ALGEBRAIC TOPOLOGICAL METHODS FOR THE SYNTHESIS OF SWITCHING SYSTEMS. I.

BY

J. PAUL ROTH

i. The problem of synthesis of switching systems is one of ever increasing importance to modern technology; it arises in the design of digital computers, telephone switching systems and control mechanisms of all sorts. Behind, in fact, every pushbutton lies a switching circuit. In spite of its considerable importance there is as yet no well developed theory for the rational design of such systems. Professor H. H. Aiken [A] states that a "lack of adequate mathematical methods for the investigation of the functional behavior of electronic control circuits represented the largest single obstacle to the rapid development of the subject." Even efforts to understand the human neural system have encountered this problem (cf. the provocative work of McCulloch and Pitts [M–P]).

Previous attempts toward building a theory have considered the problem from the point of view of symbolic logic; this paper considers the problem as being inherently of a combinatorial topological character. This point of view yields a mechanization and visualization of the problen not evident from the logical approach.

ii. The synthesis problem may be loosely described as follows. Let A be a binary variable, assuming values 0 and 1. A switch with variable A is a device having two states "closed" if $A = 1$ and "open" if $A = 0$ (see illustration).

SWITCH WITH VARIABLE A

A given variable is allowed several switches. In the switching-system synthesis problem one is given n binary variables $A_1, \cdots, A_n$ and a specification that for certain combinations of the variables the system to be designed shall be closed and for certain others, open (there may also be combinations for which we "don't care"): The problem is to devise a system of switches, a linear graph of such switches, which satisfies the specifications and at the same time utilizes a *minimum number of switches* (other types of minima are sometimes sought).

iii. *Logical formulations of the problem.* (a) The definitive correlation of this synthesis problems with Boolean algebra was due to Shannon [S]. (b) Within logic itself is the old problem [QO] of simplifying truth functions,

Presented to the Society, December 29, 1955; received by the editors July 30, 1956 and, in revised form, November 23, 1956.

more precisely, "of devising a general mechanical procedure for reducing any formula to its *simplest* equivalent" [Q]. Quine remarks that "despite the trivial character of truth-function logic" this "problem has remained curiously stubborn." Quine specializes the general problem to that of finding a simplest *normal* equivalent. A formula is *normal* if it is the disjunction of conjunctions (disjunctive normal form) with the additional requirement that no letter can occur twice in a clause. We term this *the problem of Quine*. According to Nelson [N], "the problem Quine deals with has been solved only when a *technically* manageable procedure, specifically a procedure programmable on computers has been devised," and Quine's method "turns out to be very impractical for even the largest computers now available."

iv. *Geometrical approaches.* (a) The Harvard University Computation Laboratory developed a "chart method" for improving circuit designs which involved, however, for a problem of n variables, the examination of 2^{2^n} entries. (b) A distinct approach was devised by Montgomerie [M] and developed and extended by Veitch [V] and Karnaugh [K]; this method is graphical in character involving inspections by a human operator and essentially it seeks the minimum for the problem of Quine. It may be remarked that the cubing algorithm developed in §2 supplies the mechanization of this method for n variables called for by Karnaugh.

v. *Outline of this paper.* In §1 the cubical complex which is naturally associated with a Boolean function is defined. It is of crucial importance that this complex is given a completely explicit, analytical description. Boolean functions are equivalent in the classical sense if and only if their associated complexes are identical. An algebra of cubical complexes is described which permits a direct visualization of a Boolean algebra, having several advantages, it would seem, over the Venn diagram. This algebra is in fact a faithful representation of the appropriate Boolean algebra. It is appropriate also to identify Boolean functions if their associated complexes are isomorphic, the transformation group being essentially a permutation group.

Let K be a cubical complex (defined by a Boolean function of n variables). Let L be a subcomplex of K. A *cover* of L by cubes of K consists of a collection C of cubes of K such that each vertex of L lies on a cube of C. Let q_k be the number of k-cubes of C. The problem is to find a cover C such that the form

$$\sum_{k=0}^{n} q_k(n - k)$$

is minimized. When $K=L$ we have the problem of Quine. (The vertices of $K-L$ are sometimes referred to as "don't care" vertices.)

An *elementary cocycle* of K is a cube which is not the face of a higher dimensional cube of K; Quine's terminology for the same object, when $K=L$, is *prime implicant*. §2 describes the *cubing algorithm* which is a method for computing Z, the subspace of elementary cocycles. This algorithm is com-

pletely mechanical, is designed in fact for a digital computer; a problem in n variables is shown to require *on the average* less than $3^n 2^{n+1}$ single-digit binary additions. A detailed example for $n=5$ is included.

A cube e is termed an *L-extremal* if it has a vertex of L with the property that all of its cofaces are contained in e. It is easily shown that every minimal cover must contain the subspace E of extremals. A corollary is that for the case when E is itself a cover, E constitutes the unique minimum cover. §3 describes two algorithms for locating the extremals, each suitable to a different kind of computing machine. Location of the extremals is a fast operation.

§4 is entitled "Some methods for obtaining 'near-minimal' covers." The covers have the property that they cannot be improved by Quine's dispensing operation [Q1]. A simple example is given showing where the algorithm would fail to deliver the minimum. In Part II of this paper however, we shall give an algorithm for finding a minimum(1).

In the special case when K *is* 1-*dimensional*, R. Z. Norman has devised an ingenious algorithm which always supplies a minimum. The method does not generalize for K greater than 1-dimensional, but it appears to give a new approach to the four-color problem.

The cubing algorithm requires that the problem be "fed into the machine" in the form of 0-cubes; the initial specification of the problem, however, may be more structured. In §5 the $*$-*algorithm* is described which allows higher dimensional cubes, if they are available, to be fed into the machine and thus the number of steps required for the computation is in general substantially reduced. If the initial information is in the form of 0-cubes, then this reduces to the cubing algorithm. The basic fact in the construction is the definition of the $*$-*product* between cubes (cf. fig. 5.0). §5 develops the necessary properties of the $*$-product and proves that the $*$-algorithm does in fact always deliver up Z.

§§6 and 7 describe two different methods for locating E, each involving a different kind of product between cubes.

§7 discusses the problem of generating "near-minimal" covers following the computation of Z and E by the $*$- and #-algorithms.

§8 describes a function-space formulation of the synthesis problem. Results on this problem are meagre; a so-called *tensor algorithm* is illustrated by two examples. §9 treats an extension of the synthesis problem to multivalued switches.

The author is deeply grateful to The Institute for Advanced Study for its sponsorship of this work under contracts jointly supported by the Army, Office of Naval Research and Air Research and Development Command, and

(1) *Added in proof.* Cf. *Combinatorial topological methods in the synthesis of switching circuits*, presented at the International Symposium On The Theory Of Switching, Harvard University, Cambridge, Mass., April 2, 1957 (to appear in the Proceedings of symposium). Here two algorithms are given for this problem.

to Dr. H. H. Goldstine, Acting Director of the Institute Electronic Computer Project. The author is especially indebted to Mr. James Pomerene of the Institute for Advanced Study and Dr. R. Z. Norman of Princeton University for stimulating discussions on every aspect of this development. Acknowledgment is made of profitable discussions with Mssrs. G. T. Jacobi and G. Kron of the G.E. General Engineering Laboratory, Dr. Richard Shuey of the General Electric Research Laboratory, with Drs. Deming Lewis, Director, Maurice Karnaugh and Edward F. Moore of the Switching Research Department of Bell Laboratories(2).

1. **Topological foundations of the problem.** In this section the basic structure is described upon which the whole development depends.

1.1. *The cubical complex $K(f)$.* Let f be a mapping of the nth Cartesian product $Z_2^{(n)}$ of the space Z_2 of integers modulo 2 into Z_2

$$f\colon Z_2^{(n)} \to Z_2.$$

We shall think of $Z_2^{(n)}$ as an n-cube and shall define inductively the cubical complex K derived from the set $f^{-1}(1)$ of elements mapped onto 1. The space of 0-cubes K^0 is the discrete set $f^{-1}(1)$; we shall term the elements of $f^{-1}(1)$ *0-cubes or vertices.* For instance, for $n=4$, 0100 and 0110 might be 0-cubes of $f^{-1}(1)$. Two 0-cubes are said to be the *faces* of a 1-cube if it is possible to transform from one to the other by the change of a single coordinate; thus for example

$$0111 \to 0101$$

by changing the third component. The *pair* 0111 and 0101 of two such 0-cubes is termed a 1-*cube*, which is conveniently and unambiguously designated by a symbol such as

$$01x1;$$

here x is to be thought of as a "variable" taking on the values 0 and 1. We term the component having x as the *free* component and the others as *bound.* Then the set of all such 1-cubes defined by K^0 defines the space K^1 of 1-cubes of K. Two 1-cubes of K^1 are said to form *opposite faces of a 2-cube* if their free elements occupy the same component and if exactly one of their bound components disagree; for example $0x10$ and $0x11$ are opposite faces of the 2-cube $0x1x$. The space of all distinct 2-cubes formed from K^1 is thus the *space of 2-cubes* K^2. By induction the space K^r of r-cubes is defined for all r. For instance $xx011x$ is a 3-cube in 6-space. The *cubical complex* $K(f)$ is

(2) The referee has pointed out the following pertinent references: R. Higonnet and R. Grea, *Etude logique des circuits electriques et des systemes binaires*, Paris, Editions Berger-Lerault, 1955; E. J. McCluskey, Jr., *Minimization of Boolean functions*, Bell System Tech. J. vol. 35, November, 1956, pp. 1417–1444; 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, EC-5, September, 1956, pp. 126–132.

thus defined as this collection of cubes $K^0, K^1, \cdots, K^r, \cdots$ so determined *plus* the face- and coface-operators defined below. The complex K has the important peculiarity that if it contains the vertices of a cube, then it contains the cube itself.

1.2. *Face- and coface-operators.* Let K^r be the subspace of r-cubes of such a cubical complex K; we shall define mappings of $K^r \to K^{r+1}$ called *coface operators* and of $K^r \to K^{r-1}$ termed *face operators*; these mappings tell us how the complex K is put together and we shall be concerned with an explicit construction of the coface mappings.

Let $(a_1, \cdots, a_n)$ be an r-cube, with $a_i = 0$, 1 or x, $i = 1, \cdots, n$. Then the *ith face operators* are defined as follows:

$$\partial_i^1(a_1, \cdots, a_n) = \begin{cases} (a_1, \cdots, a_{i-1}, 1, a_{i+1}, \cdots, a_n) & \text{if } a_i = x, \\ \phi & \text{if } a_i \neq x \end{cases}$$

and

$$\partial_i^0(a_1, \cdots, a_n) = \begin{cases} (a_1, \cdots, a_{i-1}, 0, a_{i+1}, \cdots, a_n) & \text{if } a_i = x, \\ \phi & \text{if } a_i \neq x. \end{cases}$$

If $a = (a_1, \cdots, a_n)$ and $a_i = x$ then $\partial_i^1 a$ and $\partial_i^0 a$ are said to be *opposite* faces. The definition of the *coface operators* is more complicated: Letting $(a_1, \cdots, a_n)$ be an r-cube, $a_i = 0$, 1 or x,

$$\delta_i(a_1, \cdots, a_n) = \begin{cases} \phi \text{ if } a_i = x, \\ (a_1, \cdots, a_{i-1}, x, a_{i+1}, \cdots, a_n) = c^{r+1} \\ \qquad\qquad\qquad\qquad \text{if } a_i \neq x \text{ and if } c^{r+1} \subset K, \\ \phi \text{ if } c^{r+1} \not\subset K. \end{cases}$$

The face- and coface-operators satisfy several obvious identities

$$\partial_i^\alpha \partial_j^\beta = \partial_j^\beta \partial_i^\alpha,$$

$$\delta_i \delta_j = \delta_j \delta_i,$$

$$\delta_j \partial_i^\alpha = \partial_i^\alpha \delta_j, \qquad i \neq j,$$

$$\delta_i \partial_i^\alpha(c^r) = c^r \qquad \text{if } \partial_i^\alpha c^r \neq \phi,$$

$$\partial_i^{a_i} \delta_i c^r = c^r \qquad \text{if } \delta_i c^r \neq \phi,$$

similar to those for the semi-simplicial complexes introduced by Eilenberg and Zilber [E–Z].

1.3. EXAMPLES. $\delta_1(011xx) = x11xx$; $\partial_4^0(011xx) = 0110x$.

1.4. *The cubical complex of a Boolean function.* Let f be a Boolean function (also termed truth function) of n variables $A_1, \cdots, A_n$. Let β_i denote the assignment of a truth value to A_i: $\beta_i = 1$ if A_i is true, $\beta_i = 0$ if A_i is false; then

the array $(\beta_1, \cdots, \beta_n)$ denotes such an assignment. The set of all assignments for which f is true defines the vertices of $K(f)$, and the construction described in 1.1 uniquely defines the complex $K(f)$ associated with f. The complex defined by the arguments for which f is false is the complex $\tilde{K}(f)$ complementary to K in the n-cube Q; it is the complex determined by the vertices *not* in $K(f)$.

In its classical sense, two Boolean functions are considered as equivalent if they have the same truth table; from the function-theoretic point of view two functions equivalent in this sense are merely different expressions for the *same* thing.

1.5. PROPOSITION. *Boolean functions are equivalent in the classical sense if and only if their associated complexes are identical.*

Thus, there is a 1-1 correspondence between Boolean functions of n variables and cubical complexes contained in the n-cube Q. The product of two such functions f, g corresponds to the intersection of their associated complexes: $K(f \cdot g) = K(f) \cap K(g)$; further $f \cup g$ corresponds to the complex generated by the vertices of $K(f)$ and $K(g)$, the universe corresponds to Q and the zero, to the null cube. Therefore, we have the following result.

1.6. PROPOSITION. *The algebra of Boolean functions of n variables is faithfully represented by this algebra of cubical complexes contained in Q.*

1.7. EXAMPLES. The complex associated with $f = \overline{A} \vee \overline{B}\overline{C} \vee BC$ is depicted below (Fig. 1). $\overline{A}$ represents the cube $0xx$, $\overline{B}\overline{C}$ is $x00$, and BC, $x11$. Figure (2) represents, among several, $\overline{B}\overline{C} \vee \overline{A}C \vee AB$. Note that this K contains a non-bounding 1-cycle. Other examples are given in §§4.4 and 5.0. The pictures

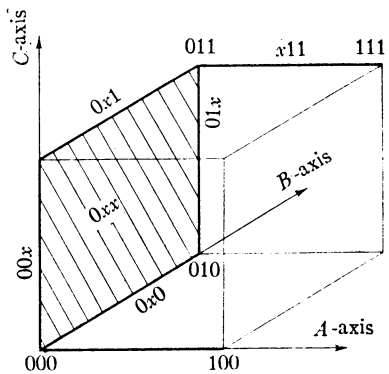


FIG. 1

$x00 = \overline{B}\overline{C}$, $0xx = \overline{A}$, $x11 = BC$
Complementary complex is $001 \cup 101$.

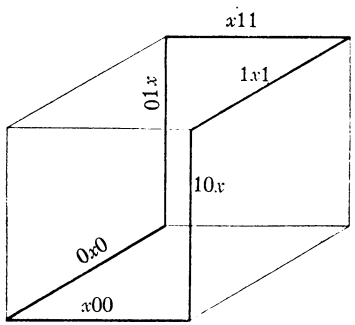


FIG. 2

$f = \overline{B}\overline{C} \vee AC \vee \overline{A}B = A\overline{B} \vee BC \vee \overline{A}\overline{C}$

are, of course, not faithful representations of the complex, but do serve the heuristic purpose of visualization.

1.8. It is natural to identify two Boolean functions if it is possible to transform one into the other by a suitable permutation of variables or negation of certain variables. Such a transformation corresponds to an automorphism or symmetry of the n-cube. The group of all such automorphisms is isomorphic to the hyper-octahedral group O_n; its order is $n!2^n$ (cf. David Slepian, *On the number of symmetry types of Boolean functions of n variables*, Canadian J. Math. vol. 5 (1953) pp. 185–193). We shall say then that two cubical complexes are isomorphic if there exists an element of O_n transforming one into the other.

1.9. PROPOSITION. *Two Boolean functions are equivalent if and only if their associated complexes are isomorphic.*

1.10. The *logical problem* is "to devise a general mechanical procedure for reducing any formula to its simplest equivalent" [Q1]. Quine's specialization of the problem is to find a simplest *normal* equivalent: A formula is *normal* if it is a disjunction of conjunctions (disjunctive normal form) with the additional requirement that no letter in a clause occur twice. We shall give Quine's problem a topological interpretation.

1.11. Let K be a cubical complex (defined by a Boolean function of n variables) contained in an n-cube Q. Let L be a subcomplex of K. A *cover* of L by cubes of K consists of a collection of cubes of K such that each vertex of L lies on a cube of C.

1.12. The *cost* of a cover C, where q_k denotes the number of k-cubes in C, is given by the formula

$$\sum q_k(n - k).$$

1.13. The problem that we consider then is to find a K-cover of L of minimum cost. When $K = L$ we have the problem of Quine. In this case, each cube of a cover corresponds to a fundamental formula in a normal-form expression, and its cost to the number of literals occurring in the given expression. Consider the following example: Let $L^0 = \{00, 01, 11\}$ and $K^0 = L^0 \cup \{10\}$. Then $C = \{0x, x1\}$ is the minimum cover of L by cubes of L, of cost 2, whereas $C^* = \{xx\}$ is the minimum cover of L by cubes of K, of cost 0. The vertex 10 is sometimes called a "don't care" vertex.

1.14. REMARK. In the synthesis of so-called "two-level 'and' and 'or' diode circuits," an apparently different kind of minimum is required. This is essentially the number of "ands" and "ors" occurring in a given expression; this amounts to the cost, as defined above, plus the number of cubes in the cover. But this becomes

$$\sum q_k(n - k) + \sum q_k = \sum q_k[(n + 1) - k].$$

This new problem is therefore interpretable as the old one, but in an $(n+1)$-

cube rather than an n-cube. In synthesis work a considerable variety of different kinds of minima are desired. We treat, however, only the minimum problem described above.

1.15. DEFINITION. An *elementary r-cocycle* of K is an r-cube z^r of K such that $\delta_i z^r = \phi$ in K for all i: It is thus a cube with no cofaces. Let Z^r denote the set of all elementary r-cocycles and

$$Z = \sum_r Z^r.$$

We shall call any element of Z a *cocycle*. It may be seen that in its logical interpretation when $K = L$ a cocycle is a *prime implicant* in the language of Quine. The following theorem is due to Quine.

1.16. THEOREM. *If C is a minimal cover, then $C \subset Z$.*

For if c were an element of C and $\delta_i c \neq \phi$, then $C^* = (C - c) + \delta_i c$ has lower cost than C. q.e.d. Thus, a computation of Z may be helpful in finding a minimal cover.

FIG. 2.3. Example of the cubing algorithm computation of Z for a synthesis problem in 5 variables cubing-1 operation. Computation of K^1 and Z^0
(Columns 6–15 are omitted.)

	K^0					1					2					3					4					5				
						1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	1	1	0	0	0	0	0	0	0	0
1	1	1	0	0	1																									
2	1	0	0	1	1	0	1	0	1																					
3	0	0	1	1	0	1	1				1	0	1																	
4	1	1	0	0	0	0	0	0	0	**1**	0	1	0	1		1	1													
5	0	0	0	0	0	1	1				1	0	0	1		0	0	1	1		1	1								
6	1	0	0	0	1	0	**1**	0	0	0	0	0	0	**1**	0	1	0	1			0	1	0	0	1	1	0	0	0	1
7	0	1	1	0	1	1	0	1			1	1				0	1	0	1		1	0	1			0	1	1		
8	0	0	1	1	1	1	1				1	0	1			0	0	0	0	**1**	1	1				0	0	1	1	
9	1	1	0	1	1	0	0	0	**1**	0	0	**1**	0	0	0	1	1				0	0	0	1	1	1	1			
10	1	0	1	0	1	0	1	1			0	0	1	1		1	0	0	1		0	1	1			1	0	1		
11	1	0	1	1	0	0	1	1			0	0	1	0	1	**1**	0	0	0	0	0	1	1			1	0	1		
12	1	0	0	0	0	0	1	0	0	1	0	0	0	1	1	1	0	1			0	**1**	0	0	0	**1**	0	0	0	0
13	1	1	1	1	1	0	0	1	1		0	1	1			1	1				0	0	1	1		1	1			
14	0	0	0	1	0	1	1				1	0	0	0	1	0	0	**1**	0	0	1	1				0	0	0	**1**	0
15	0	0	0	0	1	1	1				1	0	0	1		0	0	1	1		1	1				0	0	0	0	**1**
16	0	0	1	0	1	1	1				1	0	1			0	0	0	1	1	1	1				0	0	1	0	1

FIG. 2.3 (cont.). Cubing operation 2: computation of K^2 and Z^1

K^1

x	0	1	1	0																				
x	0	0	0	0	·	0	1	1																
x	0	0	0	1	·	0	1	1		·	0	0	0	**1**										
x	0	1	0	1	·	0	0	1	1	·	0	1	0	1	·	0	**1**	0	0					

1	x	0	0	1																				
1	x	0	1	1	0	·	0	**1**	0															
1	x	0	0	0	0	·	0	0	**1**	0	·	0	1	1										
0	x	1	0	1	1	·	1			1	·	1			1		1							

0	0	x	1	0																				
1	0	x	0	1	1	0		1																
1	1	x	1	1	1	1				0	1		1											
0	0	x	0	1	0	0		1		**1**	0		0	0	1	1								

1	1	0	x	1																				
1	0	0	x	1	0	**1**	0		0															
0	0	0	x	0	1	1				1	0	0		1										
0	0	1	x	1	1	1				1	0	1			0	0	1		1					

1	1	0	0	x																				
0	0	1	1	x	1	1																		
0	0	0	0	x	1	1				0	0	1	1											
1	0	0	0	x	0	**1**	0	0		1	0	1			**1**	0	0	0						

$K^2 = Z^2$

x 0 0 0 x
x 0 x 0 1
1 x 0 x 1
1 x 0 0 x

Cubing algorithm ceases.

2. **The cubing algorithm.** We now give an algorithm which computes Z, the set of elementary cocycles of K. This method assumes that information about the complex K is given only in a listing of all its 0-cubes. §5 describes the $*$-algorithm, which starts with cubes of any dimension.

2.1. *Description of the cubing algorithm.* The first step is to add each 0-cube to each other 0-cube componentwise and modulo 2. If exactly one component

of the sum of any particular pair is 1, then this pair forms a 1-cube of K, denoted as described above. Next we tabulate all 1-cubes of K grouping together with those with the same free component. We agree that two 1-cubes may be added if and only if they have the same free component: Two 1-cubes form a 2-cube if and only if exactly one of the bound components of their sum is 1. We compute and tabulate all 2-cubes of K grouping together those with the same free components. The algorithm proceeds inductively determining all k-cubes of K until a dimension k is reached when there are no k-cubes.

2.2. An example is chosen with five variables; while the totality of possible Boolean functions is 2^{2^5} we need consider a complex with at most 2^4 0-cubes. In the 0th column of Figure 2.3 is listed 16 5-digit random binary numbers. These 16 entries completely define the problem; they are the 0-cubes of a complex K. The first column lists the (partial) sums of the first row entry 11001 with each of the others; it will be observed that as soon as two 1's appeared the summing operation was ceased for, it will be recalled, two 0-cubes form a 1-cube only if their sum has exactly one component different from 0. This observation has important consequences with regard to the probable number of arithmetic operations required to effect the synthesis.

2.4. LEMMA. *Let u, v be infinite sequences of binary digits. If the components of u, v are added modulo 2 sequentially and addition is stopped as soon as two components of the sum are* 1, *then the expected value for the number of additions performed is* 4.

This can be verified by noting that the expected value of performing exactly m additions is $m(m-1)/2^m$ and that this is the general term in the expansion of $1/(1-x)^2$ for $x=1/2$.

2.5. LEMMA. *For a problem in n variables for the entire cubing algorithm, the probable number of binary single-digit additions is less than 3^n2^{n+1}.*

This very crude bound may be verified by a detailed examination of the steps in the cubing operation. It may be noted as experimental evidence that cubing operation 1 in the example required 434 binary additions, as compared with the bound 480 supplied by Lemma 2.4. For the higher synthesis the probable number of operations drops very sharply since two 2-cubes, for instance, can be added only if their free variables occur in the same component. The second column lists the partial sums of the second row entry with the others, and so on. Those sums for which exactly one component is 1, showing the formation of a 1-cube, are denoted by a bold 1. It will be observed by glancing down the columns and across the rows that all 0-cubes are assimilated into 1-cubes. In the "synthesis 2" chart are listed all the 1-cubes, grouped according to their free component. It will thus be observed that row

one entry was added only to row two entry; the entire "synthesis 2" required 108 elementary additions. No pair of 2-cubes was summable, all the 2-cubes are cocycles, and so the operation ceased.

3. **Algorithm for finding k-extremals.** A cube e of K is called an *L-extremal* if it contains a vertex $d \in L$ which has the property that all of its cofaces are contained in E; d is termed *distinguished*. It follows that an extremal is a cocycle. If no confusion arises an L-extremal will be referred to as an extremal. Let E denote the space of all extremals.

3.1. THEOREM. *Any minimum cover C_0 contains E.*

Proof. Let d be a distinguished vertex which is covered by a nonextremal cube c of a cover C; let e be the extremal of d. Then $e \supset c$ and $C^* = (C-c) \cup e$ is a cover of lower cost. q.e.d.

3.2. COROLLARY. *If E constitutes a cover it is the unique minimal cover.*

We describe a method for locating the k-extremals which can be effected in several ways, depending upon the characteristics of the machine used for the computation. If for instance there is unlimited storage available (as in hand computation), then the operation can be effected so that no arithmetic operations are performed, but considerable storage is used. On the other hand, if memory is limited and arithmetic operations are fast (as with an electronic computer) then the algorithm can be carried out with very limited memory facilities while involving a sizeable number of very rapid arithmetic operations. (See discussion of this question of arithmetic versus memory by Goldstine [G]).

3.3. *The algorithm.* The 0-extremals are those vertices which do not form the face of any 1-cube and can be obtained simply by writing them down by inspection from synthesis 1. They can also be located by attempting to inject each vertex of L into each 1-cube—those which do not inject are the 0-extremals. These vertices are now set aside and are no longer considered in the process for locating the higher extremals.

The 1-extremals are located by finding those vertices which have exactly 1-coface, that is, are incident to exactly one 1-cube. Again these can be located simply by inspecting synthesis 1 or by the process of injection. We now set aside all the 1-distinguished vertices and in fact all the vertices of the 1-extremals; these vertices have been "provided for" by the 1-extremals.

We determine the 2-distinguished vertices and the 2-extremals by the same process: From the vertices remaining we find those which are incident to exactly one 2-cube, this obtained either by an injection process or else from an "incidence matrix" between the remaining vertices and the 2-cubes. The process proceeds inductively to determine all k-extremals. If this method is applied to Example 2.3, it is discovered that $E = \{x0x01,\ 1x0x1,\ 1x00x,$

$0x101,\ 11x11\}$; inspection of the incidence matrix for this problem shows that if E is supplemented with $C=\{x0110,\ 000x0,\ 0011x\}$, a minimum cover is obtained of cost 29.

The k-extremals may also be located using only the synthesis charts; for instance, to find a 2-extremal we examine all the 1-cofaces of a vertex v: If each of these has exactly one and the same 2-coface, then this 2-coface is an extremal. And to find k-distinguished vertices we need restrict ourselves only to those vertices with exactly k 1-cofaces; this observation of course considerably cuts down the number of inspections that have to be made.

We are left with a collection of vertices which are incident to no k-extremal and a collection of "noncritical" cubes. The problem that remains is to select from these noncritical cubes a cover of these vertices which is minimal, such a cover is not in general unique.

4. **Some methods for generating "near-minimal" covers.** The subspace Z^r of r-cocycles contains, as shown in §3, the subspace E^r of r-extremals: Let $N^r=Z^r-E^r$. The remaining problem is to select a cover for the vertices not covered by the extremals so that the total cover has a minimum "cost." The method which we describe below yields a cover which cannot be improved by the dispensing operations of Quine [Q1]; nevertheless it does not in general yield a minimum.

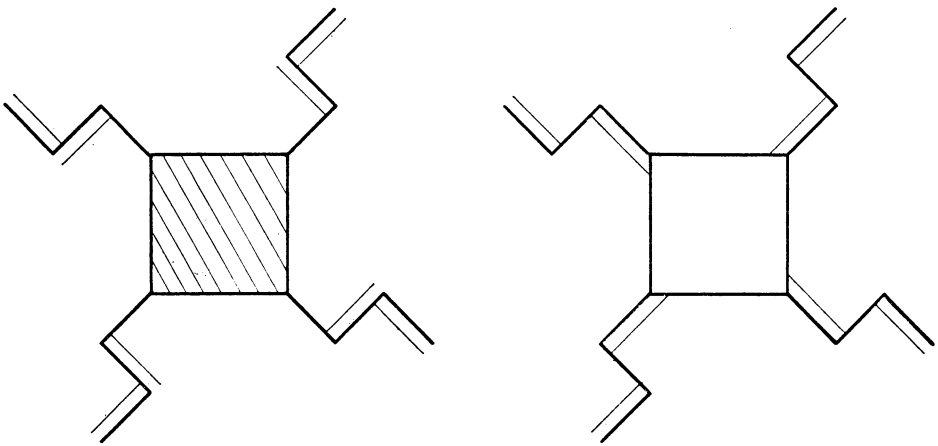
4.1. *The algorithm.* Let N^m be the highest dimensional nontrivial N^r. First we catalog the "incidence matrix" of the 0-vertices not contained in any of the extremals with the m-cubes of N^m. Of course this need not be actually constructed: For, the information can be read off of the cubing-operation charts; or else the information may be generated by the "injection" process, injecting or attempting to inject the uncovered vertices into the m-cubes of K^m, this being a very fast mchine operation. We order these m-cubes in an arbitrary fashion, for instance, in the order of their occurrence. Consider the first m-cube in the order selected: If it has a vertex which is not already covered and which is not incident to any of the remaining cubes in the ordering, we include it in the covering, otherwise not. Proceeding inductively, consider the nth cube in the ordering: If it has a vertex which is not already covered by the cubes thus far selected and which is not incident to any of the remaining cubes in the ordering, we include it in the covering, otherwise not. This completes the description of the selection of an *m-cover*. We then proceed inductively selecting the $(m-1)$th cover, etc. down to the 1st cover, at each stage dealing with the vertices that have not already been considered.

4.2. THEOREM. *The cover obtained by the above-described process cannot be simplified by Quine's dispensing operation.*

Proof. In the dispensing operation a clause is "dispensed with" if it implies the remainder of the clause: Translated into our terminology this merely

means that each vertex of a given cube (corresponding to the clause under consideration) is contained in other cubes of the cover (the cover corresponds to a particular normal form of the truth function), but by the method of selection of our cover, no cube is included in the cover unless it has a vertex not contained by any other cube. q.e.d.

4.3. The following *example* shows, however, that this method does not in general produce a minimum cover even if one considers all possible orderings of the cubes of the various N^r's:



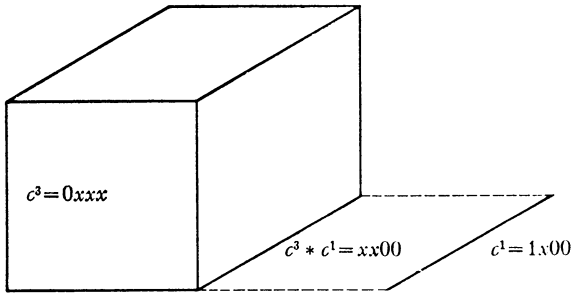
Cover selected by algorithm Minimal cover

The doubly colored cubes denote the elements included in a cover.

4.4 FIG.

5. **The $*$-algorithm.** The cubing algorithm described in §1 for computing the set of elementary cocycles of K requires one to start with the collection of all 0-cubes of K. A possible knowledge of the existence of higher dimensional cubes is not utilized to simplify the computation. Frequently, however, the initial statement of the problem supplies us with a knowledge of some of these cubes and in this section we describe another algorithm which utilizes such information if initially available. The number of steps required in general to effect the computation is considerably reduced, and when the initial information is given in the form of a collection of 0-cubes, this new algorithm reduces to the old one.

We begin with a definition of the $*$-product of two cubes c^r and c^s; geometrically this product is the largest t-cube which contains opposite $(t-1)$-faces in c^r and c^s: If there is none, then the product is said to be 0 or the empty cube.



5.0. *Illustration of* $*$*-product of cubes.* $n=4$.

First we define coordinate $*$*-multiplication*

$*$	0	1	x
0	0	y	0
1	y	1	1
x	0	1	x

5.1. *Definition of* $*$*-product of cubes.* Let $c^r=(a_1, \cdots, a_n)$ and $c^s=(b_1, \cdots, b_n)$ be cubes of a complex K, where the coordinates a_i, b_j are 0, 1 or x. Now if more than one y appears in the $*$-product of the coordinates, c^r*c^s is said to be 0 or the empty cube; if on the other hand, at most one y appears, then

$$c^r*c^s = [i(a_1*b_1), \cdots, i(a_n*b_n)]$$

where

$$i(0) = 0, \qquad i(1) = 1, \qquad i(x) = x, \qquad i(y) = x.$$

Note: if $c^r \cap c^s = c^t$ then $c^r*c^s = c^t$.

5.1′. *Arithmetic realization of the* $*$*-product.* The following scheme for representing arithmetically the $*$-product may be used[2]: Let

$$x = (0,0), \qquad 0 = (0,1), \qquad 1 = (1,0), \qquad y = (1,1)$$

then the $*$-product is effected by adding the components of the representation by means of Boolean addition; thus, e.g. $0*1=(0,1)+(1,0)=(1,1)=y$.

5.2. EXAMPLES. $10x*11x=1xx$ and $11x10*0xx11=\phi$.

PROPOSITIONS.

5.3. $*$*-multiplication is commutative and nonassociative. For instance* $(10x*11x)*0xx=xxx$ *while* $10x*(11x*0xx)=1xx$.

[2] This scheme was suggested by James Pomerene of The Institute for Advanced Study, ECP.

5.4. *Dimension* $(c^r * c^{r+s}) \leqq 1+r$, $r, s \geqq 0$, *where the dimension of the empty cube is agreed to be* -1.

5.5. *If* $c^r \subset c^s$ *then* $c^r * c^s = c^r$.

5.6. $(a_1, \cdots, a_n) \supset (b_1, \cdots, b_n)$ *if and only if* $a_i = b_i$ *or* x *for each* i.

5.7. *If* c^r *and* $\bar{c}^r$ *are opposite faces of* c^{r+1}, *then* $c^r * \bar{c}^r = c^{r+1}$.

This last proposition shows the relation of the $*$-product with the cubing algorithm.

5.8. *The actual* $*$*-algorithm.* Let $\hat{C}_0$ be an initially given cover of K. "Subsume" $\hat{C}_0$, i.e. form a cover $C_0 \subset \hat{C}_0$ with the property that no cube of C_0 is the face of any other.

5.9. LEMMA. $Z^0 = \{c \mid c \in C_0$ *and* $c * C_0 \not\supset$ *any 1-cube*$\}$.

Proof. Let U denote the set on the right side of the equality. Let $a \in Z^0$. Suppose that there is a cube b of C_0 such that $a * b = c$ is a 1-cube. Then in the coordinate $*$-product of a and b exactly one y will appear. Suppose then, without loss of generality, that $a_1 * b_1 = y$. Then for $i > 1$, $a_i * b_i = a_i$. Thus $a * b = (x, a_2, \cdots, a_n) = c$. But then $\delta_1 a = c$ so that a could not be an element of Z^0, contrary to hypothesis. Hence $Z^0 \subset U$. Next we show that $U \subset Z^0$. Let then $u = (u_1, \cdots, u_n) \in U$. Since $u * u = u$, u must be a 0-cube. Suppose, however, that $\delta_1 u = (x, u_2, \cdots, u_n)$ belongs to K. Then $\bar{u} = (\bar{u}_1, u_2, \cdots, u_n)$, where $\bar{u}_1 = u_1 + 1$ modulo 2, is a vertex of K and thus there must be a cube $v = (v_1, \cdots, v_n)$ in C_0 covering $\bar{u}$. By 5.6, for $i > 1$, $v_i = u_i$ or x. On the other hand v_1 must equal $\bar{u}_1$ for if v_1 were x, then v would contain u contrary to the hypothesis that C_0 contains no cube which is the face of another. Then $u * v = (x, u_2, \cdots, u_n)$. Thus $u * C_0$ does contain a 1-cube, contrary to hypothesis. q.e.d.

The criterion given by this lemma enables us to pick out Z^0. Let

$$\hat{C}_1 = (C_0 - Z^0) \cup (C_0 * C_0 - \text{its 0-cubes}).$$

Let $C_1 = \hat{C}_1$ subsumed. (Note: $C_1 \cup Z^0$ is a cover, C_1 being a cover only when $Z^0 = \phi$.)

5.10. LEMMA. C_1 *contains all 1-cubes or cofaces thereof.*

Proof. Let c be a 1-cube. If C_0 contains c or a coface thereof then so too does C_1. Suppose then the contrary. Without loss of generality we may assume $c = (x, c_2, \cdots, c_n)$ with $c_i = 0$ or 1 for $i > 1$. Then its faces $(1, c_2, \cdots, c_n)$ and $(0, c_2, \cdots, c_n)$ must lie on distinct cubes a and b of C_0 respectively. By 5.6 $a_i = c_i$ or x, and $b_i = c_i$ or x for $i > 1$. Furthermore it must be that $a_i = 1$ and $b_1 = 0$ for if not a or b would contain c contrary to hypothesis. Hence $a_1 * b_1 = y$, and $a_i * b_i = c_i$ or x for $i > 1$ and hence, by 5.6 $a * b$ contains c. Thus $C_0 * C_0 \supset c$ and so too then does C_1. q.e.d.

5.11. Before proceeding with the formal mathematical induction let us look at the next step in the algorithm. We form $C_1 * C_1$ and establish that

$Z^1 = \{c \mid c \in C_1 \& c * C_1 \not\supset \text{any 2-cube}\}$, we set $\hat{C}_2 = (C_1 - Z^1) \cup (C_1 * C_1 - \{\text{its 0- and 1-cubes}\})$ and $C_2 = \hat{C}_2$ subsumed.

Now let us suppose that for all $n \leqq r$, C_n, Z^n, have been computed and that C_n contains all n-cubes for K or cofaces thereof. Then we define

$$C_{r+1} = (C_r - Z^r) \cup (C_r * C_r - \{\text{its } s\text{-cubes}, s < r+1\})$$

and $C_{r+1} = \hat{C}_{r+1}$ subsumed. The proofs of the following two lemmas are slight extensions of those of 5.9 and 5.10.

5.12. LEMMA. *C_{r+1} contains all $(r+1)$-cubes of K, or cofaces thereof.*

We then locate Z^{r+1} by the following criterion.

5.13. LEMMA. $Z^{r+1} = \{c \mid c \in C_{r+1}$ *and* $c * C_{r+1} \not\supset$ *any* $(r+2)$*-cubes*$\}$.

If this follows that our algorithm determines the space Z of cocycles of K.

5.14. *Example of a computation of Z by $*$-algorithm.* The following example is discussed by Quine [Q1, 628]. In the first column of the computation for Z^0 below are the *given* cubes: for instance, $ps = 1xx1x$, $pqrt = 111x1$.

COMPUTATION OF Z^0

		(a)	(b)	(c)	(e)
(a)	$1xx1x$				
(b)	$0xx0x$	$yxxy$			
(c)	$x0xx1$	$10x11$	$00x01$		
(d)	$1x11x$	$1x11x$			
(e)	$x110x$	$111yx$	$0110x$	$xy101$	
(f)	$111x1$	11111	$y1101$	$1y1x1$	11101

$$\Rightarrow Z^0 = \phi$$

COMPUTATION OF Z^1

		(a)	(b)	(c)	(e)	(g)	(h)
(a)	$1xx1x$						
(b)	$0xx0x$						
(c)	$x0xx1$						
(e)	$x110x$						
(g)	$111xx$	$1111x$	$y110x$	$1y1x1$	$1110x$		
(h)	$xx101$	$1x1y1$	$0x101$	$x0101$	$x1101$	11101	
(i)	$1x1x1$	$1x111$	$yx101$	$101x1$	11101	$111x1$	$1x101$

$$\Rightarrow Z^1 = \phi$$

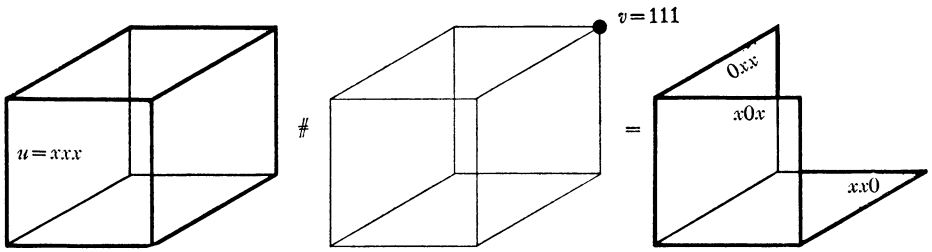
Since no new cubes appear in computation of Z^1 the algorithm stops. The first column above gives a listing of Z^3 and Z^2; Z^4 and $Z^5 = \phi$.

6. First method for locating E: the $*$-algorithm. The $*$-algorithm gives an economical method for computing Z; on the other hand the methods of §3 for locating the extremals cannot be used following the $*$-algorithm, for these methods utilized the "incidence relations" between vertices and cocycles, and for the $*$-algorithm, vertices are not determined, nor are these

incidence relations immediately available. The #-algorithm to be described in this chapter outlines another method for computing the extremals which can be used following the $*$-algorithm.

Before making a formal definition of the #-product of two cubes, we shall give some geometrical illustrations.

6.1. *Geometrical illustration of the #-product.* If u and v are cubes then $u \# v$ geometrically is the complex determined by the vertices of u that are not in v; thus # is a kind of differencing operation. In Figure 6.1′, $u = xxx$, $v = 111$ and



$$u \# v = xxx \# 111 = 0xx + x0x + xx0$$

6.1′ FIG.: Illustration of #-product.

6.2. *Definition of #-product.* The *#-product of coordinates* is first defined, in the following table

#	0	1	x
0	z	\$	z
1	\$	z	z
x	1	0	z

here z and \$ are symbols which serve to determine the actual #-product of the cubes represented by the coordinates. As would be expected from 6.1 the #-product of coordinates is noncommutative.

Let $c^r = (a_1, \cdots, a_n)$ and $c^s = (b_1, \cdots, b_n)$ be cubes of a complex K. If $a_i \# b_i = \$$ for any i, we say that

$$c^r \# c^s = c^r.$$

Geometrically the occurrence of such a product, $a_i \# b_i = \$$, among the coordinates means that c^r and c^s lie on opposite $(n-1)$-cubes.

If $a_i \# b_i \neq \$$ for all i, the product is defined as follows: Suppose $a_i \# b_i = \alpha_i$ where α_i is either 0 or 1; then

$$a \# b = \sum (a_1, \cdots, a_{i-1}, \alpha_i, a_{i+1}, \cdots, a_n),$$

the sum running over all i for which $a_i \# b_i = 0$ or 1. If $a_i \# b_i = z$ for all i, then

$$a \# b = \phi.$$

Note that if $a \# b = \phi$ then $a \subset b$. Note also that $a \# b \subset a$, so that the #-product always remains within the complex.

6.3. The following EXAMPLE indicates how this product can be mechanized: Let $u = xxx0$, $v = 00xx$; we compute $u \# v$ by the following scheme

u	$xxx0$
v	$00xx$
coordinate product	$11zz$
u	$xxx0$
$u \# v$	$1xx0 + x1x0 \leftarrow$ obtained by expanding third & fourth rows by "minors" of the third (the z entries are ignored).

6.4. LEMMA. *The #-product satisfies the following distributive law, where u, v, w are cubes,* $(u+v) \# w = u \# w + v \# w$.

It is clear that the #-product is noncommutative and nonassociative but it does have the following kind of commutativity property, which figures very importantly in the #-algorithm.

6.5. LEMMA. *If u, v and w are cubes, then* $(u \# v) \# w = (u \# w) \# v$.

6.6 EXAMPLE.

$$\begin{aligned}(xxx0 \# 1xx0) \# 000x &= 0xx0 \# 000x = 01x0 + 0x10,\\ (xxx0 \# 000x) \# 1xx0 &= (1xx0 + x1x0 + xx10) \# 1xx0\\ &= \phi + 01x0 + 0x10 = 01x0 + 0x10.\end{aligned}$$

The lemma following provides the basis for the #-algorithm for computing the extremal set E. Let the cocycle set be indexed in some fashion (say, indexing first the highest dimensional cocycles, then the next highest and so on):

$$Z = \{z_1, z_2, \cdots, z_p\}.$$

6.7. LEMMA. *z_i is an extremal if and only if*

$$(\cdots((\cdots(((z_i \# z_1) \# z_2) \# \cdots \# z_{i-1}) \# z_{i+1}) \# \cdots \# z_p \neq \phi.$$

By 6.5 the product is independent of the manner of indexing.

The lemma following provides a method for determining whether or not a random collection C_1 of cubes is a cover.

Let c_α be a cube and C_β a collection of cubes. The product

$$c_\alpha \# C_\beta$$

is to denote the iterated product of c_α with every element of C_β; by 6.5 this product is independent of the order selected for the product. Let C_λ and C_μ be collections of cubes: Then

$$C_\lambda \# C_\mu = \{c_\lambda \# C_\mu \mid c_\lambda \in C_\lambda\}.$$

6.8. LEMMA. *If C_0 is a cover, then C_1 is a cover if and only if $C_0 \# C_1 = \phi$.*

6.9. *Description of the #-algorithm.* Suppose that the cocycle set Z has already been computed, say by the $*$-algorithm. In accord with 6.7 we form the iterated product of each cocycle with all the other cocycles. If this product is not zero, then the cocycle is an extremal. This would seem to require a large number of products but again, as with the cubing algorithm, the probable number of operations is significantly less.

One property of the #-product which is useful in reducing the number of arithmetic operations of the algorithm is that if $a=(a_1, \cdots, a_n)$ and $b=(b_1, \cdots, b_n)$ are cubes and if for some i, $a_i=1$ and $b_i=0$, then $a \# b=a$.

Here is another criterion useful in reducing computations.

6.10. LEMMA. *If a and b are cocycles, $c=a*b$ and $a \neq b$, then c is not an extremal.*

6.11. *Example of #-algorithm.* We shall compute the extremals for the example discussed in 5.14. By 6.10 we need consider only cubes (a), (b), (c), (e) and (i).

COMPUTATION TO DETERMINE WHETHER a IS AN EXTREMAL

a	$1xx1x$		
b	$0xx0x$		
	$		
$a \# b$	$1xx1x$		
c	$x0xx1$		
	$\cdot 1 \cdot \cdot 0$		
$(a \# b) \# c$	$11x1x$	$+\ 1xx10$	
e	$x110x$	$x110x$	
	$\cdot\cdot 0$$	$\cdot 00$$	
$((a \# b) \# c) \# e$	$11x1x$	$+\ 1xx10$	
g	$111xx$	$111xx$	
	$\cdot\cdot 0\cdot\cdot$	$\cdot 00\cdot\cdot$	
$a \# b \# c \# e \# g$	$1101x$	$+\ 10010$	$+\ 1x010$
h	$xx101$	$xx101$	$xx101$
	$\cdot\cdot$$	$\cdot\cdot$$	$\cdot\cdot$$
$a \# b \# c \# e \# g \# h$	$1101x$	$+\ 10010$	$+1x010$
i	$1x1x1$	$1x1x1$	$1x1x1$
	$\cdot\cdot$$	$\cdot\cdot$$	$\cdot\cdot$$
$a \# (Z-a)$	$1101x$	$+\ 10x10$	$+1x010$

$\Rightarrow a$ is an extremal.

COMPUTATION TO DETERMINE WHETHER i IS AN EXTREMAL

	$1x1x1$
	$1xx1x$
	$\cdots 0\cdot$
	$1x101$
	$0xx0x$
	$
	$1x101$
	$x0xx1$
	$\cdot 1 \cdots$
	11101
	$x110x$
	$\cdot\cdot\cdot\cdot\cdot$

$\Rightarrow i$ is *not* an extremal

By similar computations one finds that b and c are extremals and that e, g, h and i are not. Thus

$$E = a + b + c.$$

7. **Second method for locating E: the $\cap$-algorithm.** We shall now describe another method, called the *$\cap$-algorithm*, for computing E, usually requiring less operations than the #-algorithm.

7.1. First the *$\cap$-product of coordinates* is defined, in the following matrix equation

$\cap$	0	1	x
0	0	ϕ	0
1	ϕ	1	1
x	0	1	x

Let $c=(a_1, \cdots, a_n)$ and $c'=(b_1, \cdots, b_s)$ be cubes; then

$$c \cap c' = \phi \text{ if for any i } a_i \cap b_i = \phi$$

$$c \cap c' = (a_1 \cap b_1, \cdots, a_n \cap b_n) \text{ otherwise.}$$

7.2. Next we observe that from any set of cubes can be constructed a unique cubical complex by applying to them the $*$-algorithm. We shall term this the cubical complex *defined* by the set of cubes.

7.3. THEOREM. *A cube e is an extremal if and only if the complex defined by the set $\{e \cap z_i \mid z_i \in Z-e\}$ coincides with e.*

7.4. Thus in the example of 5.14 $a \cap (Z-a) = 10x11+111x+1x111$, whose cubical complex does not coincide with a. Thus a is an extremal.

7.6. REMARK. Regardless of the manner in which E is computed the problem remains of how to supplement E to form a minimal cover. Part II of this paper will describe how this may be accomplished effectively. We remark, however, that the methods of article 4 for obtaining a "near-minimal" cover are easily adapted to the case when Z is computed by the $*$-algorithm. Cf. [R4, 31].

8. **Function-space formulation of switching-system synthesis problem** (Cf. [R3]).

Let K be a cubical complex contained in an n-cube Q; a *pair* (X, A) consists of a linear graph X and an *assignment* A which attaches to each branch of X an $(n-1)$-cube of Q. Let p, q be vertices of X and α a geometrical (*acyclic*) 1-chain linking p and q; A defines a corresponding algebraic 1-chain α^*, the coefficient of a cell σ of α^* being the $(n-1)$-cube assigned to α by A. The *Kronecker index* of α^* is the set-theoretic product or intersection of its coefficients. Let $\Omega(X, p, q; A)$ be the space of all such "loops" α^*. The pair

(X, A) shall be *admissible* with respect to K if the union of the Kronecker indices of the elements of Ω defines precisely the complex K. The *switching system problem* is now to find an admissible pair such that the number of branches of X is a minimum. (The reader will see that the multiple terminal problem can be given a similar though more complicated formulation.)

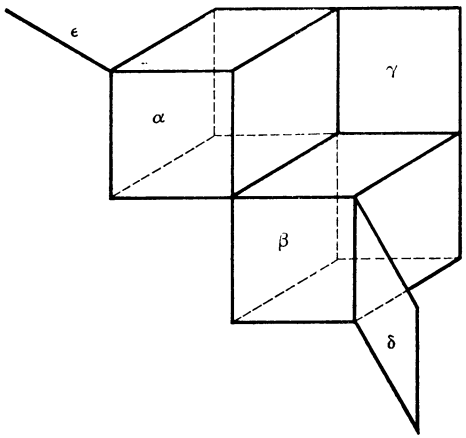
It has been shown [R4, 31–36] that the logical problem of finding a Boolean function equivalent to a given one, having a minimum number of literals, is interpretable in this context as restricting X to be of a particular form, termed a series-parallel, or block-network. In general, the minimum for the logical problem exceeds that for the problem just formulated.

REMARK. We call the number of branches of an admissible pair its *cost*. Now while there is an infinity of pairs admissible with respect to a given K, there is only a finite number of such whose cost

$$c(X, A) \leqq M$$

for any positive M. The problem then is to discover an efficient way of generating all those admissible with respect to a given K satisfying such an inequality for some M. (It is easy to show for instance that for K contained in an n-cube, M may be chosen to be not more than $n2^{n-1}$). The algorithm described below is such an attempt.

8.1. The tensor algorithm will be described by an example. In this example, figure 8.2, the complex A is presented as five cubes.



	A	B	C	D	E	F←variable
$\alpha =$	x	x	x	0	0	0
$\beta =$	0	0	x	x	x	0
$\gamma =$	x	0	0	x	0	0
$\delta =$	0	0	1	1	x	x
$\epsilon =$	1	1	1	x	0	0

FIG. 8.2

8.3. *The complex A all of which are extremals*; (this collection thus comprises the unique minimum cover for the problem of Quine, of cost 19). The

idea of the algorithm is to discover equalities of components of various sets of cubes by examining them pairwise, three-at-a-time, etc. The computation is exhibited below

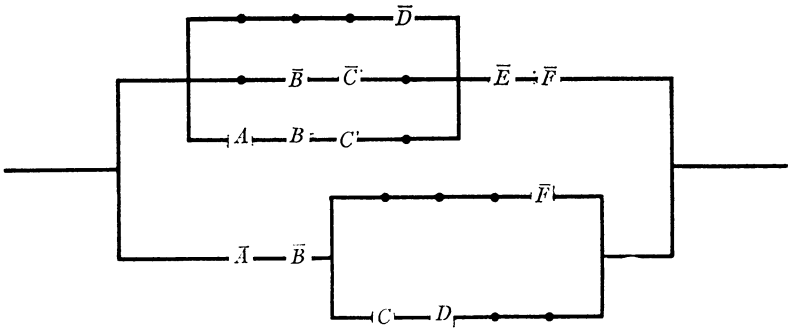
	α	β	γ	δ
α $xxx000$				
β $00xxx0$	$0x$ $0x$ xx $x0$ $x0$ 00			
γ $x00x00$	xx $0x$ $0x$ $0x$ 00 00	$x0$ 00 $0x$ $x0$ 00 00		
δ $0011xx$	$0x$ $0x$ $1x$ 10 $x0$ $x0$	00 00 $1x$ $1x$ xx $x0$	$0x$ 00 10 $1x$ $x0$ $x0$	
ϵ $111x00$	$1x$ $1x$ $1x$ $x0$ 00 00	10 10 $1x$ xx $0x$ 00	$1x$ 10 10 xx 00 00	10 10 11 $x1$ $0x$ $0x$

9.4 $A \otimes A$

We now seek an "extended cover," that is, a set of tensor products such that each cube appears in one of the elements selected; inspection of $A \otimes A$ and $A \otimes A \otimes A$ leads to the solution (higher tensor products here lead to no further simplifications)

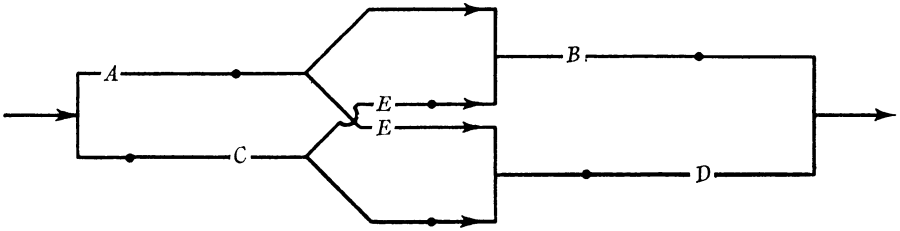
$$\alpha \otimes \gamma \otimes \epsilon = xx1 \quad x01 \quad x01 \quad 0xx \quad 000 \quad 000$$
$$\beta \otimes \delta = 00 \quad 00 \quad x1 \quad x1 \quad xx \quad 0x$$

this has the circuit realization



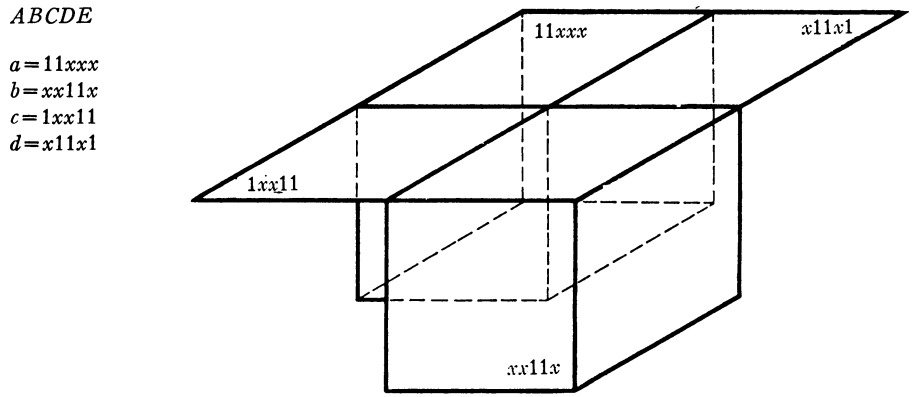
whose cost is 13. This method leaves much to be desired.

8.4. SECOND EXAMPLE. The general problem is to determine from the topological and positional properties of a cubical complex what are its admissible pairs. In Figure 8.5 a complex K is defined by set $E = \{a, b, c, d\}$ of cubes each of which are extremals. Let $[ab]$ denote the tensor product of a and b, etc. Inspection of $F = E + E \otimes E + E \otimes E \otimes E$ and $F + F \otimes F$ leads to the following extended cover, which may be shown to be minimal over the class of directed systems.



[[ac], [db]]

9. **The synthesis problem for multiple-valued switches.** Attention thus far has been restricted to consideration of problems connected with 2-valued switches, and indeed this restriction is the common one in engineering practice [K-R-W4], dictated by technological reasons. It is of interest nevertheless to see whether the methods can be extended to deal with the more general situation, and it turns out that to a remarkable extent they can be extended rather easily; this extension will only be sketched here.



8.5. A cubical complex

9.1. The problem may be described as follows. We are given variables $A, B, \cdots, Z$, each of which take on $p+1$ values $0, 1, \cdots, p$. We specify that for the following "vectors" (here the a's, b's, $\cdots$, z's are integers between 0 and p)

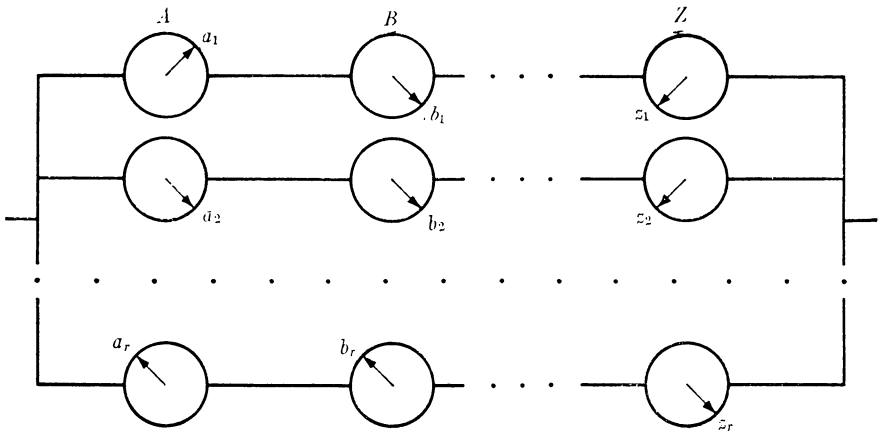
$$A\ B \cdots Z$$

$$v_1 = a_1 b_1 \cdots z_1$$

$$v_2 = a_2 b_2 \cdots z_2$$

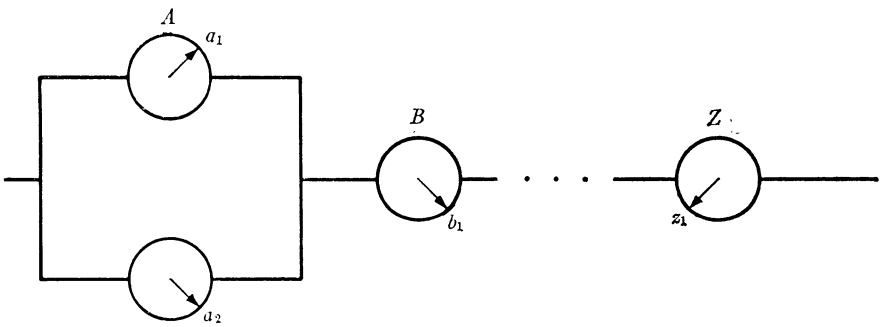
$$v_r = a_r b_r \cdots z_r$$

the "circuit shall be closed" and for no others. A circuit realization of this array would be



9.2 FIG.

Now if for example $b_1 = b_2, \cdots, z_1 = z_2$, then the first two rows of the above circuit could be simplified to

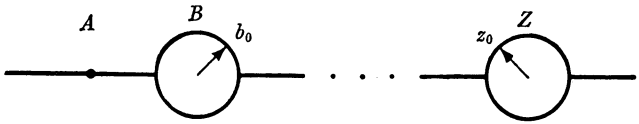


9.3 FIG.

We would say that v_1 and v_2 are *A-equivalent*, meaning that all but their A-components agree. It is immediate that A-equivalence is an equivalence relation. Now if $p = 1$, the variables would be binary, then the switches A in 9.3 could be eliminated as superfluous and in accordance with the definitions of 1.1 the *pair* (v_1, v_2) would form a 1-cube. For $p > 1$, the analogous circumstance would be $p+1$ distinct vectors $v_0, \cdots, v_p$ all A-equivalent: this set of vectors could then be identified as a "1-cube," *par un abus de langage*, denoted

$$(xb_0 \cdots z_0)$$

where x is to be thought of as a "random" variable taking on values $0, 1, \cdots, p$; this cube would be represented by the circuit



9.4. *The analogue of the cubing algorithm* for multiple-valued switches involves the following steps:

1°. The problem is assumed to be given in the form of a collection of vectors, each vector being a set of values of the domain for which "the circuit shall be closed."

2°. Each vector is subtracted componentwise from each other vector: those pairs for which exactly one component of the difference, say the K-th, is not zero is said to be *K-equivalent.*

3°. These equivalence classes are collected together: If $p+1$ distinct vectors are K-equivalent, these vectors form a 1-*cube*; these 1-cubes are determined.

4°. Exactly as with the cubing algorithm, 1-cubes may be subtracted only if their "free" variables occur in the same component.

5°. The procedure goes inductively to determine s-cubes of each dimension s.

9.5. *Example of "cubing" algorithm for multiple-valued switches,* $p=3$, $n=4$.

$A\ B\ C\ D \leftarrow$ variables											
0 1 2 0											
1 1 2 0	1 0 0 0										
1 2 2 0	1 1		0 1 0 0								
2 2 2 2	2 1		1 1		1 0 0 2						
2 2 2 0	2 1		1 1		1 0 0 0		0 0 0 2				
2 1 2 0	2 0 0 0		1 0 0 0		0 0		0 1 0 2		0 1 0 0		
↑ 0-cubes											

This computation yields the 1-cube $x120$, and the equivalence classes (1120, 1220) and (2220, 2120); this collection covers all "0-cubes," and whereas the circuit realization for the array of "0-cubes" is 24, that for this array is $3+5+5=13$. For this problem the algorithm ceases at this point.

Investigation of analogues for other algorithms developed for the binary problem will be left to the reader.

BIBLIOGRAPHY

[A] H. H. Aiken (editor), *Synthesis of electronic computing and control circuits*, Harvard University Press (1951).

[C] S. H. Caldwell, *Discussion of Karnaugh's paper*, AIEE vol. 72 Part I (1953) pp. 598–599.

[E–Z] S. Eilenberg and J. A. Zilber, *Semi-simplicial complexes and singular homology*. Ann. of Math. 51 (1950) pp. 499–513.

[G] H. H. Goldstine, *On the relation between machine developments and numerical analysis*, paper presented at Bonn Colloquium, October 20–22, 1955.

[H] D. A. Huffman, *The synthesis of sequential switching circuits*, Journal of the Franklin Institute vol. 257 (1954) pp. 161–190.

[J] G. T. Jacobi, unpublished paper.

[K] M. Karnaugh, *The map method for synthesis of combinational logic circuits*, Trans. AIEE vol. 72, Part I (1953) pp. 593–598.

[K–R–W] W. Keister, A. E. Ritchie and S. H. Washburn, *The design of switching circuits*, Chapter 5, New York, D. Van Nostrand Co., 1951.

[M–P] W. S. McCulloch and W. Pitts, *A logical calculus of the ideas immanent in nervous activity*. Bulletin of Mathematical Biophysics vol. 5 (1943) pp. 115–134.

[M] G. A. Montgomerie, *Sketch for an algebra of relay and contactor circuits*, J. IEE vol. 95, Part III (1948) pp. 303–312.

[M,E] E. F. Moore, *A table of four-relay two-terminal contact networks—case* 22108, Bell Telephone Laboratory Technical Memo. MM-52-180-45.

[N–H] A. Nakasima and M. Hanzawa, *The theory of the equivalent transformation of simple partial paths in relay circuits*. J. Inst. Elec. Comm. Engrs. Japan, 1936–1938.

[N] Raymond J. Nelson, *Review of Quine's paper*, J. Symbolic Logic vol. 18 (1953) pp. 280–283.

[N1] R. L. Nelson, *Simplest normal truth functions*, J. Symbolic Logic vol. 20 (1955) pp. 105–108.

[N,R] R. Z. Norman, unpublished paper.

[QO] W. V. Quine, *Methods of logic*, New York, 1950.

[Q] ———, *The problem of simplifying truth functions*, Amer. Math. Monthly vol. 59 (1952) pp. 521–531.

[Q1] ———, *A way to simplify truth functions*, Amer. Math. Monthly vol. 62 (1955) pp. 627–631.

[R1] J. P. Roth, *A combinatorial topological method for the synthesis of switching systems in n variables*, Bull. Amer. Math. Soc. Abstract 62-2-281.

[R2] ———, *An algorithm for the problem of Quine*, Bull. Amer. Math. Abstract 62-3-367.

[R3] ———, *A function-space formulation of the switching circuit synthesis problem*, Bull. Amer. Math. Soc. Abstract 62-3-392.

[R4] ———, *Algebraic topological methods for the synthesis of switching systems in n variables*, The Institute for Advanced Study, Princeton, ECP 56-02, April 1956.

[S] C. E. Shannon, *A symbolic analysis of relay and switching circuits*, Trans. AIEE vol. 57 (1938) pp. 713–723.

[S1] C. E. Shannon, *The synthesis of two-terminal switching circuits*, Bell System Technical Journal vol. 28 (1949) pp. 59–98.

[V] E. W. Veitch, *A chart method for simplifying truth functions*, Proceedings of the Association for Computing Machinery, Richard Rimbach Associates, Pittsburgh, 1952, pp. 127–133.

THE INSTITUTE FOR ADVANCED STUDY,
PRINCETON, N. J.