221

and the unary operation, defined by $0 = 0$ and $I = 1$. Then $B$ is a Boolean algebra.

<table>
<thead>
<tr>
<th>$0$</th>
<th>$0$</th>
<th>$0$</th>
<th>$0$</th>
<th>$0$</th>
<th>$0$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$0$</td>
<td>$1$</td>
<td>$0$</td>
<td>$1$</td>
<td>$1$</td>
<td>$1$</td>
</tr>
<tr>
<td>$1$</td>
<td>$1$</td>
<td>$*$</td>
<td>$0$</td>
<td>$1$</td>
<td>$+$</td>
</tr>
</tbody>
</table>

Let $B$ be the set with two elements $\{0,1\}$ with binary operations $+$ and $\cdot$ defined by

\begin{align*}
(q + a) & = q + a \quad (q \cdot a) = q \cdot a \\
(q \cdot a) + (q + a) & = (q + a) \cdot (q \cdot a) \\
(q + a) \cdot (q + a) & = (q \cdot a) + (q \cdot a) \\
(q + a) \cdot (q + a) & = (q + a) + (q + a)
\end{align*}

For example, \((q + a) + (q \cdot a) = q + a + q \cdot a\) and not \((q \cdot a) + (q + a) = q + a + q \cdot a\) means \(q + a \cdot q = q + a\) and not \((q \cdot a) + (q + a) = q + a + q \cdot a\)

We adopt the usual convention that unless we are advised by parentheses, the precedence of operations is $\cdot$ and $+$, and we call these product and sum, respectively.

The following axioms hold where $a, b, c, d, e, f$ are any elements in $B$.

**Complement Laws:**

\begin{align*}
0 & = 0 + a \\
1 & = a + 0
\end{align*}

**Identity Laws:**

\begin{align*}
0 & = 0 + a \\
1 & = a + 0
\end{align*}

**Distributive Laws:**

\begin{align*}
(a + b) \cdot (a + c) & = (a \cdot a) + (a \cdot c) \\
(a + b) \cdot (a + c) & = (a \cdot a) + (a \cdot c)
\end{align*}

**Commutative Laws:**

\begin{align*}
(a + b) & = (b + a) \\
(a \cdot b) & = (b \cdot a)
\end{align*}

These Basic Definitions have similar properties to the usual Boolean laws.

**Boolean Algebra**

Chapter 12
Theorem 17: Let 0, 1 be any elements in a Boolean algebra B.

Using the axioms of Boolean algebra, we prove (Problem 17a) the following theorem.

**I.3 BASIC THEOREMS**

By using the dual of each step of the proof of the original statement, then the dual is also a consequence of those axioms since the dual statement can be proven in other words, if any statement is a consequence of the axioms of a Boolean algebra, then its dual is also a theorem.

**Theorem 17 (Principle of Duality):** The dual of any theorem in a Boolean algebra is also a theorem.

Principle of duality holds in B. Namely, observe the symmetry in the axioms of a Boolean algebra B. That is, the dual of the set of axioms of B is the same as the original set of axioms. According to the important principle of duality, the important operation * is dual to the operation + and identity element 0 is dual to identity element 1. Hence, the dual of any statement in a Boolean algebra is the statement obtained by interchanging * and +.

**I.2 DUALITY**

For any elements a, b in B,

\[(a \lor b) = (a \land b)\]

and

\[(a \lor b) = (a \land b)\]


Two Boolean algebras B and B' are said to be isomorphic if there is a one-to-one correspondence f : B -> B' which preserves the three operations, i.e., such that

\[f(a \lor b) = f(a) \lor f(b)\]


For example, (1.2.3.4) in Example 11c) is a subalgebra of (1, 2, 3, 4) is a subalgebra of R. And only if C is closed under the three operations of B, i.e., +, * and 0, then C is a subalgebra of B. We note that C is a subalgebra of B if and only if C is a Boolean algebra. Hence, if C is a Boolean algebra we say C is a subalgebra of a Boolean algebra. We say C is a subalgebra of B.

Then, if C is a Boolean algebra with 0 the zero element and 1 the unity element,

\[a \lor 0 = a, \quad a \land 1 = a, \quad a \lor 0 = a, \quad \text{and} \quad a \land 1 = a\]

For example, let C be a collection of sets that is closed under union, intersection, and complement. Then C is a Boolean algebra with the empty set 0 as the zero element and the universal set C as the unity element.
Theorem 12.1

Theorem 12.2

Example 12.2
normal form for Boolean algebras which we discuss below.

Each of the eight sets of the form \(V \cup B \cup C\), where \(A\) is a \(V\) or \(B\) or \(C\), is or \(C\) is \(V\) or \(C\), is or \(C\) is \(V\) or \(C\).

For example, \(g\) is \(V\), \(A\) is \(V\), \(B\) is \(C\) and \(C\) is \(B\). Any nontrivial set expression involving \(A\), \(B\), and \(C\), for example, \(g\) or \(B\) or \(C\).

Consider the Venn diagram in Fig. 12-3 of the sets \(A\), \(B\), and \(C\). Observe that these three sets \(A\), \(B\), and \(C\) partition the universal set into the eight numbered sets which can be represented as follows:

Figure 12-2 shows the diagram of the Boolean algebra of the power set \(P(A)\) of \(A\), observe that the two diagrams are structurally the same.

Figure 12-2 shows the diagram of the Boolean algebra of the power set \(P(A)\) of \(A\), observe that the two diagrams are structurally the same.

Example 12.2. Consider the Boolean algebra \(B(\alpha)\) for \(\alpha = 0, 2, 5, \ldots, 79\). (See Example 12.1.)

Corollary 12.7: A finite Boolean algebra has \(2^n\) elements for some positive integer \(n\).

If a set \(A\) has \(n\) elements, then its power set \(P(A)\) has \(2^n\) elements. Thus the above theorem is just our next result.

Theorem 12.6: The above mapping \(f: B(P(A)) \rightarrow B(\alpha)\) is an isomorphism.

The mapping is well defined since the representation is unique.

\[
\{a_1, a_2, \ldots, a_n\} \mapsto (x) / f
\]

where \(f: B(P(A)) \rightarrow B(\alpha)\) is defined by

\[
a_1 \cdot \cdots \cdot a_n = x
\]

Consider the function \(f: B(P(A)) \rightarrow B(\alpha)\) defined by

\[
a_1 \cdot a_2 \cdots a_n = x
\]

elements of \(A\).

Say, for \(x = 0\) in \(B\) can be expressed uniquely (except for order) as the sum (join) of atoms, i.e., \(P(A)\) be the Boolean algebra of all subsets of the set \(A\) of atoms. By Theorem 10.11, each

\[\text{CHAP. 12 BOOLEAN ALGEBRA}\]

\[\text{224}\]
A Boolean expression \( G(x_1, x_2, \ldots, x_n) \) is said to be in full disjunctive normal form if it is in disjunctive normal form. Any Boolean expression in disjunctive normal form can easily be changed into this form.

\[ G = (q + \bar{q} + \bar{r} + r) \]

which is in disjunctive normal form by the absorption law.

\[ \text{If } q + \bar{q} = 1 \text{ and by the absorption law } \]

\[ 0 + \bar{a} + a + \bar{b} + b = \bar{a} + a + \bar{b} + b + \bar{c} + c = G \]

For example, by (1)

\[ (q + \bar{q})(\bar{a} + a) = \]

\[ ((\bar{a} + a)(\bar{b} + b))(\bar{c} + c) = \]

\[ ((\bar{a} + a)(\bar{b} + b))(\bar{c} + c) = G \]

For example, by (1)

\[ G \]

Using the distributive law, we can further transform \( G \) into a sum of products and then using the commutative, associative, and complementation laws we can finally transform into a sum of products of literals.

Any parenthesis and brackets in an algebraic expression may be removed provided that these do not alter the operation of the expression. In this respect we can construct an algorithm to transform any Boolean expression into its disjunctive normal form.

In another example, consider the expression \( z, x + z, b + x, x \), but the second is not in disjunctive normal form since \( z \) is contained in \( x, z, b \), but the second is in disjunctive normal form since \( x, z \) is contained in \( x, z, b \).

A Boolean expression \( F \) is said to be in disjunctive normal form (and if \( F \) is a fundamental product of two or more fundamental products of which none is included in another product, \( F \) is said to be fundamental product in an algebraic expression, \( \bar{z}, x, b, x, z, x \), and \( z + x = 1 \).

\[ F = \bar{z} + \bar{z}, x, b + x, x \]

Then by the absorption law \( F = \bar{z} + \bar{z}, x, b + x, x \) if \( z \), \( b \), \( x \), \( x, z \) are not fundamental products but \( z, x, b, x, z, x \).

\[ \text{If } z, x + (\bar{z} + z, x) = F \]

\[ (\bar{z}, x + z, x, b, x) + (z, x + z, x) = G \]

For example, \( a, +, b \) and \( a \) and \( \cdot \) are Boolean operations in \( x, a, b \).

Consider a set of variables (or letters or symbols) say \( x, z, z' \). By a Boolean

\[ \text{II. DISJUNCTIVE NORMAL FORM} \]

BOOLEAN ALGEBRA

CHAP. 12
With respect to a switching circuit, we will let

\[ a \lor (c \land d) \land (e \lor f) \]

be described by (1), and circuit (2) can be described by (2). Circuit (1) of Fig. 1-9 can be described by (1), and circuit (2) can be described by (2).

**Example 12.4.**

Circuit (1) of Fig. 1-9 can be described by (1), and circuit (2) can be described by (2). Hence, it can be seen that a Boolean switching circuit design means an arrangement of wires and switches that can be described by the use of connectives \( \land \) and \( \lor \). A Boolean switching circuit may be constructed by repeated use of series and parallel combinations; hence, it can be denoted recursively.

**Example 12.5.**

Let \( a \land b \land c \) and \( a \lor b \lor c \) be connected in series and \( a \land b \) and \( a \lor b \) be connected in parallel.

**Example 12.6.**

Let \( a \lor b \) and denote electrical switches, and let \( a \land b \) denote electrical switches. The property that if one is on, then the other is off, and vice versa, is shown in Fig. 1-9.

**Theorem 12.8.**

Every nonzero Boolean expression \( B(x_1, x_2, \ldots, x_n) \) can be put into full disjunctive normal form and such a representation is unique.

We note that \( x^+ = 1 \), so multiplying by \( x \) is permissible. The following theorems:

\[
\begin{align*}
ac^+ + ad^c + acd + adc + acd + acd + acd &= \overline{d} \\
ac^+ + ad^c + acd + adc + acd + acd + acd &= \overline{d}
\end{align*}
\]

above apply to full disjunctive normal form by fundamental product of \( B \) by \( x^+ \) if \( B \) does not involve \( x \). For example, we transform
Suppose $p$ and $q$ are fundamental propositions such that exactly one variable says "on."

<table>
<thead>
<tr>
<th>Step</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Thus current will flow only if both $A$ and $B$ are on.

<table>
<thead>
<tr>
<th>$A$</th>
<th>$B$</th>
<th>$A \land B$</th>
<th>$A \lor B$</th>
<th>$A \lor (C \land A)$</th>
<th>$A \land B \lor C$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

The behavior of circuit (2) in Fig. 12.5 is indicated by the following truth table for $(A \land B) \lor C$.

**Theorem 12.2:** The algebra of Boolean switching circuits is a Boolean algebra.

In order to find the behavior of a Boolean switching circuit, a table is constructed which is analogous to the truth tables for propositions. The only difference is that 1 and 0 are used instead of $T$ and $F$. Notice that the above three tables are identical with the tables of conjunction, disjunction, and negation for statements (and propositions).

The next table shows the relationship between a switch $A$ and a switch $B$.

<table>
<thead>
<tr>
<th>$A$</th>
<th>$B$</th>
<th>$A \land B$</th>
<th>$A \lor B$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Thus two tables describe the behavior of a series circuit $A \land B$ and a parallel circuit $A \lor B$.
For example, if \( \overline{a+b+c+d+e+f+g+h} = \mathcal{E} \) is an inclusive OR expression of the literals of \( \mathcal{E} \), then we will denote the number of summands in \( \mathcal{E} \). Consider a Boolean expression \( \mathcal{E} \) in its disjunctive normal form. We will denote minimal forms of \( \mathcal{E} \) by \( \mathcal{E} \).

There are many ways of representing the same Boolean expression \( \mathcal{E} \). Since \( \mathcal{E} \) may be a switching circuit, we may want a representation which is in some sense minimal. Other types of \( \mathcal{E} \) may be represented by the same Boolean expression \( \mathcal{E} \). Since \( \mathcal{E} \)

### 12.10 Minimal Boolean Expressions

Since neither step in the consensus method will change \( \mathcal{E} \), the prime implicants of \( \mathcal{E} \) are already in the sum of the prime implicants.

**Theorem 12.11:** The consensus method applied to any Boolean expression \( \mathcal{E} \) will eventually

**Lemma 12.10:** If \( \mathcal{E} \) is the consensus of \( \mathcal{P} \) and \( \mathcal{Q} \), then \( \mathcal{F} \) is the consensus of \( \mathcal{P} \) and \( \mathcal{Q} \), where the \( \mathcal{P} \)'s are fundamental products. Applying the following two steps to \( \mathcal{E} \) will be called the consensus method:

For finding the prime implicants of \( \mathcal{F} \), we next discuss the consensus method and \( \mathcal{F} + \mathcal{G} \).

### 12.12 Fundamental Products

A fundamental product \( \mathcal{F} \) is called a prime implicant of a Boolean expression \( \mathcal{E} \) if its appearance in one of \( \mathcal{P} \) and \( \mathcal{Q} \) implies that \( \mathcal{P} \) and \( \mathcal{Q} \) were written as the sums of \( \mathcal{F} \) and \( \mathcal{G} \), and the literal \( \mathcal{F} \) appears complemented in one of \( \mathcal{P} \) and \( \mathcal{Q} \) and uncomplemented in the other. Then the-con
which correspond in the obvious way to the four squares in the Kurzweig map in Fig. 12-6(a).

The variables $x$ and $y$ are four fundamental products, $x'y'$, $x'y$, $xy'$, and $xy$. These will sometimes use the terms squares and fundamental products interchangeably.

Note that $x'y'$ and $x'y$ are not adjacent. Also note that $z$ and $x'y'$ are not adjacent.

$$
\begin{align*}
\alpha b, x &= (1)b, x = (x + z)b, x = x, x' + x'z, x \\
\alpha x + z &= (1), xz = (x + z), xz = x, z, x + z, x
\end{align*}
$$

Further, in the same Kurzweig map, fundamental products in the same variable will be represented by squares. We will only treat the case of two squares or four variables.

In our Kurzweig maps, fundamental products in the same variable will be represented by squares. We will only treat the case of two squares or four variables.

In our Kurzweig maps, fundamental products in the same variable will be represented by squares. We will only treat the case of two squares or four variables.

Theorem 12.2: A minimal disjunctive normal form of a Boolean expression is a sum of prime implicates of the expression.

The following theorem shows the basic relationship between minimal disjunctive normal forms of $f$ and prime implicates of $f$: if the latter is simpler than the former, then $f$ is a sum of prime implicates of $f$.
In full detail, the Karnaugh map is represented on the cylinder map by choosing the appropriate squares on the cylinder would then have a side in common. As before, any Boolean expression involving the identified edge, then we would obtain the cylinder pictured in Fig. 12-7(a). Adjacent squares on the cylinder would then have a side in common. In other words, if we cut our band and glue the map along the identified edge and the right edge of the map, since and and are adjacent products and are adjacent products and are adjacent products and the right edge of the map is identical to the left edge and the right edge of the map. Since, we must identify the squares which correspond in the obvious way to the cylinder squares in the Karnaugh map in Fig. 12-7(a).

There are eight fundamental products, namely $x, z, xz, z', xz, z, x', z'$.

There are eight fundamental products, namely $x, z, xz, z', xz, z, x', z'$.

Here there are eight fundamental products, namely $x, z, xz, z', xz, z, x', z'$.

$x, z, xz, z', xz, z, x', z'$.

Here there are eight fundamental products, namely $x, z, xz, z', xz, z, x', z'$.

Here there are eight fundamental products, namely $x, z, xz, z', xz, z, x', z'$.

Here there are eight fundamental products, namely $x, z, xz, z', xz, z, x', z'$.

Here there are eight fundamental products, namely $x, z, xz, z', xz, z, x', z'$.

Here there are eight fundamental products, namely $x, z, xz, z', xz, z, x', z'$.

Here there are eight fundamental products, namely $x, z, xz, z', xz, z, x', z'$.

Here there are eight fundamental products, namely $x, z, xz, z', xz, z, x', z'$.

Here there are eight fundamental products, namely $x, z, xz, z', xz, z, x', z'$.

Here there are eight fundamental products, namely $x, z, xz, z', xz, z, x', z'$.

Here there are eight fundamental products, namely $x, z, xz, z', xz, z, x', z'$.

Here there are eight fundamental products, namely $x, z, xz, z', xz, z, x', z'$.

Here there are eight fundamental products, namely $x, z, xz, z', xz, z, x', z'$.

Here there are eight fundamental products, namely $x, z, xz, z', xz, z, x', z'$.

Here there are eight fundamental products, namely $x, z, xz, z', xz, z, x', z'$.

Here there are eight fundamental products, namely $x, z, xz, z', xz, z, x', z'$.

Here there are eight fundamental products, namely $x, z, xz, z', xz, z, x', z'$.

Here there are eight fundamental products, namely $x, z, xz, z', xz, z, x', z'$.

Here there are eight fundamental products, namely $x, z, xz, z', xz, z, x', z'$.

Here there are eight fundamental products, namely $x, z, xz, z', xz, z, x', z'$.

Here there are eight fundamental products, namely $x, z, xz, z', xz, z, x', z'$.

Here there are eight fundamental products, namely $x, z, xz, z', xz, z, x', z'$.

Here there are eight fundamental products, namely $x, z, xz, z', xz, z, x', z'$.

Here there are eight fundamental products, namely $x, z, xz, z', xz, z, x', z'$.

Here there are eight fundamental products, namely $x, z, xz, z', xz, z, x', z'$.

Here there are eight fundamental products, namely $x, z, xz, z', xz, z, x', z'$.

Here there are eight fundamental products, namely $x, z, xz, z', xz, z, x', z'$.

Here there are eight fundamental products, namely $x, z, xz, z', xz, z, x', z'$.

Here there are eight fundamental products, namely $x, z, xz, z', xz, z, x', z'$.

Here there are eight fundamental products, namely $x, z, xz, z', xz, z, x', z'$.

Here there are eight fundamental products, namely $x, z, xz, z', xz, z, x', z'$.

Here there are eight fundamental products, namely $x, z, xz, z', xz, z, x', z'$.

Here there are eight fundamental products, namely $x, z, xz, z', xz, z, x', z'$.

Here there are eight fundamental products, namely $x, z, xz, z', xz, z, x', z'$.

Here there are eight fundamental products, namely $x, z, xz, z', xz, z, x', z'$.

Here there are eight fundamental products, namely $x, z, xz, z', xz, z, x', z'$.

Here there are eight fundamental products, namely $x, z, xz, z', xz, z, x', z'$.

Here there are eight fundamental products, namely $x, z, xz, z', xz, z, x', z'$.

Here there are eight fundamental products, namely $x, z, xz, z', xz, z, x', z'$.

Here there are eight fundamental products, namely $x, z, xz, z', xz, z, x', z'$.

Here there are eight fundamental products, namely $x, z, xz, z', xz, z, x', z'$.

Here there are eight fundamental products, namely $x, z, xz, z', xz, z, x', z'$.

Here there are eight fundamental products, namely $x, z, xz, z', xz, z, x', z'$.

Here there are eight fundamental products, namely $x, z, xz, z', xz, z, x', z'$.

Here there are eight fundamental products, namely $x, z, xz, z', xz, z, x', z'$.

Here there are eight fundamental products, namely $x, z, xz, z', xz, z, x', z'$.

Here there are eight fundamental products, namely $x, z, xz, z', xz, z, x', z'$.

Here there are eight fundamental products, namely $x, z, xz, z', xz, z, x', z'$.

Here there are eight fundamental products, namely $x, z, xz, z', xz, z, x', z'$.

Here there are eight fundamental products, namely $x, z, xz, z', xz, z, x', z'$.

Here there are eight fundamental products, namely $x, z, xz, z', xz, z, x', z'$.

Here there are eight fundamental products, namely $x, z, xz, z', xz, z, x', z'$.
For a Boolean expression $g(x, y, z)$, the function $f(x, y, z)$ is the same function as $g(x, y, z)$.

Techniques for minimizing a Boolean function differ from those used for manipulating algebraic expressions. The minimum or a minimal cover of $g(x, y, z)$ is a collection of squares which form a two-by-four rectangle. These squares are the ones which contain a 1 or a 0 for each variable.

For a minimal cover of $g(x, y, z)$, the function $f(x, y, z)$ is the same function as $g(x, y, z)$.

The function $f(x, y, z)$ is defined as follows:

$$f(x, y, z) = g(x, y, z)$$

For a minimal cover of $g(x, y, z)$, the function $f(x, y, z)$ is the same function as $g(x, y, z)$.

For a minimal cover of $g(x, y, z)$, the function $f(x, y, z)$ is the same function as $g(x, y, z)$.

For a minimal cover of $g(x, y, z)$, the function $f(x, y, z)$ is the same function as $g(x, y, z)$.

For a minimal cover of $g(x, y, z)$, the function $f(x, y, z)$ is the same function as $g(x, y, z)$.

For a minimal cover of $g(x, y, z)$, the function $f(x, y, z)$ is the same function as $g(x, y, z)$.

For a minimal cover of $g(x, y, z)$, the function $f(x, y, z)$ is the same function as $g(x, y, z)$.

For a minimal cover of $g(x, y, z)$, the function $f(x, y, z)$ is the same function as $g(x, y, z)$.

For a minimal cover of $g(x, y, z)$, the function $f(x, y, z)$ is the same function as $g(x, y, z)$.

For a minimal cover of $g(x, y, z)$, the function $f(x, y, z)$ is the same function as $g(x, y, z)$.
THE Axioms of $F$ are the prime divisors of $m$.

Boolean Algebra

In the lattice-dense (upper bound) of $F$, $a$ is an atom of $F$, and $b$ is a

and $c$ are defined (lower bound) of $D$, and then that $m = \text{atom}(x)$. Hence $x \leq y$.

$1 = \text{atom}(x)$, so $x$. Since $m = m$ and $x \leq m$, let $x = m$. Since

We need only show that $F$ is commutative. Let $x, y \in F$. Then $x \cdot y = y \cdot x$,

THEOREM (Chapter 10) that the set $F$ of all divisors of $m$ is a bounded distributive lattice of

which can be written as $a 

Then the dual is $a 

The dual of each Boolean equation which are commutative and $1$.

$\overline{a + b} = \overline{a} \cdot \overline{b}$

$\overline{a \cdot b} = \overline{a} + \overline{b}$

$\overline{0} = \overline{1} = \overline{1}$

$\overline{1} = \overline{0}$

Thus $q + v = q \cdot v + v \cdot q$ (q) The dual of each Boolean equation which are

Solved Problems

(a) $F = \{0, 1, a, b, c\}$

(b) $F = \{0, 1, a, b, c\}$

(c) $F = \{0, 1, a, b, c\}$

(d) $F = \{0, 1, a, b, c\}$

\[ \overline{a + b} = \overline{a} \cdot \overline{b} \]

\[ \overline{a \cdot b} = \overline{a} + \overline{b} \]

\[ \overline{0} = \overline{1} = \overline{1} \]

\[ \overline{1} = \overline{0} \]

The set of all divisors of $m$ is a bounded distributive lattice.

EXAMPLE 12.1 Consider the lattice $F$ of all divisors of $m$. The atoms of $F$ are the prime divisors of $m$.

BOOLEAN ALGEBRAS
<table>
<thead>
<tr>
<th>C</th>
<th>E</th>
<th>Z</th>
<th>A</th>
<th>X</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Table 1
\[ z + \Delta x = (2\Delta x) \Delta \]

<table>
<thead>
<tr>
<th></th>
<th>( z )</th>
<th>( \Delta x )</th>
<th>2</th>
<th>( \Delta )</th>
<th>( x )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

**Table 2**
We will show how the Quine-McCluskey method can be used to

Figure 1. The Karnaugh map for f

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 1
Table 3

<table>
<thead>
<tr>
<th>Step 2</th>
<th>Step 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Term String</td>
<td>Term String</td>
</tr>
<tr>
<td>000</td>
<td>z̅x</td>
</tr>
<tr>
<td>100</td>
<td>z̅x</td>
</tr>
<tr>
<td>110</td>
<td>z̅x</td>
</tr>
<tr>
<td>101</td>
<td>z̅x</td>
</tr>
<tr>
<td>111</td>
<td>z̅x</td>
</tr>
</tbody>
</table>

Table 2

<table>
<thead>
<tr>
<th>0</th>
<th>000</th>
<th>z̅x</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>100</td>
<td>z̅x</td>
</tr>
<tr>
<td>2</td>
<td>110</td>
<td>x</td>
</tr>
<tr>
<td>2</td>
<td>101</td>
<td>x</td>
</tr>
<tr>
<td>3</td>
<td>111</td>
<td>x</td>
</tr>
</tbody>
</table>

Table 2

<table>
<thead>
<tr>
<th>Number of 1's</th>
<th>Bit String</th>
</tr>
</thead>
<tbody>
<tr>
<td>z̅x</td>
<td>z̅x</td>
</tr>
<tr>
<td>000</td>
<td>z̅x</td>
</tr>
<tr>
<td>100</td>
<td>z̅x</td>
</tr>
<tr>
<td>110</td>
<td>z̅x</td>
</tr>
<tr>
<td>101</td>
<td>z̅x</td>
</tr>
<tr>
<td>111</td>
<td>z̅x</td>
</tr>
</tbody>
</table>

According to the number of 1's in the corresponding bit strings, 1 if x occurs and 0 if x doesn't occur. The second bit will be 1 if x occurs and 0 if x doesn't occur. We will represent the minterms by 0's in this expansion by bit strings. The first bit will be

Next all pairs of products of two literals that can be combined are combined to form product terms from these combinations are shown in Table 3.

Hence two terms that can be combined differ by exactly one in the number of 1's.
Products in \( n \) variables that can be combined are represented by the
Boolean sum of the products in \( n - 1 \) variables formed in the previous step.

4. Determine all products in \( n - 1 \) variables that can be formed by taking the
position \( j \) where no literal involving \( x_j \) is in the product.

If position \( j \) is not no literal involving \( x_j \), then \( x_j \) occurs in the product, \( 0 \) in this position if \( x_j \), occurs, and a dash in this
place. Products in \( n - 1 \) variables with strings that have a in the position if
variables in \( n \) variables that differ in exactly one position. Represent
are represented by all strings that differ in exactly one position. Represent
Boolean sum of monomials in the expression. Monomials that can be combined
Boolean sum of monomials in the expression. Monomials that can be formed by taking the
position \( j \) is not no literal involving \( x_j \), then \( 1 \) in the
variables in \( n - 1 \) variables that can be formed by taking the
sequence of steps to simplify a sum-of-products expression.

As was illustrated in Example 6, the Quine-McCluskey method uses the following

\[
\begin{array}{ccc|c}
0 & 1 & 2 & 3 \\
\hline
x & x & x & x \\
\end{array}
\]

Table 4

\[ z + x' \]

\[ \text{From Table 4, we see that both } z \text{ and } x' \text{ are needed. Hence, the final answer is} \]

\[ x + x' \]

\[ \text{To form the candidate product in this case, we say that the candidate product} \]

\[ x + x' \]

\[ \text{To form the candidate product in this case, we say that the candidate product} \]

\[ x + x' \]

\[ \text{To form the candidate product in this case, we say that the candidate product} \]

\[ x + x' \]

\[ \text{To form the candidate product in this case, we say that the candidate product} \]

\[ x + x' \]

\[ \text{To form the candidate product in this case, we say that the candidate product} \]

\[ x + x' \]

\[ \text{To form the candidate product in this case, we say that the candidate product} \]

\[ x + x' \]

\[ \text{To form the candidate product in this case, we say that the candidate product} \]

\[ x + x' \]

\[ \text{To form the candidate product in this case, we say that the candidate product} \]

\[ x + x' \]

\[ \text{To form the candidate product in this case, we say that the candidate product} \]

\[ x + x' \]

\[ \text{To form the candidate product in this case, we say that the candidate product} \]

\[ x + x' \]

\[ \text{To form the candidate product in this case, we say that the candidate product} \]

\[ x + x' \]

\[ \text{To form the candidate product in this case, we say that the candidate product} \]

\[ x + x' \]

\[ \text{To form the candidate product in this case, we say that the candidate product} \]

\[ x + x' \]

\[ \text{To form the candidate product in this case, we say that the candidate product} \]

\[ x + x' \]

\[ \text{To form the candidate product in this case, we say that the candidate product} \]

\[ x + x' \]

\[ \text{To form the candidate product in this case, we say that the candidate product} \]

\[ x + x' \]

\[ \text{To form the candidate product in this case, we say that the candidate product} \]

\[ x + x' \]

\[ \text{To form the candidate product in this case, we say that the candidate product} \]

\[ x + x' \]

\[ \text{To form the candidate product in this case, we say that the candidate product} \]

\[ x + x' \]

\[ \text{To form the candidate product in this case, we say that the candidate product} \]

\[ x + x' \]

\[ \text{To form the candidate product in this case, we say that the candidate product} \]

\[ x + x' \]

\[ \text{To form the candidate product in this case, we say that the candidate product} \]

\[ x + x' \]

\[ \text{To form the candidate product in this case, we say that the candidate product} \]

\[ x + x' \]

\[ \text{To form the candidate product in this case, we say that the candidate product} \]

\[ x + x' \]

\[ \text{To form the candidate product in this case, we say that the candidate product} \]

\[ x + x' \]

\[ \text{To form the candidate product in this case, we say that the candidate product} \]

\[ x + x' \]

\[ \text{To form the candidate product in this case, we say that the candidate product} \]

\[ x + x' \]

\[ \text{To form the candidate product in this case, we say that the candidate product} \]

\[ x + x' \]

\[ \text{To form the candidate product in this case, we say that the candidate product} \]

\[ x + x' \]

\[ \text{To form the candidate product in this case, we say that the candidate product} \]

\[ x + x' \]
<table>
<thead>
<tr>
<th>Term</th>
<th>Bit string</th>
<th>Number of 1's</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1000</td>
<td>$\bar{x}y'z$</td>
</tr>
<tr>
<td>2</td>
<td>1100</td>
<td>$\bar{x}y'z'$</td>
</tr>
<tr>
<td>2</td>
<td>1010</td>
<td>$x'y'z'$</td>
</tr>
<tr>
<td>2</td>
<td>0101</td>
<td>$x'y'z$</td>
</tr>
<tr>
<td>3</td>
<td>1110</td>
<td>$x'y'z$</td>
</tr>
<tr>
<td>3</td>
<td>1101</td>
<td>$x'y'z'$</td>
</tr>
<tr>
<td>3</td>
<td>0111</td>
<td>$x'yz'$</td>
</tr>
</tbody>
</table>

Table 5

These products are shown in Table 6.

Table 6. All the boolean products that can be formed by taking Boolean sums of $x'y'z$, $x'y'z'$, and $x'yz'$, and $x'yz$ are shown in Table 7. We show the products covered by each of these. Of course, we could have included all of these products in Table 7, as shown in the previous table. These products are the only products that were not used to form products in fewer variables are shown in Table 7.

Solution. We first represent the minima by bit strings and then group these: $x'y'z$, $x'y'z'$, $x'yz'$, $x'yz$.

<table>
<thead>
<tr>
<th>Product</th>
<th>Bit string</th>
<th>Number of 1's</th>
</tr>
</thead>
<tbody>
<tr>
<td>$xyz'$</td>
<td>$1010$</td>
<td>2</td>
</tr>
<tr>
<td>$x'y'z'$</td>
<td>$1100$</td>
<td>2</td>
</tr>
<tr>
<td>$x'y'z$</td>
<td>$1110$</td>
<td>2</td>
</tr>
<tr>
<td>$x'y'z'$</td>
<td>$1010$</td>
<td>2</td>
</tr>
<tr>
<td>$x'yz'$</td>
<td>$0111$</td>
<td>2</td>
</tr>
<tr>
<td>$x'yz$</td>
<td>$0101$</td>
<td>2</td>
</tr>
<tr>
<td>$x'yz'$</td>
<td>$1101$</td>
<td>2</td>
</tr>
</tbody>
</table>

Table 7

In Table 7, use the Quine-McCluskey method to simplify the sum-of-products expression:

$$+ar{x}y'z + ar{x}y'z' + x'y'z + x'y'z' + x'yz + x'yz' + z'x'y'z$$

A worked example will illustrate how this procedure is used to simplify a sum-of-products expression. It can be mechanized using a backtracking procedure (Ref. 6).

1. Find the smaller set of those Boolean products so that the sum of these products in one fewer literal than the smaller set.
2. Combine the Boolean products that arise that were not used to form a Boolean product in one fewer literal.
3. Continue combining these products into products in fewer variables as long as possible.

Table 8

<table>
<thead>
<tr>
<th>Product</th>
<th>Bit string</th>
<th>Number of 1's</th>
</tr>
</thead>
<tbody>
<tr>
<td>$xyz'$</td>
<td>$1010$</td>
<td>2</td>
</tr>
<tr>
<td>$x'y'z'$</td>
<td>$1100$</td>
<td>2</td>
</tr>
<tr>
<td>$x'y'z$</td>
<td>$1110$</td>
<td>2</td>
</tr>
<tr>
<td>$x'y'z'$</td>
<td>$1010$</td>
<td>2</td>
</tr>
<tr>
<td>$x'yz'$</td>
<td>$0111$</td>
<td>2</td>
</tr>
<tr>
<td>$x'yz$</td>
<td>$0101$</td>
<td>2</td>
</tr>
<tr>
<td>$x'yz'$</td>
<td>$1101$</td>
<td>2</td>
</tr>
</tbody>
</table>

Table 8

The only products that were not used to form products in fewer variables are shown in Table 7.

These products are shown in Table 6.

Table 6. All the boolean products that can be formed by taking Boolean sums of $x'y'z$, $x'y'z'$, and $x'yz'$ are shown in Table 7. We show the products covered by each of these. Of course, we could have included all of these products in Table 7, as shown in the previous table. These products are the only products that were not used to form products in fewer variables are shown in Table 7.

Solution. We first represent the minima by bit strings and then group these: $x'y'z$, $x'y'z'$, $x'yz'$, $x'yz$.

<table>
<thead>
<tr>
<th>Product</th>
<th>Bit string</th>
<th>Number of 1's</th>
</tr>
</thead>
<tbody>
<tr>
<td>$xyz'$</td>
<td>$1010$</td>
<td>2</td>
</tr>
<tr>
<td>$x'y'z'$</td>
<td>$1100$</td>
<td>2</td>
</tr>
<tr>
<td>$x'y'z$</td>
<td>$1110$</td>
<td>2</td>
</tr>
<tr>
<td>$x'y'z'$</td>
<td>$1010$</td>
<td>2</td>
</tr>
<tr>
<td>$x'yz'$</td>
<td>$0111$</td>
<td>2</td>
</tr>
<tr>
<td>$x'yz$</td>
<td>$0101$</td>
<td>2</td>
</tr>
<tr>
<td>$x'yz'$</td>
<td>$1101$</td>
<td>2</td>
</tr>
</tbody>
</table>

Table 7

In Table 7, use the Quine-McCluskey method to simplify the sum-of-products expression:

$$+ar{x}y'z + ar{x}y'z' + x'y'z + x'y'z' + x'yz + x'yz' + z'x'y'z$$

A worked example will illustrate how this procedure is used to simplify a sum-of-products expression. It can be mechanized using a backtracking procedure (Ref. 6).

1. Find the smaller set of those Boolean products so that the sum of these products in one fewer literal than the smaller set.
2. Combine the Boolean products that arise that were not used to form a Boolean product in one fewer literal.
3. Continue combining these products into products in fewer variables as long as possible.

Table 8

<table>
<thead>
<tr>
<th>Product</th>
<th>Bit string</th>
<th>Number of 1's</th>
</tr>
</thead>
<tbody>
<tr>
<td>$xyz'$</td>
<td>$1010$</td>
<td>2</td>
</tr>
<tr>
<td>$x'y'z'$</td>
<td>$1100$</td>
<td>2</td>
</tr>
<tr>
<td>$x'y'z$</td>
<td>$1110$</td>
<td>2</td>
</tr>
<tr>
<td>$x'y'z'$</td>
<td>$1010$</td>
<td>2</td>
</tr>
<tr>
<td>$x'yz'$</td>
<td>$0111$</td>
<td>2</td>
</tr>
<tr>
<td>$x'yz$</td>
<td>$0101$</td>
<td>2</td>
</tr>
<tr>
<td>$x'yz'$</td>
<td>$1101$</td>
<td>2</td>
</tr>
</tbody>
</table>
2. Find the sum-of-products expressions represented by each of the following Karnaugh maps.

(b) What are the minterms represented by squares adjacent to the square representing \( x \)?

I. (a) Draw a Karnaugh map for a function in two variables and put a 1 in the square

as the final answer.

needled. Consequently, we can take either \( xz + w'z + wy + x'y \) or \( wy + x'y \). These two products are included, we see that only one of the two products left is

<table>
<thead>
<tr>
<th>( x )</th>
<th>( x )</th>
<th>( x )</th>
<th>( x )</th>
<th>( x )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( x )</td>
<td>( x )</td>
<td>( x )</td>
<td>( x )</td>
<td>( x )</td>
</tr>
<tr>
<td>( x )</td>
<td>( x )</td>
<td>( x )</td>
<td>( x )</td>
<td>( x )</td>
</tr>
<tr>
<td>( x )</td>
<td>( x )</td>
<td>( x )</td>
<td>( x )</td>
<td>( x )</td>
</tr>
<tr>
<td>( x )</td>
<td>( x )</td>
<td>( x )</td>
<td>( x )</td>
<td>( x )</td>
</tr>
</tbody>
</table>

Table 2

<table>
<thead>
<tr>
<th>Term String</th>
<th>Term String</th>
<th>Term String</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-00 ( zyw ) (7)</td>
<td>1000 ( z\bar{x}y )</td>
<td>7</td>
</tr>
<tr>
<td>10-0 ( z\bar{w}y ) (1)</td>
<td>1100 ( z\bar{x}m )</td>
<td>6</td>
</tr>
<tr>
<td>11-0 ( zyw ) (8)</td>
<td>1010 ( z\bar{x}m )</td>
<td>5</td>
</tr>
<tr>
<td>1-10 ( z\bar{x}w ) (3)</td>
<td>0101 ( z\bar{w}y )</td>
<td>4</td>
</tr>
<tr>
<td>110- ( z\bar{x}y ) (9)</td>
<td>1110 ( zx \bar{w} )</td>
<td>3</td>
</tr>
<tr>
<td>-101 ( z\bar{x}y ) (4)</td>
<td>1101 ( zy \bar{w} )</td>
<td>2</td>
</tr>
<tr>
<td>01-1 ( z\bar{w}m ) (14)</td>
<td>0111 ( z\bar{x}y )</td>
<td>1</td>
</tr>
</tbody>
</table>

Table 6