BOOLEAN ALGEBRA AND THE MINIMIZATION PROBLEM

by

EMMETT M. LARSON

B. S., Bethany College, 1964

A MASTER'S REPORT

submitted in partial fulfillment of the

requirements for the degree

MASTER OF SCIENCE

Department of Mathematics

KANSAS STATE UNIVERSITY
Manhattan, Kansas

1966

Approved by:

S. Thomas Parker
Major Professor

60r LD
66 2668
?34 R4
2 1966
L334

TABLE OF CONTENTS

INTRODUCTION

Boolean algebra is a mathematical system. As with other systems in mathematics, such as the algebra of real numbers and Euclidian geometry, Boolean algebra consists of a non-empty set of elements upon which are defined abstract laws or postulates and theorems derived from these laws.

Since the 1930's, Boolean algebra has developed from what was originally regarded as merely an interesting curiosity into an extensive and mature branch of mathematics. Its postulation and development by George Boole, the eminent English mathematician, took place shortly after the realization by mathematicians that an algebra is an abstract system. It was Boole's task to separate the symbols of mathematical investigation from the things upon which they operate and he proceeded to investigate these operations in their abstract setting. Indeed, Boole produced many notable works of a mathematical and logical nature, but his prime effort was the influential treatise entitled The Laws of Thought.

Much of the contemporary interest and development in Boolean algebra was inspired by its application to the design of switching circuits for telephone and control systems, and to the design of logical circuits for electronic computers. The subject has developed also, into a significant branch of abstract algebra with important applications to topology (6,ix)[1].

In any Boolean algebra, the members of the universe class, 1, are all of the elements under consideration. In the first section of this report we will examine the generalized Boolean algebra, that is, the algebra in which there is no restriction on the number of elements. If we impose the restriction

[1]In this report the first number of the ordered pair will be used to indicate the reference and the second number will indicate the page, with the references numbered in the bibliography.

that there are exactly k elements, then the resulting 2^k classes form a Boolean algebra of order 2^k, i.e., the order of an algebra means the number of distinct classes in it. For each value of k there is a different Boolean algebra where two algebras are said to be "different" if and only if some law valid in one of them is not valid in the other.

The Boolean algebra of order 2^1, called binary Boolean algebra and often written as Boolean algebra (0,1), will be the main topic of this paper. It is the algebra of greatest significance in recent years and its applications range from the propositional calculus of symbolic logic to the logical design of switching circuits.

A basic problem that arises in connection with applications of Boolean algebra to switching circuits is that of the simplification of a given circuit which is known to have the desired closure properties. The last part of this paper will be devoted to several approaches to the reduction of Boolean expressions, i.e., the "minimization problem."

There are two types of representations for logical switching functions. One may work with the disjunctive form (also known as the sum-of-products form, normal form, or alternational form) or the conjunctive form (product-of-sums form). One is the dual of the other. Ghazala (4, 171) has shown that a simple transformation will convert one to the other. Any algorithm which manipulates the disjunctive form will, with the help of the Ghazala transformation, manipulate the conjunctive form.

It was therefore decided to concentrate on one of the two possible types of representations. Arbitrarily we choose the disjunctive representation.

DEFINITION OF A BOOLEAN ALGEBRA

A generalized Boolean algebra will be developed in this section in order to indicate that the mathematical structure is independent of its applications and for ease of later reference. The definition of a Boolean algebra that will be used is the one given by Huntington in 1904. Although many other sets of postulates could be chosen that would define the system equally well, this set has the property that no one postulate can be derived from the remaining postulates (9, 27).

Consider a set B of elements for which the familiar concept of equality is introduced using the common notation $a = b$ and meaning that a and b are two names for identical objects. There are no restrictions, at this point, placed on the nature of the objects, so that one may have equality not only between numbers, but between sets, or between functions, or indeed between the names of any objects.

An algebra involves operations, and it is assumed that in B there are two binary operations, that is, operations that may be applied to any ordered pair of elements of B to yield a unique third element of B. A Boolean algebra can then be defined in the following manner (9, 28).

Definition. A class of elements B together with two binary operations $(+)$ and $(\cdot)$ (where $a \cdot b$ will be written ab) is a Boolean algebra if the following postulates hold:

P_1. The operations $(+)$ and $(\cdot)$ are commutative.

P_2. There exist in B distinct identity elements 0 and 1 relative to the operations $(+)$ and $(\cdot)$, respectively.

P_3. Each operation is distributive over the other.

P_4. For every a in B there exists an element a' in B such that

$a + a' = 1$ and $a \cdot a' = 0$.

There is no reason why the two operations in the definition must be written (+) and (·), other usual notations being o,*; $\cup,\cap$; $\vee,\wedge$. In fact, the symbol + represents the logical sum (disjunction, inclusive union, inclusive OR) and the symbol · represents the logical product (conjunction, intersection, AND).

We notice immediately the similarity with the algebra of sets; indeed, the representation theorem of Boolean algebras shows that there exists an isomorphism between every Boolean algebra and an algebra of sets for some choice of universal set. This apparently has led Paul R. Halmos in his Lectures on Boolean Algebra to dismiss the elementary algebraic relations which follow as "set-theoretic trivialities." The following theorems are, however, important and necessary for that which is to follow.

We have below a statement of the principle of duality.

Theorem 1. Let ϕ be any formula expressed in the vocabulary of a Boolean algebra, and let ϕ' result from ϕ by interchanging occurences of the operation symbols '+' and '·' and identity elements '0' and '1'. Then ϕ is derivable from the postulates of the system if and only if ϕ' is derivable from them.

Proof: The proof of this theorem follows at once from the symmetry of the postulates with respect to the two operations and the two identities.

The general theorems about Boolean algebras, and, for that matter, their proofs also, come in dual pairs. A practical consequence of this principle that will be exploited to some degree in what follows, is that in the theory of Boolean algebras it is sufficient to state and to prove only half the theorem; the other half comes gratis from the principle of duality. Pairs of the theorems will be stated, however, though only one of each pair will be proven, for the steps in one proof are the dual of those in the other, and the justification for each step is the dual in one case of that in the other. The following two theorems are known as the idempotent laws.

Theorem 2a. For every element a in a Boolean algebra B, $a + a = a$.

Theorem 2b. For every element a in B, $aa = a$.

Proof: One has

$$a = a + 0 = a + (aa') = (a + a)(a + a') =$$
$$(a + a)(1) = a + a.$$

Theorem 3a. For every a in B, $a + 1 = 1$.

Theorem 3b. For every a in B, $a0 = 0$.

Proof: One has

$$a + 1 = (a + 1)(1) = (a + 1)(a + a') =$$
$$a + (1 \cdot a') = a + a' = 1.$$

The following two theorems are known as the absorption laws.

Theorem 4a. For every a and b in B, $a + ab = a$.

Theorem 4b. For every a and b in B, $a(a + b) = a$.

Proof: One has

$$a = 1 \cdot a = (1 + b)a = 1a + ba = a + ba = a + ab.$$

Theorem 5a. The binary operation $(+)$ is associative on B.

Theorem 5b. The binary operation $(\cdot)$ is associative on B.

Proof: It is necessary to show that $a + (b + c) = (a + b) + c$ for any a, b, c in B. Let $T = a + (b + c)$ and $S = (a + b) + c$. Then $aT = a(a + (b + c)) = a$ by theorem 4b. Similarly

$$aS = a((a + b) + c) = (a(a + b)) + (ac) =$$
$$a + (ac) = a.$$

Thus $aT = aS$. Furthermore,

$$a'T = a'(a + (b + c)) = (a'a) + (a'(b + c)) =$$
$$0 + (a'(b + c)) = a'(b + c) \text{ and}$$
$$a'S = a'((a + b) + c) = (a'(a + b)) + (a'c) =$$
$$((a'a) + (a'b)) + (a'c) =$$
$$(a'b) + (a'c) = a'(b + c).$$

Thus $a'T = a'S$. Then

$$aT + a'T = aS + a'S \text{ or}$$
$$aa' + T = aa' + S$$

or $T = S$. Thus $a + (b + c) = (a + b) + c$.

Theorem 6. The element a' corresponding to a in B is unique.

Proof: Assume there are two such elements, a_1' and a_2', satisfying P_4. Then

$$a_1' = 1a_1' = (a + a_2')a_1' = aa_1' + a_2'a_1' =$$
$$0 + a_2'a_1' = a_2'a_1' = a_1'a_2' = a_1'a_2' + 0 =$$
$$a_1'a_2' + aa_2' = (a_1' + a)a_2' = 1a_2' = a_2'.$$

The following is the law of involution:

Theorem 7. For every a in B, $(a')' = a$.

Proof: We have

$$a' + a = 1 \text{ and } a'a = 0, \text{ hence by theorem 6,}$$
$$(a')' = a.$$

Theorem 8a. In any Boolean algebra $0' = 1$.

Theorem 8b. In any Boolean algebra $1' = 0$.

Proof: By theorem 3, $0 + 1 = 1$ and $0 \cdot 1 = 0$, and since theorem 6 shows that for each a there is only one element a', these equations imply that

$$0' = 1.$$

The following two theorems are known as DeMorgan's laws.

Theorem 9a. For every a and b in B, $(a + b)' = a'b'$.

Theorem 9b. For every a and b in B, $(ab)' = a' + b'$.

Proof: For arbitrary elements a and b in B we have

$$(a + b) + (a'b') = ((a + b) + a')((a + b) + b') =$$
$$((a + a') + b)(a + (b + b')) =$$
$$(1 + b)(a + 1) = 1 \cdot 1 = 1.$$

Further,

$$(a + b)(a'b') = (a(a'b')) + (b(a'b')) =$$
$$((aa')b') + (a'(bb')) =$$
$$(0b') + (a0) = 0 + 0 = 0.$$

Then by theorem 6, $(a + b)' = a'b'$.

Finally we define a binary relation "$\leqq$", which we read "equal to or less than" for simplicity's sake and which applies to ordered pairs of elements of B.

Definition. The "order" relation is defined by the statement: For every a and b in a Boolean algebra B, $a \leqq b$ if $ab' = 0$.

This differs from the relation "$\leqq$" of the algebra of real numbers in that, given any two elements a and b of B, we may have $a \leqq b$ or $b \leqq a$ or neither of these. That is, some pairs of elements may not be comparable. The following are useful consequences of the above definition:

Theorem 10a. If $x \leqq y$ and $y \leqq z$, then $x \leqq z$.

Proof: Since $x \leqq y$, we have by the above definition, that $xy' = 0$. In like manner since $y \leqq z$, we have that $yz' = 0$. Hence

$$xz' = xz'(y + y') = xyz' + xy'z' = 0 + 0 = 0.$$

But $xz' = 0$ is equivalent to $x \leqq z$, which was to be shown.

Theorem 10b. If $x \leqq y$ and $x \leqq z$, then $x \leqq yz$.

Proof: From $x \leqq y$ and $x \leqq z$ we have $xy' = 0$ and $xz' = 0$. Hence

$$xy' + xz' = 0 \text{ and } x(y' + z') = 0.$$

But by theorem 9, $y' + z' = (yz)'$ and thus $x(yz)' = 0$ or, by the above definition, $x \leqq yz$.

Theorem 10c. If $x \leqq y$, then $x \leqq y + z$ for any z.

Proof: From $x \leqq y$ we have $xy' = 0$ and hence

$$x(y + z)' = x(y'z') = (xy')z' = 0.$$

But $x(y + z)' = 0$ is, by the above definition, equivalent to $x = y + z$.

Theorem 10d. $x \leq y$ if and only if $y' \leq x'$.

Proof: Assume first that $x \leq y$ and thus $xy' = 0$. Then

$$0 = xy' = (x')'y' = y'(x')',$$

and from this we have that $y' \leq x'$. Conversely, if $y' \leq x'$, then, applying the preceding statement, $(x')' \leq (y')'$ and by theorem 7 this gives $x \leq y$.

With the preceding formal structure of a Boolean algebra now at hand we begin our inquiry into the nature of Boolean functions.

BOOLEAN FUNCTIONS

Consider the independent variables $a,b,\ldots,x_2,x_1,\ldots,x_n$, which may take on values 0 or 1. We then have the following definitions:

Definition. A "literal" is a complemented or uncomplemented variable.

Definition. A "term" is defined to be either a single literal, or an indicated logical product of two or more literals.

Definition. A "polynomial" is an indicated sum of terms. Often we shall use the term "disjoint-type" to refer to a polynomial.

Definition. By a "Boolean function" we will mean any expression which represents the combination of a finite set of symbols each representing a constant or a variable, by the operations of $(+)$, $(\cdot)$, or $(')$; thus $(a' + b)'c + b'x + 0$ is a Boolean function provided that each of the symbols a,b,c,x represents an element of a Boolean algebra.

Since we are restricting the values of the independent variables to 0 or 1 and thus the function f_i also to the values 0 or 1, we shall be dealing with the binary Boolean algebra (0, 1), i.e., switching functions.

MINTERMS

Of particular interest among switching functions of n variables are the Boolean products containing all n of the variables as factors, either complemented or not (but not both complemented and uncomplemented in any one case, of course). When n = 1, we consider x_1 and x_1' to be these "products." When n = 2, they are $x_1'x_2'$, $x_1'x_2$, x_1x_2' and x_1x_2.

<u>Definition</u>. A "minterm" (fundamental product, minimal polynomial) of n variables is a Boolean product of these n variables, with each variable present in either its own or its complemented form.

The characteristic property of a minterm is that it takes on the value 1 for exactly one set of values of x_1 to x_n, namely that combination which makes each factor of the product equal to 1. This is a direct result of theorem 3b.

If we define $x_j^0 = x_j'$; and $x_j^1 = x_j$ then a minterm may be represented in the form $x_1^{e1}x_2^{e2}x_3^{e3}\ldots x_n^{en}$, where each e_i is either 0 or 1. It is convenient to interpret the sequence of superscripts $e_1, e_2, e_3, \ldots e_n$ as integers in binary notation. Then we use the corresponding decimal integers to number the minterms as follows:

$$m_i = x_1^{e1}x_2^{e2}\ldots x_n^{en},$$

where $(i)_{decimal} = (e_1e_2\ldots e_n)_{binary}$.

Minterms for n = 3

i	x_1	x_2	x_3	Non-Vanishing Product, m_i
0	0	0	0	$x_1'x_2'x_3'$
1	0	0	1	$x_1'x_2'x_3$
2	0	1	0	$x_1'x_2x_3'$
3	0	1	1	$x_1'x_2x_3$
4	1	0	0	$x_1x_2'x_3'$
5	1	0	1	$x_1x_2'x_3$
6	1	1	0	$x_1x_2x_3'$
7	1	1	1	$x_1x_2x_3$

Table 1.

Let S be any set of n variables and let all minterms mentioned below be minterms of S.

Theorem 11. There are exactly 2^n minterms.

Proof: The proof follows from the fundamental principle of permutations since in any minterm the first variable of S is either primed or unprimed, the second is either primed or unprimed, etc., for all n variables.

DISJUNCTIVE NORMAL FORM

Among the functions of n variables $x_1,x_2,\dots,x_n$, which can be written, a particular class of functions is of special interest, namely, those written as a sum of terms in which each term is a product involving all n variables either with or without a prime, i.e., as a sum of minterms. The following definition gives a name to such functions.

Definition. A Boolean function is said to be in "disjunctive normal form" in n variables $x_1, x_2, \ldots, x_n$, for $n > 0$, if the function is the sum of minterms, i.e., terms of the type $(f_1(x_1)) \cdot (f_2(x_2)) \cdot \ldots \cdot (f_n(x_n))$, where $f_i(x_i)$ is x_i or x_i for each $i = 1, 2, \ldots, n$, and no two terms are identical. In addition, 0 and 1 are said to be in disjunctive normal form in n variables for $n = 0$.

It is a relatively simple matter to express a given switching function as a sum of minterms. For example suppose $n = 3$ and

$$f = (x_1 x_2)\ (x_1 + x_3)$$

First we express f as a union of products, not necessarily fundamental, i.e., minterms, by application of DeMorgan's law, the law of involution, the distribution property of $(\cdot)$ over $(+)$, the absorption laws, and the idempotent laws:

$$f = (x_1 + x_2')(x_1 + x_3) = x_1 + x_2' x_3.$$

Then by P_4 we have

$$f = x_1(x_2 + x_2')(x_3 + x_3') + (x_1 + x_1')x_2' x_3.$$

Now expanding and removing duplicate terms by the idempotent laws we obtain

$$f = x_1 x_2' x_3' + x_1 x_2' x_3 + x_1 x_2 x_3' + x_1 x_2 x_3 + x_1' x_2' x_3$$

which contains only distinct fundamental products.

The method just illustrated is perfectly general and indicates the validity of the following theorem, often called the basic theorem:

Theorem 12. Every function in a Boolean algebra which contains no constants is equal to a function in disjunctive normal form.

Proof: Let an arbitrary function (without constants) of the n variables $x_1,x_2,\ldots,x_n$, be denoted by f. If f contains an expression of the form $(a + b)'$ or $(ab)'$ for some variables a and b, DeMorgan's laws may be applied to yield $a'b'$ and $a' + b'$, respectively. This process may be continued until each prime which appears applies only to a single variable x_i. Next, by applying the distributive law of $(\cdot)$ over $(+)$, f can be reduced to a polynomial.

Now suppose some term does not contain either x_i or x_i'. This term may be multiplied by $(x_i + x_i')$ without changing the function. Continuing this process for each missing variable in each of the terms in f will give an equivalent function whose terms contain x_j or x_j' for each $j = 1,2,\ldots,n$. Finally, the idempotent laws allow the elimination of duplicate terms, and with this the proof is complete.

In practice a given switching function is often defined by means of a truth table as shown in the case of $n = 3$ in Table 2. Each of the values of f is given a name f_i, where the subscript is the decimal number corresponding to the binary number opposite it.

The Boolean Function f

x_1	x_2	x_3	f
0	0	0	$0 = f_0$
0	0	1	$1 = f_1$
0	1	0	$0 = f_2$
0	1	1	$0 = f_3$
1	0	0	$0 = f_4$
1	0	1	$1 = f_5$
1	1	0	$0 = f_6$
1	1	1	$1 = f_7$

Table 2.

Now suppose we want to write a Boolean expression for this new function f. We see that f must be 1 only when x_1, x_2, and x_3 have the values 001, 1C1, or 111. But when x_1, x_2, and x_3 have the values 001 for example, then $x_1'x_2'x_3 = 1$. Similarly, when they have the values 101, then $x_1x_2'x_3 = 1$; and when all three are 1, then $x_1x_2x_3 = 1$. Since these are the only circumstances under which f should be 1, we may express f as the Boolean sum of these three minterms; that is,

$$f = x_1'x_2'x_3 + x_1x_2'x_3 + x_1x_2x_3 = m_1 + m_5 + m_7.$$

This expression is correct because it has the value 1 for the correct combination of x_1, x_2, and x_3, and for no other combinations. This procedure may be translated into Boolean language very nicely by noting that the Boolean sum of Boolean products of f_i and m_i will eliminate the undesired minterms and retain the desired ones; i.e.,

$$\begin{aligned} f &= 0\cdot m_0 + 1\cdot m_1 + 0\cdot m_2 + 0\cdot m_3 + 0\cdot m_4 + 1\cdot m_5 + 0\cdot m_6 + 1\cdot m_7 \\ &= f_0m_0 + f_1m_1 + f_2m_2 + f_3m_3 + f_4m_4 + f_5m_5 + f_6m_6 + f_7m_7 \\ &= \sum_{i=0}^{2^n-1} f_im_i \end{aligned}$$

which is the statement of f in disjunctive normal form.

It was stated that a Boolean function is usually derived from some sort of truth table by means of the basic theorem. In this form the function may often be simplified. The problem of simplifying a Boolean function as formulated by Quine (2, 497) may be stated as follows: Given a (completely defined) Boolean function f, find the simplest disjunctive (and/or conjunctive) forms which are equivalent to f. As economic design of circuits in an important factor in synthesizing a switching system, this problem has received much attention in recent years and a number of solutions are now available (2, 497).

The fact that there are alternative ways of representing functions opens up the possibility that some form may be the simplest of all, and that there may be some method for arriving at this minimal expression. However, any such method must hinge on the exact definition of the minimum form.

Industry, with Bartee's help (1, 21), has accepted three criteria to determine the respective minimality between equivalent disjunctive representations. First, the minimal expression is that expression which contains the least occurrences of literals; second, the minimal expression is the expression containing the least number of product terms; and third, the minimal expression is the expression which requires the least number of diodes in an AND-to-OR circuit configuration which the function represents.

Let us then become quickly familiar with the rules which must be followed in order to count the number of diodes needed to implement a given Boolean switching function. Consider, for example, the following three functions: $f_1 = ab$, $f_2 = a + b$, and $f_3 = ab + c + d$. The circuits for f_1 and f_2 require two diodes each, one for each literal in the expression. The function f_3 requires five diodes: two in an "and" circuit to form ab, and three more in an "or" circuit whose three inputs are c, d, and the previously formed ab. In order to determine the diode count of a given switching function it is thus necessary to count the number of literals in the given Boolean polynomial and add to this the number of terms which are products of two or more literals present in the polynomial (7, 62).

Since each disjoint-type expression is made up of a logical sum of terms, it is natural that we should take a moment to examine the terms of n variables a little more carefully. In order to find the magnitude of the class of all possible terms of n variables we begin by listing all terms containing n letters. These are of course the minterms, and we already know there are 2^n of them. Then we list all possible combinations of (n - 1) letters, (n - 2) letters, (n - 3) letters, etc., until we finally display the terms containing only one letter---and there must be 2n of them. There are evidently a finite number of possible terms for each n, and any polynomial expression can only contain terms from the list corresponding to the particular n involved. This leads to the following theorem.

Theorem 13. There are $3^n - 1$ possible terms for $n = 1$.

Proof: Let us associate each term with an n-digit number, written to the base three as follows:

If a literal appears in complemented form, write 0;

literal appears uncomplemented, write 1;

literal does not appear, write 2.

The proof then follows from the fundamental principle of permutations since there are 3^n different numbers (base three) having n digits, and since all of these except the number $3^n - 1 = 222\ldots2$ correspond to a term, we see there must be $(3^n - 1)$ different terms of n variables.

The following definition is of importance to the discussion of the minimization problem which is to follow.

Definition. A "prime implicant" is a logical product which is a term of some minimal form of a Boolean function. Each prime implicant will contain a minimum number of literals.

The simplification methods which will now be presented all have the objective of selecting from the $(3^n - 1)$ possible terms one group which defines the function and contains the least number of literals, the least number of product terms, or which employs as few diodes as possible in the circuit representation.

ALGEBRAIC METHOD OF SIMPLIFICATION

The first method to be described requires that the logical designer (engineer, mathematician or logician) employ judgment, experience, and ingenuity to simplify an expression by applying the appropriate postulates and theorems as developed in the first part of the paper. Because of the frequent appearance of the following polynomial in disjunctive-type expressions we state as theorems:

Theorem 14a. $a + a'b = a + b$.

Theorem 14b. $a(a' + b) = ab$.

Proof: By means of the distributive property P_3 of (+) over (·) we have,

$$a + a'b = (a + a')(a + b);$$

and by P_4 and P_2,

$$(a + a')(a + b) = a + b.$$

The application of these Boolean theorems and postulates to a given function may not be obvious. It is usually necessary to rearrange and even modify the original function before any of the rules can be used. For example, it may be necessary to complicate the original function by adding xx' to it, or by multiplying it by $(x + x')$, if an x can be chosen which can subsequently be eliminated. The general principles are illustrated in the following examples:

Example 1. Simplify $f = ab' + c + a'c'd + bc'd$. Applying theorem 14a to the last three terms, we find

$$f = ab' + c + a'd + bd.$$

Rearranging and applying P_3,

$$f = ab' + c + d(a' + b) =$$
$$ab' + c + d(ab')'.$$

And applying theorem 14a again,

$$f = ab' + c + d.$$

Example 2. Simplify $f = abc + abd' + ac' + a'b'c'd' + a'c$. Factoring and applying theorem 14a,

$$f = a(bc + c') + a'(c + b'c'd') + abd' =$$
$$a(b + c') + a'(c + b'd') + abd'.$$

The last term may now be eliminated by combining it with the first and using theorem 4a, giving

$$f = ab + ac' + a'c + a'b'd'.$$

In the next example no obvious simplification exists.

Example 3. Simplify $f = ab' + bc' + b'c + a'b$. If the last two terms are multiplied by $(a + a')$ and $(c + c')$ respectively, the resulting terms may be rearranged and combined as follows:

$$\begin{aligned} f &= ab' + bc' + b'c(a + a') + a'b(c + c') \\ &= ab' + ab'c + bc' + a'bc' + a'b'c + a'bc \\ &= ab'(1 + c) + bc'(1 + a') + a'c(b' + b) \\ &= ab' + bc' + a'c. \end{aligned}$$

Note first that in all three examples the final expression is simpler than the original. If the diode criteria of minimality is applied, in example 1 the number of diodes necessary was reduced from twelve to five; in example 2 from nineteen to thirteen; and in example 3 from twelve to nine. The reduction is also obvious using the literal or term count criterias. However, though these reductions are satisfying it is not obvious that the final expression in each of the three examples is actually the simplest. This indicates in part, some of the restrictions of this approach to simplification.

The algebraic method is indeed, useful only for the simplest of functions. Though it is easy to apply, it has two difficulties: there is in general no way of determining whether or when one has obtained the simplest expression for a function; and if several alternative minimal forms exist, it will be difficult to find them all.

The following method has three very important advantages: the function to be simplified may be attacked directly, without being expanded into minterm form; the process of finding prime implicants is simplified by making them correspond to patterns familiar to the eye; and the ease with which essential terms may be identified makes it possible to reduce the labor involved in seeking prime implicants.

THE VEITCH-KARNAUGH MAP OF A BOOLEAN FUNCTION

Of primary importance in the study of Boolean functions of n variables are the 2^n combinations of n 0's and 1's. A special form of Venn diagram was suggested by Veitch (1952) for use in the simplification of Boolean functions and is shown in Fig. 1 for n = 2, n = 3, and n = 4 for the Boolean variables a, b, c, d.

Veitch Diagrams:

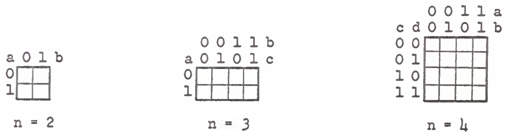


Fig. 1.

The product designated by a particular square is obtained by noting the values of the variables in the column and row that intersects the square.

A reorganization of the Veitch maps was proposed by Karnaugh in 1953. In the Veitch-Karnaugh scheme, the four combinations of values of two binary variables x_1 and x_2 are represented as a linear array of four squares (Fig. 2). It is convenient for later purposes to list the combinations in the order 00, 01, 11, 10, which is known as the Gray code. Since the squares corresponding to the combinations are used for recording information, the combinations are customarily written above the squares.

Map for Two Variables

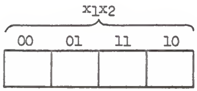


Fig. 2.

The labeling of the coordinates of these squares may be simplified as shown in Fig. 3. Here the two columns embraced by the symbol "x_1" are those in which the variable x_1 has the value 1. The columns not embraced by the same symbol are those in which x_1 is 0. Again this is the case for x_2 and also for x_3, and x_4 in Fig. 4. Thus, in the square marked x in Fig. 3 $x_1 = 0$ and $x_2 = 1$ and the combination corresponding to this square is 01.

Simplified Map for n = 2

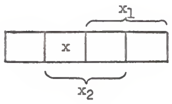
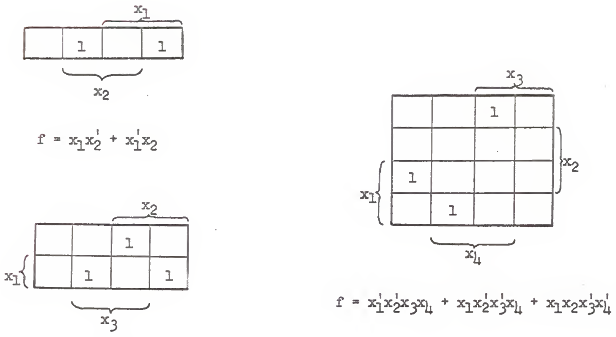


Fig. 3.

Now to each combination of values of the variables x_1, x_2, x_3, x_4, when n = 4 there corresponds a unique minterm $x_1^{e1}x_2^{e2}x_3^{e3}x_4^{e4}$ which takes on the value 1 for that combination and no others so that the squares may be put into 1-to-1 correspondence with the minterms. This correspondence enables us to construct a map of a given switching function: we simply record a 1 in each square corresponding to a combination for which the function is 1, i.e., in each square corresponding to a minterm which appears in the disjunctive normal form of the function. The same procedure applies, of course, in the case of two, or of three variables. Fig. 4 gives maps of a variety of functions.

Maps Corresponding to Several Functions:



$f = x_1x_2' + x_1'x_2$

$f = x_1'x_2'x_3x_4 + x_1x_2'x_3'x_4 + x_1x_2x_3'x_4'$

$f = x_1'x_2x_3 + x_1x_2'x_3 + x_1x_2x_3'$

Fig. 4.

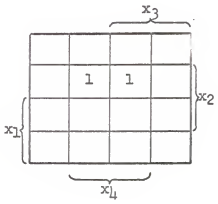
Definition. The squares in which 1's are recorded are called the "p-squares" of the function, where p means product.

Definition. If two 1's are in squares whose combinations differ by only one digit, the squares are called "adjacent."

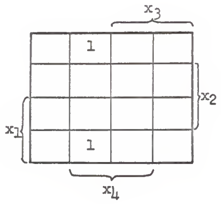
The combination of variables may be interpreted as vertices of a 4-dimensional cube and, in such cases, combinations corresponding to adjacent squares determine adjacent vertices of the cube. One should consider the left and right edges of the map as identical and also consider the top and bottom edges of the map as identical. That is, a p-square in the left hand column of the map is adjacent to the corresponding p-square in the same row of the right hand column of the map, and similarly for the top and bottom rows.

The function corresponding to two adjacent 1's is obtained by writing down the product of all those variables whose values are fixed in the two squares in question, uncomplemented if this fixed value is 1, complemented if it is 0.

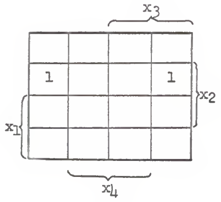
Maps Corresponding to Three-Factor Products:



$f = x_1'x_2x_4$



$f = x_2'x_3'x_4$



$f = x_1'x_2x_4'$

Fig. 5.

In the first example we wrote x_1' because x_1 is 0 in both squares, x_2 because x_2 is 1 in both, x_4 because x_4 is in both. x_3 was omitted because x_3 is 0 in one square, 1 in the other. The legitimacy of this procedure follows from the disjunctive normal expansion of the function represented by the two squares:

$$f = x_1'x_2x_3'x_4 + x_1'x_2'x_3x_4 = x_1'x_2x_4(x_3' + x_3) = x_1'x_2x_4.$$

In the second example:

$$f = x_1'x_2'x_3'x_4 + x_1x_2'x_3'x_4 = (x_1 + x_1')x_2'x_3'x_4 = x_2'x_3'x_4,$$

and in the third example:

$$f = x_1'x_2x_3'x_4' + x_1'x_2x_3x_4' = x_1'x_2x_4'(x_3 + x_3') = x_1'x_2x_4'.$$

Definition. N adjacent p-squares are said to constitute an "n-dimensional p-subcube" where $2^n = N$, since they correspond to the endpoints of an edge of an N-cube.

Thus in the above examples we have 1-dimensional p-subcubes.

In Fig. 6 the principles used to write down the functions corresponding to the maps are the same as those used in the preceding examples and may be verified in the same manner.

Maps Corresponding to Two-Factor Products:

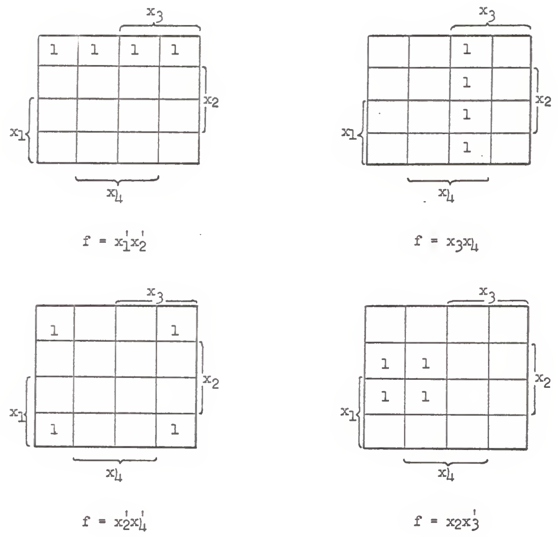
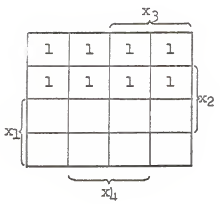


Fig. 6.

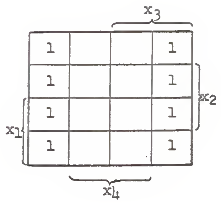
In each of the above examples, the four p-squares correspond to four vertices of a square face of the 4-cube and hence are called 2-dimensional p-subcubes.

The maps of Fig. 7 illustrate some 3-dimensional p-subcubes. Again they may be verified as in the first set of examples.

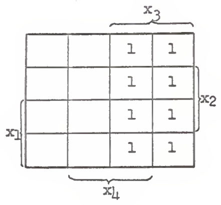
Maps Corresponding to Single Variables:



$f = x_1'$



$f = x_4'$



$f = x_3$

Fig. 7.

In review, the use of the map method requires that the function to be simplified be represented first in the form of a polynomial, i.e., disjunctive type form. For each term in the function, a 1 is placed in the square corresponding to that particular product of literals. Then the map is inspected for the purpose of recognizing which of several possible groupings of terms represents the best factoring of terms in the function. The usefulness of the map derives from the fact that patterns of p-squares containing 1's which will yield the simplest terms can, after sufficient practice, be easily and quickly determined by inspection. Larger p-subcubes correspond to products of fewer variables, since fewer variables are fixed for them. This suggests the following rules for obtaining a minimal representation of a function in disjunctive normal form:

1. Select a set of p-subcubes
   - (a) which includes every p-square at least once,
   - (b) is such that the p-subcubes are as large as possible, and
   - (c) are as few in number as possible.
2. Write the Boolean sum corresponding to these subcubes.

As an illustration we choose the function

$$f = x_1'x_2'x_3'x_4' + x_1'x_2x_3'x_4' + x_1'x_2x_3'x_4 + x_1'x_2x_3x_4 + x_1'x_2x_3x_4' + x_1x_2x_3x_4$$

corresponding to the map of Fig. 8. By inspection and recognizing the patterns of the previous examples we see that we can express f as the Boolean sum of the functions corresponding to one 2-dimensional p-subcube and to two 1-dimensional p-subcubes:

$$f = x_1'x_2 + x_1'x_3'x_4' + x_2x_3x_4.$$

Map Corresponding to the Boolean Function f:

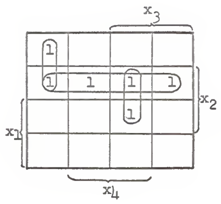


Fig. 8.

The accomplishment of 1. (b) and 1. (c) was trivial in the example given above, but this is not always the case. M. Karnaugh provided the following example:

$$f = x_1'x_2'x_3'x_4 + x_1'x_2x_3'x_4' + x_1'x_2x_3'x_4 + x_1x_2x_3'x_4' + x_1x_2x_3'x_4 +$$
$$x_1x_2x_3x_4 + x_1x_2'x_3x_4 + x_1'x_2'x_3x_4' + x_1'x_2x_3x_4' + x_1x_2x_3x_4'$$

with the following map:

Map Corresponding to Karnaugh Example:

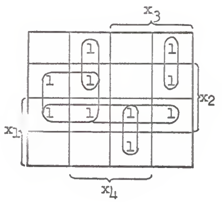


Fig. 9.

Here it is clear that we shall want to use two 2-dimensional p-subcubes corresponding, for example, to the terms x_2x_3' and x_1x_2 respectively. The problem then is to account for the 1's in the remaining p-squares. To do this, since only two of these squares are adjacent, will require three more terms. To keep these terms as simple as possible we employ two previously used adjacent p-squares which enable us to use 1-dimensional p-subcubes. These finally yield

$$f = x_1x_2 + x_2x_3' + x_1'x_3'x_4 + x_1'x_3x_4' + x_1x_3x_4.$$

A second and third possible form of f is shown in Fig. 10.

Maps Corresponding to Karnaugh Example:

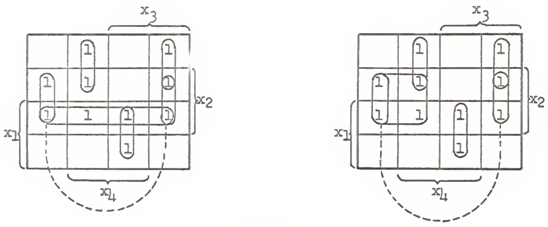


Fig. 10.

We now have in the first case of Fig. 10:

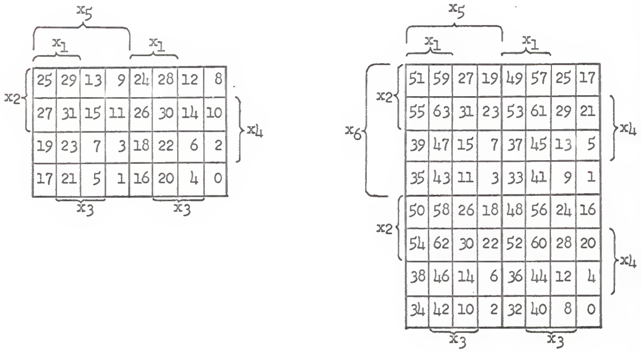
$$f = x_1x_2 + x_2x_4' + x_1'x_3'x_4 + x_1'x_3x_4' + x_1x_3x_4,$$

and in the second case of Fig. 10:

$$f = x_2x_3' + x_2x_4' + x_1'x_3'x_4 + x_1'x_3x_4' + x_1x_3x_4.$$

Any number of variables can be plotted on a Veitch-Karnaugh diagram, though the diagrams are difficult to use for more than eight variables. Representations for n = 5 and n = 6 are given in Fig. 11 with the inscribed integral representations for the corresponding minterms.

Veitch-Karnaugh Diagrams for n = 5 and n = 6:



25	29	13	9	24	28	12	8
27	31	15	11	26	30	14	10
19	23	7	3	18	22	6	2
17	21	5	1	16	20	4	0

51	59	27	19	49	57	25	17
55	63	31	23	53	61	29	21
39	47	15	7	37	45	13	5
35	43	11	3	33	41	9	1
50	58	26	18	48	56	24	16
54	62	30	22	52	60	28	20
38	46	14	6	36	44	12	4
34	42	10	2	32	40	8	0

Fig. 11.

THE QUINE SIMPLIFICATION METHOD

The Quine approach to the simplification problem is straightforward though somewhat tedious. It is valuable in the sense that the algorithm can be readily programmed for use on digital computing devices. The system leads automatically to the set of all possible minimal solutions (7, 93).

The Quine procedure will be described by specifying a set of rules, as follows:

a. Express the function in disjunctive normal form. Of course, the function may already be in this form, but if it contains terms which are not minterms they must be expanded.

b. The next step is to derive a set of "prime implicants" from the minterms. First compare each minterm with every other minterm. Whenever two minterms differ by only one variable (x in one, x' in the other), list the term formed by omitting that variable from the minterm, and put an asterisk opposite each minterm. This is the equivalent of employing P_4. If the minterms each comprise n variables, this first group of terms will each contain (n - 1) variables. Now the same procedure is followed with them; i.e., each term is compared with all the other terms of (n - 1) variables, and whenever two terms differ by only one variable, the term formed by omitting that variable is written down and each of the parent terms is marked with an asterisk. There are now three groups of terms: minterms, terms containing (n - 1) variables, and terms containing (n - 2) variables. Groups containing (n - 3), (n - 4), (n - 5),...variables are now formed in the same way from the groups containing (n - 2), (n - 3), (n - 4),...variables respectively, until finally a group of terms is formed, none of which can be combined with any other. Step b is now complete, and all the terms having no asterisks beside them are identified as prime implicants. (Note that a minterm may itself be a prime implicant if it does not combine with any other minterm.) We have at this point minimized the literal count per term.

c. Next, prepare a table (Table A) having as many rows as there are prime implicants, and as many columns as there are minterms in step a.

Identify each row with a prime implicant, and each column with a minterm. Place an asterisk in a square on the table wherever the prime implicant to the left of the square is derived, in part, from the minterm above the square. This will be true wherever the letters in the prime implicant all appear in the same form as they do in the minterm.

d. Now examine the columns in Table A. If any column in the table contains only one asterisk, the corresponding prime implicant is called an "essential term" and must be included in the final expression for f. Encircle the essential terms, together with all asterisks on the same row with essential terms. A new table, Table B, may now be drawn up having the same form as Table A, but containing only the prime implicants (rows) that have not been encircled and only the minterms (columns) which contain no encircled asterisks. The minterms omitted from Table B need not be considered further because they will be included in the final expression for f by virtue of the fact that the essential terms appear in that expression.

e. Wherever in Table B there are columns m_i and m_j such that m_i has asterisks on every row that m_j has asterisks, eliminate column m_i.

f. Form Table C from Table B by omitting the columns eliminated by step e and any rows which remain but contain no asterisks.

g. Examine Table C, and choose from it the simplest set (or sets) of prime implicants which, taken together, include at least one asterisk in each column. The Boolean sum of these prime implicants, together with the essential terms of step d, forms the simplest expression for f. Note also that this last step may be a very difficult or a very trivial one, depending on the nature of and number of entries in Table C.

Thus we arrive at the minimal term count.

Let us examine the following as an illustration of the above technique: Simplify

$$f = ab' + bc' + b'c + a'b.$$

a. Expanding f to obtain a sum of minterms:

$$\begin{aligned} f &= ab'(c + c') + bc'(a + a') + b'c(a + a') + a'b(c + c') \\ &= ab'c + ab'c' + abc' + a'bc' + ab'c + a'b'c + a'bc + a'bc' \\ &= ab'c + ab'c' + abc' + a'bc' + a'b'c + a'bc. \end{aligned}$$

b. The set of prime implicants is found as follows:

$ab'c$	*
$ab'c'$	*
abc'	*
$a'bc'$	*
$a'b'c$	*
$a'bc$	*
ab'	(combining $ab'c$ with $ab'c'$)
$b'c$	(combining $ab'c$ with $a'b'c$)
ac'	(combining abc' with $ab'c'$)
bc'	(combining abc with $a'bc'$)
$a'b$	(combining $a'bc'$ with abc)
$a'c$	(combining $a'b'c$ with $a'bc$)

None of the two-letter terms combine with one another, so there are no more prime implicants. The function f may be written as the Boolean sum of these prime implicants, but we continue with the rules, hoping to find a set of prime implicants which will form a simpler expression.

c. Table A may now be prepared, as follows:

	ab'c	ab'c'	abc'	a'bc'	a'b'c	a'bc
ab'	*	*				
b'c	*				*	
ac'		*	*			
bc'			*	*		
a'b				*		*
a'c					*	*

Table A.

d., e., and f. No column contains only one asterisk. There are therefore no essential terms, and Table B is the same as Table A. It is also evident that no two columns have the same configuration of asterisks and there does not exist any rows containing no asterisks. Thus Table C is the same as Table A.
g. Examining the table, we see there are two ways of choosing prime implicants so as to include the smallest number of prime implicants and an asterisk in every column. These two ways indicate the following minimal Boolean functions:

$$f = ab' + bc' + a'c$$

and

$$f = b'c + ac' + a'b.$$

The first of these expressions for f is the same as the result obtained in example 3. The second, which by the way, employs the same number of diodes in a switching circuit, could also be derived that way. However, not every function has alternative simplest forms, and the Quine method provides an easy way for finding any and all of them where the algebraic approach (in particular) gives no indication as to whether or not there may be more than one.

TABLE OF REFERENCES

1. Bartee, T. C.
"Computer Design of Multiple Output Logical Networks." IRE Trans. on Electronic Computers, Vol. EC-10, No. 1 (March, 1961).

2. Chu, J. T.
"Some Methods for Simplifying Switching Circuits Using 'Don't Care' Conditions." Journal of the Association for Comp. Machinary, Vol. 8, No. 4 (October, 1961).

3. Culbertson, James T.
Mathematics for Logic and Digital Devices. New Jersey: D. VanNostrand Co., Inc., 1958.

4. Ghazala, M. J.
"Irredundant Disjunctive and Conjunctive Forms of a Boolean Function." IBM Journal, (April, 1957).

5. Halmos, Paul R.
Lectures on Boolean Algebras. New Jersey: D. VanNostrand Co., Inc., 1963.

6. Hohn, Franz E.
Applied Boolean Algebra: An Elementary Introduction. New York: The MacMillan Co., 1960.

7. Phister, M. Jr.
Logical Design of Digital Computers. New York: Wiley, 1958.

8. Quine, W. V.
"The Problem of Simplifying Truth Functions." American Math Monthly, Vol. 66, (1959).

9. Whitesitt, J. Eldon.
Boolean Algebra and Its Applications. Reading, Massachusetts: Addison-Wesley Publishing Co., Inc., 1961.

ACKNOWLEDGEMENT

This writer wishes to express his appreciation to Dr. S. Thomas Parker for his valuable counsel and encouragement throughout the preparation of this report.

BOOLEAN ALGEBRA AND THE MINIMIZATION PROBLEM

by

EMMETT M. LARSON

B. S., Bethany College, 1964

AN ABSTRACT OF A MASTER'S REPORT

submitted in partial fulfillment of the

requirements for the degree

MASTER OF SCIENCE

Department of Mathematics

KANSAS STATE UNIVERSITY
Manhattan, Kansas

1966

An introductory treatment of a generalized Boolean algebra is presented in this report. The algebra is stated in terms of a set of postulates with the logical sum and logical product defined as binary operations and complementation as an unary operation. Three approaches to the minimization of completely specified Boolean functions are then indicated in the final section of this report.

A generalized Boolean algebra is defined in the first section by stating postulates concerning the operations of a Boolean algebra. Subsequently, theorems are developed which relate to these operations of a Boolean algebra and which concern particular elements of the Boolean algebra. The structure is completed by introducing an ordering relation.

In the next section the generalized Boolean algebra is restricted and the concept of a Boolean function is introduced whose range of values is 0 and 1. This is followed by a description of the representation of completely specified Boolean functions and of the necessity of arriving at a simplest form for Boolean expressions. There can be many of these irreducible expressions, so a minimality measure is assigned to each representation; these are literal count, term count, and diode count.

Finally, the concept of prime implicants is introduced and three methods of simplification and the corresponding advantages are discussed. The methods of minimization include: the direct application of algebraic operations, the Veitch-Karnaugh map, and the Quine algorithm.