Can you imagine an algebra in which the variables can take on only two values, 0 and 1, instead of the whole range of real numbers? Such an algebra is not only possible, but it turns out to be very useful. After all, if you are dealing with switch positions, you only need two numbers, a 0 to represent the open position and a 1 to represent the closed position. And when Boole applied algebra to the English language, he needed only two numbers, a 0 to represent “false” and a 1 to represent “true.” Further, we need only two values when working with binary digits, which have only two possible values.

As we develop the Boolean algebra here, we illustrate each new idea in terms of electrical switches and, because they are easy to visualize, in term of sets. We then go on to apply Boolean algebra to more complex circuits, and then to the logic gates which do the arithmetic computations in a computer.

Named for the English mathematician George Boole (1815–1864), who wrote “An Investigation of the Laws of Thought On Which Are Founded the Mathematical Theories of Logic and Probabilities” in 1854.
1 Boolean Variables and Operations

Boolean Variables

We will represent Boolean variables by the capital letters A, B, C, ... X, Y, Z. In ordinary algebra, a variable such as x will usually have a value that is one of the real numbers. For example, the solution to the equation

$$2x - 8 = 0$$

is

$$x = 4$$

In Boolean algebra a variable can never have the value 4. In fact, Boolean variables can have only one of the values 0 or 1.

These variables should not be thought of as numerical values, but rather as binary states, such as TRUE/FALSE, or YES/NO.

When applying Boolean algebra to sets, we let

0 = the empty set
1 = the universal set

When applying Boolean algebra to switching circuits, we let

0 = off
1 = on

Example 1:

(a) If our universal set U is the set of all students at a certain school, and B represents the set of all boys at that school, then

$$B = 0$$

means that there are no boys at that school, and

$$B = 1$$

means that all the students are boys.

(b) If X represents a certain switch (Fig. 1) then

$$X = 0$$

means that the switch is open (no current can flow), and

$$X = 1$$

means that the switch is closed (a current can flow).

Boolean Operations

In ordinary algebra we had the operations of addition, subtraction, and so forth. In Boolean algebra we have the three basic operations

AND OR NOT

which are represented by the multiplication dot (•), the plus sign (+), and the apostrophe (‘) or overbar (¯).

Example 2:

$$A \text{ AND } B \quad A \text{ OR } B \quad A'$$

$$A \cdot B \quad A + B \quad A' \text{ or } \overline{A}$$
The AND Operator

We represent the AND operation by $A \land B$, or simply by $AB$.

When talking about sets, the AND operator represents the intersection of two sets $A$ and $B$, which consists of those elements that are in both $A$ AND $B$. We use the symbol $\cap$ for intersection.

*** Example 3: *** If $R$ is the set of all red books in a library and $M$ is the set of all math books in the library, then

$$R \cap M$$

is the set of all red math books in the library, as in the Venn diagram (Fig. 2).

When talking about switches, the AND operator represents two switches $X$ and $Y$ in series, as in Fig. 3.

*** Example 4: *** The expression

$$X \land Y = 1$$

means that two switches $X$ and $Y$ in series are both closed. Thus, both $X$ AND $Y$ must be closed in order for current to flow from $p$ to $q$.

Postulates for the AND Operator

The postulates of Boolean algebra, those ideas that are stated without proof and from which all other theorems are derived, are given here. For each, we give an example for sets and for switching circuits.

Our first postulate is

$$0 \cdot 0 = 0$$

*** Example 5: ***

(a) The intersection of two empty sets is an empty set.
(b) Two open switches in series is an open circuit.

Our next postulate is

$$0 \cdot 1 = 1 \cdot 0 = 0$$

*** Example 6: ***

(a) The intersection of the empty set with the universal set is the empty set.
(b) An open switch in series with a closed switch is an open circuit.

Our last postulate for the AND operator is

$$1 \cdot 1 = 1$$

*** Example 7: ***

(a) The intersection of the universal set and itself is the universal set.
(b) Two closed switches in series is a closed circuit.

Truth Tables

A truth table is a convenient way of defining the Boolean operators, for all possible states of the variables. Thus $A$ can be either 0 or 1, and for each of these, $B$ can be
Example 8:
If \( A = 1, B = 0, \) and \( C = 1, \) evaluate
\[
(A \cdot B) \cdot (B \cdot C) \cdot (A \cdot C)
\]

Solution: Substituting the values for the variables,
\[
(A \cdot B) \cdot (B \cdot C) \cdot (A \cdot C) = (1 \cdot 0) \cdot (0 \cdot 1) \cdot (1 \cdot 1)
\]
Then using the truth table or the postulates gives
\[
(1 \cdot 0) \cdot (0 \cdot 1) \cdot (1 \cdot 1) = 0 \cdot 0 \cdot 1
\]
\[= 0\]

The OR Operator
When discussing two sets \( A \) and \( B, \) the OR operator gives us the union of \( A \) and \( B. \) Thus, an element of \( A + B \) is a member of sets \( A \) OR \( B. \) We use the symbol \( \cup \) for the union of two sets.

Example 9: If \( R \) is the set of all red books and \( M \) is the set of all math books, as before, then
\[R + M\]
is the set of all books that are either red or mathematical (Fig. 4).

When used for switching circuits the OR operator represents two switches in parallel, as in Fig. 5.

Example 10: The expression
\[X + Y = 1\]
means that for two switches \( X \) and \( Y \) in parallel, current flows from \( p \) to \( q \) when either \( X \) OR \( Y \) is closed.

Inclusive and Exclusive OR
In everyday speech the “or” is used in two different ways. It sometimes means “\( A \) or \( B \) or both,” as in
\[I \text{ enjoy jazz or rock.}\]

but at other times “\( A \) or \( B \) but not both,” as in
\[Shall I play jazz or rock on my iPod?\]
The first is called the inclusive OR and the second is called the exclusive OR. We will take the symbol (+) to stand for the inclusive OR.
Postulates for the OR Operator

The three postulates that apply to the OR operator are given here. The first is

**Example 11:**

(a) The union of two empty sets is an empty set.
(b) Two open switches in parallel is an open circuit.

The second postulate is

**Example 12:**

(a) The union of the universal set and the empty set, in either order, is the universal set.
(b) A closed switch in parallel with an open switch, in either order, is a closed circuit.

The last postulate for the OR operator is

**Example 13:**

(a) The union of the universal set with itself is the universal set.
(b) Two closed switches in parallel is a closed circuit.

As with the AND operator, the postulates for the OR operator can be combined into a truth table.

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>A + B</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

**Example 14:** If \( A = 1, B = 0, \) and \( C = 0, \) evaluate \((A + B) + (B + C)\)

**Solution:** Substituting and using the truth table,

\[(A + B) + (B + C) = (1 + 0) + (0 + 0) = 1 + 0 = 1\]

The NOT Operator

We represent the NOT operator with a prime or an overbar:

\( A' \) or \( \overline{A} \)

The NOT operator is defined by the simple truth table:

<table>
<thead>
<tr>
<th>A</th>
<th>( \overline{A} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>
Example 15: If \( A = 1 \), and \( C = D = 0 \), evaluate \((A + \overline{B}) \cdot (\overline{C} + D)\)

Solution: Substituting, and using the truth tables for AND, OR, and NOT, gives
\[
(A + \overline{B}) \cdot (\overline{C} + D) = (1 + 0) \cdot (1 + 0) = 1 \cdot 1 = 1
\]

When applied to a set \( A \), the NOT operator will give us the complement \( A' \) of that set, such as the shaded region in the Venn diagram (Fig. 6). We use the symbol \( A' \) to represent the complement of \( A \).

Example 16: If the universal set \( U \) consists of all the books in a library, and \( R \) is the set of all red books, then \( \overline{R} \) is the set of all book in the library that are not red.

The interpretation of the NOT operator in a switching circuit can be seen in Fig. 7, which shows a double-throw switch. If we label one position \( X \), then the other position is \( X' \). Thus, when circuit \( pq \) is continuous, \( pr \) is not, and vice versa.

EXERCISE 1 • Boolean Variables and Operations

If \( A = 1, B = 0, \) and \( C = 1 \), evaluate:

1. \((AB)(B + C)\)
2. \( A + BC + AC \)

If \( A = 1, B = 0, \) and \( C = 0 \), evaluate:

3. \((A + B)(B + C)\)
4. \( AB + BC + AC \)
5. If \( A = B = 1, \) and \( C = D = 0 \), evaluate: \((A + \overline{B})(\overline{C} + D)\).

Draw a Venn diagram for three sets \( A, B, \) and \( C, \) and shade the area representing:

6. \( AB \)
7. \( A + B \)
8. \( ABC \)
9. \( A + B + C \)
10. \( \overline{A} \)
11. \( \overline{A}B \)
12. \( \overline{A}B \)
13. \( \overline{A}B \)

Write a Boolean expression to represent each circuit.

14. Figure 8a.
15. Figure 8b.
16. Figure 8c.
17. Figure 8d.

Use a Venn diagram to decide which of the following expressions is valid.

18. \( A + \overline{B} = \overline{A} + \overline{B} \)
19. \( \overline{A} + B = \overline{A}B \)

Use a Venn diagram to simplify each expression.

20. \( AB + BC + \overline{C}A \)
21. \((A + \overline{B})(\overline{A} + C)\)

Draw a switch circuit to implement each expression.

22. \( X + YZ \)
23. \( X \overline{Y}Z \)
2 Laws of Boolean Algebra

Commutative Laws

Let us now combine the truth tables for AND, OR, and NOT into a single table:

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>AND</th>
<th>OR</th>
<th>NOT</th>
</tr>
</thead>
<tbody>
<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>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

and see what laws we can deduce. Look first at the AND and OR columns in the combined truth table. We notice that the values for $A$ and $B$ can be interchanged without changing the value of $AB$ or $A + B$. In other words,

\[
AB = BA \quad 4a
\]

\[
A + B = B + A \quad 4b
\]

The value of one variable AND another, or one variable OR another does not depend on the order in which the variables are written.

Boundedness Laws

We next notice that whenever either $A$ or $B$ is 0, then $AB$ is 0. This gives

\[
A \cdot 0 = 0 \quad 5a
\]

The value of a variable AND zero is always zero.

*** Example 17:

(a) The intersection of any set with an empty set is equal to an empty set.
(b) If switch $X$ is in series with an open switch, the circuit remains open regardless of the position of switch $X$. 

Further, whenever either $A$ or $B$ is 1, then $A + B$ is 1.

\[
A + 1 = 1 \quad 5b
\]

The value of any variable OR 1 always equals 1.

*** Example 18:

(a) The union of any set with the universal set is equal to the universal set.
(b) If a switch $X$ is in parallel with a switch that is always closed, the circuit remains closed regardless of the position of switch $X$. 

Note the similarity with the commutative laws for ordinary algebra.
Identity Laws

Next we see that whenever one variable is 1, then $AB$ takes the value of the other variable. In symbols,

\[ A \cdot 1 = A \]  \hspace{1cm} 6a

*The value of a variable AND 1 is the same as that of the original variable.*

***Example 19:***

(a) The intersection of any set $A$ and the universal set is equal to set $A$.
(b) A switch that remains closed has no effect when connected in series.

Further, when one variable is 0, then $A + B$ takes the value of the other variable.

\[ A + 0 = A \]  \hspace{1cm} 6b

*The value of a variable OR zero is the same as that of the original variable.*

***Example 20:***

(a) The union of any set $A$ and an empty set is equal simply to set $A$.
(b) A switch that is always open has no effect when connected in parallel with any circuit.

Idempotent Laws

Next we see that when $A$ and $B$ have the same value, that $AB$ has that same value.

\[ AA = A \]  \hspace{1cm} 7a

*The value of a variable AND itself is the same as the value of the original variable.*

Also, when $A$ and $B$ have the same value, then $A + B$ has that same value.

\[ A + A = A \]  \hspace{1cm} 7b

*The value of a variable OR itself is the same as the value of the original variable.*

***Example 21:***

(a) The intersection or union of a set $A$ and itself is simply $A$.
(b) When two switches that operate in unison (both open or both closed) are connected in series or in parallel, one of the switches can be eliminated.

Complement Laws

If $A = 1$, then $\bar{A} = 0$, and if $A = 0$, then $\bar{A} = 1$. Either way, we have

\[ A \cdot \bar{A} = 1 \cdot 0 \]

which, by law 6a, is equal to 0. So

\[ A \cdot \bar{A} = 0 \]  \hspace{1cm} 8a

*The value of a variable AND its complement is zero.*
Section 2  ♦ Laws of Boolean Algebra

*** Example 22:
(a) The intersection of a set and the complement of that set is the empty set.
(b) When a switch is in series with another switch that is always in the opposite state (Fig. 9), the circuit will always be open.

Similarly, for the OR,

\[ A + \bar{A} = 1 + 0 \]

which, by law 6b, equals 1. So

\[ A + \bar{A} = 1 \] 8b

The value of a variable OR its complement is 1.

*** Example 23:
(a) The union of a set and its complement is the universal set.
(b) When a switch is in parallel with another switch that is always in the opposite state (Fig. 10), the circuit will always be closed.

**Associative Laws**

One way to verify that a law of Boolean algebra is true is to see if the truth tables for each side of the equation are identical.

*** Example 24: Verify the associative law for the AND operator,

\[ A(BC) = (AB)C \] 9a

**Solution:** We make a truth table for \( A(BC) \) and for \((AB)C\).

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>BC</th>
<th>A(BC)</th>
<th>AB</th>
<th>(AB)C</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</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>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</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>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>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Try now to prove the associative law for the OR operator,

\[ A + (B + C) = (A + B) + C \] 9b

on your own, again by making a truth table for each side.

*** Example 25:
(a) When finding the union or intersection of three sets, it does not matter in which order they are taken.
(b) When three switches are wired in series or in parallel, the order of the switches does not matter.
Distributive Laws

The first distributive law,

\[ A(B + C) = AB + AC \]  

is useful because it enables us to \textit{factor} and to \textit{multiply} Boolean expressions in much the same way that Eq. 5 allowed us to factor and to multiply algebraic expressions. Let us verify law 10a by a method different than that we used before. Rather than compare the truth table for each side, we compare the Venn diagram for each, as in Fig. 11.

\[ \text{FIGURE 11} \quad \text{Use of a Venn diagram to verify law 10a.} \]

### Example 26:

Switch \( Y \) in parallel with \( Z \), both in series with \( X \) (Fig. 12a), is equivalent to switch \( X \) in series with \( Y \), both in parallel to \( X \) in series with \( Z \) (Fig. 12b).

\[ \text{FIGURE 12} \]

We now derive the distributive law

\[ A + BC = (A + B)(A + C) \]  

By law 10a, the right side becomes

\[
(A + B)(A + C) = (A + B)A + (A + B)C = AA + AB + AC + BC
\]

But by law 7a, \( AA = A \), so

\[
(A + B)(A + C) = A + AB + AC + BC
\]
Section 2  •  Laws of Boolean Algebra

Factoring by law 10a,

\[(A + B)(A + C) = A(1 + B) + AC + BC\]

But, by law 5b, \(1 + B = 1\), so

\[(A + B)(A + C) = A \cdot 1 + AC + BC\]

and by law 6a,

\[= A + AC + BC\]

Again using law 10a,

\[(A + B)(A + C) = A(1 + C) + BC\]

\[= A \cdot 1 + BC\]

\[= A + BC\]

We now prove the distributive law

\[A(\overline{A} + B) = AB\] 11a

**Proof:** By law 10a, the left side becomes

\[A(\overline{A} + B) = A\overline{A} + AB\]

Then by law 8a,

\[= 0 + AB\]

and by law 6b,

\[= AB\]

The proof of the distributive law

\[A + \overline{A}B = A + B\] 11b

is left as an exercise.

**Absorption Laws**

We prove the absorption law

\[A(A + B) = A\] 12a

**Proof:** By law 10a, the left side becomes

\[A(A + B) = AA + AB\]

Then by law 7a,

\[= A + AB\]

Factoring by law 10a,

\[A(A + B) = A(1 + B)\]

By law 5b,

\[= A \cdot 1\]

and by law 6a,

\[= A\]

The proof of the other absorption law

\[A + AB = A\] 12b

is left as an exercise.

Notice that the variable \(B\) originally on the left side has vanished, or has been absorbed.
DeMorgan’s Laws

\[ \overline{A \cdot B} = \overline{A} + \overline{B} \]  

The complement of one variable AND another is equal to the complement of one of the variables OR the complement of the other variable.

Proof: We make a truth table for each side.

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>A \cdot B</th>
<th>\overline{A} \cdot \overline{B}</th>
<th>A + B</th>
</tr>
</thead>
<tbody>
<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>0</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>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

The identical last columns of each truth table proves DeMorgan’s law.

We leave the proof of the other of DeMorgan’s laws,

\[ \overline{A + B} = \overline{A} \cdot \overline{B} \]

as an exercise.

The Involution Law

Our last law is

\[ \overline{\overline{A}} = A \]

The complement of a complement of a variable is equal to that variable.

Proof: We verify it with the following truth table:

<table>
<thead>
<tr>
<th>A</th>
<th>\overline{A}</th>
<th>\overline{\overline{A}}</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

Duality

The 21 laws are listed in Table 1. The first 12 can be divided into six that apply to the AND operator, and six to the OR. The other laws cannot be separated in that way.

Now compare each law in the left column of Table 1 with the one alongside it in the right column. Do you see that each of the pair could be obtained from the other by

1. Interchanging AND and OR operators, and
2. Interchanging 1 and 0?

One such expression is said to be the dual of the other. The principle of duality in Boolean algebra states that the dual of any law is also a law.
Section 2  ▷ Laws of Boolean Algebra

TABLE 1  Laws of Boolean Algebra.

<table>
<thead>
<tr>
<th>Law</th>
<th>AND</th>
<th>OR</th>
</tr>
</thead>
<tbody>
<tr>
<td>Commutative</td>
<td>(a) $AB = BA$</td>
<td>(b) $A + B = B + A$</td>
</tr>
<tr>
<td>Boundedness</td>
<td>(a) $A \cdot 0 = 0$</td>
<td>(b) $A + 1 = 1$</td>
</tr>
<tr>
<td>Identity</td>
<td>(a) $A \cdot 1 = A$</td>
<td>(b) $A + 0 = A$</td>
</tr>
<tr>
<td>Idempotent</td>
<td>(a) $AA = A$</td>
<td>(b) $A + A = A$</td>
</tr>
<tr>
<td>Complement</td>
<td>(a) $A \cdot \overline{A} = 0$</td>
<td>(b) $A + \overline{A} = 1$</td>
</tr>
<tr>
<td>Associative</td>
<td>(a) $A(BC) = (AB)C$</td>
<td>(b) $A + (B + C) = (A + B) + C$</td>
</tr>
<tr>
<td>Distributive</td>
<td>(a) $A(B + C) = AB + AC$</td>
<td>(b) $A + BC = (A + B)(A + C)$</td>
</tr>
<tr>
<td>Absorption</td>
<td>(a) $A(A + B) = AB$</td>
<td>(b) $A + A \cdot B = A + B$</td>
</tr>
<tr>
<td>DeMorgan’s</td>
<td>(a) $A \cdot \overline{B} = \overline{A} + \overline{B}$</td>
<td>(b) $A + \overline{B} = \overline{A} \cdot \overline{B}$</td>
</tr>
<tr>
<td>Involution</td>
<td>$\overline{\overline{A}} = A$</td>
<td></td>
</tr>
</tbody>
</table>

Example 27: Write the dual of the Boolean expression

$$(1 + BC)D + 0$$

Solution: We interchange AND and OR operators

$$\{(1 \cdot (B + C)) + D\} \cdot 0$$

and interchange 1 and 0:

$$\{(0 \cdot (B + C)) + D\} \cdot 1$$

This expression is the dual of the original.

EXERCISE 2  ▷ Laws of Boolean Algebra

Verify each law. Use a truth table, a Venn diagram, or any Boolean law having a number lower than the one you are proving.

1. The associative law for the AND operator,

$$A(BC) = (AB)C$$  \hspace{1cm} (9a)

2. The associative law for the OR operator,

$$A + (B + C) = (A + B) + C$$  \hspace{1cm} (9b)

3. The absorption law,

$$A + AB = A$$  \hspace{1cm} (12b)

4. DeMorgan’s law,

$$\overline{A + B} = \overline{A} \cdot \overline{B}$$  \hspace{1cm} (13b)
3 Simplifying Boolean Expressions

In Secs. 4 and 5 we will write Boolean expressions to represent electrical hardware. Thus if we can replace a Boolean expression with one that is exactly equivalent but simpler, we can replace the hardware with a simpler version that does the same thing.

Rearrange Terms

Laws 4 allow us to rearrange the variables in an AND or an OR expression.

Example 28:

\[ A + B + C + B + C + A = A + A + B + B + C + C \]

This rearrangement allows us to simplify further, as we see in the next example.

Identical Variables

Look for identical variables. By laws 7, one of the variables can be dropped.

Example 29: Simplify

\[ A \cdot H_11001 \]

Solution: By law 7b,

\[ A + A + B + B + C + A = A + B + C \]

Example 30: Simplify

\[ A + B \cdot B + C + A \]

Solution: By law 4b,

\[ A + B \cdot B + C + A = A + A \cdot B \cdot C \]

and by laws 7,

\[ = A + B + C \]

Expressions Containing 1 or 0

Expressions containing 1 or 0 can be simplified using laws 5 and 6.

Example 31: Simplify

\[ A + 1 + B \cdot 0 + C + 0 + D \cdot 1 \]

Solution: By laws 5 and 6,

\[ A + 1 + B \cdot 0 + C + 0 + D \cdot 1 = 1 + 0 + C + D \]

Regrouping,

\[ = (C + 1) + (D + 0) \]

By laws 5b and 6b,

\[ = 1 + D \]

and by law 5b,

\[ = 1 \]

Removing Parentheses

When possible, remove parentheses by using the distributive law (10a).
Section 3  Simplifying Boolean Expressions

*** Example 32: ** Simplify the expression

\[ A(\bar{A} + 1) \]

**Solution:** By law 10a,

\[ A(\bar{A} + 1) = A\bar{A} + A \cdot 1 \]

By laws 6a and 8a, 

\[ = 0 + A \]

and by law 6b, 

\[ = A \]

A Variable and Its Complement

When an expression contains a variable and its complement, it can often be simplified using laws 8.

*** Example 33: ** Simplify

\[ A + B\bar{B} + C + \bar{A} \]

**Solution:** By law 4b,

\[ A + B\bar{B} + C + \bar{A} = (A + \bar{A}) + B\bar{B} + C \]

By laws 8,

\[ = 1 + 0 + C \]

By law 6b,

\[ = 1 + C \]

and by law 5b,

\[ = 1 \]

Factoring

As in ordinary algebra, factoring will sometimes help us simplify an expression.

*** Example 34: ** Simplify \( ABC + ABCD \).

**Solution:** By law 6a,

\[ ABC + ABCD = ABC \cdot 1 + ABCD \]

By law 10a,

\[ = ABC(1 + D) \]

By law 5b,

\[ = ABC(1) \]

and by law 6a,

\[ = ABC \]

Notice that the variable \( D \) is not present in the final expression. Thus, if \( D \) represented a switch in a circuit, we see here that it is not needed.

*** Example 35: ** Simplify \( AB\bar{C}D + ABCD \).

**Solution:** Factoring using law 10a,

\[ AB\bar{C}D + ABCD = ABD(\bar{C} + C) \]

By law 8b,

\[ = ABD(1) \]

and by law 6a,

\[ = ABD \]

As in the preceding example, one variable has vanished.

DeMorgan’s Laws

When we have the complement of two variables joined by an AND operator, such as

\[ (A \cdot B)' \]

also written \( \bar{A} \cdot \bar{B} \)
DeMorgan’s laws enable us to separate the variables, provided that we change the operator to OR.

\[ A \cdot \overline{B} = \overline{A + B} \]

If we use the overbar to represent a complement, we can make use of the rhyme *Break the line, change the sign.*

\[ \overline{A \cdot B} = \overline{A} + \overline{B} \]

change the sign.

The same rule holds if we start with the complement of \( A \) OR \( B \). When we break the overbar, we change the sign from \( \overline{+} \) to \( \cdot \).

### Example 36: Simplify

\[ A(\overline{B + \overline{C + D}}) \]

**Solution:** We first break the line between \( C \) and \( D \),

\[ A(\overline{B + \overline{C + D}}) = A(\overline{B} \cdot \overline{C} + D) \]

Then between \( B \) and \( C \),

\[ A(\overline{B} \cdot \overline{C}) = A(\overline{B} \cdot \overline{C}) \]

and between \( \overline{C} \) and \( D \),

\[ A(\overline{D}) = A \overline{B} \overline{C} \overline{D} \]

Then by law 14

\[ A \overline{B} \overline{C} \overline{D} = \overline{A} \overline{B} \overline{C} \overline{D} \]

### EXERCISE 3 • Simplifying Boolean Expressions

Expand each expression and simplify.

1. \((A + \overline{B})(\overline{A} + \overline{B})\)
2. \((A + BC)(A + A\overline{B})\)
3. \((A + B)(\overline{A} + B)\)
4. \((A + \overline{B})(\overline{A} + \overline{B})(A + B)(\overline{A} + B)\)

Factor each expression and simplify.

5. \(XYZ + XZ\)
6. \(\overline{AB} + AB\)
7. \(X\overline{Z} + XYZ + XZ\)
8. \(XY + XY + \overline{XY} + \overline{XY}\)

Simplify each expression.

9. \((\overline{X} + Y)(X + Y)\)
10. \(\overline{X}Y\overline{X}\)
11. \(XY + XYZ + XY\overline{Z}\)
12. \(\overline{X} + \overline{Y} + \overline{Z} + XYZ\)
13. \(X + Z + X + Y \cdot Y\)
14. \(X + Y \cdot 0 + Z + 0 + W \cdot 1 + 1\)
15. \((\overline{X} + 1)X\)
16. \(\overline{X} + Z + X + Y\overline{Y}\)
17. \(WXYZ + WYZ\)
18. \(WXY\overline{Z} + WXYZ\)
19. \((Y + \overline{W}Z)X\)
20. \(\overline{X}Y + XY + X\overline{Y}\)
21. \((XYZ)(XYZ)\)
22. \(XYZ + X\overline{Y}Z + XYZ\overline{Z}\)

### 4 Switching Circuits

**A Switch as a Boolean Variable**

As we did in Sec. 1, we represent the closed position of a switch by 1 and the open position by 0. As before, we designate a switch by a capital letter, \( X, Y \), and so on.
Section 4 * Switching Circuits

If two switches operate so that they open and close simultaneously, we will give
them the same letter. If they operate so that one is always open when the other is
closed, and vice versa, we label one switch as the complement of the other, say, $X$ and $\bar{X}$.

The AND, OR, and NOT Operators

As in Sec. 1, two switches $X$ and $Y$ in series are denoted by $XY$, and in parallel are
denoted by $X + Y$ (Fig. 13a and b). In Fig. 13c, the switch position $\bar{X}$ is always
opposite to that of $X$, and hence represents the NOT operator.

Representing a Switch Circuit by a Boolean Expression

Now that we can represent a pair of switches by a Boolean expression, we go on to
represent circuits with many switches in combination by more complex Boolean ex-
pressions.

*** Example 37: Write a Boolean expression for the circuit of Fig. 14a.

Solution: Note that switch $C$ appears twice. This means that they work in unison,
and that they are always in the same state. Also, switches $C$ and $\bar{C}$ work in unison,
but are always in opposite states.

![circuit_diagram](image)

FIGURE 14

We first combine switches $B$ and $C$ in the upper branch. Since they are in
series, we write $B \cdot C$ or simply $BC$. This pair is in parallel with switch $\bar{C}$ in the
lower branch so we write

$$BC + \bar{C}$$

This entire combination is in series with switches $A$ and $C$, so we write

$$A(BC + \bar{C})C$$

or

$$AC(BC + \bar{C})$$

Simplifying Circuits with Boolean Algebra

Once a circuit has been represented by a Boolean expression, simplification of the
expression may lead to simplification of the circuit.

George Boole's book on the algebra of
propositions was published in 1854,
before the invention of electricity. Eighty-four years later, Claude E.
Shannon wrote a paper showing how
Boole's algebra could be applied to
relay and switching circuits. Boolean
algebra is now used more in this
application than for the analysis of
propositions or the algebra of sets.

The symbols now in current use for AND and OR, as well as the meanings of 0
and 1, are the reverse of those used by Shannon.
**Example 38:** Simplify the circuit of Fig. 14a.

**Solution:** Starting with the Boolean expression from the preceding problem, we use law 10a to get

\[ AC(BC + \overline{C}) = A(BCC + C\overline{C}) \]

But by laws 7a and 8a,

\[ CC = C \quad \text{and} \quad C\overline{C} = 0 \]

so

\[ A(BCC + C\overline{C}) = A(BC + 0) = A(BC) = ABC \]

But \( ABC \) is the Boolean expression for three switches in series (Fig. 14b). This simpler circuit will perform identically to the one in Fig. 14a.

---

**Drawing a Circuit from a Boolean Expression**

Earlier we wrote a Boolean expression for a given switch circuit, and now we do the opposite. Starting with a Boolean expression, we draw a corresponding switch circuit. Any variables connected by the AND operator are placed in series, and those connected by the OR operator are placed in parallel.

**Example 39:** Draw the circuit represented by the expression

\[ X(\overline{Y} + Z) + \overline{X}YZ \]

**Solution:** The first term has switch \( \overline{Y} \) in parallel with \( Z \), and both of these in series with \( X \) (Fig. 15a). The second term, \( \overline{X}YZ \), represents three switches \( \overline{X}, Y, \) and \( Z \) in series (Fig. 15b). Then both these groups are connected in parallel with each other (Fig. 15c).

---

**Designing a Switch Circuit to Perform a Given Function**

1. Make a truth table for the circuit.
2. Write a Boolean expression from the truth table.
3. Simplify the expression.
4. Draw a circuit to match the expression.
Example 40:
A stairway (Fig. 16) is to have a light that can be turned on or off by the switch at
the head or at the foot of the stairs. Design the circuit.

Solution:
1. Make a truth table: Let us start with both switches in position 0, with the light
off. Then if we put switch B into position 1, the light should come on.

<table>
<thead>
<tr>
<th>Switch A</th>
<th>Switch B</th>
<th>Light</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>Off</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>On</td>
</tr>
</tbody>
</table>

Putting switch A into position 1 should now turn the light off.

<table>
<thead>
<tr>
<th>Switch A</th>
<th>Switch B</th>
<th>Light</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>Off</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>On</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>Off</td>
</tr>
</tbody>
</table>

Then putting switch B into position 0 should turn the light back on.

<table>
<thead>
<tr>
<th>Switch A</th>
<th>Switch B</th>
<th>Light</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>Off</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>On</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>Off</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>On</td>
</tr>
</tbody>
</table>

If we represent “light on” by 1 and “light off” by 0, we get the truth table,

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>Light</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

We will see in Sec. 5 that this is the truth table for the exclusive OR.

2. Write the Boolean expression: If we now represent the 0 positions of the
switches by $\bar{A}$ and $\bar{B}$, and their 1 positions by $A$ and $B$, the second row of the
truth table shows that the switch combination $\bar{A}B$ will turn the light on. Further,
the fourth row shows that the combination $A\bar{B}$ will also turn the light on. Since
either combination will work, we connect them with the OR operator. Thus the
truth table for the expression

$$\bar{A}B + A\bar{B}$$

will be identical to the one for our switch problem.
3. **Simplify**: The expression is already as simple as possible.
4. **Draw the circuit**: Our Boolean expression calls for switches $\bar{A}$ and $B$ in series, and $A$ and $\bar{B}$ in series, with each of these pairs in parallel (Fig. 17a). Then switches $\bar{A}$ and $A$ can be combined into a single switch, and $B$ and $\bar{B}$ into a single switch (Fig. 17b).

![Figures 17a and 17b showing a stairway light circuit.](image)

**EXERCISE 4 • Switching Circuits**

Write a Boolean expression for each circuit.

1. Figure 18a.
2. Figure 18b.
3. Figure 18c.
4. Figure 18d.

![Figures 18a, 18b, 18c, and 18d showing different circuits.](image)
Section 5  Logic Gates

Simplify each circuit.

5. Figure 19a.  6. Figure 19b.
7. Figure 19c.  8. Figure 19d.

![Circuit Diagrams](image)

**FIGURE 19**

Without simplifying, draw the circuit represented by each expression.

9. \(X(Y + Z) + XYZ\)  10. \(XY(ZW + AB) + AC\)
11. \(X + YZ + XZ\)  12. \((X + \overline{Y})(\overline{X} + Y)Z\)

Draw a truth table for each Boolean expression.

13. \(XY + XZ\)  14. \(X(\overline{Y} + \overline{Z})\)

Draw a truth table for each circuit.

15. Figure 18a.  16. Figure 18b.
17. Figure 18c.  18. Figure 18d.

5  Logic Gates

In our chapter on binary numbers we said that computers perform arithmetic using binary rather than decimal numbers, and then we did binary addition and subtraction, and so forth. But how does a computer actually add or subtract binary numbers? We will see how in this section. First we describe the operation of four types of devices called logic gates.

The AND Gate

An AND gate is shown symbolically in Fig. 20. We will use the Boolean values 0 and 1 to represent the voltage level at any terminal of a logic gate. The actual voltages are not important here; what we are concerned with is only whether a terminal is at one state or another.

An AND gate can have any number of input terminals. The output \(C\) is 1 only when all the inputs are 1.
Actual AND gates may be constructed using relays, diodes, transistors, and so forth. We will not be concerned here with the actual circuitry used for the various gates, only with the logic of their operation.

The OR Gate

An OR gate (Fig. 21) is one whose output is 1 if either of the inputs is 1. As with the AND gate, an OR gate can have any number of inputs. The output is 1 if any one of the inputs is 1.

Exclusive OR Gate

The output of the exclusive OR gate (Fig. 22) is 1 when either input is 1, as for the ordinary OR gate. This gate differs, however, in that the output is 0 when both inputs are 1, as in the following truth table.

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>A ⊕ B</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Note the symbol ⊕ for the exclusive OR.

The NOT Gate, or Inverter

The NOT gate, also called an inverter, is usually represented by a triangle with a circle at its vertex (Fig. 23). The output of an inverter always has the opposite state as the input. Inversion is often shown by placing a circle at a terminal of another logic gate, as in Fig. 24.

Converting a Boolean Expression to a Logic Circuit

Suppose that we have a Boolean expression that describes some operation we wish to carry out, and we need to design the circuit to do it.

Example 41: Design the circuit to implement the Boolean expression

\[ \overline{AB} + CD \]

Solution: We start at the output and work backward. Our expression has two terms joined by an OR operator, so we draw an OR gate with inputs \( \overline{AB} \) and \( CD \) (Fig. 25a). We next show \( \overline{AB} \) as the output an AND gate with inputs \( \overline{A} \) and \( B \), and \( CD \) as the inverted output of an AND gate with inputs \( C \) and \( D \) (Fig. 25b). We then use an inverter to convert \( A \) to \( \overline{A} \) (Fig. 25c). Finally, we draw the circuit more simply by showing each inversion as a circle at the input or output of the other devices (Fig. 26).
Writing a Boolean Expression for a Given Logic Circuit

Suppose now that we have a logic circuit and wish to write a Boolean expression for the output. We proceed as in the following example.

*** Example 42: *** Write a Boolean expression for the output of the logic circuit in Fig. 27.

Solution: We start by labeling each input. Then we proceed from left to right, putting the output of each gate right on the circuit diagram until the output of the last gate is reached. In this example, the output is

\[(A + B)\bar{C} + DE + \bar{F} + \bar{G}\]
Binary Addition

There are two general methods for binary addition, serial and parallel.

Serial addition requires just one input wire for each number, with the digits being represented by a pulse train, usually with the least significant bit first.

Parallel addition requires one input wire for each binary digit. Thus to add two 8-bit binary numbers would require 16 input wires. Each input would be high (1) or low (0) depending on whether the digit is 1 or 0. We will show only parallel addition here.

Parallel Addition of Two Binary Digits

When both bits are 0 the sum is zero, when one bit is 1 the sum is 1, and when both bits are 1, the sum is 0 with a carry of 1. The circuit used to accomplish this addition is called a half-adder (Fig. 28). The two input lines carry the two bits to be added, and the two outputs are the sum and the carry.

Notice now that the carry is 1 only when both X and Y are 1. Thus the carry can be obtained by using an AND gate.

Further, the sum is 1 if either X OR Y is 1, but not when both are 1. This is exactly the function of the exclusive OR gate. Thus the half-adder can be made by combining two basic logic gates, the AND gate for the carry and the exclusive OR gate for the sum, as in Fig. 29.

Addition of Three Binary Digits

A circuit called a full adder (Fig. 30) is used to add three 1-bit binary numbers. The three inputs are usually the bits X and Y from two numbers to be added, and a carry from a preceding addition. The outputs are a sum and a carry, as with the half-adder.

We want sum and carry = 0 when all inputs are 0, sum = 1 and carry = 0 when any one input is 1, sum = 0 and carry = 1 when any two inputs are 1, and sum = 1 and carry = 1 when all three inputs are 1.

<table>
<thead>
<tr>
<th>Carry-in</th>
<th>X</th>
<th>Y</th>
<th>Sum</th>
<th>Carry-out</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</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>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</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>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

To add three numbers, we can add any two of them, then add their sum to the third. We do that here, using a half-adder to add X and Y, and another half-adder to add the carry-in to the sum of X and Y, as in Fig. 31.
When either half-adder has a carry, the OR gate passes it along to the carry-out terminal. Note that both half-adders cannot have a carry, for if half-adder 1 had a carry it must have had a sum of 0, thus making it impossible for half-adder 2 to have a carry.

**Addition of Binary Numbers of Any Length**

The next step is to add two binary numbers $X$ and $Y$, each of which has many bits. Let us assume numbers having a length of four bits.

Let us indicate the bit position of each number with a subscript. Thus, $X_1$ is the least significant bit in the number $X$ and $Y_1$ is the most significant bit in the number $Y$. Our addition problem would then look like this:

```
X_4 \ X_3 \ X_2 \ X_1
+ Y_4 \ Y_3 \ Y_2 \ Y_1
```

where each $S$ stands for a sum.

Figure 32 shows a circuit to add two 4-bit binary numbers $X$ and $Y$. The least significant bits $X_1$ and $Y_1$ are added with a half-adder. The next higher bits $X_2$ and $Y_2$ and the carry $C_1$ from the first addition are combined with a full adder, and so on, the carry from each addition becoming an input for the next addition.

**EXERCISE 5 • Logic Gates**

Without simplifying, design the circuit to implement each Boolean expression.

1. $A + BC$
2. $(X + Y)Z$
3. $(A + B\overline{C}) + \overline{A}$
4. $X\overline{Y} + \overline{X}Y$
5. $X + XY + X\overline{Y}$
6. $X\overline{Y}Z + \overline{Y}Z + X\overline{Y}$

Simplify each expression and draw a logic circuit that will implement it.

7. $XY + X\overline{Y} + \overline{X}Y + X\overline{Y}$
8. $(XY + Z)(W + XY)$
9. $X\overline{Y}Z + X\overline{Y}$
10. $XYZ + \overline{X}YZ + X\overline{Y}$

Write a Boolean expression for the output of each logic circuit.

11. Figure 33a.
12. Figure 33b.
13. Figure 33c.
14. Figure 33d.

**FIGURE 33**

(a)  (b)  (c)  (d)
### REVIEW PROBLEMS

1. If \( X = 1, Y = 0, \) and \( Z = 1, \) evaluate \((X \cdot Y) \cdot (Y \cdot Z) \cdot (X \cdot Z)\)
2. Simplify: \( X + Z + X + Y \cdot Y \)
3. Draw the switch circuit represented by the expression \( \bar{A}BC + A(B + \bar{C}) \)
4. Simplify: \( XYZ + WXYZ \)
5. Write a Boolean expression for the circuit of Fig. 34.

![FIGURE 34]

6. Simplify: \( XYZ + WXYZ \)
7. If \( A = 1, B = 1, \) and \( C = 0, \) evaluate \((A + C) + AB\).
8. Simplify: \( (Y + ZW)X \)
9. Simplify the circuit of Fig. 35.

![FIGURE 35]

10. Simplify: \( Y \cdot 0 + Z \cdot 0 + W \cdot 1 + 1 + X \)
11. Design the switch circuit to implement the Boolean expression \( \overline{WZ} + XY. \)
12. Design a logic circuit to implement the expression in Problem 11.
13. Simplify the expression: \( \bar{X}(1 + X) \)
14. Write a Boolean expression for the output of the logic circuit in Fig. 36.
15. Simplify: \( Z + X + Y \bar{Y} + Z + \bar{X} \)
16. If \( X = Y = 0, \) and \( Z = W = 1, \) evaluate \((\bar{X} + \bar{Y}) \cdot (\bar{Z} + W)\)

![FIGURE 36]