—

```

3
SUBROUTINE OPDEC(AA,A,B,C,NV,NWPC,ISW1,ISW4)
INTEGER A(1),B(1),C(1),AA(1)

C
C   INITIALISATION OF ISW4 DEFINE BELOW
C
C   ISW4=0
C
C   SET THE HEADER OF C
C
C   C(1)=A(1)
C
C   NWPCP1 = TOTAL NUMBER OF WORDS IN CUBES A OR B OR C INCLUDING THE
C   HEADER
C
C   NWPCP1=NWPC+1
C
C   INITIALISATION OF THE COUNTER KSUB OF THE OCCURENCES OF
C   K(SHARPS)*I OR K(SHARPS)*J
C   THE COUNTER KSUB IS USED AS ONE OF THE TEST OF SUBSUMING
C   WHEN THE SHARPS OPERATOR IS USED
C
C   KSUB=0
C
C   INITIALISATION OF IPHIC THE COUNTER OF THE OCCURENCES OF PHI
C
C   IPHIC=0
C
C   ILAST=THE INDEX OF THE POSITION OF THE FIRST BIT OF THE LAST BYTE
C   OF THE LAST WORD OF ANY OF THE CUBES
C
C   ILAST=(NV-1-(NWPC-1)*12)*3
C
C   LOOP 20 EXECUTES THE OPERATIONS OVER THE CUBES ELEMENTS ACCORDING
C   TO THE OPERATOR CORRESPONDING TO ISW1
C
C   DO 20 N=2,NWPCP1
C   IF(N.EQ.NWPCP1) GO TO 21
C   IMAX=33
C   GO TO 22
21  IMAX=ILAST
22  DO 20 I=0,IMAX,3
C
C   COPY OF A INTO AA , WHERE AA IS USED WHEN NEW CUBES
C   ARE GENERATED AS ONE OF THE THREE POSSIBLE OUTPUT OF SHARPS OPERATOR
C
C   FLD(I,3,AA(N))=FLD(I,3,A(N))
C
C
C   COMPUTATIONS OF THE ELEMENTS OF C FOR THE FOLLOWING CASES
C   POSSIBLE FOR ALL OPERATORS
C   I*J=PHI      J*I=PHI
C   I*E=I        J*E=J
C   I*I=I        J*J=J
C
C   IF(FLD(I,3,A(N)).EQ.FLD(33,3,1)).AND.

```

-FROM E ULUG-PH D THESIS-CARLETON UNIVERSITY-

```

$FLD(I,3,P(N)).EQ.FLD(33,3,2)) GO TO 23
IF(FLD(I,3,A(N)).EQ.FLD(33,3,2).AND.
$FLD(I,3,P(N)).EQ.FLD(33,3,1)) GO TO 23
IF(FLD(I,3,A(N)).EQ.FLD(33,3,1).AND.
$FLD(I,3,P(N)).EQ.FLD(33,3,7)) GO TO 24
IF(FLD(I,3,A(N)).EQ.FLD(33,3,2).AND.
$FLD(I,3,P(N)).EQ.FLD(33,3,7)) GO TO 25
IF(FLD(I,3,A(N)).EQ.FLD(33,3,1).AND.
$FLD(I,3,P(N)).EQ.FLD(33,3,1)) GO TO 24
IF(FLD(I,3,A(N)).EQ.FLD(33,3,2).AND.
$FLD(I,3,P(N)).EQ.FLD(33,3,2)) GO TO 25

```

```

C
C COMPUTATION OF C FOR THE CASES PROPER TO A GIVEN OPERATORS
C
GO TO (26,27,27,28),ISW1

```

```

C
C COMPUTATIONS OF THE ELEMENTS OF C FOR THE CASES POSSIBLES ONLY FOR SU
C K(SUBSUM)*I=PHI
C K(SUBSUM)*J=PHI
C K(SUBSUM)*E=E
C
26 IF(FLD(I,3,B(N)).EQ.FLD(33,3,1).OR.
$ FLD(I,3,B(N)).EQ.FLD(33,3,2)) GO TO 23
GO TO 29

```

```

C
C COMPUTATION OF THE ELEMENTS OF C FOR THE CASES POSSIBLES ONLY
C IN INTERSECTION AND CONSENSUS
C
C E*I=I
C E*J=J
C E*E=E
C

```

```

27 IF(FLD(I,3,P(N)).EQ.FLD(33,3,1)) GO TO 24
IF(FLD(I,3,B(N)).EQ.FLD(33,3,2)) GO TO 25
GO TO 29

```

```

C
C COMPUTATION OF THE ELEMENT OF C FOR THE CASES POSSIBLES ONLY
C IN SHARPS OPERATION AND DECODING OF K(SHARPS) INTO E
C IN THE CURE AA AND ALSO IN THE CASE K(SHARPS)*E THE INCREMENTATION
C OF THE COUNTER KSUB
C K(SHARPS)*I=J
C K(SHARPS)*J=I
C K(SHARPS)*E=E
C

```

```

28 FLD(I,3,AA(3))=FLD(33,3,7)
KSUB=KSUR+1
IF(FLD(I,3,P(N)).EQ.FLD(33,3,1)) GO TO 25
IF(FLD(I,3,B(N)).EQ.FLD(33,3,2)) GO TO 24
KSUB=KSUR-1
GO TO 29

```

```

C
C SET PHI INTO C , INCREMENT COUNTER IPHC AND RECORD
C INDEX I AND N RESPECTIVELY INTO IPOS AND IWOR
C

```

```

23 IPHC=IPHC+1
IPOS=I
IWOR=N

```

-MR M E ULUG-PH D THESIS-CARLETON UNIVERSITY-

```

      FLD(I,3,C(N))=FLD(0,3,0)
      GO TO 20
C
C      SET I INTO C
C
24      FLD(I,3,C(N))=FLD(33,3,1)
      GO TO 20
C
C      SET J INTO C
C
25      FLD(I,3,C(N))=FLD(33,3,2)
      GO TO 20
C
C      SET E INTO C
C
29      FLD(I,3,C(N))=FLD(33,3,7)
C
C      EVALUATE CUBE C WHITH RESPECT TO THE OPERATORS
C
20      CONTINUE
      GO TO (30,31,30,32),ISW1
C
C      A SUBSUM P IF IPHIC=0 THEN SET ISW4=1
C      A DOES NOT SURSUM B IF IPHIC >0 THEN SET ISW4=0
C
C
C      A INTERSECTION B IS EMPTY IF IPHIC > 0 AND THEN SET ISW4=0
C      A INTERSECTION B IS NOT EMPTY IF IPHIC =0 AND THEN SET ISW4=1
C
30      IF(IPHIC.EQ.0) GO TO 34
      GO TO 35
C
C      A CONSENSUS B IS EMPTY IF IPHIC > 1 AND THEN SET ISW4=0
C      A CONSENSUS B IS NOT EMPTY IF IPHIC = 0, 1 AND THEN SET ISW4=1
C
31      IF(IPHIC.GT.1) GO TO 35
      IF(IPHIC.EQ.0) GO TO 34
C
C      DECODE THE PHI INTO E SINCE IPHIC=1
C
      FLD(IPOS,3,C(IWOR))=FLD(33,3,7)
      GO TO 34
C
C      A SHARPS P IS EQUAL TO A IF IPHIC > n ,THEN SET ISW4=1
C
32      IF(IPHIC.GT.0) GO TO 36
C
C      A SHARPS P IS EMPTY IF IPHIC = 0 AND KSUB = 0 ,THEN SET ISW4=0
C
      IF(KSUB.EQ.0) GO TO 35
C
C      A SHARPS P IS AN ARRAY OF NEW CUBES CONTAINING DEPENDING
C      OF THE CASES FROM 1 TO NV CUBES
C
      K=0
      DO 100 N=2,NWPCP1
      IF(N.EQ.NWPCP1) GO TO 101

```

-MR M F ULUG-PH D THESIS-CARLETON UNIVERSITY-

```
      TMAX=33
      GO TO 102
101    TMAX=ILAST
102    DO 100 I=0,TMAX,3
      IF((FLD(I,3,A(N)).EQ.FLD(33,3,3).AND.FLD(I,3,B(N)).EQ.FLD(33,3,7))
      S.OR.FLD(I,3,A(N)).NE.FLD(33,3,3)) GO TO 100
      DO 300 NN=2,NWPCP1
100    C(NN+K)=AA(NN)
      IF(FLD(I,3,B(N)).EQ.FLD(33,3,1)) GO TO 303
      FLD(I,3,C(N+K))=FLD(33,3,1)
      GO TO 302
103    FLD(I,3,C(N+K))=FLD(33,3,2)
102    K=K+NWPC
100    CONTINUE
C
C    SET THE HEADER OF ARRAY C
C
      FLD(18,9,C(1))=K/NWPC
      FLD(27,9,C(1))=K+1
34    ISW4=1
      GO TO 37
35    ISW4=0
C
C    SET THE HEADER OF C
C
      FLD(18,9,C(1))=0
      FLD(27,9,C(1))=1
      GO TO 37
C
C    COPY AA INTO C AS AS NEEDED IN THE CASE OF SHARPS, WHEN
C    A INTERSECTION B IS EMPTY
C
36    ISW4=1
      DO 38 NN=2,NWPCP1
38    C(NN)=AA(NN)
37    CONTINUE
      RETURN
      END
```

-MR M E ULUG-PH D THESIS-CARLETON UNIVERSITY-

```

      3
      SUBROUTINE FNC(D,NV,ISW1,ISW2)
      INTEGER D(1)
      NC=1
      IWP=2
12     IFP=0
      C
      C     TEST IF THE CUBE ELEMENT IS I = OCT. 1
      C     OR J = OCT. 2, IF YES SKIP ENCODING THE CUBE ELEMENT
      C
122    IF(FLD(IFP,3,D(IWP)).EQ.FLD(33,3,1).OR.
      S   FLD(IFP,3,D(IWP)).EQ.FLD(33,3,2)) GO TO 9
      C
      C     ENCODE AS A CUBE IF ISW2 = 1
      C     ENCODE AS B CUBE IF ISW2 = 2
      C
      C     IF(ISW2.EQ.2) GO TO 9
      C
      C     ECODE ACCORDING TO OPERATORS AS FOLLOWS
      C     SUBSUM..... K(SUBSUM) = OCT. 4
      C     CNSSUS..... E = OCT. 7
      C     INTERX..... E = OCT. 7
      C     SHARPS ..... K(SHARPS) = OCT. 3
      C
      C     GO TO (8,9,9,6),ISW1
      6   FLD(IFP,3,D(IWP))=FLD(33,3,3)
      C     GO TO 9
      8   FLD(IFP,3,D(IWP))=FLD(33,3,4)
      9   IF(NC.EQ.NV) GO TO 11
      C     IF(IFP.EQ.33) GO TO 13
      C     IFP=IFP+3
      C     NC=NC+1
      C     GO TO 122
13     IWP=IWP+1
      C     GO TO 12
11     CONTINUE
      RETURN
      END
```

```

3
SUBROUTINE PRINTER(A, NIV)
COMMON /OZ/I3,I6,I12,I33,I36,IBK,K1,K6,K7,KEY,IPCL(8),IRCL(5),
1 MASK(12),I,J,K,L,M,N,NWPC,NV,NC,NW,IA,IB,IC,NWA,NWB,NWC,MSK,IO(80)
INTEGER A(1)
N = NIV
CALL RHEAD(A)
IF(NC .EQ. 0) GO TO 5
IF(N .LE. 0) N = NIV
IF(N .GT. NV) N = NV
N = N + 1
IO(N) = IBK
M = NV + 1
NW = 1
DO 3 J = 1,IC
I = 0
NW = NW + 1
DO 2 K = 1,M
IF(K .EQ. N) GO TO 2
L = FLD(T, I3, A(NW))
IO(K) = IPCL(L+1)
IF(K .LT. N) GO TO 1
IF(L .EQ. 7) IO(K) = '- '
1 I = I + I3
IF(I .LT. I36) GO TO 2
IF(K .EQ. M) GO TO 2
I = 0
NW = NW + 1
2 CONTINUE
3 PRINT 4, (IO(K), K = 1,M)
4 FORMAT(1X, 80A1)
RETURN
5 PRINT 6
6 FORMAT(/' EMPTY ARRAY'/)
RETURN
END

```


1	
1	2
1	3
1	3
1	3
1	3
1	3
1	3
1	3
1	3
2	3
2	3
2	3
2	3
2	3
2	3
2	3
	4
	4
	4
	4
	4

-MR M E ULUG-PH D THESIS-CARLETON UNIVERSITY-

```
      J = J + 1
      DO 15 N = 1,80
      IF(I0(N) .NE. I2(K)) GO TO 16
15  CONTINUE
      GO TO 14
16  DO 17 K = 1,5
      IF(I0(N) .EQ. IRCL(K)) GO TO 14
17  CONTINUE
      PRINT 18, J
18  FORMAT(I4, ' CARDS SKIPPED')
      RETURN
19  PRINT 20
20  FORMAT('DREADAR--ARRAY OVERFLOW')
      GO TO 13
21  CALL WHEAD(A)
      RETURN
      END
```

4
4
4
4
5
5
5
5
5
5
5
5
5
6

-MR M E ULUG-PH D THESIS-CARLETON UNIVERSITY-

```

JEL      3
SUBROUTINE ERRORS
C      I12 = 12 OCTAL ENCODED LOGIC ARRAY ELEMENTS (TRITS) PER WORD
C      I6 = 6 ALPHANUMERIC CHARACTERS PER WORD
C      I30 = 36 BITS PER WORD
C      I3 = 3 BITS PER TRIT
C      IBK BLANK CHARACTER
C      IPCL PRINTER CHARACTER LIST
C      IRCL READER CHARACTER LIST
C      NWPC NUMBER OF WORDS PER CUBE
C      NV NUMBER OF VARIABLES
C      NC NUMBER OF CUBES IN ARRAY
C      NW NUMBER OF WORDS IN ARRAY
COMMON /OZ/I3,I6,I12,I33,I36,IBK,K1,K6,K7,KEY,IPCL(8),IRCL(5),
1MASK(12),I,J,K,L,M,N,NWPC,NV,NC,NW,IA,IB,IC,NWA,NWR,NWC,MSK,IO(80)
DATA I3/3/,I6/6/,I12/12/,I33/33/,I36/36/,I2/00/,IBK/' '/,KEY/052/
DATA K1/01111111111111/, K6/06666666666666/, K7/07777777777777/

C
C      MODIFICATION OF DIETMEYER ROUTINE
C      SUCH THAT ....
C      IPCL(3) = '1' BECAUSE ONE = OCT. 2
C
DATA IPCL/'0','0','1','0','0','0','0','1','X'/

C
C      END OF MODIFICATION
C
DATA IRCL/'0','1','2','X','D'/
DATA MASK/07,070,0700,07000,070000,0700000,07000000,070000000,
1 0700000000,07000000000,070000000000,0700000000000/
INTEGER A(1)
GO TO 3
ENTRY RHFEAD(A)
A(1) = 0
FLD(0, 6, A(1)) = KEY
IF(NV .EQ. 0) RETURN
IF(NC .EQ. 0) RETURN
IX = NV - 1
IX = IX/I12
NWPC = IX + 1
IX = NWPC*NC
NW = IX + 1
FLD(6, 4, A(1)) = NWPC
FLD(10, 8, A(1)) = NV
FLD(18, 9, A(1)) = NC
FLD(27, 9, A(1)) = NW
RETURN
ENTRY RHFEAD(A)
IX = FLD(0, 6, A(1))
IF(IX .NE. KEY) GO TO 1
NWPC = FLD(6, 4, A(1))
NV = FLD(10, 8, A(1))
NC = FLD(18, 9, A(1))
NW = FLD(27, 9, A(1))
IF(NV .EQ. 0) RETURN
IF(NW .EQ. NWPC*NC+1) RETURN
1 PRINT 2

```

-MR M E ULUG-PH D THESIS-CARLETON UNIVERSITY-

2 FORMAT('INVALID ARRAY HEADER RECORD')

3 I = 0

GO TO (1, 1), 1

END

4
4
5
5

-MR M F OLUG-PH D THESIS-CARLETON UNIVERSITY-

```

EL      3
        SUBROUTINE CPYCUB(A, B, NCUBE)
        COMMON /CZ/I3,I6,I12,I33,I36,IRK,K1,K6,K7,KEY,IPCL(8),IPCL(5),
1 MASK(12),I,J,K,L,M,N,NWPC,NV,NC,NW,IA,IB,IC,NWA,NWB,NWC,MSK,IO(80)
        INTEGER A(1), B(1)
        M = 0
        GO TO 1
        ENTRY MOVCUR(A, B, NCUBE)
        M = 1
1 CALL RHEAD(A)
        IF(NCUBE .LE. 0) GO TO 5
        IF(NCUBE .GT. NC) GO TO 5
        J = NCUBE*NWPC - NWPC + 1
        DO 2 N = 1,NWPC
2 B(N+1) = A(N+J)
        K = NC
        L = NW
        NC = 1
        CALL WHEAD(I)
        IF(M .EQ. 0) RETURN
        NW = L - NWPC
        NC = K - 1
        IF(NC .EQ. 0) GO TO 4
        IF(NCUBE .GT. NC) GO TO 4
        J = J + 1
        DO 3 N = J,NW
2 A(N) = A(N+NWPC)
3 CALL WHEAD(A)
        RETURN
5 NC = 0
        CALL WHEAD(I)
        PRINT 6
6 FORMAT('CPYCUB--NONEXISTENT CUBE, A ARRAY SET EMPTY')
        RETURN
        END

```

[illegible]

-MR M E ULUG-PH D THESIS-CARLETON UNIVERSITY-

```

3
INTEGER A(100),B(100),C(100),D(100),E(100),AA(100)

THIS PROGRAM IS AN EXAMPLE ILLUSTRATING THE USE OF THE MAIN LOGIC
OPERATORS. A VALUE FROM 1 TO 7 IS ASSIGNED TO ISW1,I.E.:

      ISW1 = 1 ABSORB                      - UNARY OPERATOR
      = 2 CONSENSUS                      - UNARY OPERATOR
      = 3 CUBE INTERSECTION              - BINARY OPERATOR
      = 4 SHARP PRODUCT                  - BINARY OPERATOR

READ INPUT PARAMETERS

      N      = NUMBER OF INPUT VARIABLES
      M      = NUMBER OF OUTPUT VARIABLES
      NV     = TOTAL NUMBER OF VARIABLES
      ISW1   = CODE OF THE LOGIC OPERATORS

READ 1,N,M,ISW1
FORMAT(3I2)
      MX     = DIMENSIONED SIZE OF ARRAY A,B,C,D

      MX=100
      NV=N+M
      PRINT 2,N,M,ISW1,NV,MX
2      FORMAT('PROGRAM LOGOP IS A DEMONSTRATION PROGRAM OF THE LOGIC OPE
$RATORS'/' INPUT PARAMETERS :/' N = ',I2,' M = ',I2,' ISW1 = ',I2
$, ' NV = ',I2,' MX = ',I4/' INPUT A - ARRAY'/)

      ARRAY A IS READ BY READAR IN ORDER TO BE OPERATED BY AN UNARY
      OPERATOR (ISW1 = 1 TO 2) OR TO BE OPERATED AS THE LEFT SIDE ARRAY
      BY A BINARY OPERATOR (ISW1 = 3 TO 4).ARRAY A IS PRINTED BY PRNTAR

      CALL READAR(A,NV,MX)
      CALL PRNTAR(A,N)

      ACCORDING TO ISW1 ,THE RIGHT SIDE ARRAY B OPERATED BY A BINARY
      OPERATOR IS READ BY READAR AND PRINTED BY PRINTAR
      IF (ISW1.LE.2) GO TO 3
      PRINT 13
13     FORMAT('/' INPUT B - ARRAY'/)
      CALL READAR(B,NV,MX)
      CALL PRNTAR(B,N)
      GO TO(4,5,8,9),ISW1

      ARRAY A IS ABSORBED AND REPLACED IN A * A <-- A(A) *
4      CALL SUBSUM(A,B,C,D,E,AA)
      PRINT 15

```

-MR M E ULUG-PH D THESIS-CARLETON UNIVERSITY-

```

15  FORMAT(/' ABSORB - UNARY OPERATOR : A <-- A(A)'/
    $' ABSORBED ARRAY A'/)
    GO TO 16

C
C  THE CUBES OF ARRAY A ARE CONVERTED INTO PRIME IMPLICANTS USING
C  THE ITERATIVE CONSENSUS ALGORITHM, ASSUMING BARTEE CODING, I.E.
C  FOR M>L EACH '1' IN THE M-UPLES PART OF EACH CUBE OF A FUNCTION
C  ARRAY IS REPLACED WITH AN 'X' ' A <-- *(A) '
C
5    CALL CNSSUS(A,B,C,D,E,AA,MX)
    PRINT 17
17  FORMAT(/' CONSENSUS - UNARY OPERATOR : A <-- *(A)'/
    $' PRIME IMPLICANTS ARRAY'/)
    GO TO 16

C
C  THE INTERSECTION OF ARRAYS A AND B IS PLACED IN ARRAY C ' C <-- A\B
C  NCC RETURNS THE NUMBER OF CUBES IN ARRAY C
C
8    CALL INTERX(A,B,C,MX,NCC,AA,D,E)
    PRINT 20,NCC
20  FORMAT(/' CUBE INTERSECTION - BINARY OPERATOR : C <-- A\B'/
    $' NCC = ',I3/)
    GO TO 24

C
C  THE SHARP PRODUCT OF ARRAY A AND B IS PLACED IN ARRAY C ' C <-- A#B '
C  NCC RETURNS THE NUMBER OF CUBES IN ARRAY C
C
9    CALL SHAPPS(A,B,C,D,MX,NCC,E,AA)
    PRINT 26,NCC
26  FORMAT(/' SHARP PRODUCT - BINARY OPERATOR : C <-- A#B'/
    $' NCC = ',I3/)
    GO TO 24
16  CALL PRNTAR(A,N)
    GO TO 23
24  CALL PRNTAR(C,N)
23  CALL EXIT
    END

```


-MR M E ULUG-PI D THESIS-CARLETON UNIVERSITY-

```

EL      3
        SUBROUTINE SUBSUM(A,B,C,D,E,AA)
        COMMON /OZ/I3,I6,I12,I33,I36,IBK,K1,K6,K7,KEY,IPCL(8),IRCL(5),
1MASK(12),I,J,K,L,M,N,NWPC,NV,NC,NW,IA,IB,IC,NWA,NWR,NWC,MSK,IO(80)
        INTEGER A(1),B(1),C(1),D(1),E(1),AA(1)
        CALL RHEAD(A)
        NWPCA=NWPC
        NWA=NW
        IF(NC.LE.1) RETURN
        LA=NWA-NWPCA
        IAA=0
        DO 4 IA=2,LA,NWPCA
        IAA=IAA+1
        IF(A(IA).EQ.0) GO TO 4
        CALL CPYCUB(A,B,IAA)
        NA=IA+NWPCA
        JAA=IAA+1
        DO 2 JA=NA,NWA,NWPCA
        IF(A(JA).EQ.0) GO TO 2
        CALL CPYCUB(A,C,JAA)
        CALL ENC(B,NV,1,1)
        CALL ENC(C,NV,1,2)
        CALL OPDEC(AA,B,C,D,NV,NWPC,1,ISW4)
        IF(ISW4.EQ.1) GO TO 5
        CALL ENC(C,NV,1,1)
        CALL ENC(B,NV,1,2)
        CALL OPDEC(AA,C,B,D,NV,NWPC,1,ISW4)
        IF(ISW4.EQ.1) GO TO 6
        GO TO 2
6        A(JA)=0
2        JAA=JAA+1
        GO TO 4
5        A(IA)=0
4        CONTINUE
        NC = 0
        J = 2
        DO 7 I=2,NW,NWPCA
        IF(FLD(0,18,A(I)).EQ.FLD(0,18,0).AND.
$FLD(17,18,A(I)).EQ.FLD(0,18,0)) GO TO 7
        DO 8 K = 1,NWPCA
8        A(J+K-1) = A(I+K-1)
        NC = NC + 1
        J = J + NWPCA
7        CONTINUE
        CALL WHEAD(A)
        RETURN
        END

```

2
2

3
3
3
3

3
3
3

-MR M E ULUG-PH D THESIS-CARLETON UNIVERSITY-

```

/EL      3
SUBROUTINE INTERX(A,B,C,MXC,NCC,AA,n,E)
COMMON /OZ/I3,I6,I12,I33,I36,IBK,K1,K6,K7,KEY,IPCL(8),IRCL(5),
1MASK(12),I,J,K,L,M,N,NWPC,NV,NC,NW,IA,IB,IC,NWA,NWR,NWC,MSK,IO(80)
INTEGER A(1),B(1),C(1),D(1),E(1),AA(1),CC(10)
NC=0
CALL WHEAD(C)
CALL RHEAD(A)
NWPCA=NWPC
NWA=NW
NCA=NC
IF(NCA.EQ.0) GO TO 10
CALL RHEAD(B)
NWPCB=NWPC
NCB=NC
NWB=NW
IF(NCB.EQ.0) GO TO 10
DO 20 IA=1,NCA
CALL CPYCUB(A,D,IA)
CALL ENC(D,NV,2,1)
DO 20 IB=1,NCB
CALL CPYCUB(B,E,IB)
CALL ENC(E,NV,2,2)
CALL OPDEC(AA,D,E,CC,NV,NWPC,3,TSW4)
IF(ISW4.FQ.0) GO TO 20
CALL COMPIN(C,CC,MXC)
NCC=FLD(18,9,C(1))
20  CONTINUE
30  CALL SUBSUM(C,B,A,D,E,AA)
RETURN
10  NC=0
NCC=0
CALL WHEAD(C)
GO TO 30
END

```

-R P M F ULUG-PH D THESIS-CARLETON UNIVERSITY-

```

1EL      3
          SUBROUTINE SHARPS(A,BB,C,D,MXC,NCC,F,AA)
          COMMON /QZ/I3,I6,I12,I33,I36,IBK,K1,K6,K7,KEY,IPCL(8),IRCL(5),
1MASK(12),I,J,K,L,M,N,NWPC,NV,NC,NW,IA,IB,IC,NWA,NWB,NWC,MSK,IO(80)
          INTEGER A(1),B(1),C(1),D(1),E(1),AA(1),CC(100),BB(100)
          NC=0
          CALL WHEAD(C)
          CALL RHEAD(A)
          IF(NC.EQ.0) GO TO 19
          NCA=NC
          CALL RHEAD(BB)
          IF(NC.EQ.0) GO TO 21
          NCB=NC
          DO 20 IB=1,NCB
          CALL CPYCUB(BB,D,IB)
          CALL ENC(D,NV,4,2)
          DO 30 IA=1,NCA
          CALL CPYCUB(A,E,IA)
          CALL ENC(E,NV,4,1)
          CALL OPDEC(AA,E,D,CC,NV,NWPC,4,TSW4)
          IF(ISW4.EQ.0) GO TO 30
          CALL COMBIN(C,CC,MXC)
30        CONTINUE
          CALL SUBSUM(C,A,B,D,E,AA)
          NCA=FLD(18,9,C(1))
          IF(IB.EQ.NCB) GO TO 20
          CALL COPYAR(A,C,MXC)
          NC=0
          CALL WHEAD(C)
20        CONTINUE
          NCC=NCA
          RETURN
19        CALL WHEAD(C)
22        NCC=NC
          RETURN
21        CALL WHEAD(C)
          CALL SUBSUM(A,B,C,D,E,AA)
          CALL COPYAR(C,A,MXC)
          GO TO 22
          END

```

-PR M E ULUG-PH D THESIS-CARLETON UNIVERSITY-

```

VEL      3
          SUBROUTINE CNSSUS(A,B,C,D,E,AA,MXA)
          COMMON /OZ/I3,I6,I12,I35,I36,IBK,K1,K6,K7,KEY,IPCL(8),IPCL(5),
1MASK(12),I,J,K,L,M,N,NWPC,NV,NC,NW,IA,IB,IC,NWA,NWR,NWC,MSK,IO(8)
          INTEGER A(1),B(1),C(1),D(1),E(1),AA(1),BB(100),CC(100)
          CALL RHEAD(A)
          IF(NC.LE.1) RETURN
          NCA=NC
20        NCAM1=NCA-1
          DO 10 II=1,NCAM1
            NN=II+1
            CALL CPYCUB(A,BB,II)
            CALL ENC(BB,NV,3,1)
            DO 10 JJ=NN,NCA
              CALL CPYCUB(A,C,JJ)
              CALL ENC(C,NV,3,2)
              CALL OPDFC(AA,BB,C,D,NV,NWPC,2,TSW4)
              IF(ISA4.FQ.0) GO TO 10
            CALL COMBIN(A,D,MXA)
10        CONTINUE
          CALL SUBSUM(A,B,C,D,E,AA)
          CALL RHEAD(A)
          NCA=NC
          DO 30 KK=1,NV
            IF(A(KK).NE.CC(KK)) GO TO 40
30        CONTINUE
          RETURN
40        CALL COPYAR(CC,A,MXA)
          GO TO 20
          END

```

ACCOUNT: D8015

PROJECT: CADMDN

12.941 IN: 16 OUT: 0 PAGES: 19 COST: \$ 0.77

TIME: 12:48:06-MAR 2,1972

IME: 12:49:03-MAR 2,1972

APPENDIX E

IDEMPOTENT SCALAR QUATERNIANS

APPENDIX E

IDEMPOTENT SCALAR QUATERNIANS

Let the set G be $X_{i=1}^4 Z_2^{(i)}$, the external direct product of Z_2 under addition with itself four times. Let us re-name certain elements of G. we shall let

$$1 = (1, 0, 0, 0), i = (0, 1, 0, 0), j = (0, 0, 1, 0), k = (0, 0, 0, 1)$$

We also let

$$a_1 = (a_1, 0, 0, 0), a_2 i = (0, a_2, 0, 0), a_3 j = (0, 0, a_3, 0) \text{ and } a_4 k = (0, 0, 0, a_4)$$

we then have

$$(a_1, a_2, a_3, a_4) = a_1 + a_2 i + a_3 j + a_4 k \text{ where } a_i \in Z_2$$

thus if

$$\alpha = a_1 + a_2 i + a_3 j + a_4 k \quad \text{and}$$

$$\alpha' = a'_1 + a'_2 i + a'_3 j + a'_4 k$$

We define

$$\alpha + \alpha' = (a_1 + a'_1) + (a_2 + a'_2)i + (a_3 + a'_3)j + (a_4 + a'_4)k$$

Multiplication on G is defined as follows:

$$1 \cdot a_i = a_i \cdot 1 = a_i \quad \text{and} \quad 0 \cdot a_i = a_i \cdot 0 = 0$$

and using u_x, u_y , and u_z for i, j, k

$$0 \cdot u_i = u_i \cdot 0 = 0 \quad \text{and} \quad 1 \cdot u_i = u_i \cdot 1 = 1$$

and

$$u_i \cdot u_j = \delta_i^j \quad \text{where} \quad \delta_i^j = \begin{cases} u_i & \text{if } i = j \\ 0 & \text{if } i \neq j \end{cases}$$

δ_i^j is similar to Kronocker's delta, a mixed tensor of rank two, and represents the idempotent scalar property of i, j, k .

It also implies that

$$i + i = 0, \quad j + j = 0, \quad k + k = 0$$

Now we shall show that $\langle X_{i=1}^4 Z_2^{(i)}, +, \cdot \rangle$ is a sixteen element Boolean ring with unity. The elements of the set $G = X_{i=1}^4 Z_2^{(i)}$ are the vertices of a four dimensional cube. Using the binary notation these vertices will be numbered

0 - 15 as follows:

(0, 0, 0, 0) → 0	(1, 0, 0, 0) → 8
(0, 0, 0, 1) → 1	(1, 0, 0, 1) → 9
(0, 0, 1, 0) → 2	(1, 0, 1, 0) → 10
(0, 0, 1, 1) → 3	(1, 0, 1, 1) → 11
(0, 1, 0, 0) → 4	(1, 1, 0, 0) → 12
(0, 1, 0, 1) → 5	(1, 1, 0, 1) → 13
(0, 1, 1, 0) → 6	(1, 1, 1, 0) → 14
(0, 1, 1, 1) → 7	(1, 1, 1, 1) → 15

Since the addition is by components the additive closure follows from the idempotent scalar property of i, j, k and from the fact that $a_i \in Z_2$. From this it is clear that all sixteen elements are of order two.

There exists an additive identity element "e" such that

$$e = 0 + 0 \cdot i + 0 \cdot j + 0 \cdot k$$

This is the vertex designated 0, namely (0, 0, 0, 0)

Every element is its own additive inverse. Namely

$$(\alpha + \alpha) = (a_1 + a_1) + (a_2 + a_2)i + (a_3 + a_3)j + (a_4 + a_4)k$$

since $a_i \in Z_2$

$$\alpha + \alpha = 0$$

Additive associativity can be shown as follows:

$$\alpha + (\alpha' + \alpha'') = [a_1 + (a'_1 + a''_1)] + [a_2 + (a'_2 + a''_2)]i + [a_3 + (a'_3 + a''_3)]j + [a_4 + (a'_4 + a''_4)]k$$

since $\{a_i, a'_i, a''_i\} \in Z_2$

$$\alpha + (\alpha' + \alpha'') = [(a_1 + a'_1) + a''_1] + [(a_2 + a'_2) + a''_2]i + [(a_3 + a'_3) + a''_3]j + [(a_4 + a'_4) + a''_4]k = (\alpha + \alpha') + \alpha''$$

Hence

$$\alpha + (\alpha' + \alpha'') = (\alpha + \alpha') + \alpha''$$

The addition is also commutative since

$$\alpha + \alpha' = [a_1 + a'_1] + [a_2 + a'_2]i + [a_3 + a'_3]j + [a_4 + a'_4]k$$

$$\alpha + \alpha' = [a'_1 + a_1] + [a'_2 + a_2]i + [a'_3 + a_3]j + [a'_4 + a_4]k$$

$$\alpha + \alpha = \alpha' + \alpha$$

Hence $\langle X_{i=1}^4 Z_2^{(i)}, + \rangle$ is an abelian group.

Multiplicative closure can be shown as follows:

$$\alpha_1 \cdot \alpha_2 = (a_1 + a_2i + a_3j + a_4k) \cdot (a'_1 + a'_2i + a'_3j + a'_4k)$$

$$\alpha_1 \cdot \alpha_2 = a_1 a'_1 + (a_2 a'_2 + a_1 a'_2 + a_2 a'_1)i + (a_3 a'_3 + a_1 a'_3 + a_3 a'_1)j + (a_4 a'_4 + a_1 a'_4 + a_4 a'_1)k \dots (1)$$

Since $\{a_i, a'_i\} \in Z_2$, multiplication is closed and commutative.

There exists a unity such that

$$u = 1 + 0 \cdot i + 0 \cdot j + 0 \cdot k$$

This is the vertex designated 8, namely (1, 0, 0, 0).

Since $\{a_i, a'_i\} \in Z_2$ there is no multiplicative inverse.

Multiplicative Associativity can be proven as follows: From the equation (1)

we have

$$\alpha_1 \cdot \alpha_2 = a_1 \cdot a'_1 + Ai + Bj + Ck$$

where

$$A = (a_2 \cdot a_2' + a_1 a_2' + a_2 a_1')$$

$$B = (a_3 a_3' + a_1 a_3' + a_3 a_1')$$

$$C = (a_4 \cdot a_4' + a_1 a_4' + a_4 a_1')$$

$$(\alpha_1 \cdot \alpha_2) \alpha_3 = (a_1 a_1' + A i + B j + C k) (a_1'' + a_2'' i + a_3'' j + a_4'' k) \dots (2)$$

We shall show that

$$(\alpha_1 \cdot \alpha_2) \alpha_3 = \alpha_1 (\alpha_2 \cdot \alpha_3) = (a_1 + a_2 i + a_3 j + a_4 k) (a_1' a_1'' + A' i + B' j + C' k) \dots (3)$$

Where

$$A' = a_2' a_2'' + a_1' a_2'' + a_2' a_1''$$

$$B' = a_3' a_3'' + a_1' a_3'' + a_3' a_1''$$

$$C' = a_4' a_4'' + a_1' a_4'' + a_4' a_1''$$

We find that the products of the first terms in each bracket of the equations (2) and (3) are equal, namely $(a_1 a_1') : a_1'' = a_1 \cdot (a_1' \cdot a_1'')$

Since

$$\{a_i, a_i', a_i''\} \in Z_2$$

Continuing with multiplication we find that the coefficient of i in equation (2) is given by

$$A a_2'' + a_1 a_1' a_2'' + a_1'' A = A (a_2'' + a_1'') + a_1 a_1' a_2'' \dots (4)$$

Similarly the coefficient of A in equation 3 is given by

$$a_2 A' + a_1' a_1'' a_2 + a_1 A' = A' (a_2 + a_1) + a_1' a_1'' a_2 \dots (5)$$

Substituting the value of A in equation (4) we get

$$\begin{aligned} & (a_2 \cdot a_2' + a_1 a_2' + a_2 a_1') (a_2'' + a_1'') + a_1 a_1' a_2'' = \\ & a_2 a_2' a_2'' + a_2 a_2' a_1'' + a_1 a_2' a_2'' + a_1 a_2' a_1'' + a_2 a_1' a_2'' \\ & a_2 a_1' a_1'' + a_1 a_1' a_2'' = a_1' a_1'' a_2 + (a_2 + a_1) [a_2' a_2'' + a_2' a_1'' + a_1' a_2''] \\ & = a_1' a_1'' a_2 + A' (a_2 + a_1) \end{aligned}$$

Hence the coefficient of i in equations (2) and (3) are identical.

Similarly it can be shown that the coefficients of j and k in equations (2) and (3) are also identical. Hence $(\alpha_1 \cdot \alpha_2) \alpha_3 = \alpha_1 \cdot (\alpha_2 \cdot \alpha_3)$

The proof of the left distributive law under addition is as follows:

$$\begin{aligned} \alpha_1 (\alpha_2 + \alpha_3) &= (a_1 + a_2 i + a_3 j + a_4 k) [(a_1' + a_1'') + (a_2' + a_2'') i \\ &+ (a_3' + a_3'') j + (a_4' + a_4'') k] = [a_1 a_1' + (a_2 a_2' + a_1 a_2' + a_2 a_1') i \\ &+ (a_3 a_3' + a_1 a_3' + a_3 a_1') j + (a_4 a_4' + a_1 a_4' + a_4 a_1') k] \\ &+ [(a_1 a_1'' + (a_2 a_2'' + a_1 a_2'' + a_2 a_1'') i + (a_3 a_3'' + a_1 a_3'' + a_3 a_1'') j \\ &+ (a_4 a_4'' + a_1 a_4'' + a_4 a_1'') k] = \alpha_1 \cdot \alpha_2 + \alpha_1 \cdot \alpha_3 \end{aligned}$$

the right distributive law under addition can be demonstrated in a similar manner.

Thus it has been shown that $\langle X_{i=1}^4 Z_2^{(i)}, +, \cdot \rangle$ is a sixteen element Boolean ring with unity. The abelian group $\langle X_{i=1}^4 Z_2^{(i)}, + \rangle$ and abelian semigroup with unit $\langle X_{i=1}^4 Z_2^{(i)}, \cdot \rangle$ are given in tables XXXI and XXXII.

Inspection of the table XXXI reveals that abelian group $\langle X_{i=1}^4 Z_2^{(i)}, + \rangle$ splits into blocks with the induced operation well defined and the blocks form a group under this induced operation. Then the block containing the identity "e" must be a subgroup of $\langle X_{i=1}^4 Z_2^{(i)}, + \rangle$. Since $\langle G, + \rangle$ is an abelian group, its subgroup is a normal subgroup. Let us call this normal subgroup K, hence

$$K = \{0, 1, 2, 3\} \quad \text{or}$$

$$K = \{(0, 0, 0, 0), (0, 0, 0, 1), (0, 0, 1, 0), (0, 0, 1, 1)\}$$

By inspecting of the table XXXI it can be seen that $\langle K, + \rangle$ is isomorphic $Z_2 \times Z_2$.

The number of cosets of K in G is given by Lagrange theorem.

$$|G| = |K| \cdot [G : K]$$

where $[G : K]$ is the number of right (left) cosets of K in G and called the "index" of K in G.

Hence

$$|G| = 4 \cdot [G : K]$$

$$[G : K] = 4$$

These cosets are as follows

$$K = \{0, 1, 2, 3\}$$

$$4 + K = \{4, 5, 6, 7\}$$

$$8 + K = \{8, 9, 10, 11\}$$

$$12 + K = \{12, 13, 14, 15\}$$

As can be seen from the table XXXI each coset forms a group isomorphic to $Z_2 \times Z_2$. Moreover the factor group of $\langle X_{i=1}^4 Z_2^{(i)}, + \rangle$ modulo K, denoted by "G/K" is isomorphic to $Z_2 \times Z_2$.

The group formed by the coset $\{12, 13, 14, 15\}$ is of particular interest to us since the vertex designated 14 contains the elements of the set $B = \{1, i, j, 0\}$ which is used in the ring structure of the unified theory. The elements of this group are as follows:

$$(1, 1, 0, 0), (1, 1, 0, 1), (1, 1, 1, 0), (1, 1, 1, 1)$$

As it was mentioned before the sixteen elements of G are the vertices of a four dimensional cube. Each coset represent a face of this cube as shown below.

+	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	1	0	3	2	5	4	7	6	9	8	11	10	13	12	15	14
2	2	3	0	1	6	7	4	5	10	11	8	9	14	15	12	13
3	3	2	1	0	7	6	5	4	11	10	9	8	15	14	13	12
4	4	5	6	7	0	1	2	3	12	13	14	15	8	9	10	11
5	5	4	7	6	1	0	3	2	13	12	15	14	9	8	11	10
6	6	7	4	5	2	3	0	1	14	15	12	13	10	11	8	9
7	7	6	5	4	3	2	1	0	15	14	13	12	11	10	9	8
8	8	9	10	11	12	13	14	15	0	1	2	3	4	5	6	7
9	9	8	11	10	13	12	15	14	1	0	3	2	5	4	7	6
10	10	11	8	9	14	15	12	13	2	3	0	1	6	7	4	5
11	11	10	9	8	15	14	13	12	3	2	1	0	7	6	5	4
12	12	13	14	15	8	9	10	11	4	5	6	7	0	1	2	3
13	13	12	15	14	9	8	11	10	5	4	7	6	1	0	3	2
14	14	15	12	13	10	11	8	9	6	7	4	5	2	3	0	1
15	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

$$\langle X_{i=1}^4 \mathbb{Z}_2, + \rangle$$

Abelian Group

Table XXXI

.	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	1	0	1	0	1	0	1	1	0	1	0	1	0	1	0
2	0	0	2	2	0	0	2	2	2	2	0	0	2	2	0	0
3	0	1	2	3	0	1	2	3	3	2	1	0	3	2	1	0
4	0	0	0	0	4	4	4	4	4	4	4	4	0	0	0	0
5	0	1	0	1	4	5	4	5	5	4	5	4	1	0	1	0
6	0	0	2	2	4	4	6	6	6	6	4	4	2	2	0	0
7	0	1	2	3	4	5	6	7	7	6	5	4	3	2	1	0
8	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
9	0	0	2	2	4	4	6	6	9	9	11	11	13	13	15	15
10	0	1	0	1	4	5	4	5	10	11	10	11	14	15	14	15
11	0	0	0	0	4	4	4	4	11	11	11	11	15	14	15	15
12	0	1	2	3	0	1	2	3	12	13	14	15	12	13	14	15
13	0	0	2	2	0	0	2	2	13	13	15	14	13	13	15	15
14	0	1	0	1	0	1	0	1	14	15	14	15	14	15	14	15
15	0	0	0	0	0	0	0	0	15	15	15	15	15	15	15	15

$$\langle X_{i=1}^4 Z_2, \cdot \rangle$$

Abelian Semi group with unity

Table XXXII

Fig 4

Hence

K = {0, 1, 2, 3}	→	0 0 X X
4 + K = {4, 5, 6, 7}	→	0 1 X X
8 + K = {8, 9, 10, 11}	→	1 0 X X
12 + K = {12, 13, 14, 15}	→	1 1 X X

As can be seen the cosets $\{0, 1, 2, 3\}$ and $\{4, 5, 6, 7\}$ form the two opposite faces of the 4D. cube. Similarly the cosets $\{8, 9, 10, 11\}$ and $\{12, 13, 14, 15\}$ form another set of opposite faces.

Fig 4. Cosets of $\langle X_{i=1}^4 Z_2^{(i)}, + \rangle$

Let us now investigate the quotient rings of $\langle G, +, \cdot \rangle$. By inspecting the tables XXXI and XXXII it can be readily seen that 16 element Boolean ring with unity splits up into four blocks with induced operations well defined.

Let

$$N = \{0, 1, 2, 3, 4, 5, 6, 7\} \quad \text{and}$$

$$M = \{8, 9, 10, 11, 12, 13, 14, 15\}$$

Let us show that N is an ideal of the 16 element Boolean ring with unity. For this, let us first show that $\langle N, +, \cdot \rangle$ is a subring of $\langle G, +, \cdot \rangle$. N is a subring of G , if and only if, $a - b \in N$ and $a \cdot b = b \cdot a \in N$ for every $a, b \in N$. The first condition, $a - b \in N$ for every $a, b \in N$, implies that $\langle N, + \rangle$ is an additive subgroup of $\langle G, +, \cdot \rangle$. This can be shown with the help of table XXXI. As can be seen $\langle N, + \rangle$ is Abelian groupoid of order eight with an identity element $e = 0$. Also each element is its own inverse. The associativity can be demonstrated as follows:

$$\begin{array}{ll} \text{(a)} & 3 + (7 + 6) = 3 + 1 = 2 \quad \quad \quad \text{(b)} \quad 7 + (2 + 4) = 7 + 6 = 1 \\ & (3 + 7) + 6 = 4 + 6 = 2 \quad \quad \quad (7 + 2) + 4 = 5 + 4 = 1 \\ \\ \text{(c)} & 5 + (3 + 1) = 5 + 2 = 7 \quad \quad \quad \text{(d)} \quad 6 + (5 + 7) = 6 + 2 = 4 \\ & (5 + 3) + 1 = 6 + 1 = 7 \quad \quad \quad (6 + 5) + 7 = 3 + 7 = 4 \\ \\ \text{(e)} & 1 + (7 + 2) = 1 + 5 = 4 \\ & (1 + 7) + 2 = 6 + 2 = 4 \end{array}$$

Hence $\langle N, + \rangle$ is an Abelian group of order eight. There are three Abelian groups of order eight up to isomorphism, namely Z_8 , $Z_4 \times Z_2$, and $Z_2 \times Z_2 \times Z_2$. By inspection it can be seen that $\langle N, + \rangle$ is not a cyclic group. It is well known that every element ($\neq 1$) of $Z_2 \times Z_2 \times Z_2$ is of order two, while $Z_2 \times Z_4$ has at least one element of order four. Since all of its elements are idempotent, $\langle N, + \rangle$ must be isomorphic to $Z_2 \times Z_2 \times Z_2$. Since N is a sub-set of G , such that the group operation G is closed on N , and N is a group under this induced operation, $\langle N, + \rangle$ is an additive subgroup of $\langle G, +, \cdot \rangle$. The second requirement, $a \cdot b = b \cdot a \in N$ for every $a, b \in N$, is also satisfied. As can be seen from the table XXXII the elements of $\langle N, \cdot \rangle$ are symmetrical about the diagonal. Hence $\langle N, +, \cdot \rangle$ is a subring of $\langle G, +, \cdot \rangle$.

Now for N to be an ideal of $\langle G, +, \cdot \rangle$ it must be shown that $b \cdot N \subseteq N$ and $N \cdot b \subseteq N$ for every $b \in G$. Again this can be readily seen by inspecting the table XXXII.

Hence the ideal N and its coset M split the 16 element Boolean ring with unity into four blocks with the induced operation well defined. By observing the tables XXXI and XXXII and comparing them with the tables XXIX and XXX it can be seen that the resultant quotient ring, G/N , is isomorphic to Z_2 field. Therefore N is a maximal and prime ideal of $\langle G, +, \cdot \rangle$.

Previously it was pointed out that the factor group of $\langle G, + \rangle$ modulo K , G/K , was isomorphic to $Z_2 \times Z_2$. By inspecting the table XXXII it can be seen that multiplicative part of the $\langle G, +, \cdot \rangle$ also splits into sixteen blocks with the induced operations well defined.

Let us show that $K = \{0, 1, 2, 3\}$ is also an ideal of the sixteen element Boolean ring with unity. For this let us first show that $\langle K, +, \cdot \rangle$ is a subring of $\langle G, +, \cdot \rangle$. K is a subring of G , if and only if, $a - b \in K$ and $a \cdot b = b \cdot a \in K$ for every $a, b \in K$. The first condition, $a - b \in K$ for every $a, b \in K$, implies that $\langle K, + \rangle$ is an additive subgroup of $\langle G, +, \cdot \rangle$. It was already shown that $\langle K, + \rangle$ was isomorphic to the Klein 4-group. Since K is a subset of G , such that the group operation G is closed on K , and K is a group under this induced operation, $\langle K, + \rangle$ is an additive subgroup of $\langle G, +, \cdot \rangle$.

The second requirement, $a \cdot b = b \cdot a \in K$ for every $a, b \in K$, is also satisfied. As can be seen from the table XXXII the elements of $\langle K, \cdot \rangle$ are symmetrical about the diagonal. Hence $\langle K, +, \cdot \rangle$ is a subring of $\langle G, +, \cdot \rangle$.

Now for K to be an ideal of $\langle G, +, \cdot \rangle$ it must be shown that $b \cdot K \subseteq K$ and $K \cdot b \subseteq K$ for every $b \in G$. Again this can be readily seen by inspecting the table XXXII.

Hence the ideal $K = \{0, 1, 2, 3\}$ and its cosets

$$J = 4 + K = \{4, 5, 6, 7\}$$

$$P = 8 + K = \{8, 9, 10, 11\}$$

$$T = 12 + K = \{12, 13, 14, 15\}$$

split $\langle G, +, \cdot \rangle$ into sixteen blocks with the induced operations well defined. Let us show that the resultant quotient ring, G/K , is isomorphic to a four element Boolean ring with unity.

Tables XXXIX and XXXX show how $\langle G, +, \cdot \rangle$ forms the quotient ring, G/K .

+	K	J	P	T
K	K	J	P	T
J	J	K	T	P
P	P	T	K	J
T	T	P	J	K

Table XXXIX

$\langle K, + \rangle$ Group

$\cdot$	K	J	P	T
K	K	K	K	K
J	K	J	J	K
P	K	J	P	T
T	K	K	T	T

Table XXXX

$\langle K, \cdot \rangle$ Semigroup

It has already been shown that $\langle K, + \rangle$ is isomorphic to the Klein 4-group.

By inspecting the table XXXX it can be seen that:

- 1°. Binary operation is closed.
- 2°. There is a unity $u = K$
- 3°. Groupoid is commutative.

The associativity can be proven as follows:

$$(a) \quad K \cdot (J \cdot P) = K \cdot J = K$$

$$(K \cdot J) \cdot P = K \cdot P = K$$

$$(b) \quad T \cdot (K \cdot J) = T \cdot K = K$$

$$(T \cdot K) \cdot J = K \cdot J = K$$

$$(c) \quad T \cdot (P \cdot J) = T \cdot J = K$$

$$(T \cdot P) \cdot J = T \cdot J = K$$

$$(d) \quad P \cdot (K \cdot T) = P \cdot K = K \\ (P \cdot K) \cdot T = K \cdot T = K$$

$$(e) \quad P \cdot (J \cdot T) = P \cdot K = K \\ (P \cdot J) \cdot T = J \cdot T = K$$

Hence it can be seen that the algebraic structure given in table XXXX is a commutative semigroup with unity $u = K$. Furthermore all of its elements are idempotent.

At this point, the only axiom which is left to be proven to show that G/K is a four element Boolean with unity, is the right and left distributivity under addition, namely

$$a \cdot (b + c) = a \cdot b + a \cdot c \quad \text{and} \\ (b + c) \cdot a = b \cdot a + c \cdot a$$

The fact that the left distributive law holds can be demonstrated as follows:

$$(a) \quad T \cdot (P + J) = T \cdot T = T \\ T \cdot P + T \cdot J = T + K = T$$

$$(b) \quad K \cdot (J + T) = K \cdot P = K \\ K \cdot J + K \cdot T = K + K = K$$

$$(c) \quad P \cdot (T + K) = P \cdot T = T \\ P \cdot T + P \cdot K = T + K = T$$

$$(d) \quad J \cdot (T + T) = J \cdot K = K \\ J \cdot T + J \cdot T = K \cdot K = K$$

Similarly the right distributive law holds as shown below:

$$(a) \quad (K + P) \cdot J = P \cdot J = J \\ K \cdot J + P \cdot J = K + J = J$$

$$(b) \quad (P + T) \cdot P = J \cdot P = J \\ P \cdot P + T \cdot P = P + T = J$$

$$(c) \quad (T + K) \cdot J = T \cdot J = K \\ T \cdot J + K \cdot J = K + K = K$$

$$\begin{aligned} \text{(d)} \quad (J + J) \cdot P &= K \cdot P = K \\ J \cdot P + J \cdot P &= J + J = K \end{aligned}$$

Hence G/K is isomorphic to a four element Boolean ring with unity.

It has been shown that "Idempotent Scalar Quaternians" is a sixteen element Boolean ring with unity. This new quaternion and two of its quotient rings play a very important part in the development of the unified theory, since one of its quotient rings, G/N , is isomorphic to Z_2 field and the other, G/K , is a four element Boolean ring with unity.

APPENDIX F

BOOLEAN ALGEBRA OF THE UNIFIED THEORY

Appendix F

BOOLEAN ALGEBRA of the UNIFIED THEORY

In the theory of the switching circuits the smallest sub-algebra of Boolean algebras, $\langle \{0, 1\}, V, \wedge, \neg, 0, 1 \rangle$, is used throughout. It is based on the axioms of the two element Boolean ring with unity under $\oplus$ and $\wedge$ operations.

On the other hand the four element Boolean ring with unity described in the section 4.5 relates to the following four state Boolean algebra:

$$\langle \{00, 01, 10, 11\}, V, \wedge, \neg, 00, 11 \rangle.$$

It can be shown [4] that 00 and 11 correspond to the empty set $\emptyset$ and the universe U_n of the set theory. In addition, the elements of the set $Q = \{0, 1, x\}$ will be assigned to the following states:

$$0 = 01, \quad 1 = 10, \quad x = 11$$

Since this algebra is in B space, it is necessary and desirable to relate its $\wedge, V, \neg$ operations to the topology of Roth's space. This will be done by examining each operation in detail.

Complement Operation:

On bit by bit bases this operation resembles sharpening out from the universe, i.e., $x \# a_i$.

Hence we have

$$\begin{aligned} \bar{j} &= \overline{10} = 01 = i & \text{or} & \quad K \oplus j = 11 \oplus 10 = 01 = i \\ \text{and} \end{aligned}$$

$$\bar{i} = \overline{01} = 10 = j \quad \text{or} \quad K \oplus i = 11 \oplus 01 = 10 = j$$

"AND" Operation:

Since this is the same binary operation which is used in multiplicative part of the four element Boolean ring with unity it represents $\cap$, $\sqsubseteq$ and $*$ operators. Hence

$$a \wedge b = \begin{cases} \emptyset \Rightarrow a \not\sqsupseteq b \\ b \Rightarrow b \sqsubseteq a \\ a \Rightarrow a \sqsubseteq b \\ c \Rightarrow a \cap b = c \end{cases}$$

In $*$ operation one $\emptyset$ is allowed and a new "c" cube is formed by replacing $\emptyset$ by an X.

In other words the "AND" operation indicates the position of the two cubes with respect to each other by stating whether or not they intersect or cover each other. In the star product the "AND" operation yields the largest "c" cube between "a" and "b" cubes.

"OR" Operation:

This is a new operation in Roth space and fits into the unified theory as follows:

$$a \vee b = (a \oplus b) \oplus (a \wedge b) = c$$

Like in the Roth's sharp algorithm both the multiplicative and additive parts of the four element Boolean ring with unity is used.

Before interpreting the topological significance of the new operator in the Roth space, let us determine its algebraic structure. The results of applying bit by bit "OR" binary operation to the subset $B = \{0, 1, i, j\}$ is shown in Table XXXXI

$\vee$	0	1	i	j
0	0	1	i	j
1	1	1	1	1
i	i	1	i	1
j	j	1	1	j

Table XXXXI

$\langle B, \vee \rangle$ Semigroup

By inspecting the table it can be seen that:

- 1°. Binary operation is closed.
- 2°. There is an identity $e = 0$
- 3°. Groupoid is Abelian
- 4°. Elements of the groupoid are idempotent.

In addition, it can be shown that the binary operation is associative:

For example:

- (a) $(i \vee j) \vee 1 = 1 \vee 1 = 1$
 $i \vee (j \vee 1) = i \vee 1 = 1$
- (b) $(i \vee i) \vee j = i \vee j = 1$
 $i \vee (i \vee j) = i \vee 1 = 1$
- (c) $(0 \vee i) \vee j = i \vee j = 1$
 $0 \vee (i \vee j) = 0 \vee 1 = 1$
- (d) $(1 \vee i) \vee 0 = 1 \vee 0 = 1$
 $1 \vee (i \vee 0) = 1 \vee i = 1$
- (e) $(1 \vee 1) \vee 0 = 1 \vee 0 = 1$
 $1 \vee (1 \vee 0) = 1 \vee 1 = 1$

Hence it can be seen that the algebraic structure given in table XXXXI is an Abelian semigroup with identity $e = 0$. Furthermore all of its elements are idempotent.

Let us define the "OR" operator as follows:

Mapping: The mapping from the Q space to the B space is the same as the one used for $\sqcap$, $\sqcup$ and $*$ operators. In other words the coordinates of the upper and lower cubes are mapped as follows:

$$0 \rightarrow i = 01$$

$$1 \rightarrow j = 10$$

$$x \rightarrow l = 11$$

Purpose: To find the smallest "c" cube which covers both "a" and "b" cubes.

Examples

Topology in Roth's Space

$$\begin{array}{lcl} 1.) \{1, 1, x\} & \rightarrow & \{j, j, l\} \\ \{1, 1, 0\} & \rightarrow & \{j, j, i\} \\ * & & \underline{V} \\ \{1, 1, x\} & \leftarrow & \{j, j, l\} \end{array}$$

$$\begin{array}{lcl} 2.) \{1, 0, x\} & \rightarrow & \{j, i, l\} \\ \{1, 1, 1\} & \rightarrow & \{j, j, j\} \\ * & & \underline{V} \\ \{1, x, x\} & \leftarrow & \{j, l, i\} \end{array}$$

$$\begin{array}{lcl} 3.) \{1, x, x\} & \rightarrow & \{j, l, l\} \\ \{0, 0, x\} & \rightarrow & \{i, i, l\} \\ * & & \underline{V} \\ \{x, x, x\} & \leftarrow & \{l, l, l\} \end{array}$$

Let us now prove the following theorem of Boolean algebra using all three operators in B space.

$$A + \bar{A} \cdot B = A + B \quad \text{Let} \quad A = 001 \quad B = 11x$$

$$A + \bar{A} \cdot B = 001 + 110 \cdot 11x \rightarrow iij \vee (jji \wedge jjl)$$

$$= iij \vee jji = 111 \rightarrow xxx$$

also

$$A + B = 001 + 11x \rightarrow iij \vee jjl = 111 \rightarrow xxx$$

Hence $A + \bar{A} \cdot B = A + B$

In other words xxx cube which covers 001 and 110 vertices also cover 001 vertex and 11x edge.

BIBLIOGRAPHY

1. R. L. Ashenhurst, "The Decomposition of Switching Functions," Proc. International Symposium on the Theory of Switching, April 2-5, 1957, Vol. 29 of Annals of Computation Laboratory of Harvard University, 1959, pp. 74-116.
2. H. A. Curtis, "A New Approach to the Design of Switching Circuits," Princeton : Van Nostrand, 1962.
3. D. L. Dietmeyer and P. R. Schneider, "Identification of Symmetry, Redundancy, and Equivalence of Boolean Functions," IEEE Trans. on Electronic Computers, Vol. EC-16, Dec. 1967, pp. 804-817.
4. D. L. Dietmeyer, "Logic Design of Digital Systems," Allyn and Bacon Inc., Boston, Mass., 1971.
5. J. B. Fraleigh, "A First Course in Abstract Algebra," Addison-Wesley Publishing Co., Reading, Mass., 1967.
6. S-T Hu, "Threshold Logic," University of California Press, 1965.
7. R. M. Karp, F. E. McFarlin, J. P. Roth and J. R. Wilts, "A Computer Program for the Synthesis of Combinational Switching Circuits," Proc. Second Annual AIEE Symposium on Switching Circuit Theory and Logical Design, Oct. 1961, pp. 182-194.
8. E. J. McCluskey, "Minimization of Boolean Functions," Bell System Tech. Journal, Vol. 35, No. 6, Nov. 1956, pp. 1417-1444.
9. E. J. McCluskey, Jr. and T. C. Bartee, et al, "A Survey of Switching Circuit Theory," New York:McGraw Hill, 1962.
10. E. J. McCluskey, "Introduction to the Theory of Switching Circuits," New York:McGraw Hill, 1965.
11. R. E. Miller, "Switching Theory Volume I: Combinational Circuits," New York: John Wiley and Sons, 1965.
12. A. Mukhopadhyay, "Detection of Total or Partial Symmetry of a Switching Function with the Use of Decomposition Charts," IEEE Trans. on Electronic Computers, Vol. EC-12, Oct. 1963, pp. 553-557.
13. S. R. Petrick, "A Direct Determination of the Irredundant Forms of a Boolean Function from the Set of Prime Implicants," Air Force Cambridge Research Centre, Cambridge, Mass., Tech Report AFCRC-TR-56-110, 1956.
14. W. V. Quine, "The Problem of Simplifying Truth Functions," Am. Math. Monthly, Vol. 59, 1952, pp. 521-531.

15. W. V. Quine, "A Way to Simplify Truth Functions," Am. Math. Monthly, Vol. 62, 1955, pp. 627-631.
16. J. P. Roth, "Algebraic Topological Methods for the Synthesis of Switching Systems I," Trans. Am. Math. Soc., Vol. 88, No.2, July 1958, pp. 301-326.
17. J. P. Roth and E. G. Wagner, "Algebraic Topological Methods for the Synthesis of Switching Systems, Part III. Minimization of Nonsingular Boolean Trees," IBM Journal, Vol. 4, No. 4, Oct. 1959, pp. 326-345.
18. J. P. Roth, "Algebraic Topological Methods for the Synthesis of Switching Systems in n variables," The Institute for Advanced Study, Princeton, ECP 56-02, April 1956.
19. J. P. Roth, "A Combinatorial Topological Method for the Synthesis of Switching Systems in n variables," Bull. Amer. Math. Soc. Abstract 62-2-281.
20. J. P. Roth, "An Algorithm for the Problem of Quine," Bull. Amer. Math. Abstract 62-3-367.
21. J. P. Roth, "A Function-Space Formulation of the Switching Circuit Synthesis Problem," Bull. Amer. Math. Soc. Abstract 62-2-392.
22. J. P. Roth, "Minimization over Boolean Trees," IBM Journal, Vol. 4, No. 5, Nov. 1960, pp. 543-558.
23. J. P. Roth and R. M. Karp, "Minimization over Boolean Graphs," IBM Journal, Vol. 6, No. 2, April 1962, pp. 227-238.
24. R. H. Urbano and R. K. Mueller, "A Topological Method for the Determination of the Minimal Forms of a Boolean Function," IRE Trans. on Electronic Computers, Vol. EC-5, No. 3, Sept. 1956, pp. 126-132.
25. P. Weiner and T. F. Dwyer, "Discussion of Some Flaws in the Classical Theory of Two-Level Minimization of Multiple-Output Switching Networks," IEEE Trans. on Computers, Vol. C-17, Feb. 1968, pp. 184-186.
26. P. E. Wood, Jr., "Switching Theory," New York: McGraw Hill Book Co. 1968.
27. Dietmeyer, D.L. and P.R. Schneider, "A Subroutine Set for Automation of Logic Circuit Design" Proc. of COMMON Anniversary Meeting, New Orleans, La., November 30, 1966.
28. Dietmeyer, D.L. and P.R. Schneider, "Combinational Switching Function Subroutine Set," 9.4.062 program release to COMMON library, IBM Corp. DP. Program Information Dept., Hawthorn, New York, July 1966.

29. Dietmeyer, D.L. and P.R. Schneider, "Aids for Logic Design Algorithm Development," IEEE Transactions on Education, Vol. E-10, September 1967, pp. 127-131.
30. Knuth, D.E., "The Art of Computer Programming," Addison-Wesley Publishing Co., Reading, Mass. 1968. (Vol. I)
31. Mendelson, E., "Theory and Problems of Boolean Algebra and Switching Circuits, Schaum Outline Series, McGraw-Hill Book Co., 1970 .
32. Baumslag, B. and Chandler, B., "Theory and Problems of Group Theory " Schaum's Outline Series, McGraw-Hill Book Co., 1968.