UCRL-9227

UNIVERSITY OF CALIFORNIA

Lawrence Radiation Laboratory
Berkeley, California

Contract No. W-7405-eng-48

THE REDUCTION OF BOOLEAN TRUTH FUNCTIONS TO MINIMAL FORM

Alan Natapoff

May 20, 1960

DISCLAIMER

THE REDUCTION OF BOOLEAN TRUTH FUNCTIONS TO MINIMAL FORM

Contents

Abstract 3

Introduction 4

I. The Unique Portion of the Reduction of the Original Truth Table . 8

II. The Generation of the Class of Minimal Coverings

A. Outline of the Algorithm 14

B. Proof of the Central Theorem 15

C. Development of the Algorithm and Partition of the Residue into Components 17

D. Formal Characterization of the Partition in Terms of an Equivalence Relation 19

E. Uniqueness of the Maximal Additive Partition 21

F. Formal Analogy Between a Component and an Irreducible Representation of a Group 23

G. Illustration of the Algorithm by a Computation on a Sample Truth Table 24

Conclusion. 32

Acknowledgments 33

Appendix - Description of IBM 704 Program Salomé 34

References 38

THE REDUCTION OF BOOLEAN TRUTH FUNCTIONS TO MINIMAL FORM

Alan Natapoff

Lawrence Radiation Laboratory
University of California
Berkeley, California

May 20, 1960

ABSTRACT

The problem of the reduction of an arbitrary truth function to the minimal union of basic cells is discussed. The solution to this problem has applications to pattern recognition and logical circuit design. An algorithm is presented that solves the problem and generates the class of minimal unions. It partitions an arbitrary truth function into a well-defined set of subfunctions (components) in such a way that the partition is invariant under all transformations that preserve the topology of the original truth function. It is shown that this reduction exhausts the minimal coverings of each subfunction and generates from these all minimal coverings of the original function. The theorem that "A union of cells basic to a vertex contains no further cells basic to that vertex" is proved and is used in the algorithm. The IBM 704 program (SALOMÉ) that performs the reduction is described in the Appendix.

THE REDUCTION OF BOOLEAN TRUTH FUNCTIONS TO MINIMAL FORM

Alan Natapoff

Lawrence Radiation Laboratory
University of California
Berkeley, California

May 20, 1960

INTRODUCTION

The algebra of Boolean truth functions is interesting because it is, roughly speaking, the algebra obeyed by logical elements in computers and by patterns of light and dark dots. Abstractly, it is the algebra of the sets of vertices of an N-dimensional Euclidean cube.[1] Its applications derive from the fact that any set of Boolean truth functions can be reduced to a unique class of minimal functions. The minimal property is of interest in the design of economical logical circuitry, and the uniqueness property, in the simple characterization of sets of similar patterns of light and dark dots. We examine some of the general aspects of the problem of simplifying truth functions in order to illustrate applications to computers and to define a procedure for finding the class of minimal Boolean functions.

The definitions of the terms given below are important for the understanding of what follows. They are those generally used in applications to logical circuitry (where elements can be so gated as to realize logical "and" and "or" operations) and may differ from usages current in abstract Boolean algebra.

Definitions

Boolean Variable: A symbol that can be assigned the value 0 (false) or the value 1 (true).

Boolean Expression (or vertex): An assignment, to each of a set of N Boolean variables, of the value 0 or the value 1 in each case.

(If the variables are written as $X_N \cdots X_3 X_2 X_1$ and the values assigned are written in the same order $1 \cdots 010$, each Boolean expression on N variables corresponds to a unique ordered N-tuple, each of whose components is either 0 or 1. These may be read as binary numbers for convenience. They may also be read as the coordinates of a point in an N-dimensional Euclidean space and, since each coordinate can take on only the value 0 or 1, the set of such expressions on N variables corresponds to the 2^N vertices of an N-dimensional cube. This correspondence explains the alternate term "vertex".)

Truth Table: A set of vertices on N variables whose minimal covering is demanded. (For definiteness, we assume that we are referring, in what follows, to a particular problem of a truth table on N variables $X_N \cdots X_3 X_2 X_1$.)

Don't Care Vertex: A vertex that may be included in the covering of the truth table where convenient. (The set of such vertices is prescribed at the same time as the truth table. Together they specify the problem to be solved. Every minimal covering covers the entire truth table. Where a more compact covering can be achieved by allowing one or more "don't care" vertices to be included, they are also covered.) We call the table of "true" vertices the short truth table, and the combined table of "true" and "don't care" vertices the long truth table.

Cell: If each of N Boolean variables may be either affirmed or negated, then 2^N distinct Boolean expressions may be constructed from them. If N-k of the N variables are kept fixed (that is, either specifically affirmed or specifically negated) and the other k are allowed to vary at will, then 2^k distinct Boolean expressions can be generated. Whenever N-k variables are specified to be kept fixed, the set of 2^k Boolean expressions generated by varying the remaining k is called a cell (of dimension k).

Complete Cell: A cell is complete if and only if each Boolean expression it can generate is on the long truth table.

Basic Cell (or Maximal Cell): A cell, C, is basic to the covering of a given vertex, V, if and only if C (1) contains V, (2) is compete, and (3) is not a proper subcell of any other complete cell. (It follows immediately that if C is basic to one vertex, it is basic to all vertices contained in it.)

Covering: A set of complete cells is a covering of a given truth table T if and only if each vertex of T is contained in at least one cell of the set.

Irredundant covering: A covering is irredundant if and only if no proper subset of it is also a covering.

Minimal Covering: A covering is minimal (with respect to some given criterion) if its norm (with respect to that criterion) is less than, or equal to, the norm of any other covering.

In logical circuit design, the norm usually adopted is the number of diodes required for the instrumentation of the logic. This can be expressed as a weighted sum of the number of cells and their dimensions. If K_i is the dimension of the ith cell, and m is the number of cells, the norm for the covering is expressed as

$$am + b \sum_{i=1}^{m} (N - K_i), \quad \text{for } a, b > 0,$$

where the particular values of a and b depend on the kind of logic used. The procedure outlined here gives the minimum circuitry only for two-stage logic of the N-input, 1-output type. We are concerned only with norms of this type. The solution of the general multistage N-input, M-output problem has been examined, for example, by Prather.[2] His solution depends on the prior solution of the problem we are considering.

Adjacent Vertex: A vertex, W, is adjacent to another vertex, V, if and only if there is exactly one Boolean variable, the ith, such that the components V_i and W_i corresponding to it differ, i.e. $V_i \neq W_i$.

Geometrically, this corresponds to the situation in which V and W are connected by an edge of the N cube, hence the name adjacent.

Essential Cell: A cell, E, is essential to the covering of the vertex, V, if and only if E (1) contains every true or don't care vertex that is adjacent to V, and (2) is complete. If we relax condition (2), this definition becomes that of cell L given in Part II. We prove there that L (and hence every essential cell) has the property: All complete cells covering V are subsets of L. If L is itself complete, it is called essential.

If V has an essential cell E, then every other cell covering V has a higher norm than E. We see this by noting that all other complete cells covering V are of smaller dimension, k_i, because they are subsets of E. Thus if a vertex has an essential cell, E, every minimal covering of that vertex must contain E.

I. THE UNIQUE PORTION OF THE REDUCTION OF THE ORIGINAL TRUTH TABLE

The first step in the reduction of the truth table is the inspection of each vertex of the short truth table, T, to determine whether or not it has an essential cell, E. If it has, we delete all the vertices covered by E from T, and examine the remaining vertices for essential cells. No cells of smaller norm are ignored by this procedure, because we know that essential cells, where they exist, are the cells of smallest norm covering the vertex in question. If a vertex has no essential cell of its own, it may still be covered by a cell essential to some other vertex.

After we have deleted all possible essential cells from T, we are left with a residue, **R**, of vertices not contained in any cell that is essential to a vertex. We now have the unique minimal covering of the portion of T apart from R. Supposing that **R** is not empty, as in general it is not, we then find a whole class of minimal coverings of **R**, each of which, when united with the essential cells for T, forms a minimal covering of T. Further, each such minimal covering is composed entirely of basic cells, since any nonbasic cell in a covering is always replaceable by a basic cell containing it, and the covering thus constructed is of smaller norm than the original.

The theory developed so far is illustrated in the following example in pattern recognition. (Exploitation of this approach, incidentally, has been meager). Suppose that we have a 2 by 2 grid on which each of the four points can be either dark ("off" or 0) or light ("on" or 1). There are exactly $2^4 = 16$ possible distinct configurations on such a grid. We can characterize, for example, that subset comprising all the patterns possible on the "main diagonal" of the grid. The patterns are:

Numbering the positions 1 2 we can symbolize the four patterns by

3 4

1000, 0001, 0000, 1001. These now represent the vertices on a four-dimensional cube, and form a truth table which we must now cover. The cell of dimension 2 specified by fixing bits 2 and 3 at 0 and allowing bits 1 and 4 to range at will, is essential to the covering of the first vertex of the table (in fact, to all the vertices). The set of patterns on the main diagonal can thus be characterized by the notation -00- (where - means that either zero or one may be substituted at will). For convenience we may adopt the notation of Harris,[3] in which a "2" replaces the symbol - and the essential cell 2002 (or -00-) stands for the set of vertices generated by all possible substitutions of 0 and 1 into the arguments where "2"'s appear. In this case, 2002 stands for the set of vertices (1000, 0001, 0000, 1001). If, for compactness, we read these binary expressions as decimal numbers, then we have the set (8, 1, 0, 9) for our truth table. Its minimal covering is 2002.

A short example and a detailed calculation of minimization of logical circuitry are given below.

If some computer circuitry had to examine two input bits and give as output the logical "or" of these bits, the required output for all possible input combinations would be as follows:

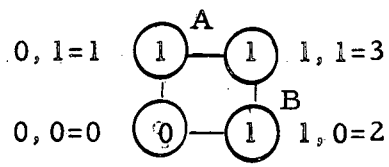
INPUT COMBINATIONS

Vertex Label *	Input X	Input Y	Output = X "or" Y
0	0	0	0
1	0	1	1
2	1	0	1
3	1	1	1

*The label for each input combination is its numerical value read as a binary number, e.g., for the case input X=1, input Y = 0, the label is $(10)_2 = 2$.

The table prescribes that vertices 1, 2, 3 be covered and that vertex 0 be left uncovered. (We cover those and only those vertices or input combinations whose corresponding output is "1".)

Representing the truth table geometrically, we have



The encircled number gives the output prescribed for that vertex.

Cell A is the cell of dimension 1 represented by 21

Cell B is the cell of dimension 1 represented by 12

Note that cell A is essential to vertex 1

B is essential to vertex 2 .

Of the vertices adjacent to vertex 1, one is on the long list. A contains this vertex and is complete, hence A is essential to vertex 1. Neither cell A nor cell B is essential to vertex 3. (In fact, vertex 3 has no essential cell, because no complete cell containing vertex 3 also contains all vertices adjacent to vertex 3 that are on the long list. The long list is identical with the short list here. In this case vertices 1, 2 are adjacent to vertex 3 and no complete cell contains vertices 1, 2, 3, implying immediately that vertex 3 has no essential cell.)

Together, A and B cover all of T, hence **R** is empty and our solution is complete and unique. Notice, also, that cells A and B are both 1 cubes (cubes of dimension 1) and that each contains one "2" in its representation (A = 21, B = 12). In general, the representation of a k cube has k "2"'s and N-k fixed arguments.

From this example, which gave a unique minimal covering, we proceed to the detailed calculation in a problem in which there is a class of minimal coverings.[4] There is a logical problem for six inputs (N = 6) which leads to the following truth table (Table I). (Each vertex is again named by the numerical equivalent of its binary representations.)

Our prescription requires that we examine each vertex of T to see whether or not it has an essential cell.

The first vertex of T is 4 = 000100.

The vertices adjacent to 4 are:

5 = 000101

6 = 000110

0 = 000000

12 = 001100

20 = 010100

36 = 100100

Each of these differs in exactly one bit from the representation of vertex 4 in accordance with our definition of "adjacent." In addition, we note that of these vertices:

6 is on the true list and differs from 4 in bit 2;

12 is on the true list and differs from 4 in bit 4;

36 is on the true list and differs from 4 in bit 6;

5 is on the don't-care list and differs from 4 in bit 1.

The other adjacent vertices, 0 and 20 are on neither list.

Accordingly, we construct the smallest cell containing all the vertices 4, 6, 12, 36 and 5. Vertex 4 has an essential cell if and only if this cell (our L or PELL, discussed in Part II; PELL stands for Potential Essential cELL) is complete.

Table I

Long list

True (Short List)	Don't Care
4 = 000100	1 = 000001
6 = 000110	3 = 000011
12 = 001100	5 = 000101
14 = 001110	7 = 000111
33 = 100001	9 = 001001
34 = 100010	11 = 001011
36 = 100100	13 = 001101
39 = 100111	15 = 001111
41 = 101001	17 = 010001
42 = 101010	19 = 010011
44 = 101100	21 = 010101
47 = 101111	22 = 010110
50 = 110010	23 = 010111
54 = 110110	24 = 011000
57 = 111001	25 = 011001
58 = 111010	26 = 011010
61 = 111101	27 = 011011
62 = 111110	28 = 011100
	29 = 011101
	30 = 011110
	31 = 011111
	49 = 110001
	51 = 110011
	53 = 110101
	55 = 110111

The vertex 4 is represented by 4 = 00100.

The smallest cell containing 4, 6, 12, 5 is represented by 202122, and can be constructed by replacing the original value of an argument of vertex 4 by "2" whenever a vertex adjacent to 4 and differing from 4 in that argument is on the long truth table.

In cell 202122, the 16 vertices that can be generated by varying the arguments in which "2" 's appear (and hence the vertices contained the cell) are 4, 5, 6, 7, 12, 14, 15, 36, 37, 38, 39, 44, 45, 46, 47. The vertex 37, for example, is not on the long truth table, therefore the cell is not complete and the vertex 4 does not have an essential cell.

The next vertex on the short truth table is vertex 6. The appropriate cell to examine is 022122. This cell contains, among others, vertex 20, which is not on the long truth table; 6, therefore has no essential cell either.

We repeat the process successively on the other vertices of T, the short table, and we find in each case that the cell appropriate to the vertex is incomplete until we reach vertex 33. The appropriate cell, 222001, for vertex 33 is complete and is thus the essential cell for vertex 33. The vertices it contains (1, 9, 17, 25, 33, 41, 49, 57) are all deleted from our short list, and placed in the don't care list. We then examine the remaining vertices of the short list and find that vertices 34, 36, 47, 61 have essential cells containing, respectively, the vertices, (34, 42, 50, 58), (4, 12, 36, 44), (7, 15, 39, 47), (17, 21, 25, 29, 49, 53, 57, 61).

When the elements of T contained in all five essential cells are deleted from T and placed on the don't care list the vertices 6, 14, 54, 62, are the only ones remaining uncovered. They comprise R, and the construction of the set of minimal coverings for them is discussed in II. We have completed the first step referred to in II.

II. THE GENERATION OF THE CLASS OF MINIMAL COVERINGS

A. Outline of the Algorithm

The class of minimal coverings (in terms of sums of products) of a given set (or short truth table*) of Boolean expressions can be generated by the following algorithm.

First: Find the maximal set of partial coverings which must appear in every minimal covering (the set of essential cells). Remove from the truth table of expressions to be covered all expressions that are covered by this maximal set, leaving a residue **R**, of expressions remaining to be covered. (The expressions covered by the essential cells are thereafter treated exactly as don't care vertices).

Second: Partition **R** into clusters[5] of uncovered expressions that we call the "components" of **R**. Each component has the property that if any complete cell covers two vertices of **R**, the vertices belong to the same component.

Third: Construct the class of minimal coverings of each component. Any selection of one minimal covering from each class constitutes a minimal covering of the entire residue **R**.

Each covering of **R**, taken together with the essential cells of step one, forms a minimal covering of the entire short truth table.

We now show that each part of the algorithm is feasible. The first part has already been successfully examined by various investigators.[6] The second part is developed below.

*The short truth table is the set of true statements; the long truth table = short truth table plus the set of "don't care" statements. Every covering must cover the short, and be covered by, the long truth table.

B. Proof of the Central Theorem

Sublemma: If C is a cell containing (or covering) a vertex V, then for every argument i such that $C_i \neq 2$ we have $C_i = V_i$.

Proof of sublemma: The cell C covers those and only those vertices generable by its variable arguments while keeping its fixed arguments constant. Hence all vertices of C agree with C in all of its (C's) fixed arguments. Thus, no vertex differing from the representation of C in any fixed argument can be covered by C.

Lemma 1: If B and C are distinct cells, each basic to the covering of a particular vertex V, then there is:

1. At least one argument, say the mth, such that $C_m \neq 2$ and $B_m \neq 2$ and in addition,

2. At least one argument, say the nth, such that $B_n = 2$ and $C_n \neq 2$.

Proof of Lemma: From the sublemma above we deduce that if $B_i \neq C_i$ for some i, then $B_i \neq 2$ or $C_i \neq 2$ and not both, since both B and C contain vertex V. Consider the set of i's for which $B_i \neq C_i$. This set is not empty, since B and C are distinct cells.

Assume that for each such i, $B_i = 2$ and $C_i \neq 2$. This implies that B contains C as a subcell, contradicting our assumption that C is basic to the covering of V. This possibility is thus excluded. Similarly, the possibility is also excluded that $C_i = 2$ and $B_i \neq 2$ for each i where $B_i \neq C_i$. We conclude that there is at least one m such that $C_m = 2$ and $B_m \neq 2$ and one n such that $B_n = 2$ and $C_n \neq 2$.

Theorem:

If $B^{(1)}$, $B^{(2)}$, $B^{(3)}$, ... $B^{(r)}$ are distinct cells each basic to the covering of a vertex V, and if C is a cell basic to V which is contained in the union of the $B^{(i)}$'s, then C is identical with one of $B^{(i)}$.

Proof of Theorem: Assume C is distinct from each $B^{(j)}$. For each $B^{(j)}$, then, there is (by Lemma 1), at least one argument i for which $B_i^{(j)} = V_i \neq 2$ and $C_i = 2$. These i's taken together, for all the cells $B^{(j)}$, form a set of arguments whose typical member we shall denote by k. The values of these arguments may be varied at will to give vertices covered by cell C.

If we fix all the arguments k so that they disagree with the corresponding values of the arguments of the vertex V, and specify that the remaining arguments agree with V, we have constructed a vertex W which lies in C but not in any of the $B^{(j)}$'s. (This is clear because $W_k \neq V_k$ for every k, but for at least one k, $B_k^{(j)} = V_k$, by construction. Hence, W differs in a fixed argument from $B^{(j)}$ for each $B^{(j)}$ and is not covered by it.

Vertex W (and hence, Cell C) does not, therefore, lie in the union of the $B^{(j)}$'s.

C. Development of the Algorithm and the Partition of the Residue Into Components

Definition: A PELL (Potential Essential ceLL) of a vertex V is any cell such that:

1. All complete cells covering V are subsets of it.
2. No proper subcell of it has property (1).

The PELL of a vertex V is the smallest cell containing all cells basic to V, and in that sense can be called the cell-wise least upper bound of the unions of basic cells covering V.

Lemma 2: For each V there exists a unique PELL. Construct the set, S, of vertices on the long truth table which differ from V in exactly one argument. If these arguments are then changed to variable ones in the representation of V, a cell, L, is specified which is the smallest cell containing V and all adjacent true and don't-care vertices. No complete cell can contain V and have any variable argument not also variable in L, since that would imply the existence of an adjacent vertex which is on the long truth table and is not contained in S. Since any cell whatever must agree with V in fixed arguments if it covers V, all complete cells containing V are subsets of L.

It is clear from the construction that L is the smallest cell covering V and containing all of S. If any element, s, of S is not covered by a subcell L', of L, the complete cell composed of s and V is not covered by L'. Then no proper subcell of L can be a PELL for V. The construction of a PELL for V is unique.

Corollary: The PELL for V contains all cells basic to V.

Having proved the central theorems and lemmas, we can proceed to the development of the second part of the algorithm, the partitioning of the residue **R** into clusters.

1. Choose from **R** a vertex V.

2. Construct all basic cells containing V.

a. Construct L, the PELL of V. L cannot be complete or it would be an essential cell and V would not lie in **R**.

b. Choose a second vertex, V', in L. There is a smallest cell, G, containing V and V'. If G is complete, construct any basic cell containing V and V' and delete all vertices covered by it from L. If G is not complete, delete V' from L.

In either case, after the appropriate vertex or vertices have been deleted from L, select a new V' from the undeleted portion of L and repeat Step 2b, until L is completely exhausted.

Assertion: All the basic cells covering V have not been computed.

Proof of Assertion: The union of basic cells computed by procedure B contains no further cell basic to V (by our theorem), and the rest of L contains none, by construction. Since every cell basic to V is a subcell of L, we have computed all cells basic to V.

3. After all the basic cells covering V have been computed, determine which vertices of **R** have been covered by them. Keep all such vertices in CORRAL, return to Step 2 and repeat the subsequent processes until every vertex placed in CORRAL at any stage has been so processed. When this is done, we have computed a component of R and have a list of basic cells making it up. It is clear from the procedure that the same component would be generated no matter which vertex was chosen as initial V.

It follows from the above that we have computed all cells basic to any vertex of any component, A. Since all minimal coverings of A are composed entirely of cells which are basic to some vertex of A, every minimal covering of A is a subset of the basic cells thus computed.

D. Formal Characterization of the Partition in Terms of an Equivalence Relation

We now prove that the components form a partition of R (that is, we prove that R is covered by the union of the components and that distinct components are disjoint.) Every vertex of R is covered by at least one component, since the process of construction does not terminate until this is true. We prove the disjointness of components by giving a formal characterization as follows: A nonempty subset, A, of R is a component of R if and only if:

1. Every pair V, W, of vertices of A obeys the condition $V \cong W$ where $V \cong W$ if and only if there exists a sequence (chain) of cells $C_1, C_2, \cdots C_t$ such that

   a. C_i is complete for $1 \leqslant i \leqslant t$

   b. V is contained in C_1

      W is contained in C_t

   c. $C_i \cap C_{i+1} \cap R$ is not empty, for $1 \leqslant i < t$

2. A is not a proper subset of any subset of R that satisfies condition(1). The relation $\cong$ gives a partition of a set R in general if $\cong$, is:

   Symmetric ($V \cong W$ implies $W \cong V$)

   Reflexive ($V \cong V$)

   Transitive ($V \cong W$, $W \cong Z$ implies $V \cong Z$)

   for all V, W, and Z in R.

We need to prove that the relation $\cong$ as defined by conditions (1) (a) - (1) (c) is symmetric, reflexive and transitive.

To be proved:

(i) $\cong$ is symmetric:

$V \cong W$ implies the existence of a chain $C_1, C_2, \cdots C_t$ satisfying the conditions 1 a-c given above. If we renumber these C's in reverse order, we get a chain whose existence implies $W \cong V$.

(ii) $\cong$ is reflexive:

There is always at least one complete cell, C, covering every vertex V. This forms a chain satisfying conditions 1 (a) - (c) hence $V \cong V$.

(iii) $\cong$ is transitive:

$V \cong W$ implies the existence of a chain $C_1, C_2, \cdots, C_t$ for V, W.

$W \cong Z$ implies the existence of a chain $D_1, D_2, \cdots, D_s$ for W, Z where both chains satisfy conditions (a)-(c).

Thus C_1 contains V, C_t contains W

D_1 contains W, D_s contains Z.

If C_t and D_1 contain W, since W is an element of R we conclude $C_t \cap D_1 \cap R$ is not empty and the chain $C_1, C_2, \cdots C_t, D_1, D_2, \cdots D_s$ satisfies conditions 1 (a)-(c) for V, Z.

We conclude $V \cong Z$ and that $\cong$ is a transitive relation.

It follows immediately that any two distinct components, A and A', of R are disjoint, for if they have a vertex V in common, a contradiction results: Assume a and a' are arbitrary vertices of A and A' respectively and that V lies in both. This means $a \cong V$ and $a' \cong V$ (or $V \cong a'$, since $\cong$ is symmetric.)

From the transitivity of relation $\cong$, $a \cong V$, $V \cong a'$ implies $a \cong a'$, and hence components A and A' are identical, contradicting our assumption. We conclude that the set of components forms a partition of R.

The foregoing justifies the use of the name "component" for the subsets forming on partition of R. In topology, a component of a set R is a maximal connected subset of it. Definitions (D, 1) and (D, 2) confer these properties on the subsets we have called "components."

E. Discussion of the Uniqueness of the Maximal Additive Partition

To each component there corresponds a set of basic cells that is disjoint from each of the sets of basic cells corresponding to the other components. (If a basic cell appeared in two such sets, we could conclude that the two corresponding components intersected.) Since our norm is linear, the norm for any covering of R by a set U of basic cells equals the sum of the norms for the subsets of U covering the components. Conversely, we can consider those partitions P of R such that the norm for any union U of basic cells covering R equals the sum (over all the subsets P_i comprising P) of the norms for the subunions of U covering P_i. A subunion S of U covers P_i if each basic cell B of U is in S if and only if B covers a vertex of P_i.

We see that every such partition P must insure that every basic cell covers elements of R belonging to a single subset P_i: (If a basic cell B covered vertices in distinct subsets in any covering of R containing B, B would contribute twice to the norm of subunions, but only once to the norm of the original union, U. The sum of norms of the subunions would not equal the norm of U.) From this it follows that distinct subsets P_i and P_j must be disjoint. (For any vertex V of R, there is some basic cell that covers it. If V lies in two components, such a basic cell covers vertices in two distinct components, a contradiction. We have seen that each subset P_i of R under P contains, with any vertex V, all vertices V' of R such that there is a complete cell covering V and V'. If we specify that each subset be the minimal subset having this property, we have reproduced the partition into components as defined above.

To summarize the results of the preceding paragraphs:

1. The decomposition of R into components is a partition of R and has the additive property Q. The norm for any covering of R by a union, U, of basic cells is equal to the sum of the norms for the subcoverings of the components. (A subcovering of A is the subset of cells, C, of U such that C covers a vertex of A).

2. Conversely, any decomposition of R into minimal subsets having additive property Q is identical with the decomposition into components. In this sense, the partition into components is the maximal additive partition.

The property Q implies that any minimal covering, M, of R is composed entirely of minimal subcoverings of components. (Every covering of R decomposes into subcoverings of components and if any such subcovering is not minimal, a subcovering of lower norm could be substituted, giving by property Q, a covering of R having a norm lower than that of M. This contradicts our assumptions of the minimality of M.

Finally, since all minimal subcoverings of the components are generated by the algorithm, it follows that all minimal coverings of R are also generated. If there are q clusters and p_i minimal coverings of the ith component, there will be $p_1 p_2 \cdots p_q$ minimal coverings of R. These will be generated, however, by $p_1 + p_2 + \cdots + p_q$ coverings, a much smaller number in general.

F. Remarks on the Formal Analogy Between a Component and an Irreducible Representation of a Group

The partition of R into components A is such that each component has the important property that any covering of a vertex of A covers a portion of R contained entirely within A. Stated slightly differently, A is an invariant subspace of T under all covering operations.

This procedure bears a striking resemblance to a procedure in the theory of representation of groups by which the matrices representing the elements of the group are reduced (by linear transformations on the base vectors) to the form of squares appearing along the matrix diagonal. There, each such square corresponds to a subspace of the entire linear space which is invariant under the action of the group being represented. Our components correspond to these invariant subspaces with the basic cells covering the vertices of the component corresponding to the base vectors of the invariant subspace. The decomposition is unique in both cases. A component containing a vertex V is also the closure of V within R under the operation of covering by complete cells and is a precise analog to an irreducible Markov chain with vertices playing the role of states.[7]

G. Illustration of Algorithm by a Computation on a Sample Truth Table

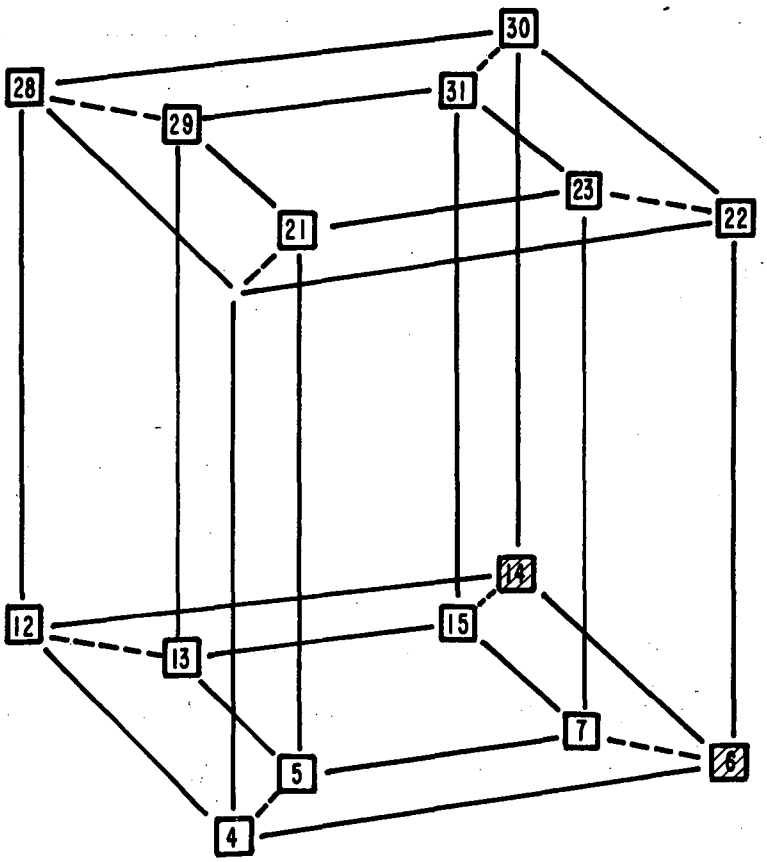
As an example, suppose, for a given truth table, that the vertices uncovered by essential cells are:

Vertices	Basic Cells Covering These Vertices	
	Cell No.	Cell
6 = 000110	1	002122
14 = 001110	2	022112
54 = 110110	3	021122
62 = 111110	4	212110
	5	210112
	6	112210
	7	110212
	8	211210

Let us form the square, symmetric matrix **M** whose rows and columns are labeled by the vertices V_i of **R** and whose elements are specified by the rule $M_{ij} = 1$ if there is a complete cell which covers V_i, and V_j, $M_{ij} = 0$ if not.

For our example, **R** = ~~(14, 6,~~ 6, 14, 54, 62).

	6	14	54	62
6	1	1		
14	1	1		
54			1	1
62			1	1

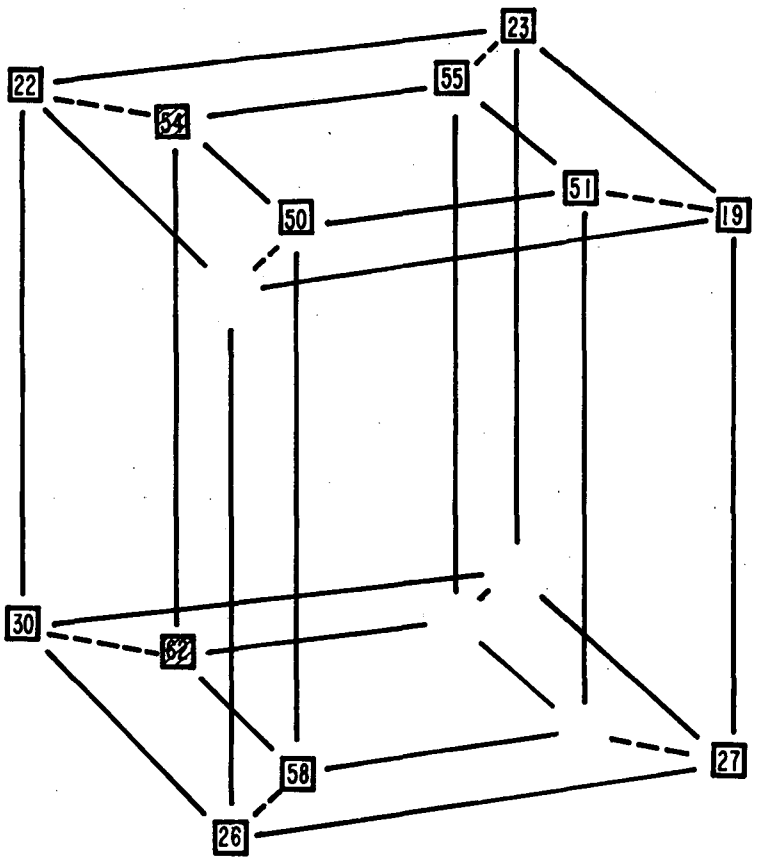


MU-20984

Fig. 1a. Diagram of Cluster 1: Cell 022122, which contains it, is shown.
Cluster 1 contains vertices (6, 14) of R and basic cells 022112 (covering vertices 6, 14), 021122 (6, 14) and 002122 (14).

Key for Fig. 1. ☐ at a vertex denotes that the corresponding vertex is on the long list, but not in R. ▨ at a vertex denotes that the corresponding vertex is in R.

No box at a vertex denotes that the corresponding vertex is not on the long list.



MU-20985

Fig. 1b. Diagram of Cluster 2: Cell 212212 which contains it is shown.
Cluster 2 contains vertices(54, 62) of R and basic cells 212110 (covering vertices 54, 62), 210112(54), 110212(54), 112210(54, 62), 211210(62).

(Note that vertex 22 lies both in Cluster 1 and in Cluster 2. This is no contradiction because 22 is a don't care vertex. If 22 lay in R, Clusters 1 and 2 would become a single cluster).

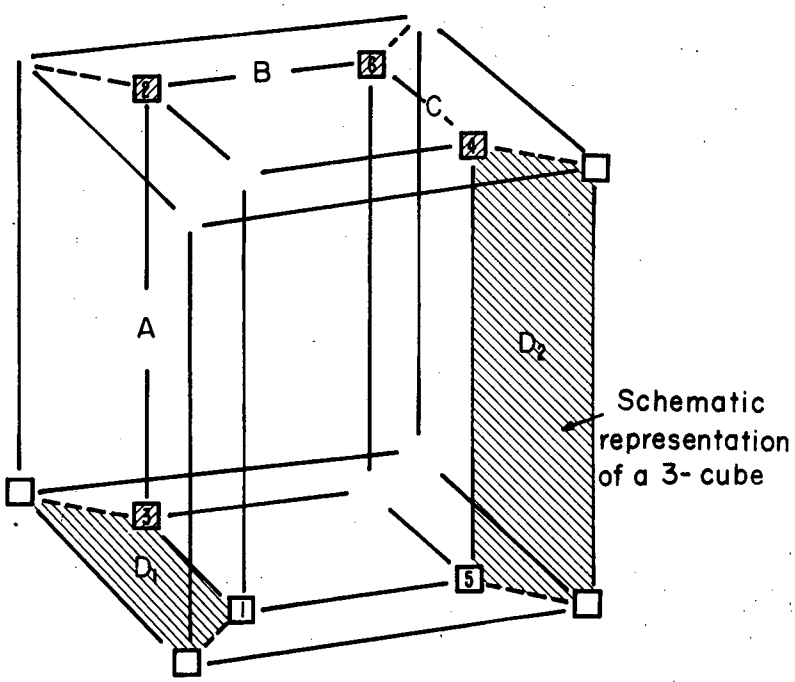


Fig. 1c. Schematic diagram of a truth table, T, having two minimal coverings each with a different number of basic cells.

Short list: (2, 3, 4, 6)
Long list: (2, 6 + cells D_1 and D_2)

Cells A = (2, 3) = 000012
B = (2, 6) = 000210
C = (4, 6) = 000120
D_1 = = 222021
D_2 = = 222102

The minimal coverings are (A, C), (D_1, D_2, B). Each has norm 12.

It is always possible to "diagonalize" M by permutations of rows and columns. The sets of vertices corresponding to each subsquare are the components of R. The procedure of diagonalization geometrically is simple. Choose a vertex V_i and place its row at the top, its column to the left. Next permute again so that all other vertices that have entries in their first row (or in their first column, since M is symmetric) have their corresponding rows (columns) brought up (left) past all the rows (columns) of vertices not having this property. After this is done perform the corresponding operation on the vertex row corresponding to row and column 2. When the last vertex row and column is reached, the process will be complete.

It is useful to note that if V_i and V_j are covered by a complete cell, then there is at least one basic cell covering them. We have seen how to generate all the basic cells covering the vertices of a component. It is important that although a cluster A may be covered by cells containing vertices not in A, such cells do not contain any vertices of R that are not in A.

In our example, we now write down the table U whose rows are labeled to correspond to the basic cells C_i and the columns, to the vertices V_j, of the cluster. $U_{ij} = 1$ if C_i contains V_j, if not, $U_{ij} = 0$.

Basic Cell No.	6	14	54	62
1	1	1	0	0
2	1	1	0	0
3	0	1	0	0
4	0	0	1	1
5	0	0	1	0
6	0	0	1	1
7	0	0	1	0
8	0	0	0	1

Vertex

Notice that U has the same diagonal structure as M.

The process of diagonalization (or component formation) can now be visualized. The steps for this specific case are reviewed (referring to the chart):

1. Choose the lowest numbered vertex V, in **R** (Vertex 6).
2. Find all basic cells covering V (cells numbered 1, 2).
3. Set aside all new vertices in **R** that are covered by these cells (vertex 14).
4. Repeat Step 2 with each new vertex found (cell 3 is now added).
5. Repeat Step 3 with each cell found in Step 4. (No new vertices of **R** are covered). If new vertices are covered, repeat Step 4, if not,
6. When all basic cells about all vertices are computed, the component is completed. Delete from **R** all vertices of the component (6, 14 are deleted, leaving 54, 62) and perform Step 1 again on the remainder of **R** (which gives us vertex 54) and continue through Steps 2 through 6 until **R** is empty. **R** has then been partitioned into components (two in this case), with the set of basic cells and the vertices they must cover both available. We now go to the relatively straightforward, third part of our algorithm.

Given a list of all the basic cells covering a component we wish to select all the subsets of this list which form minimal coverings of A. We first compute all possible coverings of A containing cell 1. This is done by adding cell 2 to the partial covering of A begun by cell 1, if and only if it covers at least one vertex that 1 leaves uncovered. If cell 2 does not have this property, cell 3 is examined, and so on. If cell 2 does have this property, cell 3 is added to the partial covering begun by cells 1 and 2, if and only if

it covers a vertex left uncovered by the union of cells 1 and 2. The process continues until all the vertices of A are covered. Suppose cells 1, 2, 6, 9 comprise the covering so constructed, we next exhaust all coverings containing cells 1, 2, 6 by trying to replace cell 9 with cell 10 or some other cells of higher number. When we have reached the highest numbered basic cell covering A, we will have exhausted all coverings containing 1, 2, 6. We then exhaust all coverings containing cells 1, 2 by the analogous procedure of seeking to continue the partial covering begun by cells 1, 2 with a cell numbered greater than 6 and proceeded as before. Finally, we exhaust all cells containing cell 1 by seeking to continue the partial covering begun by cell 1 with a cell numbered higher than 2 and proceeding as before. When all coverings containing cell 1 have been covered, we start a new covering with cell 2 until we have computed all coverings containing cell 2 but not cell 1. This procedure, too, is repeated until the end of the list of basic cells for A is reached.

At that point we have a set of coverings of the component A that contains as a subset all irredundant coverings by basic cells and hence all minimal coverings of A. By inspection of the norm for each covering we can select out the minimal covers of A. We could also select the irredundant subset of these coverings.

To illustrate this procedure of selection of coverings, we use the example given in part G. There, component 1 has 3 basic cells and 2 vertices to be covered, vertices 6. 14.

Cell 1 alone constitutes a covering of component 1, as does cell 2. Cell 3 does not appear in any minimal or irredundant covering of component 1.

Component 2 can be covered by the following sets of cells (as generated by the procedure cited):

(4), (5, 6), (5, 8), (6), (7, 8).

The minimal and irredundant covers are

	Component 1	Component 2
Minimal Covers	Cells (1), (2)	Cells (4), (6)
Irredundant Covers	Cells (1), (2)	Cells (4), (6) (5, 8), (7, 8)

There are 2 minimal coverings of component 1, and 2 minimal coverings of component 2; hence there are 4 minimal coverings of **R**.

CONCLUSION

The procedure we have examined gives, for an arbitrary Boolean truth function, the class of sums of Boolean products, each member of which is minimal under what we have called a "linear norm". One application of this solution would lead, by a procedure devised by Prather,[2] to a solution of the N-input, M-output problem. Our solution is of the N-input, 1-output problem. Our scheme is desirable because: (1) It is well adapted for mechanization by digital computers--this has been accomplished in a program entitled SALOMÉ--(see Appendix). (2) It treats the problem in terms of fundamental topological units, simplifying the description of results. (3) It provides a framework of structural sets on which to build further investigations, in contrast to the unstructured set that is the starting point of the problem. (4) It gives the entire class of minimal (as well as the entire class of "Irredundant") coverings by basic cells. This is of use in any fundamental research into the structure of Boolean functions. (5) In common with many earlier schemes, it employs only notions that are invariant under choice of description. If we transform the description of our logic, for example, by interchanging 0's and 1's everywhere, the solution to our problem in the transformed coordinates is generated by transforming our original solutions similarly.

ACKNOWLEDGMENTS

It is a pleasure to thank Professor Bruce H. McCormick, Department of Physics and Digital Computer Laboratory, University of Illinois, Urbana, for the contributions he has made to the research embodied in this paper. Apart from his original conjecture, acknowledged previously, that a simplifying partition into clusters was possible for an arbitrary truth table, his acute criticism and constant encouragement were of invaluable assistance.

This work was done under the auspices of the U. S. Atomic Energy Commission.

APPENDIX

DESCRIPTION OF SALOMÉ

IBM 704 program SALOMÉ takes a truth table T of vertices and a table of don't-care vertices as input. Its output is a set of coverings of T by basic cells with the subset of minimal coverings noted from among these. Unless it is altered (as can easily be done), the norm assumed is $M + \sum_{i=1}^{M} (N - k_i)$, where M = the number of cells in the covering and k_i is the dimension of the ith cell.

Specifications

Running time. About 12 seconds for a typical table for N = 6. For higher N, running time should go up more slowly than 2^N, and would depend on the structure and density of the truth table.

Storage Needed. For N = 17, a 32-K memory is needed. This would not be adequate if any component had more than 1024 cells in it.

Input. Vertices are specified by punching one vertex on a card starting in column 12 (column 13 for BCD). The decimal, octal, or BCD representation of the vertex should be preceded in columns 8-11 by the appropriate three-letter code (DEC, OCT, BCD). In BCD, the symbols T, F are used for 1, 0 and entire cells may be entered, using the BCD mode, by the use of the symbol U (unspecified) for the notation "2". The number of arguments, N, in the truth table is loaded into the decrement of octal location 4000 by a single card preceding the transfer card.

Output. The entire true and don't-card lists are printed out on tape 9, with representation in BCD as well as in decimal (if octal is desired in place of decimal, sense switch 6 should be down).

All the basic cells and their norms are listed by component, as well as the class of coverings of each component with norms for each.

Sense Switches and Options

Sense Switch 1.

Must be down until all data cards have been loaded, then it is superfluous.

Sense Switch 2.

Until data cards have been loaded:

UP: Input in OCT or DEC form

DOWN: Input in BCD form

After data cards have been loaded:

UP: Does not fill in flag positions

DOWN: Counts the number of traverses of program part a set of flags and stores this number in region FLAG in storage.

Sense Switch 3.

UP: (Always)

Sense Switch 4.

UP: Gives a list of coverings containing only a few per cent of redundant coverings. (Second slowest option)

DOWN: Redundant coverings will comprise half or more of output unless Sense Switch 5 is up.

Sense Switch 5.

UP: Only irredundant coverings are listed as output (slowest option) All irredundant coverings by basic cells are given.

DOWN: Redundant coverings comprise half or more of output unless Sense Switch 4 is up.

Sense Switch 6.

UP: Numerical representations of vertices printed in decimal.

DOWN: Numerical representations of vertices printed in octal.

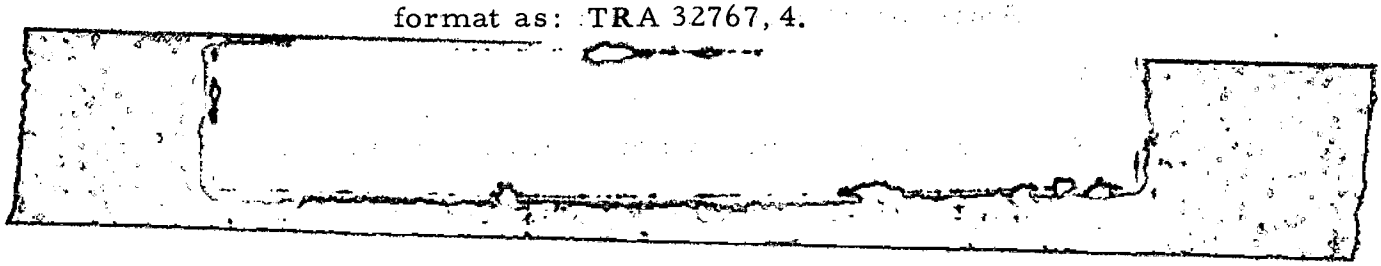
Option.

All the vertices in the input cards will have been listed when the program stops just after all the data cards have been read. This allows use of the program not only to reduce a truth table, but also to list vertices in cells already computed without going through the entire reduction. The cells in question are loaded in the BCD format specified above and the program is run only until the program stop after all the cards have been loaded.

Order of deck.

1. Deck of instructions (without transfer card).
2. N card: In absolute loading format, a card containing the value of N (number of arguments in input truth table) in the decrement of first word (8L), and specified to be loaded into octal location 4000.
3. Transfer card.
4. True vertices, listed one per card. (Starting in column 12 if listed in DEC or OCT; starting in column 13 with a blank in column 12 if listed in BCD). In columns 8-11 the appropriate 3 letter code. (DEC, OCT, BCD), DEC and OCT cards may be mixed if desired and the order of the vertices is irrevelant. If repetitions occur, they will be ignored.

5. Don't-card vertices (if any) preceded by a card punched in SAP format as: TRA 32767, 4.



Note - Every covering given by SALOMÉ is composed entirely of basic cells.

REFERENCES

. W. V. Quine, The Problem of Simplifying Truth Functions, Amer. Math. Monthly 59, 521-531 (1952);

R. K. Mueller, On the Synthesis of a Minimal Representation of a Logic Function, AFCRC-TR-55-104, April 1955.

R. H. Urbano and R. K. Mueller, A Topological Method for the Determination of the Minimal Forms of a Boolean Function, Proc. IRE, EC-5, 126-132 (1956).

R. Prather, The Simplification of Logical Truth Functions, (M. S. Thesis), University of California, Berkeley, Department of Electrical Engineering Sept., 1958.

B. Harris, An Algorithm for Determining Minimal Representations of a Logic Function, Proc. IRE, EC-6, 103-108, (1957).

R. Prather, Command Structure Proposed Computer, (unpublished).

The initial suggestion that a simplifying partition into clusters of an arbitrary truth table might be constructed was made by Dr. Bruce H. McCormick, Department of Physics, University of Illinois, Urbana, (private communication).

6. See especially Refs. 1, 2. Reference 2 contains the procedure for construction of essential cells which most closely resembles the one we have used.

7. William Feller, An Introduction to Probability Theory and Its Applications, Second Edition (John Wiley, New York, 1957), Ch. XV, especially Part 4.